Development of Electric Formula SAE Real-time Telemetry Software

Frank Yi Tan

This report is submitted as partial fulfilment
of the requirements for the Master Degree of the
School of Electrical, Electronic & Computer Engineering,
The University of Western Australia,
2011

Abstract

The increased popularity of Formula SAE event all over the world has brought an intensive need for an open standard race car telemetry system. The Renewable Energy Vehicle team (REV) at the University of Western Australia is keen on developing such a system for electric Formula SAE cars.

The software part of the telemetry system (named *Crystal Ball*) has been developed to receive, interpret and visualize real-time data collected from the Formula SAE car. It is designed to provide race engineers an insight into all aspects of the race car. It is expected that *Crystal Ball* would be released as open source software, to enable collaboration among different teams for further development, and finally to substitute its commercial counterparts.

This dissertation documents the development process of *Crystal Ball*. It begins by defining the role of the project in the telemetry system, an analysis of existing solutions, followed by an evaluation of essential features and limitations of these solutions. The scope of the project and the requirement is then specified based on iterative requirement engineering, existing system evaluation and resource constraints. The dissertation presents the design and implementation of *Crystal Ball*, including other design and implementation options. The rationale of the selections is discussed. The software is validated by data simulation, user experience survey and expert evaluation.

Keywords: Telemetry, real-time, simulation, visualization, electric vehicle, Formula SAE

Acknowledgements

I would like to formally forward my sincere thanks to the following people, who have made this project and dissertation possible.

First and foremost, I would like to thank my supervisor Prof. Thomas Bräunl for offering this project, and his guidance throughout the year.

Many thanks to Prof. Terrence L. Woodings for his guidance on software process and his help on the dissertation.

Thanks to all members of the REV team. It has been a great pleasure to work together with them. Special thanks to Thomas Walter and Paul Holmes for their assistance on the project, Ian Hooper and Matthew Webster for proof reading of the dissertation.

Appreciation extends to Winthrop Prof. Andreas Wicenec for his suggestions on *Crystal Ball* telemetry software. Thanks to many students who have joined the user experience experiment in the validation phase of the project.

Furthermore, many thanks to GNU free software foundation and many open source communities. The project would not be possible without their work.

Finally, I would like to thank my dear Vivian, my family and friends for all their support, care and encouragement.

Contents

List of Figures

List of Tables

Listings

Glossary

2D	Two Dimensional.
3G	3rd Generation Mobile Telecommunications.
API	Application Programming Interface.
ASCII	American Standard Code for Information Interchange.
CIIPS	Centre for Intelligent Information Processing Systems at UWA.
CSV	Comma Separated Values.
GNU/GPL	GNU General Public License.
GPS	Global Positioning System.
GPX	GPS Exchange Format.
GSM	Global System for Mobile Communications.
GUI	Graphical User Interface.
HCI	Human Computer Interface.
HDOP	Horizontal Dilution of Precision.
ICT	Information and Communications Technology.
IDE	Integrated Development Environment.
IP	Internet Protocol.
LAN	Local Area Network.
LGPL	GNU Lesser General Public License.
MCU	Micro-controller Unit.
MDI	Multiple Document Interface.

NMEA	The National Marine Electronics Association.
OS	Operating System.
PC	Personal Computer.
RC	Release Candidate.
REV	Renewable Energy Vehicles Laboratory at UWA.
RF	Radio Frequency.
RPM	Revolutions per Minute.
SAE	Society of Automotive Engineers.
SD	Secure Digital.
SVN	Apache Subversion.
TCP	Transmission Control Protocol.
UDP	User Datagram Protocol.
USB	Universal Serial Bus.
UTC	Universal Time Coordinated.
UWA	The University of Western Australia.

CHAPTER 1

Introduction

1.1 Formula SAE

Formula SAE is an international Formula-style race car competition organized by the Society of Automotive Engineers. It aims to promote engineering excellency of university students. This competition dates back to 1978 and was originally named SAE Mini Indy [24]. The Formula SAE name was adopted to reflect the road-racing nature of the event and the increased engineering content [25].

The scenario of the Formula SAE project is that a student design team is contracted with a fictional manufacturing company to develop a small Formula-style race car.

"The prototype race car is to be evaluated for its potential as a production item. The target marketing group for the race car is the non-professional weekend auto-cross racer." [25]

The Formula SAE competition encompasses all aspects of automotive industry. The student team must go through research, design, manufacturing, testing, marketing, management and financing stages to build a competitive solution for the competition event.

1.2 Formula SAE Electric

Formula SAE Electric started from Formula SAE Hybrid, which is sponsored by SAE International as a spin-off of the Formula SAE competition based on hybrid vehicle technology. Formula SAE Hybrid emphasizes drive train innovation and fuel efficiency in a high-performance applications [25].

Originally, electric-only race cars in the competition were defined as "Hybrid-in-Progress" cars, which needed to be upgraded to full hybrid power-trains the next year. The constraint is relieved in *Formula Student Electric* event in Germany, which becomes a competition event for electric-only cars.

As electricity is considered to be a clean and sustainable form of energy, many university teams are beginning to build Formula SAE cars powered solely by electricity to explore electric vehicle technology and attract public awareness for renewable and sustainable energy. The REV lab at the University of Western Australia is one of them.

1.3 Race Car Telemetry

To optimize the performance of a race car, a range of information about the car must be able to be measured and analyzed. In a professional Formula One car, hundreds of sensors are embedded there, which make a race car one of the most heavily instrumented objects in the world [28]. The sensor data is constantly gathered and transmitted to the rest of the racing team for real-time analysis. This process is known as telemetry.

A telemetry system in racing world is not only a group of sensors. It also includes a data gathering system, a wireless communication system and data diagnostic software [2]. Telemetry is formally defined as the use of telecommunications for automatically indicating or recording measurements at a distance from the measuring instruments [14]. Telemetry allows the race engineers to interpret a vast amount of data collected during the race, and use this information to properly tune the car for optimum performance.

There are three parts in a telemetry system. The first part is an embedded controller unit facilitating sensor data collection and logging. The second part is the wireless communication module, which can be based on GSM, 3G or RF. It transmits data packets from the car to a PC for further analysis. The third part is a PC application that receives, interprets and analyzes the data for race engineers' assistance.

1.4 The Project & This Dissertation

The ”Formula SAE Telemetry Software” project fits into the third part of the telemetry system. It aims to develop a desktop application (named *Crystal Ball*), which can provide race engineers a simple interface to analyze telemetry data visually in real-time. It should conform to an easy-to-follow communication standard and is intended to release under GNU/GPL license to replace its commercial counterparts. This dissertation documents the development process of the software.

The following chapter (Chapter 2) outlines the design of two pieces of main stream telemetry software for motor sports developed by commercial organizations, and a web based telemetry system developed for urban vehicles. Their features and limitations are discussed, followed by a summary of the survey. Chapter 3 specifies the requirement of this project based on iterative requirement elicitation and validation, survey of existing systems and available resources.

Chapter 4 demonstrates the system design of *Crystal Ball* with discussions on design choices. Chapter 5 explains the implementation of the design and discusses the possible ways for future development. Other options in design and implementation, and the rationale behind choices made are also discussed in these two chapters.

System evaluation is given in Chapter 6, where data simulation, user experience survey and expert evaluation are used for the validation process. Chapter 7 comes with a conclusion for this project, along with suggestions for future work.

Test cases designed for the *Crystal Ball* testing are provided in Appendix A for reference of future development or maintenance. Appendix B provides the questionnaire used in user experience experiment. Appendix C shows example data files for load and export used in the software.

CHAPTER 2

Survey of Existing Systems

This chapter overviews two pieces of main stream telemetry software for motor sports developed by commercial organizations, and a web based telemetry system developed for urban vehicles. The dissertation analyzes their features, functionalities and user interfaces. The advantages and limitations of these systems are discussed.

2.1 MoTeC Telemetry System

MoTeC Pty Ltd is an Australian company focusing on motor sport technology. MoTeC has a number of data acquisition and telemetry solutions for motor sport, including both hardware and software. Sensor measurement includes temperature, pressure, forces, etc. Telemetry equipments use radio frequency or GSM networks. Radio frequency relies upon a set of spread spectrum radio modems, which transmit data at up to 115200 baud [18]; GSM telemetry makes use of GSM mobile phone network for data communications at speeds up to 9600 baud, which is approximately 1.2 KBytes per second [18].

MoTeC i2 data analysis software (figure 2.1) provides 2D data visualization (including waveform, scatter, circuit, etc.), driving simulation, and complex mathematical tools for both real time data play and log file replay.

In GUI design, MoTeC i2 group graphic objects in pages. The available objects [17] includes:

a. Graphs (waveform, scatter, etc.)

b. Dial Gauge

c. Numeric Display

d. Status Light

e. Steering Wheel

f. Track Map

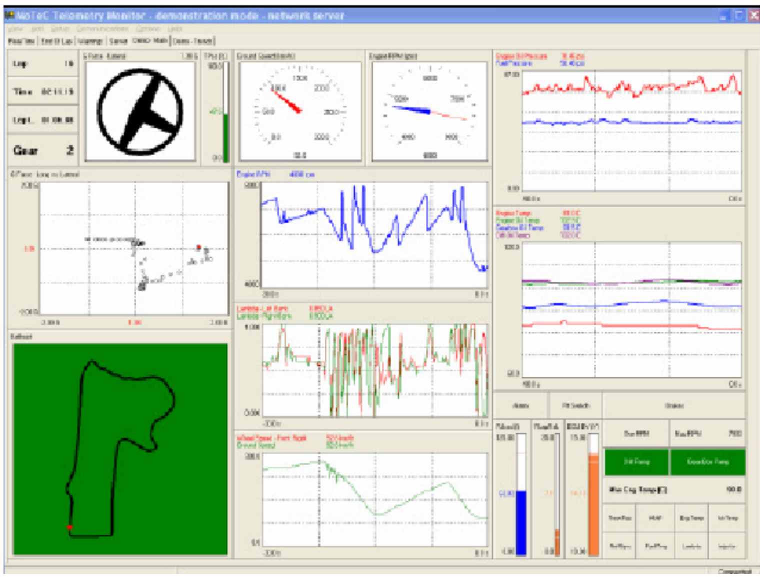


Figure 2.1: MoteC i2 Telemetry Software GUI Overview

The general features of MoTeC i2 software is:

a. Objects display one or more channel values.

b. Tab based page selection.

c. Configurable objects display on each page.

d. Copy/paste objects between pages.

e. Dragable and resizeable objects.

f. Configurable warning alarm setting.

g. Snapshot for each view.

h. Reference data set creation.

i. Data share between PCs.

j. Data export.

MoTeC i2 software is a powerful application. It meets most of the needs of a motor sport team, but it also has several limitations.

First of all, MoTeC telemetry system is a proprietary system. The communications protocol and meta-data standard is not open to the general public. As a result, the telemetry software relies on MoTeC telemetry hardware, and cannot work on self-designed systems. This limitation greatly eliminates a race car's design choices.

Second, MoTec i2 aims to include all features for professional motor sports and make them customizable. However, the design of the user interface is so complex, which can easily overwhelms users (figure 2.2). This goes against the modern HCI and GUI design concept [15].

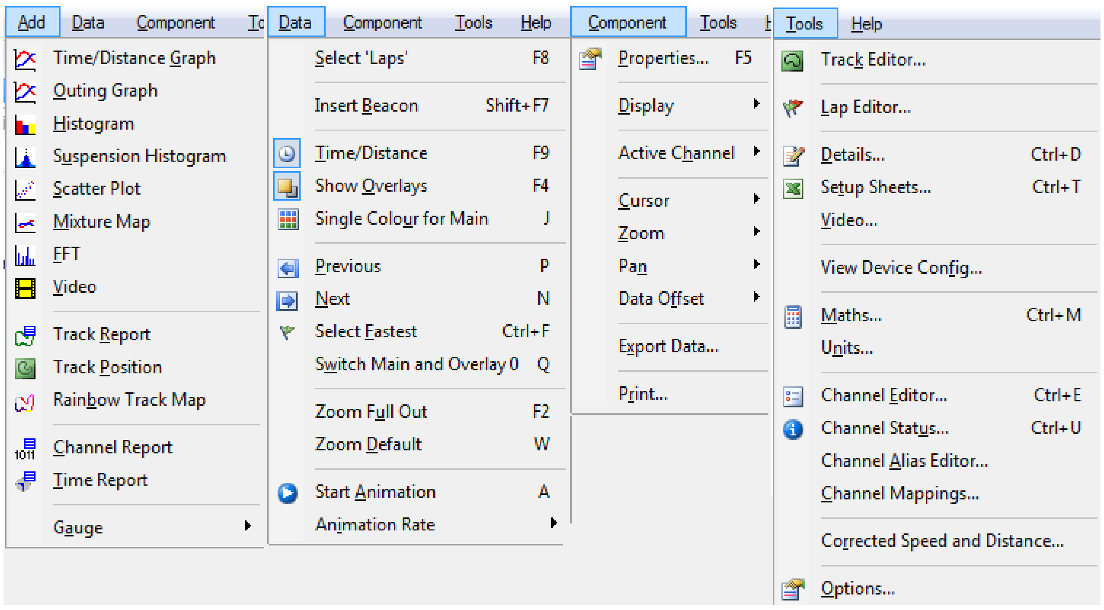


Figure 2.2: Complex Menus in MoteC i2 Telemetry Software

2.2 ATLAS Telemetry Software

ATLAS (Advanced Telemetry Linked Acquisition System) is a software package developed by Mclaren Electronics to obtain, display and analyze data from control systems, including those used in motor sport applications [16].

"ATLAS is used by the professional data analyst working with data acquired by

telemetry or uploaded from a data logger. ATLAS is appropriate for an individual data analyst or for many engineers all monitoring telemetry together [16]."

The features of ATLAS includes:

a. Highly customizable, Workbook containing Pages and Displays [16].

b. Time line based display for easy navigation.

c. Reference data for live data comparison.

d. Fast data handling.

e. Extensive help with context sensitive links [16].

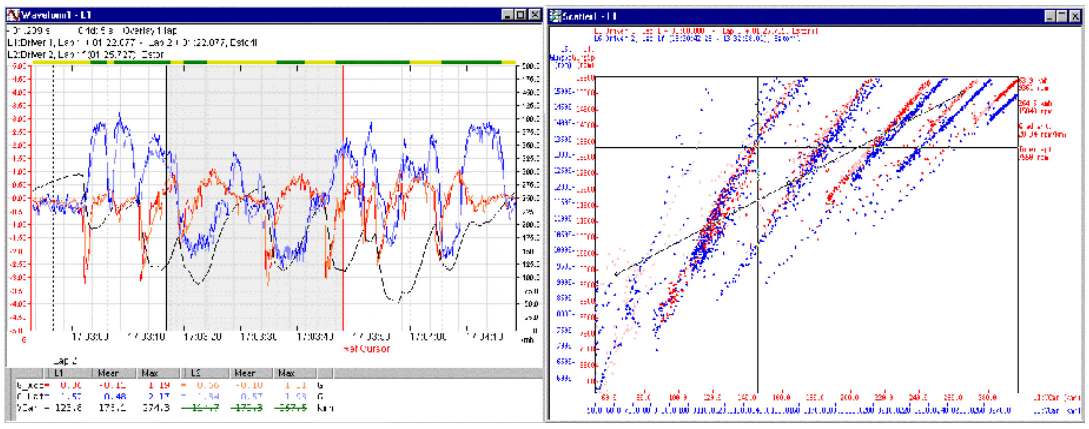


Figure 2.3: ATLAS Telemetry Software: Waveform & Scatter

2.3 REV Web Based Vehicle Telemetry

The REV web based vehicle telemetry system [23] is developed by John Pearce in 2010. REV Getz and Lutos, two electric urban vehicles converted in REV, are equipped with data loggers that store sensor data and transmit it via the GSM network. Recorded/transmitted sensor data includes the vehicle's GPS position, charge status, current, voltage, and activation of heater, air-conditioner and head lights [23]. The user interface web pages for PCs and mobile device are shown in Figure 2.4 and 2.5.

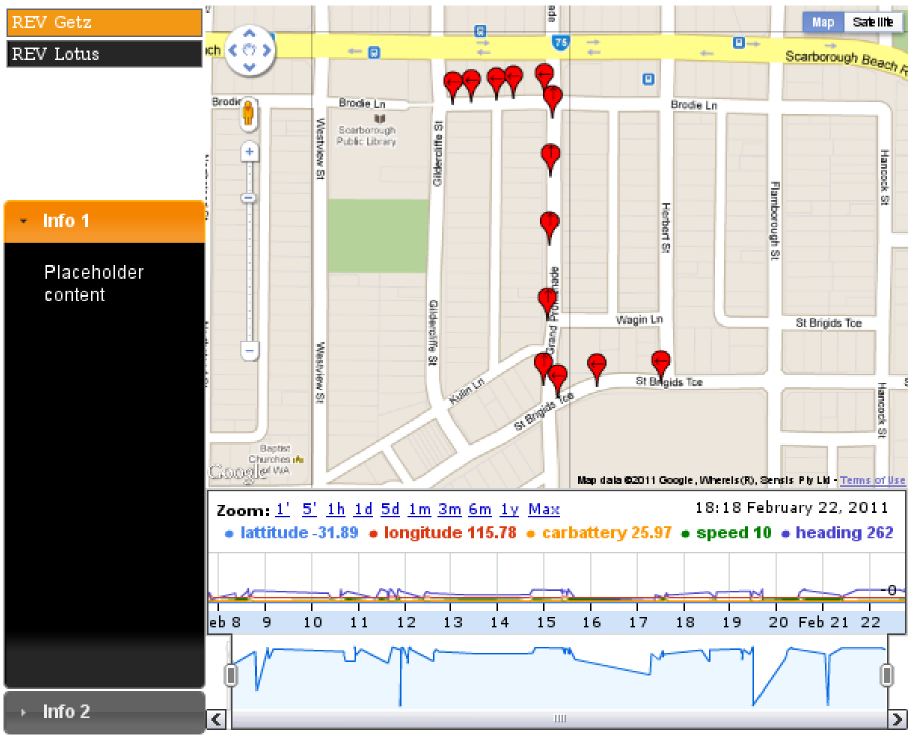


Figure 2.4: REV Urban Vehicle Telemetry Web Page for PCs

The REV web based vehicle telemetry system is excellent for monitoring urban vehicles. However it is not enough for Formula SAE telemetry for the following reasons.

a. Formula SAE is always running at a much higher speed than urban vehicles. The update rate of REV web based vehicle telemetry system is upto 0.2 Hz, which is too low compared to what Formula SAE real-time measurement requires.

b. Formula SAE cars requires much broader measurement of the vehicle than what the REV web based vehicle telemetry system offers. Examples of the measurement include three way G-forces, yaw rate and the torque of each electric motor. These are very important for race car monitoring.

c. On the contrary, some properties such as heater, air-conditioner and head lights are not applicable to motor sports vehicles.

d. REV web based vehicle telemetry system has a strong focus on city map, as the geographic position is very important to an urban vehicle. However, the Formula SAE car, which always runs on a track in a specific area, is not designed for urban traveling. It does not require the absolute geographic location of the car. Instead, a track make more sense to motor sports than a city map.

