Marcus Pham

4 Hannans Street

Morley, WA, 6062

27 October 2014

Winthrop Professor John Dell

Dean

Faculty of Engineering, Computing and Mathematics

University of Western Australia

35 Stirling Highway

Crawley, WA, 6009

Dear Sir,

I submit to you this final report entitled Renewable Energy Vehicles' User Interface in partial fulfillment of the requirement of the award of Bachelor of Engineering.

Yours Faithfully,

Marcus Pham

Renewable Energy Vehicles' User Interface

Redesign of REV GUI and Instrumentation
&
Road Testing Of GM Volt

Marcus Pham, University Of Western Australia

***Abstract* - In any vehicle, the first point of human to machine interaction is to the instrumentation inside of the vehicle's cockpit.**

Instrumentation is particularly important in an electric vehicle, as a user will need to receive information to gauge the current state of the vehicle whilst not overloading the driver with information and causing a distraction to the actual driving of the vehicle.

This project will focus on creating a more robust and improved version of the Graphical User Interface (GUI) inside of the two Renewable Energy Vehicles (REV). The new interface will predominantly aim to satisfy the likely users of electric vehicles. As such, the new user interface will aim to give the user what is most likely to be the main reason behind their choice of an electric vehicle over and internal combustion vehicle, efficiency.

Improvements will enable users to measure and see their driving efficiency and compare this with previous trips in real time. This will encourage drivers to more effectively reduce energy use, which will reduce both cost and benefit the environment. This task will involve the addition of new features as well as removing redundant and unrequired features to maintain ease of use.

I. Introduction

THERE is a growing need to find alternative methods of transport. With the ever-depleting sources of fossil fuels, the growing concerns of ozone layer depletion and the new awareness of the impact of internal combustion engine (ICE) vehicles on carbon emissions, there has never been a more suitable time to search for other methods for propulsion.

Due to the steadily increasing emissions restrictions on passenger vehicles, many car manufacturers are researching other viable methods to create cleaner and most efficient vehicles.

This has slowly led electric vehicles slowly becoming a more and more suitable alternative to the conventional ICE vehicle.

This paper is submitted in partial completion of a Bachelor of Electrical Engineering/Bachelor of Computer and Math Science at the University of Western Australia. Submitted 27/10/2014.

M. Pham, 20495924@student.uwa.edu.au is with the University of Western Australia, Crawley, WA 6009 Australia.

Supervised by Prof. Thomas Bräunl, Electrical, Electronic and Computer Engineering, University of Western Australia.

A. Electric Vehicles

Electric vehicles (EV) have been around for over 100 years [1], since the early 1800's, being even more popular than ICE vehicles at one point in time. However, due to various problems and limitations of an electric motor, the electric vehicle lost popularity since the 20th century. The major contributing factor to the loss in popularity is the speed and range limitations on battery-powered vehicles.

Modern EV's are still limited by range issues and the biggest concern of an EV owner is range-anxiety. Even with all the advances in technology over the years, the maximum range for an EV that has been commercially sold is ~400km, the Tesla Model S. [2]

Despite this, there are a whole host of benefits to having an EV in comparison with an ICE vehicle. Some benefits include:

Cutting down on CO_2 emissions. EVs do not produce any CO_2 emissions when in use. The carbon footprint of EVs is from the energy source to charge them. Additionally, the conversion form electricity to kinetic energy is much more efficient than chemical to kinetic, further reducing energy usage.

Cheaper running costs. With the steadily decreasing stores of fossil fuels, fuel prices are on the rise. Additionally, recharging an EV is much cheaper than refueling a vehicle for the same distance travelled.

Lower maintenance costs. EVs have less moving parts and hence are much simpler compared with ICE vehicles. This relates to less services and less maintenance required to keep the cars on the road.

Less noise pollution. EVs don't produce much, if any, noise when running. Major sources of noise come from the tire noise present at higher speeds. [3]

B. The REV Project

The UWA Renewable Energy Vehicle (REV) project is a project by the University of Western Australia (UWA) that was restarted in 2008 with aims to prove the viability and 'revolutionise personal transport' by building zero emission vehicles, charged from renewable sources [4]. Currently UWA has 2 road-legal fully electrical vehicles that provides a solution to the before mentioned problem.

The REV Eco was the first plug-in electric conversion project that was started and completed in 2008. The REV Eco was originally a Hyundai Getz 2008 [5], which had been converted to electric drive by replacing the internal combustion engine with an electric drive motor, batteries wiring and instrumentation.

Fig. 1. The REV Eco, an electrically converted Hyundai Getz converted as part of the REV Project in 2008.

The second conversion made by UWA to convert an ICE to electric drive vehicle is the REV Racer. This car was based on a 2002 Lotus Elise S2 [6] and like the Getz involved complete replacement of the ICE by a motor controller, batteries and instrumentation.

Fig. 2. The REV Racer, the second car converted by the REV Project in 2009.

C. Instrumentation

In any vehicle, the first point of person to machine interaction is to the instrumentation inside of the vehicle's cockpit. Over the years, there have been many changes in the display mode and method with regards to vehicle instrumentation. Improvements and upgrades have evolved over the years through improvements of technology and changing views on the required elements.

The basic requirements that are present in almost every modern vehicle is: a speedometer, tachometer, odometer, fuel level gauge, water temperature gauge and warning indicators. This instrumentation can vary from vehicle to vehicle varies depending on the needs and desires of the target driver, but the interface must be there to inform the user of current statistics as well as any problems or issues present.