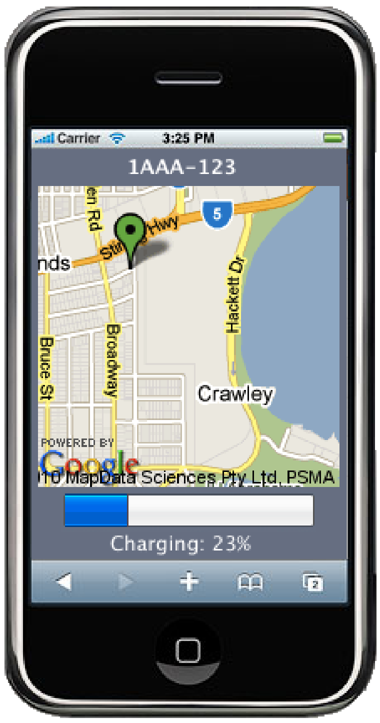


Figure 2.5: REV Urban Vehicle Telemetry Web Page for Mobile Devices

2.4 Summary

The REV web based vehicle telemetry system is not appropriate for real-time monitoring of a Formula SAE car. MoTec i2 and ATLAS are both very powerful telemetry software in race car system monitoring. From the feature analysis of both applications, their functionalities can be summarized as follows:

a. Real-time data visualization mode and simulation mode.

b. Graph plotting functionality including waveform and scatter.

c. Race track and car trajectory display.

d. Dial gauge visualization.

e. Numeric display.

f. Reference data creation and display concurrently with live data.

g. Configurable alarm warning setting.

h. Snapshot for each view.

i. Time line based display and navigation.

j. Configurable objects display.

Among these features, (a), (b), (c), (d), (e), (f) and (h) are considered essential by REV Formula SAE members [27] for a real-time telemetry application.

The key disadvantages of existing telemetry software are price, hardware bounding and a learning curve for users.

The most important reason for price and hardware bounding issue is that telemetry software is not general purpose software. It aims for professionals in different domains and the requirement for different domains are usually very different from each other. Lots of these types of software are designed and customized for specific purposes, and in this occasion, race car monitoring. The lack of massive needs makes it hard for open source communities to step in or survive in this market, which makes this type of software highly commercial.

Further more, the specific design and customization for certain purposes makes software development teams prefer to design their own protocols for data communication. This on the one hand optimizes the performance of the system as a whole, and on the other hand, promotes the sales of their own hardware.

The learning curve of such software is caused by the user group aimed by the software development teams. As telemetry software are usually designed for professionals in a specific field, the existing software is designed to include everything inside to accommodate all different needs. Moreover, to organize so many features and functionalities, the existing software tries to provide many configuration options for users to customize the software according to their own preference. These made the software even more difficult to use.

To address the first two problem is where this project is originated. The increased popularity of Formula SAE events all over the world has brought intensive needs for wireless and real-time car monitoring. an open source telemetry system with a widely used standard is apparently the best choice for these university teams.

From another perspective, the Formula SAE teams from universities all around the world have many talented software engineers who are keen on joining the development of an open source telemetry software. This makes the project's future promising.

Making the telemetry software easy to use is also a key aspect in this project. The author optimized and proved the GUI design of the software in several ways. This is further discussed in Section 4.3 and 6.3 of this dissertation.

CHAPTER 3

System Requirement

This chapter specifies the requirement of the telemetry software project based on iterative requirement elicitation and validation, survey of existing systems and the constraints of time and resources. References to design, implementation and test cases for critical features are provided for forward traceability of the development.

3.1 Requirement Engineering

The development of *Crystal Ball* telemetry software adopted Extreme Programming [12] software process. In Extreme Programming, requirement is treated as a list of user stories, which is subject to change from time to time.

The user stories in this project are mainly obtained and balanced from the following ways.

- Interview of Formula SAE race engineers, instrumentation engineers and an experienced software developer.
- Observation of Formula SAE race engineers and instrumentation engineers.
- Survey on existing commercial telemetry software.
- Iterative validation of software prototypes.
- Constraints on time and resources.

A series of prototypes and releases have been distributed on CIIPS SVN server during the development life cycle of the software. These releases have been evaluated by Formula SAE team members for validation purpose. New user stories and scope have been added to the next iterations of development from all evaluation events.

The software is still under development following the iterative process, which means the requirement and data format is subject to change according to what customer requires in the future. By the time the dissertation is written, a series of user stories have been finalized, which are specified in the following sections.

3.2 Operation Modes

Three operation modes, real-time mode, simulation mode and network mode, are required in the software.

a. Real-time mode of Crystal Ball is to obtain data from the serial port, which connects to the *XBee pro* Zigbee transceiver, interpret and visualize the data in real time. The visualization methods are specified in Section 3.4. The implementation of this user story is discussed in Section 5.1.1, and the test case is presented in Appendix A.2.

b. Simulation mode is to read a file, which is logged in an SD card in a Formula SAE race car when it was running on the track. The software shall be able to simulate and replay the logged data with methods specified in Section 3.4. The implementation of this user story is discussed in Section 5.1.2, and the test case is presented in Appendix A.3.

c. Network mode is to set the program as a network server or client to enable real-time data share between different desktops. This mode is meant to be used for the PC which directly connects to the RF transceiver to share data with other PCs. The visualization methods are specified in Section 3.4. The implementation of this user story is discussed in Section 4.5 and 5.1.1, and the test case is presented in Appendix A.4.

3.3 Meta Data

The data involved in the telemetry system are collected by the embedded sensors [30] installed in a Formula SAE car.

3.3.1 GPS Data

a. UTC date and time is the date and time standard based on International Atomic Time by which the world regulates clocks and time. It shall be converted into local date and time in the program.

b. Latitude and Longitude are the coordinate of the car position.

c. Speed is the speed of the car over ground. It shall be converted to km/h in the program.

d. Bearing is the Magnetic variation in degree.

e. Altitude is the distance over the mean sea level in meters.

f. HDOP is the horizontal dilution of precision, the relative accuracy of horizontal position.

g. Satellites is the number of satellites in view.

GPS data processing is explained in Section 5.2.

3.3.2 Other Embedded Sensor Data

a. Milliseconds since the car MCU was turned on.

b. Adjusted speed of the car. It shall be converted into km/h in the program.

c. Percentage pushed of the acceleration pedal.

d. Percentage pushed of the brake pedal.

e. Forward G-force of the car.

f. Lateral G-force of the car.

g. Yaw rate of the car in degree per second.

h. Back left motor torque.

i. Back right motor torque.

j. Front left motor torque.

k. Front right motor torque.

The design of data format is explained in Section 4.4.

3.4 Visualization Methods

Data visualization is an essential task of *Crystal Ball* telemetry software. According to the features discussed in existing systems analysis in Section 2.4, *Crystal Ball* shall encompass the following data visualization methods.

3.4.1 Waveform Plotting

Waveform plotting is to provide a curve plotting functionality for a series of 2D data sets according to time.

a. Time line based data plotting.

b. Reference curve shall be able to set underlaid by the data curve for comparison.

c. Reference curve shall be able to move sideways to remove the time line difference from real data curves.

d. Each plot canvas shall be able to display two curves for separate data sets excluding reference curves.

e. Customizable colors for each data curve.

f. Customizable number of plot canvas shall be provided: up to five canvas shall be displayed at the same time.

g. Screen shot for any plot canvas and for all plot canvases.

The implementation of this feature is explained in section 5.5, and the test case is presented in Appendix A.7.

3.4.2 Scatter Plotting

Scatter plotting is to enable 2D dots plotting for a group of data sets according to any data property.

a. Both axises shall be customizable.

b. The plotting canvas shall be able to display two sets of value on vertical axis paired with horizontal axis.

c. Customizable colors for each set of data dots.

d. Screen shot for the plot canvas.

The test case for scatter plotting is presented in Appendix A.8.

3.4.3 Circular Motion

Circular Motion graph is to display real-time force and velocity decomposition, angular velocity and the curvature of the movement.

a. Visualize ground plane G-force decomposition.

b. Visualize curvature of the movement.

c. Display angular velocity.

The implementation of this feature is explained in section 5.6, and the test case is presented in Appendix A.9.

3.4.4 Track and Car Trajectory

a. Race track shall be able to be loaded from a track file.

b. Customizable display of tracks from the track file.

c. Customizable display of comment points from the track file.

d. Auto focusing functionality of the track.

e. Car trajectory shall be able to be plotted in real-time with data received from the Zigbee transceiver, simulated with data loaded from a log file or received from network.

f. Customizable display of car trajectory.

g. Auto focusing functionality of the car.

h. Zoom in & out functionalities shall be provided to zoom the track.

i. Speed, precision & number of satellites display.

The test case for track and car trajectory is presented in Appendix A.11.

3.4.5 Gauges

a. RPM & speed gauges.

b. Steering wheel gauge.

c. Battery statues.

d. Acceleration pedal & brake pedal statues.

e. Torques requested from each motors.

The test case for gauges is presented in Appendix A.6.

3.4.6 Numeric Display

All the data listed in Section 3.3 shall be converted to proper metric and displayed numerically in real-time. The test case for numeric display is presented in Appendix A.10.

3.5 Data Processing

As spikes and high frequency noise always exist in sensor data, and the MCU embedded in the car has limited computation power, the program shall provide several options to process sensor data to reduce high frequency noise and spikes. The test case for data processing is presented in Appendix A.13.

3.6 Track Summary

Data summary such as maximum, minimum, average and area for each track or track segment shall be calculated automatically. Track segment shall be customizable for users. The test case for track summary is presented in Appendix A.12.

3.7 Data Export

The program shall embodies functionalities of exporting data for external programs. The data includes

a. Processed data.

b. Processed track file and

c. Track summary file.

The exported file shall be in a format that is accessible for external programs such as Microsoft Excel and Matlab. Exported data format is designed in Section 4.4.4. Test cases for data export are presented in Appendix A.10, A.11 and A.12.

In summary, this chapter listed the requirement or user stories for the *Crystal Ball* real-time telemetry software. The test cases for each of the requirement is provided in Appendix A. In development process, the design and implementation shall be rigorously tested and verified against the requirements listed in this Chapter. The requirements list shall also keep updated for newly introduced requirements or changed requirements.

CHAPTER 4

System Design

This chapter demonstrates the design of some important parts of Crystal Ball telemetry software, including the development tool, architectural design, GUI design, data format design, and network data sharing. Other design options and the rationale behind each design choice are discussed.

4.1 Development Tool

The choice of programming language and development tool is the first thing to decide in the software design stage. This project chooses Qt Creator 4.6 as the development tool. It has the following advantages.

4.1.1 Portability

Qt Creator is a cross-platform C++ IDE using Qt libraries. It runs on Windows, Linux/X11 and Mac OS X desktop operating systems [22]. With Qt build-in libraries, the developer can do programming and testing in one platform, while re-compile the code in another platform for multiple distributions. This is important in engineering teams, where future developers and users have different preference in operating systems.

4.1.2 Performance

C++ is regarded as an intermediate-level language. It is fast in performance and therefore is widely used in systems software, application software, device drivers, embedded software, high-performance server and client applications, and hardware design. Compared to Java which runs a virtual machine between the OS and the application, C++ code is compiled into binary code, which provides an enhanced performance. This is appreciated in high speed real-time applications.

4.1.3 High Level Build-in Library

High level API from Qt C++ library and the integrated GUI designer makes it easier for programmers to write less code while making a better program. The build-in functions also reduce the need for developers to write their own low level code, which is highly error prone, and therefore improves the robustness of programs.

4.1.4 Open Source Community Support

Qt Creator LGPL version is an open source IDE with free Qt libraries. This allows many open source communities and individuals to develop open source 3rd party libraries and graphical widgets, which makes GUI design easier.

4.1.5 Popularity in REV Team

The last but a very important reason for choosing Qt Creator is its popularity in REV team. "REV_HMI" [29], an application for Lotus Electric car PC, is developed by Thomas Walter in 2010. It provides a good guidance for software development in Qt in REV team, and offers re-usable components and test data for software development. This is also an important reason why Qt C++ is preferred over Java in this project.

4.2 Architectural Design

The architectural design of *Crystal Ball* and the functionality blocks of the program are shown in Figure 4.1.

There are six input/output sources for the application.

a. Serial port which connects to the RF transceiver as input

b. Data log file as input

c. Processed log file as output

d. GPX track file can be both input and output

e. Track summary file as output

f. Network data (in data share mode) can be both input and output

Among these six input/output, serial port data comes from the RF transceiver. Data log file is input from original logged data. The program processes the original data, and export the processed data into a processed log file. GPX file can be load into the program for track display. A car's trajectory can be exported into a new GPX track file. GPX file operations such as adding comments is also available in the program. Track summary file is generated from the program and can be exported as an output. Network data always flows from the server to the clients.

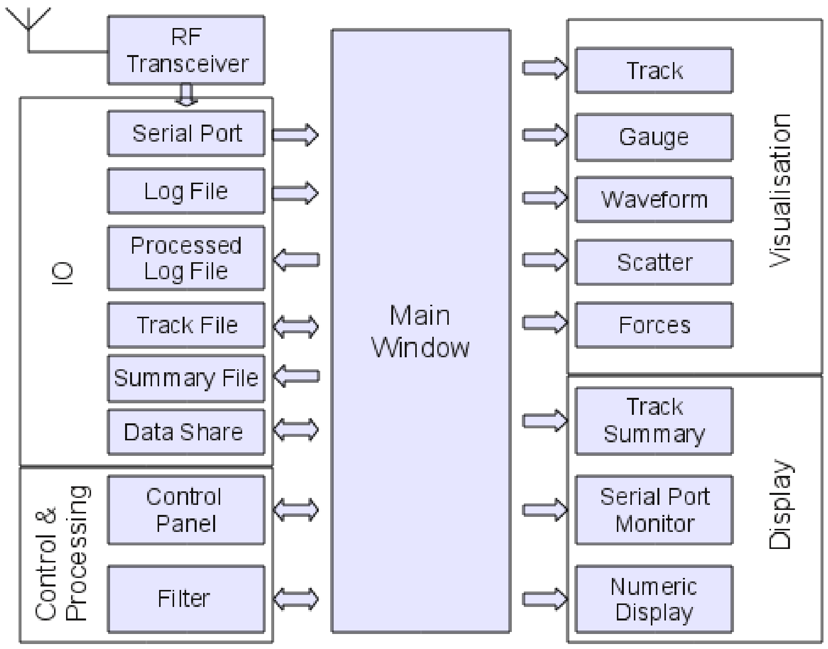


Figure 4.1: Functionality Blocks in *Crystal Ball*

The control block, which is named *Control Panel* in the program, is a dock widget that gathers all control widgets for universal control of the program. Filters is used to process data before it is displayed and visualized. Five visualization module, track, gauges, waveform, scatter and forces, illustrate data with visual effects. Three display modules, track summary, serial port monitor and numeric display, show data in original or processed way. All these modules are grouped into separate classes in the program, and are connected with each other through the Mainwindow module, which coordinate the behavior of the program and each module.

4.3 GUI Design

It is important for a desktop application to have user-friendly system interfaces [5]. *Crystal Ball* has emphasized the GUI design to simplify user operation and enhance the program's usability.

4.3.1 Application Framework

Before going to the GUI design for the application framework, it is necessary to analyze the layout of a modern GUI window. A main window provides a framework for building an application's user interface. Figure 4.2 shows the layout of a main window. It has areas reserved for a menu bar, a status bar and tool bars. The center area, named MDI area, can be occupied by any other kind of widget, such as dock widgets (discussed in Section 4.3.2) or sub-windows (Section 4.3.3).

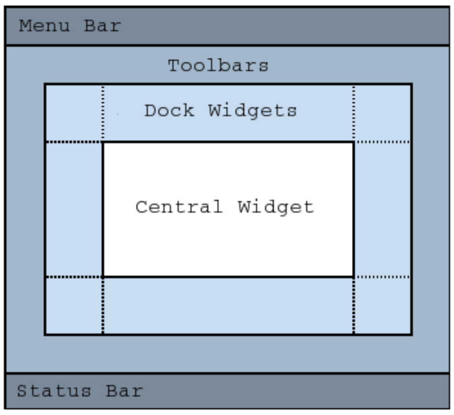


Figure 4.2: The Layout of a Standard Main Window [19]

To avoid the complex menus problem in MoTeC i2 software discussed in Section 2.1, *Crystal Ball* adopts a design, in which no menu bar exists. Figure 4.3 shows a screen shot of *Crystal Ball* main window framework. Area ① in this figure is the control panel, which gathers the control widgets of all docks and sub-windows. It is always shown on the front layer to provide user with control of the program. Similar idea is the tabbed tool bars used in Microsoft Office 2007 (Figure 4.4).

It is proved to provide users with a more accessible interface [15]. This idea has gained enormous popularity ever since, and has been adopted by many softwares.

Area ② in Figure 4.3 is a traditional status bar. It informs the user with important actions the program has done. ③ is the MDI area of the main window, where multiple sub-windows or tabbed windows reside. Sub-windows are discussed in detail in Section 4.3.3. Area ④ is the window border. Qt makes use of the system window manager to decide the appearance of this area, which makes the application the same style with other applications in the operating system. Area ⑤ is sub-windows, which is explained in Section 4.3.3. Area ⑥ is dock widgets like the control panel. Dock widgets are explained in Section 4.3.2.

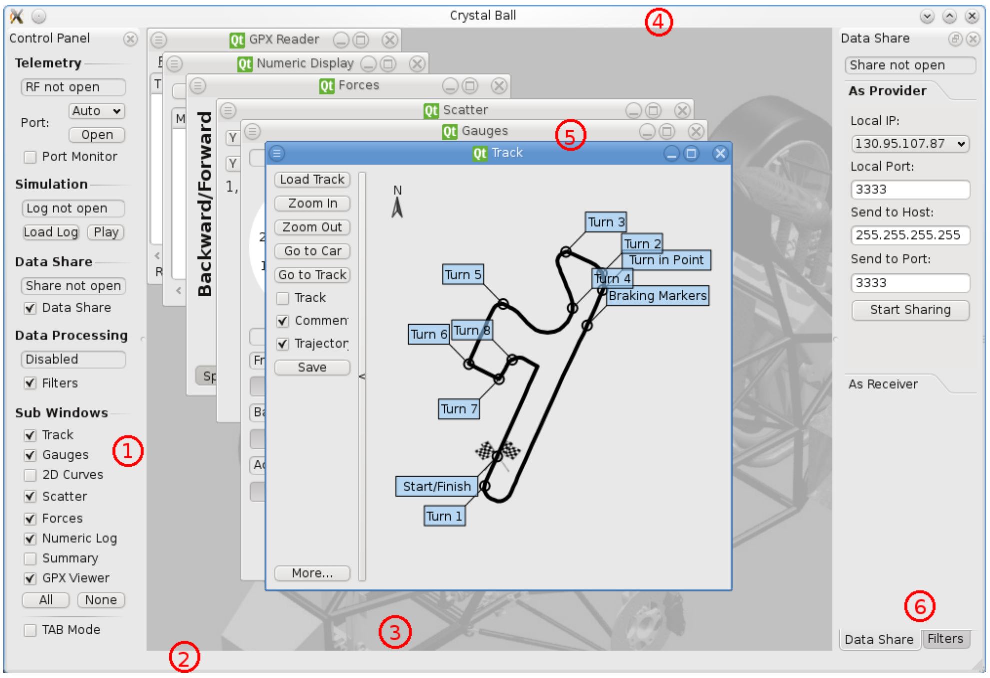


Figure 4.3: *Crystal Ball* GUI Design

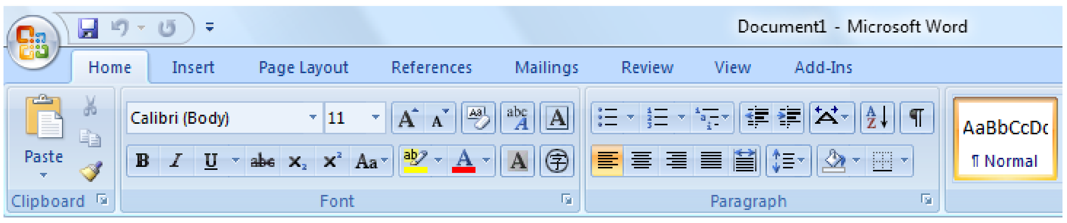


Figure 4.4: Tabbed Tool Bar in Microsoft Word 2007

4.3.2 Docks

The control panel shown in ① of Figure 4.3 is actually a so called dock widget in GUI design. A dock widget is a special window that can be docked or attached on the border of the main window. This property makes it suitable for grouping control widgets or configuration options.

In *Crystal Ball*, three docks are used for these purposes. Control panel groups the setup options for the three operation modes of the program, the sub-windows and other dock widgets (Area ① in Figure 4.3). "Data Share" dock provides options for user to setup a server or client for data sharing. "Filters" dock offers methods for data processing.

4.3.3 Sub-windows

A sub-window is a top-level window in MDI area that affiliates to the main window framework of the program. Using sub-windows offers great advantages for programs like *Crystal Ball*. First of all, it provides a way of grouping different widgets and functionalities. Moreover, it makes it possible for showing several groups of widgets at the same time, which is a great advantage over page or tabbed windows. Last but not least, it complies to the human logic of how things are organized. This significantly improves the usability of the program.

For supplement, *Crystal Ball* provides a toggle button to change sub-window display into tabbed mode for users who has other preferences.

4.4 Data Format

Data format is essential for a data process applications. It is a determining factor for the application's performance. More importantly, a format that conforms to a widely used standard makes data sharing with another programs easier. The types of data interact with the program includes communication data, logging data, track data and some exported data.

4.4.1 Communication Data

Crystal Ball is designed to receive real-time data from the Formula SAE race car. A compact data format is essential for the wireless communication. Thus, binary format is highly appreciated to reduce the size of the telemetry data [7]. Binary format for communication is the final goal of the program. Nevertheless, at this stage of the development, CSV ASCII strings are used for testing and communication system verification, quick prototyping and frequent releases purposes. The meta data is explained in section 4.4.2.

The binary format transition is scheduled in the next stage of development, which is discussed in future work in Chapter 7.

4.4.2 Logging Data

Data logging format is the format the MCU used to store collected data in the SD card installed in the car. *Crystal Ball* need to be able to access these log file in its simulation mode (Section 3.2). Data logging format goes for the least computation solution, as the MCU embedded in the car has limited computational power, while the storage of a SD card is huge compared to the data to be logged. As a result, CSV ASCII strings are the choice of the data format. The following example explains the meaning of a logged data line.

```
1    2   3   4   5     6   7    8  9   10
|    |   |   |   |     |   |    |  |   |
4756,927,258,595,1121,-93,-1879,64,64,<optional GPS sentence>
```

1) milliseconds, 4.765 seconds since turn on.
2) car speed*100, 9.27 km/h.
3) acceleration pedal value out of 1023, 258/1024 throttle.
4) brake pedal value out of 1023, 595/1024 brake.
5) forward G-force *1000, 1.1 times G-force forward.
6) lateral G-force * 1000, -0.093 times to the left.
7) yaw Rate * 1000, 1.88 degrees rotation per second (to the left).
8) back left motor output value out of 255, 64/255 torque requested from back left motor.
9) back right motor output value out of 255, 64/255 torque requested from back right motor.

10) $GPRMC or $GPGGA sentence. This is optional because the update rate of the current GPS is much slower than other sensors.

An example data log file is shown in Listing C.1 of Appendix C.

4.4.3 Track Data

Track file is used to record track points and waypoints of a race track. A track point includes the latitude and longitude of a point and optionally a comment for this point. It is used to identified a point on a race track. A waypoint is a set of latitude and longitude together with a name or comment of the point, and it is used to mark interesting points with extra text information.

Crystal Ball adopts GPX file format, which is an open and light-weight XML standard data format for the interchange of GPS data between applications and Web services on the Internet [4]. GPX format makes it easy for the program to share track data with other desktop or web applications. Race engineers can generate a GPX file from Google Map easily and load it as a track into *Crystal Ball*. It is also simple to implement reading and writing operations, as XML libraries exist for almost every popular programming language.

4.4.4 Exported Data

To make telemetry data accessible from more advanced data analysis tools such as Matlab, *Crystal Ball* is designed with functions to export different aspects of data into files. These data files include

- Processed log data file
- GPX track file
- Track summary file

The original log data is the raw data measured by the sensors embedded in the car. To let the data make more sense to race engineers, the program converts the data into numbers by standard metric unit for displaying and exportation. The processed log data file is in CSV ASCII format. An example file is shown in Appendix C.

```
1     2     3    4    5     6    7     8     9 101112 13
|     |     |    |    |     |    |     |     | | | |  |
13401,31.32,6687,0.99,0.32,-0.84,0.25,-1.976,0,0,1,1,<GPS data>
```

1) milliseconds, 13.401 seconds since turn on.
2) car speed, 31.32 km/h.
3) RPM, 6687 rpm.
4) acceleration pedal, 99%.
5) brake pedal, 32%.
6) forward G-force, -0.84 times G-force forward.
7) lateral G-force, 0.25 times G-force to the left.
8) yaw Rate * 1000, -1.976 degrees rotation per second to the left.
9) front left motor value, 0 as motor was not installed when data was collected.
10) front right motor value, 0 as motor was not installed when data was collected.
11) back left motor output value, 100% torque requested from back left motor.
12) back right motor output value, 100% torque requested from back right motor.
13) GPS data is shown below.

```
1                       2           3          4     5    6 7 8
|                       |           |          |     |    | | |
10/09/2010-10:09:32.313,-31.979283,115.815852,31.32,65.5,1,5,3.21
```

1) Local time, 10:09:32.313 on 10th September, 2010.
2) latitude, -31.979283 degree.
3) longitude, 115.815852 degree.
4) GPS speed, 31.32 km/h.
5) GPS bearing, 65.5 degree.

6) HDOP, 1 (explained in Section 5.2.3).

7) Number of satellite, 5.

8) Distance traveled in km since the first data set, 3.21 km.

The function of exporting GPX track files allows users to make track files from car trajectory, or adding comments to an existing track file. The output GPX file is in exactly the same format as the input GPX track file. An example GPX file is presented in Appendix C.

Track summary is to show users summary of different aspects of data organized by laps. The output of track summary files is also in CSV ASCII format for easy access from other programs. An example track summary file is presented in Appendix C.

```
1          2     3     4   5  6
|          |     |     |   |  |
Milisecond,45211,45211,627,NA,44584
```

1) Property of the current line of data, Millisecond.

2) The current value of the property, 45211 milisecond.

3) The maximum value of the property since the first data set, 45211 milisecond.

4) The minimum value of the property since the first data set, 627 milisecond.

5) The average value of the property since the first data set, NA for milisecond.

6) The span from the minimum to the maximum value, 44584 milisecond.

4.5 Network Data Sharing

In real-time mode, only the PC which directly connects to the RF transceiver can receive data from the Formula SAE race car. However, it is usually desirable that a telemetry software can expand to multiple desktops for several race engineers to look at different aspects of the data sets separately at the same time. *Crystal Ball* makes this possible by setting itself as a data server on one PC and a data client on other PCs.

4.5.1 Transmission Protocols

Choosing an appropriate transportation layer protocol is critical for building a server-client application, and real-time applications have specific requirements in protocol selection [11]. The most often used protocols are TCP and UDP.

- **TCP** is a connection-oriented, reliable transportation protocol. It relies on the underlying principles including error detection, retransmission, cumulative acknowledgement, timer, header field for sequence and acknowledgement numbers [10]. The connection-orient of TCP implies a hand shake process before the communication starts, which means some preliminary segments need to be sent to each other to establish a connection.
- **UDP** is a connectionless, unreliable transportation protocol [8]. It add the least necessary parts to IP, including multiplexing/demultiplexing function and some light error checking. There is no handshake or acknowledgement mechanism in UDP, and segments delivery is best-effort based, which means the communication is unreliable.

Crystal Ball chooses UDP as its transportation layer protocol for the following reasons.

a. TCP has a congestion control mechanism that throttles the transportation layer TCP sender when one or more links between the destination host become excessively congested [9]. However, real-time applications require a maximum sending rate, and do not expect to overly delay segments transmission. UDP, which has no congestion control mechanism, meets this requirement much better.

b. TCP also has an acknowledgement mechanism, in which the sender waits for acknowledgement from the receiver for every segment it sends [9]. It retransmit a data segment if a that segment gets an error or lost. UDP, which is a best-effort transmission protocol, doesn't retransmit data at any circumstances. As low latency is essential in real-time applications, some data loss can be tolerated and out-of-time data doesn't make much sense, UDP is highly preferred over TCP.

c. TCP need a three-way handshake before it establishes a connection and starts transmitting data [9]. In contrast, UDP just blasts away without any formal preliminaries [10]. So UDP does not introduce any delay to establish a connection, which is highly desirable in real-time applications.

d. A TCP segment has 20 bytes of header overhead in every segment, where UDP has only 8 bytes [10]. For an application which transmits huge amount of small data sets like *Crystal Ball*, UDP provides a better bandwidth efficiency.

e. TCP connection is point to point. Only one sender and one receiver is permitted in a connection. To transfer data from one sender to multiple receivers is not possible [10]. This means to use TCP, the application needs to setup a front server daemon which takes cares of all incoming clients requests and fork these requests to the back-end servers which do the real service. This is much more complex than UDP, in which the server can broadcast the data to all clients at the same time.

4.5.2 Application Design

As discussed at the beginning of this section, data sharing is originally designed for the PC which connects directly to the RF transceiver sharing data with other PCs. It also works for log data simulation mode, in which the PC simulating the log data can be set as a server and share its log data. However, a PC are not supposed to work as a client and server at the same time in *Crystal Ball*.

Figure 4.5 shows how *Crystal Ball* on different desktops communicate with each other. When the sever is set up, the sending process can be triggered by new coming data sets. The program packs up the data into UDP segments, and send them to the receiver or broadcast to all receivers. The receiver, which is set up as a UDP client, listens on the port specified by user, and receives UDP segments arrived at that port. When a data set is ready, the program reads the data, and does processing as required.

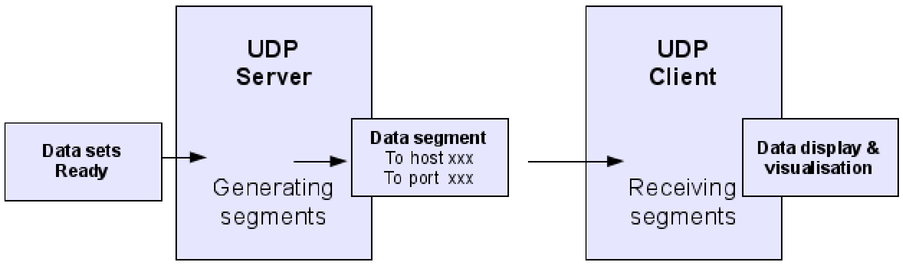


Figure 4.5: UDP Server & Client Communication

CHAPTER 5

Implementation

This chapter expounds the implementation of some critical parts of Crystal Ball, including operation modes, GPS data processing, sensor data processing, serial port access, real-time graph plotting, and automatic user settings storage. The advantages behind implementation choices are discussed. Scalability considerations and possible ways for future development are explained.

5.1 Operation Modes

According to system requirement (Section 3.2), *Crystal Ball* is supposed to have three operation modes: real-time mode, simulation mode and data share mode. This section explains how data is processed in these three modes.

Real-time mode and data share mode have the same responding mechanism for handling data. This design on the one hand simplifies the implementation of the program; on the other hand, helps to simulate and test real-time mode when telemetry hardware is not available. Simulation mode is implemented using timer's expiration mechanism.

5.1.1 Real-time & Data Share Mode

In real-time and data share mode, the length of data sets is always unknown. Each data set has a probability to get lost during transmission. Therefore, data sets received should be processed on a set by set basis. The program is set as event driven in data reading and processing. The pseudo code for the mechanism is shown in Listing 5.1.

```
    // in the program main event loop
    callDataHandlingFunctionUpon(EVENT_DATARECEIVED);
```

```
...
// in module's event loop
callProcessDataFunctionUpon(EVENT DATAPASSED);
...
// fucntions in main program
MainWindow::dataHandling()
{
  processReceivedData();
  passDataToEachModule();
}
...
// fucntions of each module
EachModule::processData()
{
  doSomethingWithReceivedData();
}
```

Listing 5.1: Pseudo Code of Data Handling in Real-time & Data Share Mode

5.1.2 Simulation Mode

```
// function of passing data sets to each module
simulateNextDataSet()
{
  stopTimer();
  passCurrentDataToEachModule();

  readNextDataSets();
  calculateTimeDifferenceBetweenTwoDataSets();

  connectTimerWithThisFunction();
  startTimerWithCalculatedTimeDifference();
}
```

Listing 5.2: Pseudo Code of Data Handling in Simulation Mode

In simulation mode, data comes from log files. The programs knows exactly how many data sets are there, and no data set is possible to get lost. This enables the program to send all data to each module once, which makes it possible for some more accurate data processing. Another important part in simulation mode is accurate time control between each data sets. Listing 5.2 shows the pseudo code for time control in simulation mode.

5.2 GPS Data Processing

5.2.1 Data Format

" GPS data format is defined by NMEA in "The NMEA 0183 Protocol" [13]. NMEA 0183 is a combined electrical and data specification for communication between marine electronic devices including GPS device. All data is transmitted in the form of sentences. Only printable ASCII characters are allowed, plus CR (carriage return) and LF (line feed). Each sentence starts with a "$" sign and ends with <CR><LF>[13].

The data sentences used in the telemetry system involves $GPRMC and $GPGGA. *GP* in the symbol stands for "Global Positioning System"; *RMC* means "Recommended Minimum Navigation Information"; *GGA* is "Global Positioning System Fix Data, Time, Position and fix related data for a GPS receiver". The protocol of these two sentences [13] is presented as follows. Only sections used in the program are explained.

```
       1         2 3       4 5         6 7   8   9
       |         | |       | |         | |   |   |
$--RMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a*hh
```

1) UTC time hh:mm:ss.sss

2) Fix statue: A - Active, V - Void

3) Absolute latitude

4) N - North hemisphere, S - South hemisphere

5) Absolute longitude

6) E - East hemisphere, W - West hemisphere

7) Speed over ground in knots

8) Heading in degree

9) Date: dd/mm/20yy

```
                                                1 2   3    4
                                                | |   |    |
$--GGA,hhmmss.ss,llll.ll,a,yyyyy.yy,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh
```

1) GPS Quality Indicator: 0 - fix not available; 1 - GPS fix; 2 - Differential GPS fix.

2) Number of satellites in view, 00 - 12

3) Horizontal Dilution of precision (HDOP)

4) Antenna Altitude above/below mean-sea-level

5.2.2 Data Extraction

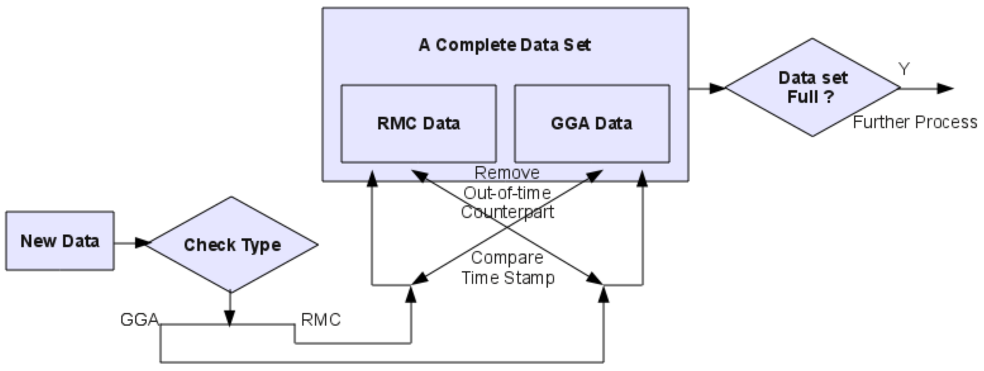


Figure 5.1: GPS Data Extraction

According to the communication data format explained in Section 4.4.1, $GPRMC and $GPGGA sentences are grouped with other telemetry data separately, and sent from the race car to a PC. Either of the two sentences are then processed by *Crystal Ball* for GPS information extraction. This process is illustrated in Figure 5.1.

As data packets may get lost during transmission, in order to keep different aspects of data synchronous with each other, the data extraction process checks if $GPRMC and $GPGGA data are generated at the same time or with very little time interval (less than 0.2 second in current design). If one of the data is out of date, it is removed from storage and the program waits for the next up-to-date data coming in.

5.2.3 Data Processing

- **Speed Adjustment**: Speed obtained from GPS usually contains bias. Transformed to km/h,

  $$V/(km/h) = V/(knots) * 1.851999$$

  The speed less than 5 km/h can be regarded as bias of GPS. As the Formula SAE race car usually drives at a speed much higher than 5 km/h, the real speed less than 5 km/h can be safely ignored.

- **Distance Calculation**: Distance traveled can be calculated in an accumulated way with speed, as the time interval between each GPS data set is very short. The GPS currently used in REV lab has an update rate of 5 Hz, which means the time interval between each GPS data is 0.2 second. The speed of a car is impossible to change to much in such a short duration. The pseudo code for calculating distance is shown in Listing 5.3.

```
dt = currentData.time - lastData.time;
meanSpeed = (currentData.speed + lastData.speed)/2;
ds = meanSpeed * dt
distance += ds;
```

Listing 5.3: Pseudo Code of Calculating Distance with GPS Speed

- **GPS Accuracy**: The GPS accuracy from satellites positions can be found in HDOP. It is a indication of 2D-coordinate accuracy.

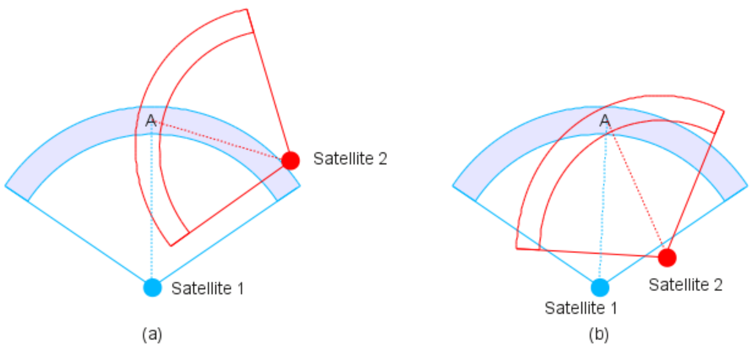


Figure 5.2: Accuracy Issue Caused by Satellites Positions

Figure 5.2 illustrates how satellites positions impact on GPS accuracy. In figure (a), two satellites are in an angle of approximately 90 degree to each other from the receiver. The area they can defines, shown as ”A” in the figure, is relatively small. However, in figure (b), the two satellites has a very small angel with each other. As a result, the area “A“ is much broader compared to the area in figure (a).

Besides of satellite position, HDOP is also relevant to some other factors [3], which are not discussed here. The explanation of HDOP number is shown in Table 5.1.

HDOP	**Rating**	**Description**
1	Ideal	This is the highest possible confidence level to be used for applications demanding the highest possible precision at all times.
2-3	Excellent	At this confidence level, positional measurements are considered accurate enough to meet all but the most sensitive applications.
4-6	Good	Represents a level that marks the minimum appropriate for making business decisions. Positional measurements could be used to make reliable in-route navigation suggestions to the user.
7-8	Moderate	Positional measurements could be used for calculations, but the fix quality could still be improved. A more open view of the sky is recommended.
9-20	Fair	Represents a low confidence level. Positional measurements should be discarded or used only to indicate a very rough estimate of the current location.
21-50	Poor	At this level, measurements are inaccurate by half a football field or more and should be discarded.

Table 5.1: Explanation of HDOP Values [31]

5.3 Sensor Data Processing

Noise and bias errors are always an issue in systems using sensor data. Figure 5.3 shows a set of speed data measured from REV’s Formula SAE Electric prototype car on 9th September, 2010. It can be seen from the figure that the speed data is bouncing heavily due to oversensitivity of hardware, as the true speed of a car is not supposed to vary so quickly. Moreover, the parts labeled with red boxes vary tremendously from the adjacent data sets. This is called a spike in signal processing. As race engineers usually tend to have a smooth curve to analyze the performance of the car and driver, removing high frequency noise and spike is a necessary feature for *Crystal Ball* telemetry software.

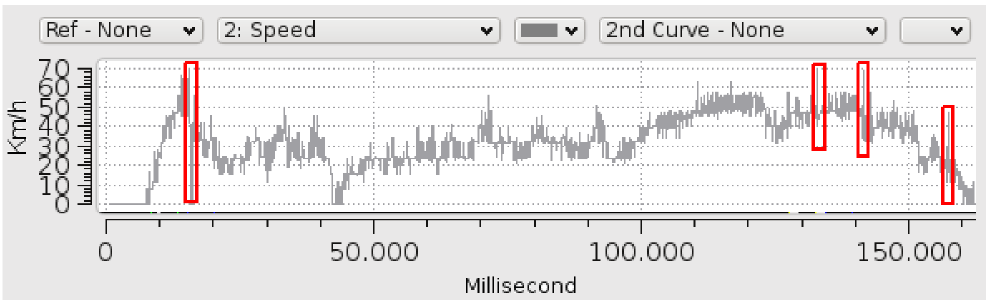


Figure 5.3: Bias and Spikes in Unprocessed Sensor Data

5.3.1 Filtering & Smoothing

The techniques of high frequency noise removal include filtering and smoothing. Filtering is to use a discrete low-pass filter to eliminate range of changes for the current sampled data. Previous data are used for the prediction. With smoothing, the data of interest is calculated with both preceding data sets and appending data sets. That means smoothing essentially brings in a longer delay in data processing. As taking the appending data into account, smoothing usually provides a better result than filtering.

5.3.2 Delay Analysis

Both filtering and smoothing cause delay in data analysis, as processing takes time. Heavier filtering (longer time constant in the exponential filter case) results in increased noise rejection, but also increase delay in response to changes [26]. So to get quick diagnosis results, filtering should go for light weight options in a real-time system.

In REV's Formula SAE Electric prototype car, the embedded MCU sends 20 data sets every second. If two data sets after the data of interest is used for smoothing, the delay introduced by processing is 0.1 second in theory, if 1 data set is used, the delay is only 0.05 seconds. This delay is acceptable in Formula SAE's real-time monitoring.

5.3.3 Scenarios Requiring Original Data

Filtering and smoothing is not applicable for all scenarios. They are usually desirable to reduce the effects of noise; however, some diagnosis depends on recognizing the presence or absence of noise, or unusual dynamic behavior, as a symptom of a fault or failing mode [30]. In these cases, the unfiltered data must be used.

5.3.4 Processing Design

As some scenarios make use of unfiltered data, the data processing function in *Crystal Ball* is not compulsory. It is disabled by default. However, the user can easily enable preferred processing options on the "Filter Dock" (see Section 4.3.2). The options includes the following data processing methods.

An example on comparison of different data processing methods are presented in Figure A.13 to A.18 in Appendix A.

a. **Average value of three adjacent data sets**

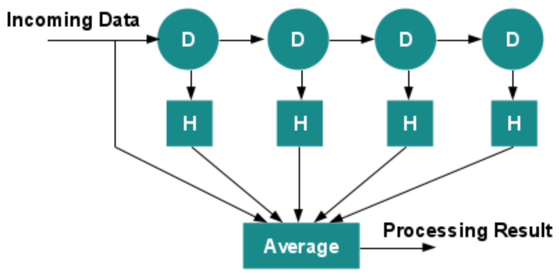


Figure 5.4: Smoothing by Averaging Five Adjacent Data Sets

In Figure 5.4, $\textcircled{\scriptsize D}$ means to delay one time slot, $\textcircled{\scriptsize H}$ means to hold the value of the delay module. These symbols are also applicable to Figure 5.5, 5.6 and 5.7. "Average" means to output the average value of all inputs (also for Figure 5.5).

b. **Average value of five adjacent data sets**

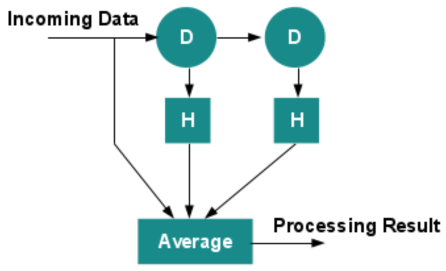


Figure 5.5: Smoothing by Averaging Three Adjacent Data Sets

c. **Median value of three adjacent data sets**

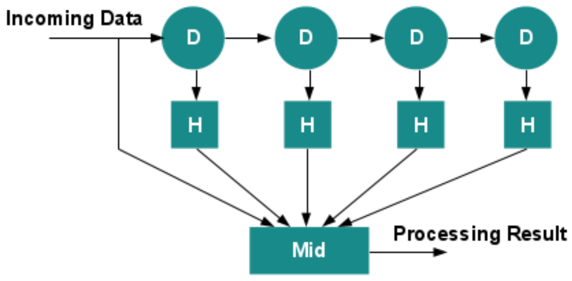


Figure 5.6: Smoothing by Using the Median Value of Five Adjacent Data Sets

Mid in the figure means to output the median value of all inputs. This is also applicable for Figure 5.7.

d. **Median value of five adjacent data sets**

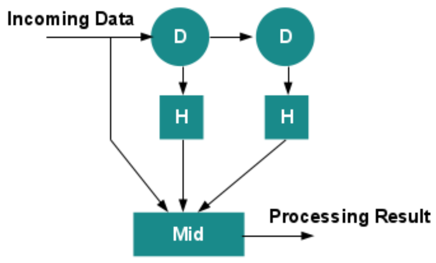


Figure 5.7: Smoothing by Using the Median Value of Three Adjacent Data Sets

e. **Spike removal**

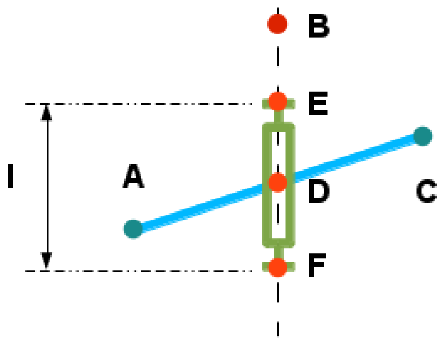


Figure 5.8: Spikes Removal

Spike removal makes use the slope between three adjacent data sets [6]. In the example shown in Figure 5.8, point B is processed by calculating the slope of the line between A and C. Suppose three points are all on the same line, B would be at the place of D. But this is usually not true in reality, so an interval "I" is applied to set the expected range of B. B is expected to appear somewhere between E and F. If B is out of this range (as shown in the example), the program treats the measurement as a spike, and use either end of the expected range which is closer to the original B to substitute the B, in this case, E.

The range "I" can be set by user by changing the ratio coefficient which multiple the distance between A and C on vertical axis. It is currently set as the vertical distance between A and C.

5.4 Serial Port Access

As RF transceiver connects to a PC through a USB port, *Crystal Ball* makes use of *QextSerialPort* 3rd party library to access data from a the USB port. *QextSerialPort* is a cross-platform serial port class, which encapsulates a serial port on both POSIX and Windows systems. It provides an abstraction layer between serial port and the OS, and offers high level API for data access.

5.4.1 Implementation Choice

The reason of using *QextSerialPort* as opposed to writing a new serial port class is explained as follows.

- **High reliability**: *QextSerialPort* is an mature and active project, which is maintained by senior system programmers. It has been proven working well in many programs. Any bugs or defects that may appear are supposed to be fixed by the maintainer soon. This greatly improves the maintainability of the software. In contrast, programming with serial port involves lots of low level coding, which is highly error prone. That significantly increases the risk of a software project.
- **Easy to use**: the high level API makes programming much easier.
- **Cross-platform capability**: *QextSerialPort* has good cross-platform capability, which is greatly appreciated in *Crystal Ball*. Self-written C++ libraries can hardly bridge the differences between different platforms.

All in all, the choice of *QextSerialPort* is not only a technical choice, but also an engineering choice. Software project nowadays is not possible to avoid code reusing, and that is also how large software can be built.

5.4.2 Data Access

The procedure of serial port access is shown in pseudo code in Listing 5.4.

```
    Get port name from user
    if(no port name specified)
        start trying from port 0
    else
        Set specified name

    Set event driven     // or "interrupt"
    Set baud rate        // usually 115200,
                         // depending on the RF transceiver
    Set flow control
    Set set parity
    Set data bits
    Set parity
    Set stop bits

    Open port
    if(success)
        waiting for data coming
    else if(number of try less than 10)
```

```
        set another port, go to open port again
    else
        report error to user
```

Listing 5.4: Pseudo Code for Serial Port Access

In the implementation, the program tries to help user if he/she doesn't know which port the RF connects to. The program simply attempt opening every ports on the PC until a port can be opened, or 10 ports are tried but no one can be opened. The limitation with this implementation is that some other device connects to the PC may be recognized as a RF transceiver by mistake. So a specified port name is a safer way of port access. An potential solution for the drawback is to access the device information to achieve a more accurate device recognition. This is discussed in future work section in Chapter 7.

5.5 Graph Plotting

Graph plotting makes use of *Qwt* 3rd party library as the back end for plotting curves and scatters. The *Qwt* library contains classes which are designed for programs with a technical background. Its 2D plot widget can plot arrays of C++ double type values. *Qwt* can be built into dynamic link libraries in Win32 or shared libraries in Unix/X11 environments. This is essential for portability of *Crystal Ball* telemetry software.

In implementation, as *Qwt* API takes arrays of double type values as inputs, all data structure received in the plotter class need to be added into arrays of double values. In simulation mode, when a list of data structure is passed to the plotter class, it iteratively transverses the received data structure list, converts data into arrays of double values, and then plot the curves. When simulation start, the plotter class, upon receiving a data structure, sets position of curve marker, which is provided by *Qwt* and take paired double values as inputs, to depict the process of how data values are changed. The pseudo code is shown in Listing 5.5. The whole process is event driven.

```
    // in the event loop
  callProcessDataListFunctionUpon(EVENT DATALISTRECEIVED,
                                                DATALIST list)
  {
    processDataList(list);
  }
```

```
callProcessDataFunctionUpon(EVENT DATARECEIVED,
                                            DATA data)
{
  processData(data);
}
...
// functions
processDataList(DATALIST list)
{
  ARRAY array = convertIntoArray(list);
  plotArray(array);
}
processData(DATA data)
{
  Position position = convertIntoPosition(data);
  plotMarker(position);
}
```

Listing 5.5: Pseudo Code for Graph Plotting

There are six situations that will cause a re-plot of the plotting area.

- Change of reference file.
- Change of property for reference curve.
- Change of data log file.
- Change of property of either data curves.
- Change of color of either data curves.
- New data structure comes.

5.6 Circular Motion

Force decomposition and velocity analysis for a circular motion for differential drive can be very accurate and complex. *Crystal Ball* only does a simplified analysis in the current implementation. An advanced analysis for differential drive can be implemented in the future. The implemented real-time analysis is shown in Figure 5.9.

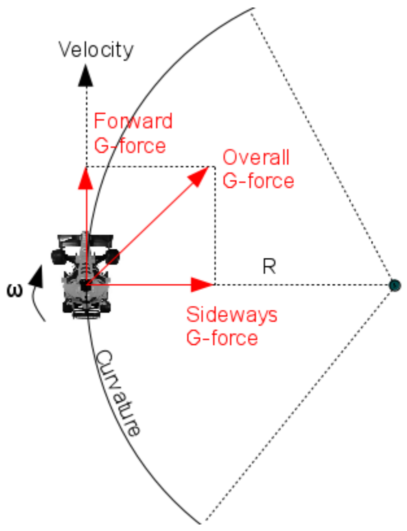


Figure 5.9: Circular Motion Analysis

Among all variables, car velocity, angular velocity, forward G-force and lateral G-force are measured by sensors in the car. For simplicity, assume car velocity is on the direction heading forward and is vertical to the line connecting to center of the circle, the radius can be calculated as

$$R = v/\omega$$

The overall G-force on the car is

$$F = \sqrt{(F_{forward}^2 + F_{lateral}^2)}$$

5.7 User Settings

Users normally expect an application to remember its settings, for example, what sub-windows they opened last time, what data curves they plotted last time, etc. *Crystal Ball* makes use of *QSettings* class, which provides persistent platform-independent application settings [20].

As *Crystal Ball* is designed for running on different platforms, the greatest advantage of using *QSettings* class is cross-platform capability. Another advantage is it allow the program to access settings from multiple threads or processes simultaneously.

QSettings is reentrant [21]. This means distinct *QSettings* object in different threads can simultaneously refer to the same files on disk (or to the same entries in the system registry). If a setting is modified through one QSettings object, the change will immediately be visible in any other QSettings objects that operate on the same location and that live in the same process [20]. This is greatly appreciated for ensuring data integrity.

The information generated by *QSettings* class is often stored in the system registry on Windows, and in XML preferences files on Mac OS X. On Unix systems, in the absence of a standard, many applications (including the KDE applications) use INI text files [20]. So for the *Crystal Ball* program, when recompiled and run on another platform, may result in different configuration files. This is transparent to users, but should be noted to future developers. An example INI file is shown in listing 5.6.

```
[TrackWindow]
bottomHiden=false
leftHiden=false
carCentered=false
showTrajectory=true
showTrack=true
showComments=true
latitude=-31.979283
longitude=115.815852
zoom=15
```

Listing 5.6: An Example INI Setting File

CHAPTER 6

Validation

This chapter presents the validation process of Crystal Ball telemetry software. The methods used in validation include data simulation, user experience survey and expert evaluation.

6.1 Validation Methods

By the time this dissertation is written, the transmitting part of the telemetry system is still under development by REV instrumentation engineers. So the whole system is not ready for use in real world scenarios. As a consequence, the real-time mode of *Crystal Ball* cannot be tested with a Formula SAE car on track.

However, the developer designed a pilot testing, which tests the serial port communication with the real RF transceiver, and tests the real-time data processing with data simulation through network (see implementation of both modes in Section 5.1). The network module redirects the data to exactly the same function block used by real-time mode. Therefore this is valid simulation testing.

All other parts of *Crystal Ball* telemetry software are tested in ways they are expected to be used. The *Crystal Ball β* version is tested by REV members including a number of experienced electrical/electronic engineers, software engineers and mechanical engineers. The software is also evaluated by professors in software and ICT domain at UWA.

6.2 Data Simulation

Data simulation is used by the developer to test all aspects of *Crystal Ball* telemetry software in the validation process. The data used for testing is obtained from two on-track driving tests of Formula SAE Electric Prototype. Both events were organized in RAC driver training and education center in Perth, Western Australia. The first test driving was organized on 9th of September 2010 and the second event was on 23rd September of the same year.

Sample test data used in simulation is provided in Appendix C. Test cases are specified in Appendix A. The data simulation test results are shown in Table 6.1.

Test Case Groups	**Success**	**Failure**	**Notes**
Framework	6	0	All passed
RF & port viewer	4	0	All passed
Log file simulation	7	0	All passed
Data share	3	0	All passed
Track	8	0	All passed
Gauges	1	0	All passed
Waveform	11	1	Reference curve shifting not implemented due to time limit & low priority.
Scatter	4	0	All passed
Circular Motion	2	0	All passed
Numeric Log	3	0	Speed is low. Refactoring required.
Track Marker	1	4	Adding comments not implemented due to time limit & low priority.
Summary	4	0	All passed
Filters	2	0	All passed
Total	56	5	5 failed test cases and one feature refactoring shall be scheduled to the next iteration.

Table 6.1: Data Simulation Results in System Testing

According to the tested cases, which are mapped from the system requirements (or user stories), the *Crystal Ball* α has achieved all the key features described in project scope in Chapter 3, and is competent for β releases. The failed test case including shifting reference curves, adding comment to track files, and numeric log refactoring can be scheduled to the next release.

6.3 User Experience

User experience is to distribute *Crystal Ball β* to the aimed user group, and let them use *Crystal Ball β* freely in scenarios that the software is designed for. The rationale behind user testing is that it is impossible for the designed test cases to cover all usage scenarios. The feedback from user experience could bring forward the problems that are ignored by the developer, and provide the developer an insight to the future development of the application.

A user experience questionnaire is designed to gather information from the user experience survey. According to the iterative nature of Extreme Programming, the questionnaire is designed for both validation and future development purposes. The questionnaire is presented in Appendix B.

The subjects involved in *user experience survey* include eight electrical/electronic engineers, seven software engineers and four mechanical engineers from school of engineering at UWA. In the experiment, the developer gives an introduction of the telemetry software. The subjects then play with the software randomly. The developer is allowed to answer questions from the subjects. Finally, the subjects are asked to fill in the questionnaire anonymously.

The result of user experience survey is categorized according to the their specialized domains. These domains include electrical/electronic (EE) engineering, software engineering and mechanical engineering. The results are presented in Table 6.2 to 6.6.

Features	**EE Engineers**	**Software Engineers**	**Mechanical Engineers**
RF Mode	Need - must have	Need - must have	Need - must have
Simulation Mode	Need - must have	Must have	Need - must have
Data Share Mode	Neutral - Need to have	Need to have	Need to have
Data Processing	Need - must have	Need - must have	Need - must have

Table 6.2: Rank on the Importance of Implemented Features

Features	**EE Engineers**	**Software Engineers**	**Mechanical Engineers**
RF Mode	Good	Good - excellent	Good
Simulation Mode	Good - excellent	Good - excellent	Good - excellent
Data Share Mode	Good - excellent	Good - excellent	Good - excellent
Data Processing	Good - excellent	Good - excellent	Good - excellent

Table 6.3: Rank on the Achievement of Implemented Features

Features	**EE Engineers**	**Software Engineers**	**Mechanical Engineers**
Framework	Good - excellent	Good - excellent	Good - excellent
Control Panel	Good - excellent	Good - excellent	Good - excellent
Tab Mode	Good - excellent	Good - excellent	Good - excellent
Dock Widgets	Good - excellent	Good - excellent	Good - excellent

Table 6.4: Rank on the GUI Design

Visualization	**EE Engineers**	**Software Engineers**	**Mechanical Engineers**
Track	Need - must have	Need - must have	Must have
Gauge	Need - must have	Need to have	Must have
Curves	Need - must have	Need - must have	Need - must have
Scatter	Neutral - Need to have	Need - must have	Need to have
Forces	Need - must have	Neutral - Need to have	Need - must have
Numeric	Need - must have	Need to have	Neutral
Summary	Need - must have	Need - must have	Need - must have

Table 6.5: Rank on the Importance of Implemented Visualization Methods

Visualization	**EE Engineers**	**Software Engineers**	**Mechanical Engineers**
Track	Good - excellent	Good - excellent	Good - excellent
Gauge	Good - excellent	Good - excellent	Good - excellent
Curves	Good - excellent	Good - excellent	Good - excellent
Scatter	Good - excellent	Good - excellent	Basically Achieved
Forces	Good - excellent	Good - excellent	Good - excellent
Numeric	Good - excellent	Good - excellent	Basically Achieved
Summary	Good - excellent	Good - excellent	Good - excellent

Table 6.6: Rank on the Achievement of Implemented Visualization Methods

The user experience result shows that real-time and simulation modes are considered as very important features in the program by all user groups. Data share mode is not treated as important by electrical/electronic engineers as by other user groups. The three working mode and data processing features are proved well implemented by all user groups.

GUI design is a success in *Crystal Ball*. Four GUI components are all ranked "Good - excellent" by the subjects in all user groups.

All visualization methods are considered very important except "scatter" and "forces analysis". "Scatter" is treated "neutral to needed" by electric/electronic engineers. "Forces analysis" is considered "neutral to needed" by software engineers. "Numeric log" and "scatter" are considered basically achieved by mechanical engineers. All other visualization method are ranked "Good - excellent" for current implementation.

In summary, most features implemented so far are considered necessary to users. They have achieved great customer satisfaction as shown in user experience survey. Therefore, a RC of *Crystal Ball* is released for expert evaluation.

6.4 Expert Evaluation

Expert evaluation is to let senior academic or engineers to evaluate the software. Experts look at the complete system from many perspectives and reveals potential problems such as inconsistency, support for different ways of working, etc.

It is a fact that a small number of users in a user test are not likely to cover all the ways that every user will do with the software. Expert can usually look over the interface and reveal some issues that may cause problems. As a result, expert evaluation is a necessary way for validation of the telemetry software.

Two professors in ICT at the University of Western Australia have helped to evaluate *Crystal Ball* telemetry software. The following suggestions are provided for the RC of *Crystal Ball*.

- Research shall be done on Zig-bee protocol used for RF communication. If error checking is not included in the protocol stack, a checksum subroutine should be implemented for wireless communication to filter incomplete and wrong data.
- Latency of the wireless transmission, which can be coarsely calculated by the difference between GPS time and system time can be calculated in real-time mode to monitor the performance of the RF module.
- Time difference between each data set shall be monitored in real-time mode. Properties like distance traveled, data processing like filtering, averaging, etc are based on the assumption that data sets are coming into the system fairly frequently (no less than 1 Hz). Lower data rate will result in low accuracy of the calculated data.
- Design of GPX marker is easy enough for users. A graphic based GPX marker makes is necessary for making tracks.
- Doxygen in-line documentation is well done, but class and variable naming can be optimized for an enhanced comprehension level and system maintainability.

Among these suggestions, the first three are regarding to real-time communication issues. The future developer should keep attention on the suggestions in further development. A more comprehensive GPX marker GUI should be designed in the next release. Naming issues in the source code shall be addressed before the coming release. Future development should keep the practice of Doxygen in-line documentation and comprehensive naming for classes and variables.

6.5 Validation Summary

The validation process of *Crystal Ball* comprises data simulation, user experience and expert evaluation. The designed test cases are mapped from the system requirements (or user stories) specified in Chapter 3. Together with user experience and expert evaluation, the process is valid for the evaluation of *Crystal Ball* telemetry software.

Crystal Ball has passed all three validation methods, and is proved to be a valid design and implementation of the required telemetry software. *Crystal Ball* is competent for a stable release with existing features. A few features in the implementation are missing due to the low priority and the time limit. These features can be added to the existing software in next release.

CHAPTER 7

Conclusion

This chapter summaries the "Formula SAE Telemetry Software" project and explains what it means to Formula SAE teams all around the world. The dissertation is reviewed in this chapter. Future work on Crystal Ball is discussed.

7.1 The Project

The "Formula SAE Telemetry Software" project has developed *Crystal Ball*, a desktop application which provides race engineers a simple interface to analyze telemetry data visually in real-time, simulation or data share (network) mode.

Crystal Ball is an open source real-time telemetry software. It works with data acquired from real-time telemetry system, stored in a log file or received from network. It encompasses functions of data processing, track and trajectory display, gauges visualization, curves and scatter plotting, circular motion analysis, numeric display and track summary. *Crystal Ball* is suitable for an individual data analyst or for many race engineers all monitoring telemetry data on different desktops at the same time.

Crystal Ball conforms to an easy-to-follow communication standard and several popular data storage format. It is scheduled to be released under GNU/GPL license to replace its commercial counterparts for REV Formula SAE team and other motor sport teams around the world. The open source and open standard of *Crystal Ball* is supposed to release the student teams from buying bounded hardwares or paid softwares from commercial organizations; therefore allows student teams to have more options on Formula SAE's hardware design, and have more freedom on customization of their telemetry software.

7.2 The Dissertation

This dissertation documents the design, implementation and validation process of the "Formula SAE Telemetry Software" project. Chapter 1 introduces the Formula SAE event, real-time telemetry and the structure of the dissertation. Chapter 2 outlines the design of two main stream telemetry applications for motor sports developed by commercial organizations and a web based telemetry system developed for urban vehicles. Their features and limitations are discussed followed by a summary of the survey. Chapter 3 specifies the requirement of this project based on iterative requirement elicitation and validation, survey of existing systems and available resources.

Chapter 4 demonstrates the system design of the *Crystal Ball* with discussions on different designed choices. Chapter 5 explains the implementation of the design and discusses the possible ways for future improvement. Other options in implementation, and the rationale behind the choices made are also discussed in this chapter.

System validation is presented in Chapter 6, where data simulation, user experience survey and expert evaluation are used for the validation process.

Test cased designed for the validation of *Crystal Ball* is provided in Appendix A for reference of system maintenance, future development and improvement. User experience questionnaire is presented in Appendix B. Appendix C shows examples of data file formats used in the program.

7.3 Future Work

7.3.1 Uncompleted Features

Due to time limit, a few planned features with lower priority has not been fully implemented. So the first task for future development is to fully implement the missing features. This work could take a test driven development methodology [1], where the the system is developed against the failed test cases until all test cases are passed.

The features to be implemented includes

a. Reference curve shifting in the *Curves* module.

b. Design and Implement a comprehensive user interface for GPX track marker.

c. Code refactoring for the *Numeric Log* module for enhanced performance.

Some implementations, which were proposed but were not scheduled in the coming release due to resource limit, should be implemented when the resources are ready. These implementations are

a. Fitting *Crystal Ball* into the telemetry system.

b. Binary data format for real-time data transmission (Section 4.4.1).

c. Research shall be done on Zig-bee protocol stack. A checksum subroutine shall be implemented if no error checking is provided in Zig-bee stack. (Section 6.4)

d. Latency monitoring for wireless data. (Section 6.4)

e. Time interval monitoring for adjacent wireless data sets. (Section 6.4)

f. Rigorous testing on automatic RF recognition in real-time mode (Section 5.1).

g. A more accurate model for force and velocity decomposition in circular motion analysis (Section 5.6).

7.3.2 Port to Windows OS

Crystal Ball is developed in Qt Creator 4.6 on Ubuntu 10.04 i386. The latest RC release is compiled on Ubuntu 10.04 i386 including *Qwt* and *QextSerialport* 3rd party libraries. The version compiled for Windows OS has not been ready for the coming release.

As originally planed, *Crystal Ball* is supposed to run on both Linux and Windows OS to meet the different preference of race engineers or data analysts. So the Windows binary shall be scheduled for the next release.

The work of porting *Crystal Ball* to Windows involves compiling *Qwt* and *QextSerialport* 3rd party libraries on Windows, setting platform awareness options in compilation configuration file, and compiling the program. Once the work is done, the SVN repository should update to the platform awareness version. Future development on both platforms can start from the new version.

7.3.3 Plug-in Awareness

At this stage, functional blocks in *Crystal Ball* are hard coded into classes and integrated to the main window as a whole framework.

Another option to organize these functional blocks is to use a plug-in mechanism. A plug-in is a set of software components that adds specific abilities to a larger software application [32]. Plug-ins can be enabled to customize the functionality of an application. The advantages of using a plug-in mechanism include the following aspects.

- It enables third-party developers to easily create new functional blocks to extend the application, or add new features without having to know the internal design of the program. This can greatly enhance the scalability of *Crystal Ball*.
- It makes the program adaptable to future changing requirement. When requirement changes for a particular functional module, only the corresponding plug-in has to be changed. The whole program can be left as it was. This significantly improves the maintainability of the program.
- It allows users to customize the application according to their own need. Unnecessary modules can be easily removed from the program to improve the overall performance.

APPENDIX A

Crystal Ball Test Cases

The test cases listed here are designed to be used for the following purposes.

a. Test driven development.

b. Validation of the implementation.

c. Validation for code refactoring.

d. A baseline for further development.

e. A supplement for requirement and design document.

The purpose of each module, the usage scenarios and expected outcomes are described in these test cases.

A.1 Program Framework

The program framework provides a main window and control panel based system for organizing functionality modules in the program.

a. Pointing the mouse to each widget on the control panel, tool tips should pop up showing the basic information about that widget.

b. Click "X" button on the window boarder of control panel. There should pup up a message box saying control panel cannot be closed.

c. Check/uncheck each dock and sub-window check box on the control panel, corresponding dock or sub-window should show/hide accordingly.

d. Click "X" button on the window boarder of each sub-window, the sub-window should be closed and the corresponding check box on the control panel should be cleared automatically.

e. Click ”All” or ”None” button on the control panel, all sub-windows should show/hide accordingly.

f. Check/uncheck ”TAB Mode” check box, all sub-windows should be presented into TAB/sub-window mode accordingly.

A.2 RF & Port Viewer

The RF module is used for communication with the RF transceiver for obtaining real-time data measured from the Formula SAE race car.

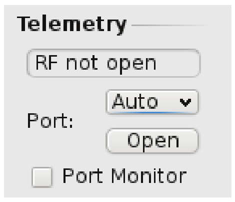


Figure A.1: RF Port GUI

a. Connect the RF transceiver to the PC, choose ”Auto” from the Port combo box, and click ”Open”. The program should be able to find a the RF transceiver automatically and open it successfully. The statue label should display ”Port Opened”.

b. Upon success of opening RF transceiver, the ”Port Monitor” sub-window should be able to open, and display raw data received from the RF transceiver.

c. Close the RF port. Disconnect the RF transceiver, and use ”Auto” mode to open port again. The program should show a message box saying port cannot open.

d. When RF transceiver is not open, trying to open the ”Port Monitor” sub-window will come across a message box saying no device/port is open.

A.3 Log File Simulation

Log file simulation module is to provide a functionality of simulating data file logged in the Formula SAE race car.

a. Load a log file from ”HOME/saeLog/” directory by clicking ”Load Log” button on the control panel. The statue box should display ”Log Loaded”.

b. Start simulation by clicking ”start” button. The statue box should display ”In simulation”. A couple of new widgets should appear below the simulation area providing information and control.

c. Click ”Pause” button. The simulation should be paused, and the ”Pause” button should becomes a ”Cont.” button. The statue box should display ”Simulation Paused”.

d. Click ”Cont.” button. The simulation should continue. The ”Cont.” button should becomes a ”Pause” button, and the statue box should display ”In Simulation”.

e. Change simulation speed by choosing another speed factor from the simulation speed combo box. The speed of simulation should change accordingly.

f. Click ”Stop” button to stop simulation before it finishes. The simulation should stop. A message box should pop up saying simulation finished.

g. Restart the simulation again. This time, allow it to finish. Upon completion, a message box should pop up informing user that simulation finished.

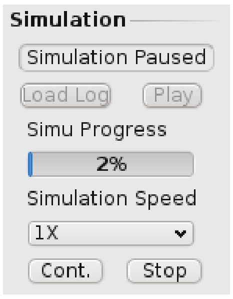

Simulation
Simulation Paused
Simu Progress
2%
Simulation Speed
1X

Figure A.2: Simulation Widgets on Control Panel

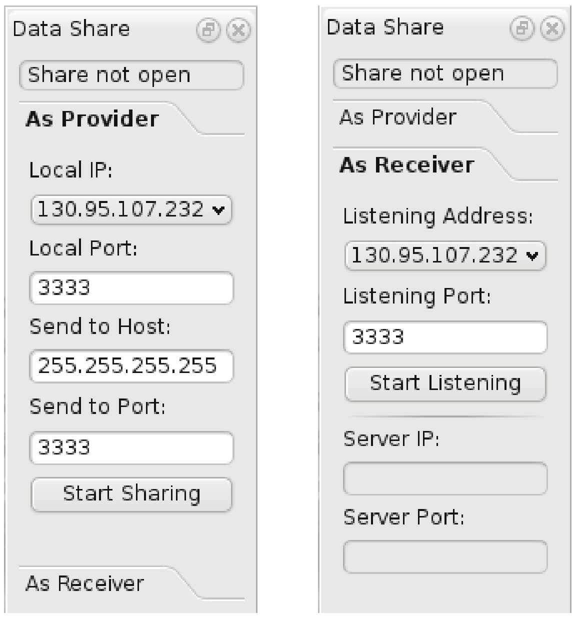
A.4 Data Share

Data share module is to allow the PC which connect directly to the RF transceiver to share information with other PCs on Internet or in the same LAN. It also works for data simulation mode.

a. Open two *Crystal Ball* programs on two PCs connecting to the Internet or the same LAN. Use the default IP address and port number to set *Crystal Ball* as data provider on one PC, and as data receiver on another PC. The statue boxes should display ”Now as Provider” or ”Now as Receiver” accordingly.

b. Load and start simulating a log file on the data provider PC. On the receiver PC, the server IP and server Port should display server information.

c. Open the Curves sub-window on the receiver PC. Choose a property for plotting. Start simulation on provider PC, the receiver PC should have data curves plotting as well.



Data Share

Share not open

As Provider

Local IP: 130.95.107.232

Local Port: 3333

Send to Host: 255.255.255.255

Send to Port: 3333

Start Sharing

As Receiver

Data Share

Share not open

As Provider

As Receiver

Listening Address: 130.95.107.232

Listening Port: 3333

Start Listening

Server IP:

Server Port:

Figure A.3: Network Dock GUI

A.5 Track Sub-window

Track sub-window is to show the race track and car trajectory in real-time or simulation mode. It also allows user to save car trajectory as a race track file.

a. Click the ”Load Track” button in the track sub-window, and load ”RAC.gpx” file from local disk. The display area should display a car track with comments on it.

b. Click ”Zoom in” button. The track should become bigger.

c. Click ”Zoom out” button. The track should become smaller.

d. Pan the display area to make the car arrow out of view. Click ”Go to Car” button. The car arrow should be centered in the display area.

e. Pan the display area to make the track out of view. Click ”Go to Track” button. The middle of the track should be centered in the display area.

f. Check/uncheck on ”Track”, ”Trajectory” and ”Comments” check boxes separately. The track, car trajectory and track comments should show or hide accordingly.

g. Click ”Save” button to save the trajectory into a track file. If no track is available, there should be a message box popping up to inform the user.

h. Click on ”More/Hide” button, and ”</>” button, the bottom and left control buttons should show and hide accordingly.

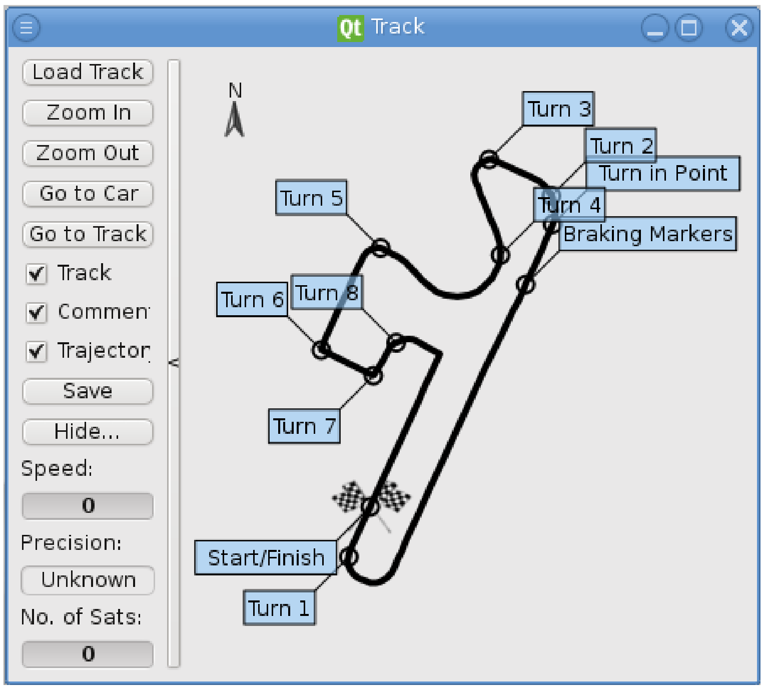


Figure A.4: Track Sub-window GUI

A.6 Gauges Sub-window

Gauges sub-window provides several analog widgets showing the information of driver's operation.

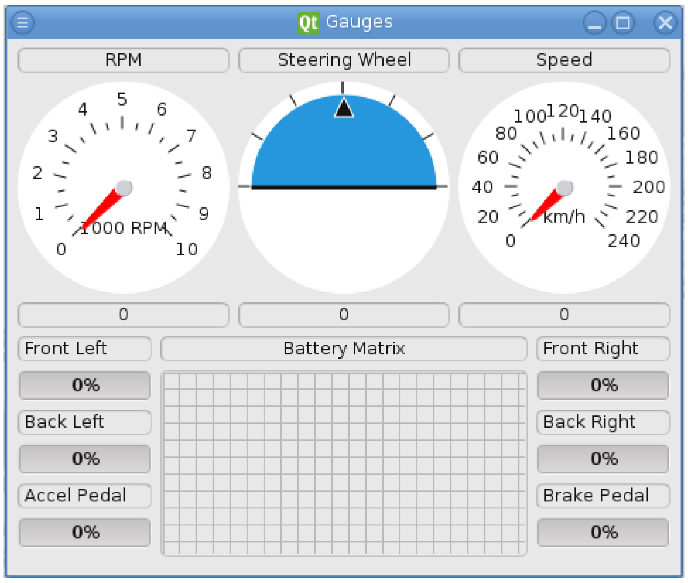


Figure A.5: Gauges Sub-window GUI

a. Start data simulation. The gauges widgets in the sub-window should display simulated data visually (Figure A.5).

A.7 Waveform Sub-window

Waveform sub-window allows user to plot different aspects of data against time line. Reference curves are allowed to be set underlying. Several plotting widgets can be used at the same time.

a. Click "Graphs" spin box and change the number. The number of plotting widgets should change accordingly.

b. Click "Preset" combo box and choose "Over view". The plotting widgets should change their curves and reference curves options into specific preset properties.

c. Click "Load Ref File" button and load a reference file. The plotting widgets should plot a specific reference curve in the color of gray according to what property is chosen.

d. Choose other properties from the "Ref" combo box. The previous curve should be cleared and new curve should be plotted.

e. Click "<-" or "->" button. The reference curve should shift left or right on time line accordingly.

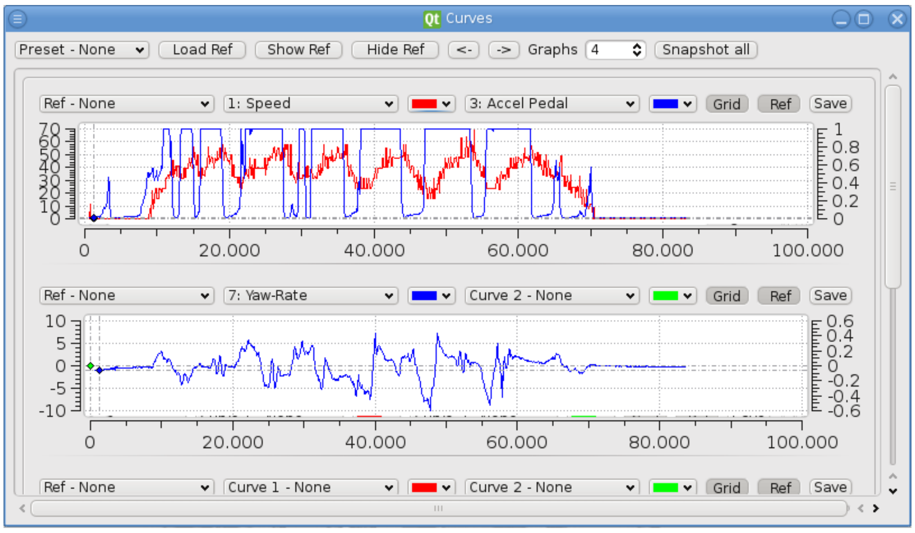


Figure A.6: Curves Sub-window GUI

Note: the graph does not show metric unit as it is implicit to users. All metric unit is described in Section 4.4 of the dissertation and the user guide of *Crystal Ball*.

f. Click "Ref" toggle button. The reference curve should hide/show accordingly.

g. Click "Grid" toggle button. The grid in the plotting area should hide/show accordingly.

h. Load a log file. Choose different properties types in the "Curve 1" and "Curve 2" combo box. The plotting widget should display curves accordingly.

i. Choose different color from the combo box of each curve. The color of the curves should change accordingly.

j. Click "Save" button. There should pop up a "Save image" file manager window for saving the graph into an image file.

k. Click "Snapshot all" button. There should pop up a "Save image" file manager window for saving all graphs into an image file.

l. Start log file simulation. There should be a marker on each curve showing how data changes along the time line.

A.8 Scatter Sub-window

Scatter sub-window allows user to assemble any two data in the measurement and plot them in a 2D scatter form.

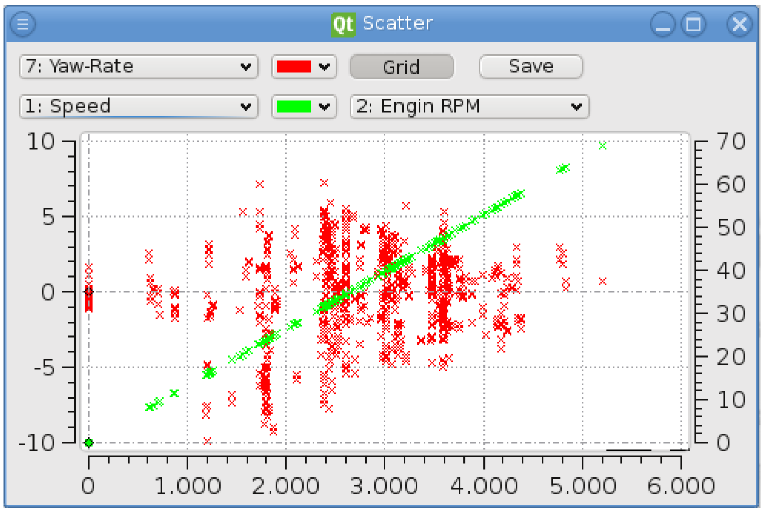


Figure A.7: Scatter Sub-window GUI

a. Click "Grid" toggle button. The grid in the plotting area should hide/show accordingly.

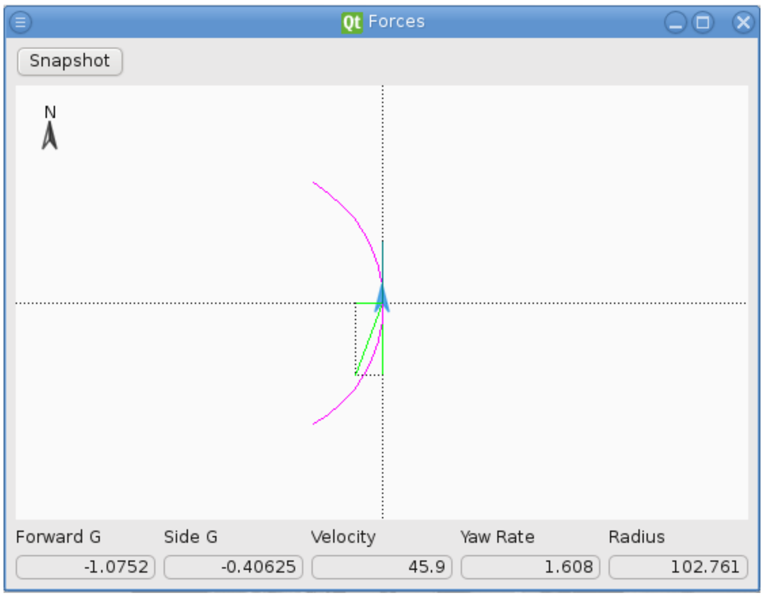
b. Load a log file. Choose different properties types for y-left y-right and x axis in separate combo boxes. The canvas should display scatter graphs accordingly.

c. Choose different color from the combo box of each scatter graph. The color of the dots should change accordingly.

d. Click "Save" button. There should pop up a "Save image" file manager window for saving the graph into an image file.

A.9 Circular Motion Sub-window

Circular Motion sub-window provides real-time force decomposition and angular movement analysis.



Forward G: -1.0752
Side G: -0.40625
Velocity: 45.9
Yaw Rate: 1.608
Radius: 102.761

Figure A.8: Circular Motion Sub-window GUI

a. Start log file simulation. There should visualize real-time forces decomposition and curvature radius. Forward and sideways G-force, speed, angular velocity and radius should be displayed below.

b. Click the "Snapshot" button, there should pop up a "Save image" file manager window and the plot can be saved as an image file.

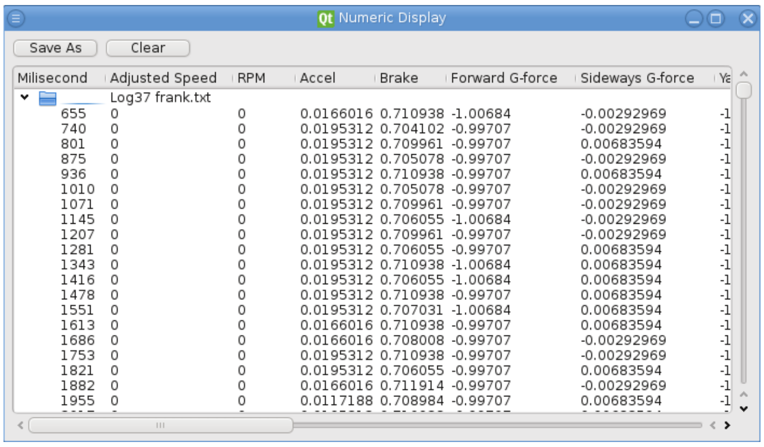
A.10 Numeric Log Sub-window

Numeric sub-window show the measured data in a processed way. It also allows user to export the processed data that can be used in other programs.

a. Start log file simulation. The data should append to the tree view area in real-time.

b. Click "Save" button, there should pop up a "Save processed data" file manager window, and the data listed in the tree view area should be saved into an CSV file.

c. Click "Clear" button, the data in the tree view widget should be cleared.

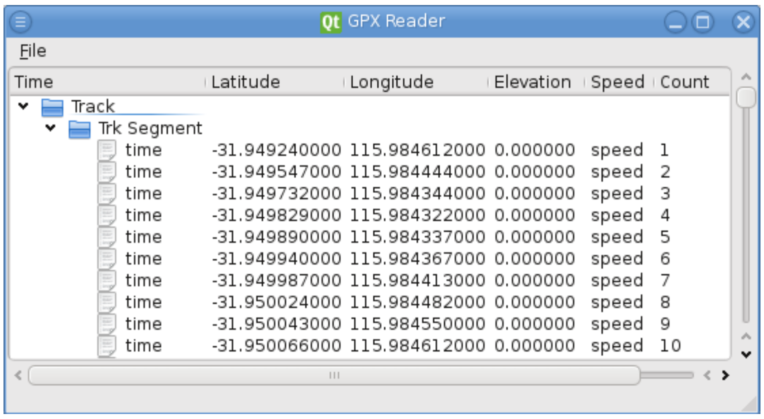


Milisecond	Adjusted Speed	RPM	Accel	Brake	Forward G-force	Sideways G-force	Ya
	Log37 frank.txt						
655	0	0	0.0166016	0.710938	-1.00684	-0.00292969	-1
740	0	0	0.0195312	0.704102	-0.99707	-0.00292969	-1
801	0	0	0.0195312	0.709961	-0.99707	0.00683594	-1
875	0	0	0.0195312	0.705078	-0.99707	-0.00292969	-1
936	0	0	0.0195312	0.710938	-0.99707	0.00683594	-1
1010	0	0	0.0195312	0.705078	-0.99707	-0.00292969	-1
1071	0	0	0.0195312	0.709961	-0.99707	-0.00292969	-1
1145	0	0	0.0195312	0.706055	-1.00684	-0.00292969	-1
1207	0	0	0.0195312	0.709961	-0.99707	-0.00292969	-1
1281	0	0	0.0195312	0.706055	-0.99707	0.00683594	-1
1343	0	0	0.0195312	0.710938	-1.00684	0.00683594	-1
1416	0	0	0.0195312	0.706055	-1.00684	0.00683594	-1
1478	0	0	0.0195312	0.710938	-0.99707	0.00683594	-1
1551	0	0	0.0195312	0.707031	-1.00684	0.00683594	-1
1613	0	0	0.0166016	0.710938	-0.99707	0.00683594	-1
1686	0	0	0.0166016	0.708008	-0.99707	-0.00292969	-1
1753	0	0	0.0195312	0.710938	-0.99707	-0.00292969	-1
1821	0	0	0.0195312	0.706055	-0.99707	0.00683594	-1
1882	0	0	0.0166016	0.711914	-0.99707	-0.00292969	-1
1955	0	0	0.0117188	0.708984	-0.99707	0.00683594	-1

Figure A.9: Numeric Display Sub-window GUI

A.11 Track Marker Sub-window

Track marker sub-window provides users an comprehensive way of editing GPX file. User is allowed to add comment or name for track points or way points.



Time	Latitude	Longitude	Elevation	Speed	Count
Track					
Trk Segment					
time	-31.949240000	115.984612000	0.000000	speed	1
time	-31.949547000	115.984444000	0.000000	speed	2
time	-31.949732000	115.984344000	0.000000	speed	3
time	-31.949829000	115.984322000	0.000000	speed	4
time	-31.949890000	115.984337000	0.000000	speed	5
time	-31.949940000	115.984367000	0.000000	speed	6
time	-31.949987000	115.984413000	0.000000	speed	7
time	-31.950024000	115.984482000	0.000000	speed	8
time	-31.950043000	115.984550000	0.000000	speed	9
time	-31.950066000	115.984612000	0.000000	speed	10

Figure A.10: Track Marker Sub-window GUI

a. Load data from any log file or GPX file, the tree view area should display track points data.

b. In the column of "comment", it should accept user input of characters.

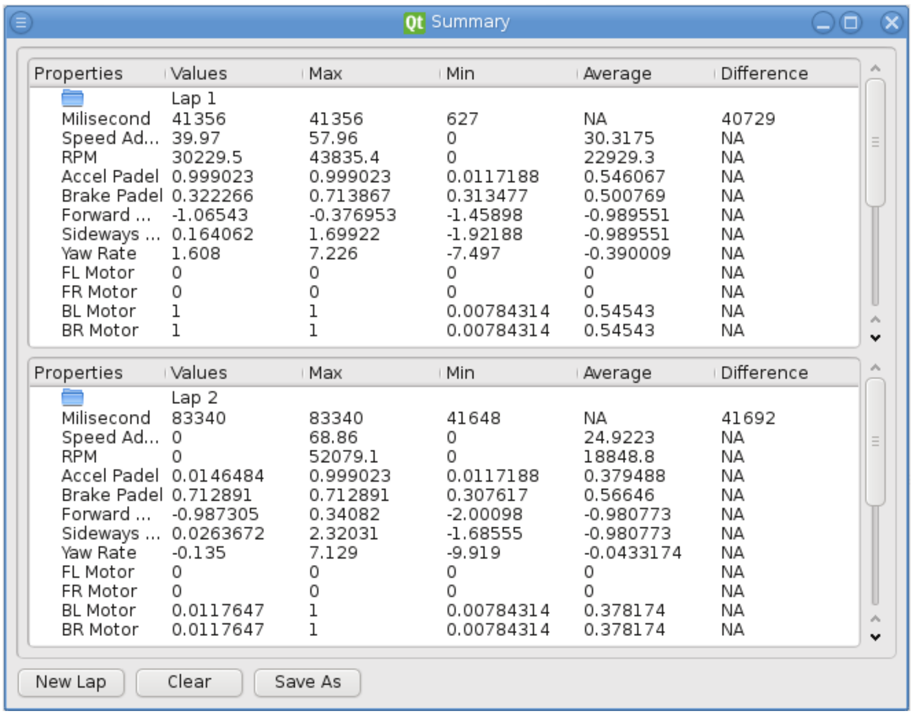
c. Click ”Save As” button, the data in tree view area should be able to save into a new GPX file.

d. Load the save the GPX file in ”Track” sub-window, check ”Comments” check box. The new added comments should be presented there.

e. Click ”Clear” button, the data in the tree view widget should be cleared.

A.12 Summary Sub-window

The summary sub-window provides track summaries for each part of track defined by user. Summary sub-window is used to calculate and display the summary of each lap, including time span, maximum, minimum and average speed, RPM, acceleration, brake pedal, distance traveled, etc.



Properties	Values	Max	Min	Average	Difference
🗀	Lap 1				
Milisecond	41356	41356	627	NA	40729
Speed Ad...	39.97	57.96	0	30.3175	NA
RPM	30229.5	43835.4	0	22929.3	NA
Accel Padel	0.999023	0.999023	0.0117188	0.546067	NA
Brake Padel	0.322266	0.713867	0.313477	0.500769	NA
Forward ...	-1.06543	-0.376953	-1.45898	-0.989551	NA
Sideways ...	0.164062	1.69922	-1.92188	-0.989551	NA
Yaw Rate	1.608	7.226	-7.497	-0.390009	NA
FL Motor	0	0	0	0	NA
FR Motor	0	0	0	0	NA
BL Motor	1	1	0.00784314	0.54543	NA
BR Motor	1	1	0.00784314	0.54543	NA

Properties	Values	Max	Min	Average	Difference
🗀	Lap 2				
Milisecond	83340	83340	41648	NA	41692
Speed Ad...	0	68.86	0	24.9223	NA
RPM	0	52079.1	0	18848.8	NA
Accel Padel	0.0146484	0.999023	0.0117188	0.379488	NA
Brake Padel	0.712891	0.712891	0.307617	0.56646	NA
Forward ...	-0.987305	0.34082	-2.00098	-0.980773	NA
Sideways ...	0.0263672	2.32031	-1.68555	-0.980773	NA
Yaw Rate	-0.135	7.129	-9.919	-0.0433174	NA
FL Motor	0	0	0	0	NA
FR Motor	0	0	0	0	NA
BL Motor	0.0117647	1	0.00784314	0.378174	NA
BR Motor	0.0117647	1	0.00784314	0.378174	NA

Figure A.11: Summary Sub-window GUI

a. Start simulation. The tree view area should start display data and summary information.

b. Click ”New Lap” button, the current summary tree view should stop updating, and a new tree view widget should appended underneath. New data and summary for the new lap should start displaying there.

c. Click ”Save As” button, the data in tree view area should be able to save into a CSV file.

d. Click ”Clear” button, the data in the tree view widgets should be cleared. The sub-window returns to the original state.

A.13 Filters

Filters are designed to smooth the measured data sets by remove suspicious bias and spikes in the data.

a. Log a data log file. Open Graphs sub-window and choose some data curves to display. Check ”Enable filters” check box, and choose a filtering method. The curves in the graph window should becomes smooth according to which filter is chosen.

b. Uncheck the ”Enable filters” check box, the curves should returns to the original look.

Figure A.13 to A.18 compares the unprocessed data curve with different types of processed curves.

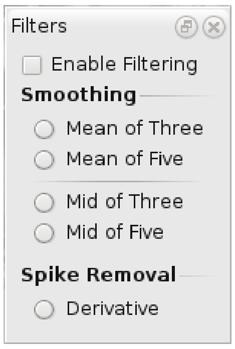


Figure A.12: Filter Dock GUI

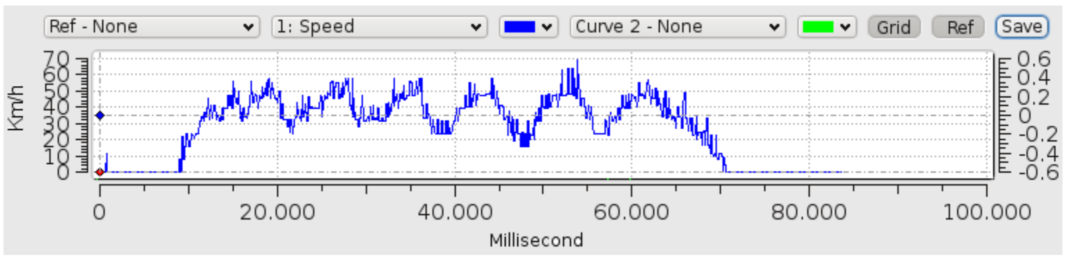


Figure A.13: Speed Data Before Processing

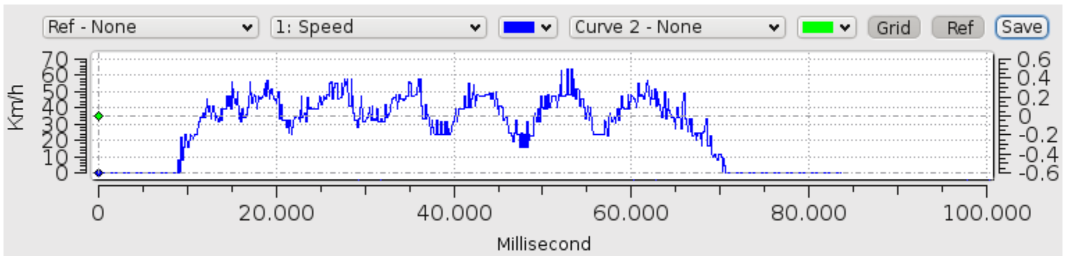


Figure A.14: Speed Data with Spike Removal Processing

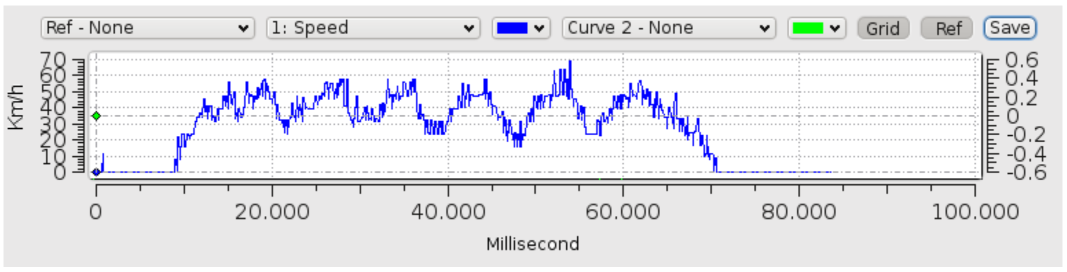


Figure A.15: Speed Data with Median Three Processing

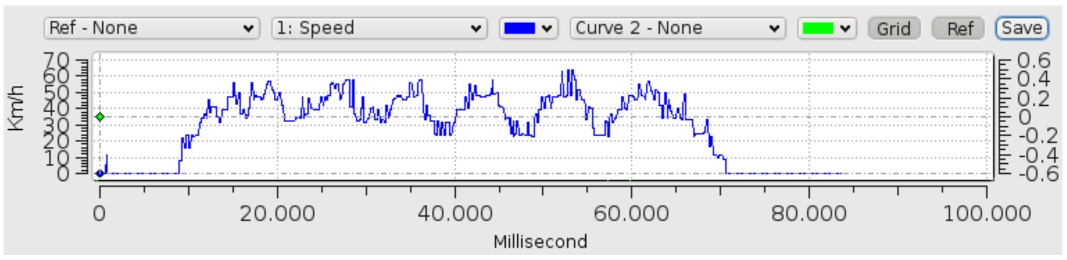


Figure A.16: Speed Data with Median Five Processing

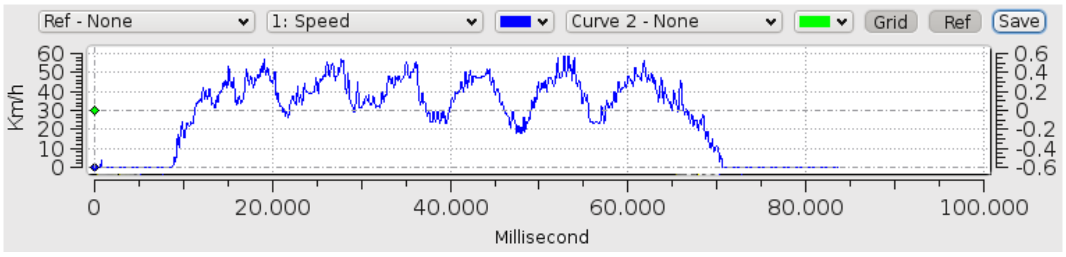


Figure A.17: Speed Data with Average Three Processing

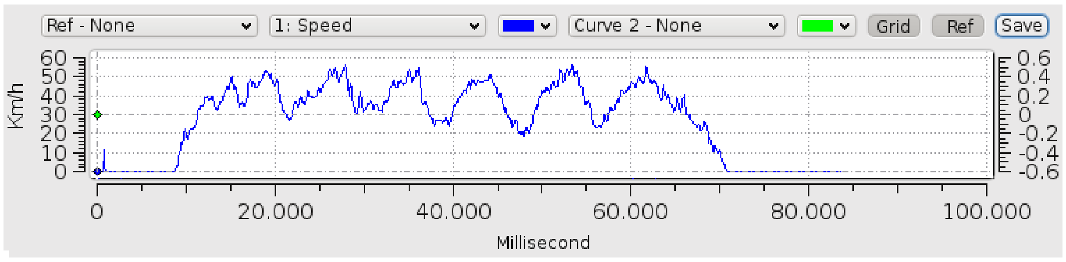
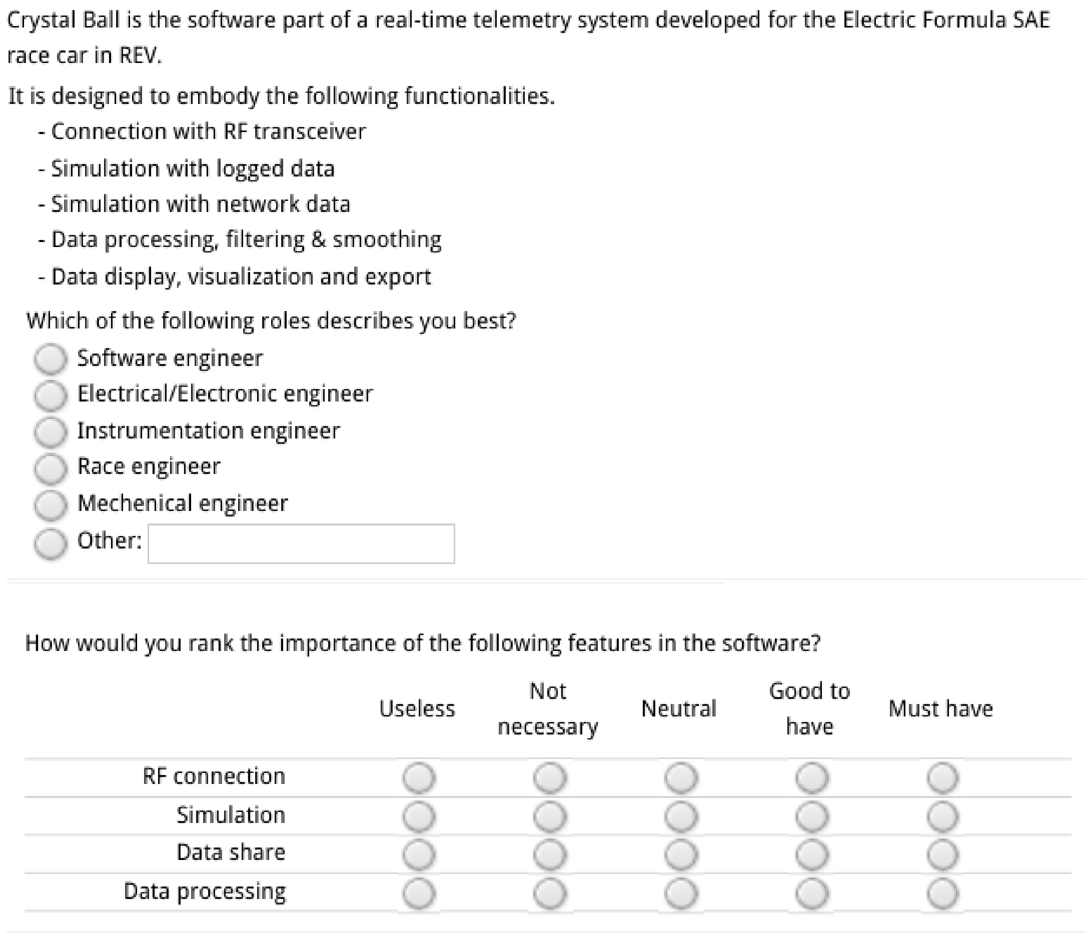


Figure A.18: Speed Data with Average Five Processing

APPENDIX B

User Experience Questionnaire

User Experience is the one of the methods used for validation of Crystal Ball real-time telemetry software. This process comprises user testing and answering a questionnaire anonymously. According to the iterative nature of Extreme Programming, the questionnaire is designed for both validation purpose and future development.


Crystal Ball is the software part of a real-time telemetry system developed for the Electric Formula SAE race car in REV.

It is designed to embody the following functionalities.

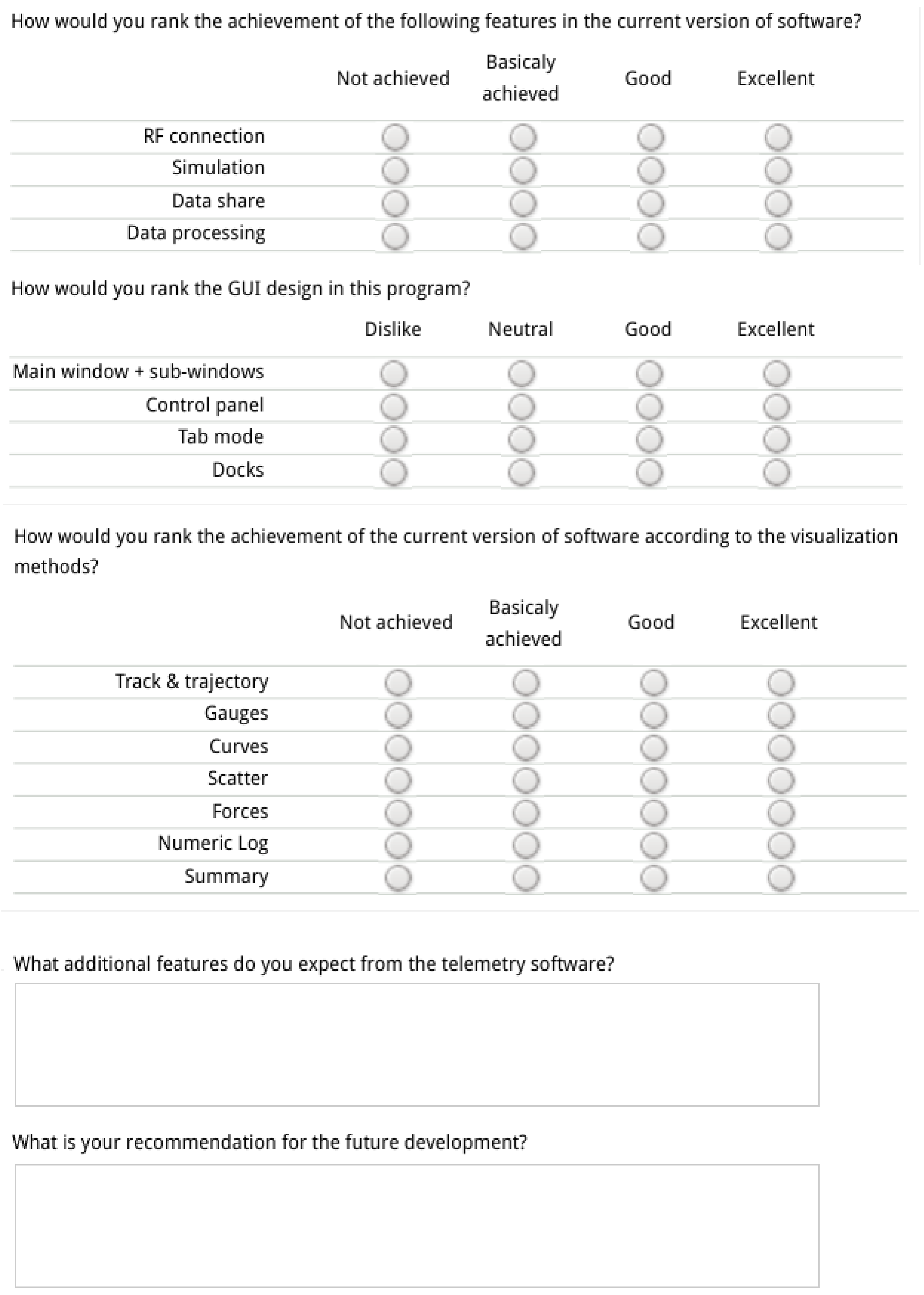
- Connection with RF transceiver
- Simulation with logged data
- Simulation with network data
- Data processing, filtering & smoothing
- Data display, visualization and export

Which of the following roles describes you best?

- ◯ Software engineer
- ◯ Electrical/Electronic engineer
- ◯ Instrumentation engineer
- ◯ Race engineer
- ◯ Mechenical engineer
- ◯ Other:

How would you rank the importance of the following features in the software?

	Useless	Not necessary	Neutral	Good to have	Must have
RF connection	◯	◯	◯	◯	◯
Simulation	◯	◯	◯	◯	◯
Data share	◯	◯	◯	◯	◯
Data processing	◯	◯	◯	◯	◯



How would you rank the achievement of the following features in the current version of software?

	Not achieved	Basicaly achieved	Good	Excellent
RF connection	○	○	○	○
Simulation	○	○	○	○
Data share	○	○	○	○
Data processing	○	○	○	○

How would you rank the GUI design in this program?

	Dislike	Neutral	Good	Excellent
Main window + sub-windows	○	○	○	○
Control panel	○	○	○	○
Tab mode	○	○	○	○
Docks	○	○	○	○

How would you rank the achievement of the current version of software according to the visualization methods?

	Not achieved	Basicaly achieved	Good	Excellent
Track & trajectory	○	○	○	○
Gauges	○	○	○	○
Curves	○	○	○	○
Scatter	○	○	○	○
Forces	○	○	○	○
Numeric Log	○	○	○	○
Summary	○	○	○	○

What additional features do you expect from the telemetry software?

What is your recommendation for the future development?

APPENDIX C

Data File Examples

```
13401,3132,1023,334,−869,259,−1976,255,255
13468,3132,1023,328,−990,591,−2073,255,255
13548,3154,1023,333,−1162,−144,−2751,255,255
13616,3154,1023,328,−1444,259,−3138,255,255
13694,3969,1023,332,−759,1378,−3816,255,255
13762,3969,1023,326,−960,1700,−3526,255,255
13838,3220,1023,329,−970,−123,−3138,255,255
13905,3220,1023,326,−1101,289,−2654,255,255
13984,3969,1023,329,−567,682,−2460,255,255
14053,3969,1023,325,−1464,884,−2266,255,255
14130,3969,1023,330,−1142,480,−2266,255,255
14197,3969,1023,326,−708,228,−1782,255,255
14276,3969,1023,328,−1021,440,−1491,255,255
14345,3969,1023,324,−950,269,−1104,255,255
14424,4699,1023,329,−1011,269,−813,255,255
14491,4699,1023,325,−920,57,−619,255,255
14599,4115,1023,327,−1192,178,−329,255,255
14668,4115,1023,325,−970,−3,−523,255,255
14739,4763,1023,328,−1253,117,−426,255,255
14806,4763,1023,325,−970,−113,−426,255,255
14885,3969,936,354,−890,27,−232,233,233
14962,3969,766,437,−1061,−93,−135,190,190
15031,5557,459,511,−1081,−3,−232,114,114
```

Listing C.1: Data Log File Example

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx version="1.0" creator="GPSBabel">
<time>2011-04-20T17:04:42Z</time>
<bounds minlat="-31.950066000" minlon="115.983986000"
    maxlat="-31.945496000" maxlon="115.986969000"/>
<wpt lat="-31.949785000" lon="115.984337000">
  <ele>0.000000</ele>
  <name>Turn 1</name>
  <cmt>Turn 1</cmt>
  <desc>Turn 1</desc>
</wpt>
<wpt lat="-31.946846000" lon="115.986588000">
  <ele>0.000000</ele>
  <name>Braking Markers</name>
  <cmt>Braking Markers</cmt>
  <desc>Braking Markers</desc>
</wpt>
<wpt lat="-31.946196000" lon="115.986931000">
  <ele>0.000000</ele>
  <name>Turn in Point</name>
  <cmt>Turn in Point</cmt>
  <desc>Turn in Point</desc>
</wpt>
<trk>
  <name>RAC</name>
<trkseg>
<trkpt lat="-31.949240000" lon="115.984612000">
  <ele>0.000000</ele>
</trkpt>
<trkpt lat="-31.949547000" lon="115.984444000">
  <ele>0.000000</ele>
</trkpt>
<trkpt lat="-31.949732000" lon="115.984344000">
  <ele>0.000000</ele>
</trkpt>
</trkseg>
</trk>
</gpx>
```

Listing C.2: GPX Track File Example

```
13401,31.32,23687.4,0.999023,0.326172,−0.8486330.25293,−1.976,0,0,1,1,,360,360,0,0,0,0,0,0
13468,31.32,23687.4,0.999023,0.320312,−0.9667970.577148,−2.073,0,0,1,1,,360,360,0,0,0,0,0,0
13548,31.54,23853.8,0.999023,0.325195,−1.13477−0.140625,−2.751,0,0,1,1,,360,360,0,0,0,0,0,0
13616,31.54,23853.8,0.999023,0.320312,−1.410160.25293,−3.138,0,0,1,1,,360,360,0,0,0,0,0,0
13694,39.69,30017.7,0.999023,0.324219,−0.7412111.3457,−3.816,0,0,1,1,,360,360,0,0,0,0,0,0
13762,39.69,30017.7,0.999023,0.318359,−0.93751.66016,−3.526,0,0,1,1,,360,360,0,0,0,0,0,0
13838,32.2,24353,0.999023,0.321289,−0.947266−0.120117,−3.138,0,0,1,1,,360,360,0,0,0,0,0,0
13905,32.2,24353,0.999023,0.318359,−1.07520.282227,−2.654,0,0,1,1,,360,360,0,0,0,0,0,0
13984,39.69,30017.7,0.999023,0.321289,−0.5537110.666016,−2.46,0,0,1,1,,360,360,0,0,0,0,0,0
14053,39.69,30017.7,0.999023,0.317383,−1.429690.863281,−2.266,0,0,1,1,,360,360,0,0,0,0,0,0
14130,39.69,30017.7,0.999023,0.322266,−1.115230.46875,−2.266,0,0,1,1,,360,360,0,0,0,0,0,0
14197,39.69,30017.7,0.999023,0.318359,−0.6914060.222656,−1.782,0,0,1,1,,360,360,0,0,0,0,0,0
14276,39.69,30017.7,0.999023,0.320312,−0.997070.429688,−1.491,0,0,1,1,,360,360,0,0,0,0,0,0
14345,39.69,30017.7,0.999023,0.316406,−0.9277340.262695,−1.104,0,0,1,1,,360,360,0,0,0,0,0,0
14424,46.99,35538.7,0.999023,0.321289,−0.9873050.262695,−0.813,0,0,1,1,,360,360,0,0,0,0,0,0
14491,46.99,35538.7,0.999023,0.317383,−0.8984380.0556641,−0.619,0,0,1,1,,360,360,0,0,0,0,0,0
14599,41.15,31121.9,0.999023,0.319336,−1.164060.173828,−0.329,0,0,1,1,,360,360,0,0,0,0,0,0
14668,41.15,31121.9,0.999023,0.317383,−0.947266−0.00292969,−0.523,0,0,1,1,,360,360,0,0,0,0,0,0
14739,47.63,36022.8,0.999023,0.320312,−1.223630.114258,−0.426,0,0,1,1,,360,360,0,0,0,0,0,0
14806,47.63,36022.8,0.999023,0.317383,−0.947266−0.110352,−0.426,0,0,1,1,,360,360,0,0,0,0,0,0
14885,39.69,30017.7,0.914062,0.345703,−0.8691410.0263672,−0.232,0,0,0.913725,0.913725,,360,360,0,0,0,0,0,0
14962,39.69,30017.7,0.748047,0.426758,−1.03613−0.0908203,−0.135,0,0,0.745098,0.745098,,360,360,0,0,0,0,0,0
15031,55.57,42027.8,0.448242,0.499023,−1.05566−0.00292969,−0.232,0,0,0.447059,0.447059,,360,360,0,0,0,0,0,0
15098,55.57,42027.8,0.225586,0.625,−0.966797−0.208984,−0.038,0,0,0.223529,0.223529,,360,360,0,0,0,0,0,0
15175,48.3,36529.5,0.0908203,0.665039,−0.701172−0.179688,−0.038,0,0,0.0901961,0.0901961,,360,360,0,0,0,0,0,0
15243,48.3,36529.5,0.0292969,0.706055,−1.18359−0.228516,0.154,0,0,0.027451,0.027451,,360,360,0,0,0,0,0,0
15312,42.61,32226.1,0.0117188,0.706055,−0.652344−0.258789,0.445,0,0,0.00784314,0.00784314,,360,360,0,0,0,0,0,0
15380,42.61,32226.1,0.0146484,0.711914,−1.24316−0.249023,0.445,0,0,0.0117647,0.0117647,,360,360,0,0,0,0,0,0
15450,42,31764.8,0.0244141,0.704102,−0.819336−0.228516,0.348,0,0,0.0235294,0.0235294,,360,360,0,0,0,0,0,0
15521,42,31764.8,0.0292969,0.707031,−0.8193360.499023,0.154,0,0,0.027451,0.027451,,360,360,0,0,0,0,0,0
```

Listing C.3: Processed Data File Example

```
Lap 1
Milisecond ,45211 ,45211 ,627 ,NA,44584
Speed Adjusted ,41.4 ,57.96 ,0 ,31.6913 ,NA
RPM,31311 ,43835.4 ,0 ,23968.3 ,NA
Accel Padel ,0.0341797 ,0.999023 ,0.0117188 ,0.552654 ,NA
Brake Padel ,0.705078 ,0.713867 ,0.313477 ,0.498073 ,NA
Forward G-force , -1.06543 , -0.376953 , -1.45898 , -0.989536 ,NA
Sideways G-force , -0.53418 ,1.69922 , -1.92188 , -0.989536 ,NA
Yaw Rate ,2.867 ,7.226 , -7.497 , -0.27523 ,NA
FL Motor ,0 ,0 ,0 ,0 ,NA
FR Motor ,0 ,0 ,0 ,0 ,NA
BL Motor ,0.0313725 ,1 ,0.00784314 ,0.552045 ,NA
BR Motor ,0.0313725 ,1 ,0.00784314 ,0.552045 ,NA

Lap 2
Milisecond ,83407 ,83407 ,45287 ,NA,38120
Speed Adjusted ,0 ,68.86 ,0 ,22.799 ,NA
RPM,0 ,52079.1 ,0 ,17243 ,NA
Accel Padel ,0.0166016 ,0.999023 ,0.0117188 ,0.357838 ,NA
Brake Padel ,0.708984 ,0.712891 ,0.307617 ,0.575106 ,NA
Forward G-force , -1.00684 ,0.34082 , -2.00098 , -0.980488 ,NA
Sideways G-force ,0.0166016 ,2.32031 , -1.68555 , -0.980488 ,NA
Yaw Rate , -0.135 ,7.129 , -9.919 , -0.135703 ,NA
FL Motor ,0 ,0 ,0 ,0 ,NA
FR Motor ,0 ,0 ,0 ,0 ,NA
BL Motor ,0.0156863 ,1 ,0.00784314 ,0.356437 ,NA
BR Motor ,0.0156863 ,1 ,0.00784314 ,0.356437 ,NA
```

Listing C.4: Track Summary File Example

Bibliography

[1] Astels, D. *Test-driven Development: A Practical Guide*. The Coad series. Prentice Hall PTR, 2003.

[2] Cocco, L., and Daponte, P. Metrology and Formula One Car. In *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008. IEEE* (may 2008), pp. 755 –760.

[3] Dah-Jing Jwo. Efficient DOP Calculation for GPS with and without Altimeter Aiding. *The Jounal of Navigation 54*, 2 (2001), 269–279.

[4] Foster, D. GPX: the GPS Exchange Format. `http://www.topografix.com/gpx.asp`, April 2011. Accessed on 3rd May, 2011.

[5] Galitz, W. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. John Wiley, 2002.

[6] HE, J. Data Cleaning for Environmental Sensor Networks. Master thesis, The University of Western Australia, December 2010.

[7] Isenburg, M., and Snoeyink, J. Binary Compression Rates for ASCII Formats. In *Proceedings of the eighth international conference on 3D Web technology* (New York, NY, USA, 2003), Web3D '03, ACM, pp. 173–ff.

[8] J Postel. *User Datagram Protocol*. Internet Engineering Task Force, ietf.org, August 1980. RFC 768.

[9] J Postel. *Transmission Control Protocol*. Internet Engineering Task Force, ietf.org, September 1981. RFC 793.

[10] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*, 4th ed. Addison Wesley, Reading, Massachusetts, US, April 2007.

[11] Jung, D.-H., Jeong, G.-M., Ahn, H.-S., Ryu, M., and Tomizuka, M. Remote Diagnostic Protocol and System for U-car. In *Proceedings of the 1st international conference on Ubiquitous convergence technology* (Berlin, Heidelberg, 2007), ICUCT'06, Springer-Verlag, pp. 60–68.

[12] KENT BECK AND CYNTHIA ANDRES. *Extreme Programming Explained: Embrace Change*, 2nd ed. Addison-Wesley, Reading, Massachusetts, US, November 2004.

[13] KLAUS BETKE. *The NMEA 0183 Protocol.* The National Marine Electronics Association, 7 Riggs Avenue Severna Park, MD 21146, May 2000. http://www.nmea.org/.

[14] KNISLEY, J., AND WERCHAN, H. Telemetry: Enhancing Tactical Network Management. In *Tactical Communications Conference, 1994. Vol. 1. Digital Technology for the Tactical Communicator., Proceedings of the 1994* (may 1994), pp. 235 –244.

[15] KORTUM, P. *HCI Beyond the GUI: Design for Haptic, Speech, Olfactory, and Other Nontraditional Interfaces (Interactive Technologies) (Interactive Technologies).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[16] MCLAREN-ELECTRONICS. Advanced Telemetry Linked Acquisition System. `http://www.mclarenelectronics.com/Products/All/SW_Atlas.asp`, November 2009. Accessed on 15th April, 2011.

[17] MOTEC. MoTeC i2 Telemetry Monitor. `http://www.motec.com.au/telemetry/telemetryoverview/`, March 2008. Accessed on 15th April, 2011.

[18] MOTEC. MoTeC Parts Catalogue. `http://www.motec.com.au/filedownload.php/MoTeC_Catalog%201.40.pdf?docid=2185`, June 2008. Accessed on 15th April, 2011.

[19] NOKIA. *Qt 4.6: QMainWindow Class Reference.* Nokia Corporation, Keilalahdentie 2-4, FI-02150 Espoo, December 2009. Technical Manual.

[20] NOKIA. *Qt 4.6: QSettings Class Reference.* Nokia Corporation, Keilalahdentie 2-4, FI-02150 Espoo, December 2009. Technical Manual.

[21] NOKIA. *Qt 4.6: Reentrancy and Thread-Safety.* Nokia Corporation, Keilalahdentie 2-4, FI-02150 Espoo, December 2009. Technical Manual.

[22] NOKIA. Qt Creator IDE and tools. `http://qt.nokia.com/products/developer-tools/`, December 2009. Accessed on 4th May, 2011.

[23] PEARCE, J. Electric Vehicle Telemetry. Honour thesis, The University of Western Australia, October 2010.

[24] SAE-International. Competition History. http://www.sae.org/students/fsaehistory.pdf, April 2005. Accessed on 12th April, 2011.

[25] SAE-International. About Formula SAE. http://students.sae.org/competitions/formulaseries/about.htm, April 2011. Accessed on 12th April, 2011.

[26] Stanley, G. Fault Detection and Diagnosis. http://www.gregstanleyandassociates.com/whitepapers/FaultDiagnosis/faultdiagnosis.htm, December 2010. Accessed on 6th May, 2011.

[27] TAN, F. Y., Holms, P., Hooper, I., and Walter, T. Discussion of The Reqirement of FSAE Telemetry Software, November 2010.

[28] Waldo, J. Embedded computing and Formula One racing. *Pervasive Computing, IEEE 4*, 3 (july-sept. 2005), 18 – 21.

[29] Walter, T. Development of a User Interface for Electric Cars. Master thesis, University of Stuttgart, December 2010.

[30] Webster, J. *The Measurement, Instrumentation, and Sensors Handbook.* The electrical engineering handbook series. CRC Press published in cooperation with IEEE Press, 1999.

[31] Wikipedia. Dilution of precision (GPS). http://en.wikipedia.org/wiki/Dilution_of_precision_%28GPS%29, April 2011. Accessed on 8th May, 2011.

[32] Wikipedia. Plug-in (computing). http://en.wikipedia.org/wiki/Plug-in_%28computing%29, May 2011. Accessed on 15th May, 2011.