In an electric vehicle the importance of the instrumentation, especially the user interface, is even more important as statistics on battery charge, battery condition and distance remaining are critical to the user.

However, the instrumentation has to be carefully designed to give the user sufficient information to gauge the state of the vehicle whilst not overloading the user, causing a distraction from the actual driving of the vehicle. Hence the interface is integral with the safety of a vehicle.

In general, there are many similarities to an ICE vehicle in terms of the vehicle instrumentation. Both cars require a speedometer and basic warning alerts. Also similar is the state of charge level in an electric vehicle in comparison to an ICE vehicle.

However there are a few differences that are common in most electric vehicle instrumentation that is different to an ICE vehicle. Most electric vehicles are automatic, due to the nature of the electric motor, and hence don't require a tachometer as gear changes are not required. Distance remaining estimates are critically important in electric vehicles due to the difficulty in recharging the batteries in contrast to re-fueling an ICE. Hence almost all electric vehicles are equipped with these whereas this trend has only started appearing on ICE vehicles. Additionally, most EVs have multiple instrument panels, usually involving a main dash and a secondary interface to get additional details. [7]

Over the last few years, students have developed and improved the instrumentation inside the two REV cars as shown in Appendix I.

Though there have been various improvements over the years, from ergonomics to aesthetics and information, there has not been a major focus on providing what the likely drivers of an EV would want in a GUI.

One of the biggest reasons that drivers would want an EV is to reduce their carbon footprint. As such, driver would likely want improve efficiency from the vehicle, further reducing the energy usage and hence decreasing carbon emissions to the environment.

With this in mind, a specially designed GUI for EVs should employ a variety of driver aids to encourage drivers to drive more efficiently, both in the short term and in the long term.

Finally, the GUI should aim to be simple yet provide all required information needed by the driver.

II. Road Testing of Commercial Electric Vehicle

In addition to the work on the instrumentation of the two REV cars, there was additional research done. This testing involved the read-world testing of a currently mass produced full electric vehicle, the Holden/Chevrolet Volt.

A. Background

The General Motors (GM) Volt entered the Australian car market in 2013, and is the first 'long range' electric vehicle to be sold in the Australian market. GM state the Volt can drive petrol free for up to 87km on a fully charged battery, and upon depleting the battery, automatically switches to the petrol generator giving the vehicles a total range of over 600km.

However, these result are under ideal conditions and not under 'every-day' driving. Hence testing was done to investigate the performance of the Volt under a range of different conditions. [8]

B. Testing

The University of Western Australia (UWA) and Murdoch University (MU) did this research as a collaborative investigation into the actual performance of the Volt. This was done under a wide variety of conditions and compared with the REV Eco, which was done in 2013 by Jonathan Oakley and Thomas Bräunl. [9]

Efforts were made to keep testing conditions as similar to tests done on the REV Eco as possible. This included travelling on the same test track during similar times, however, there are some slight differences in the tests done due to differences in the vehicles. Main differences include transmission and regenerative braking.
The conditions tested are shown below in Table 1.

Volt tests	REV Eco tests
City vs. Highway Off-peak vs. Peak 0 passengers vs. driver plus 2 Electric air-conditioning on/off 'Normal' (D) vs. 'Low' (L) vs. 'Sport' Modes 'Hold' Mode (continuous petrol engine usage) "Long range' test.	City vs. Highway Off-peak vs. Peak 0 passengers vs. driver plus 2 Electric air-conditioning on/off Headlights & radio on/off Electric heater on/off

Table 1. Tests conducted on the REV Eco in 2013 against the test conducted on the Volt in 2014

These tests were repeated 3 times by UWA and another 5 times by MU to reduce the influence of varying traffic conditions.

C. Results

Upon examination of the results obtained from the trials, clear conclusions can be made on the energy usage of the Volt. Though there are some expected results, significant increases in energy usage under the use of air-conditioning and extra load, there were some surprising results obtained.

One result that was interesting to note was the small, almost insignificant increase in energy use using the 'Sports Mode'. This mode enabled a more 'sporty' feel to the vehicle, enabling drivers to get more torque and acceleration from the vehicle. However, results showed a 2% increase in energy usage, much less than expected.

This was in contract to the noticeable increase in energy consumption using 'Low'. Due to the regenerative braking abilities of the vehicle, the expectation was to increase efficiency by allowing the vehicle to brake the car automatically in the 'Low' setting, hence regenerating more energy than in 'Normal'. However, this was met with a significant 10% increase in energy usage. More research will need to be done to confirm these results.

All results are more thoroughly discussed in the report [10].

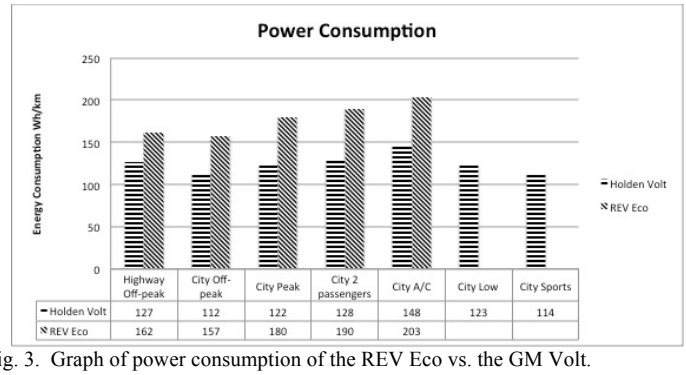

Fig. 3. Graph of power consumption of the REV Eco vs. the GM Volt. Energy is measured in Wh per km.

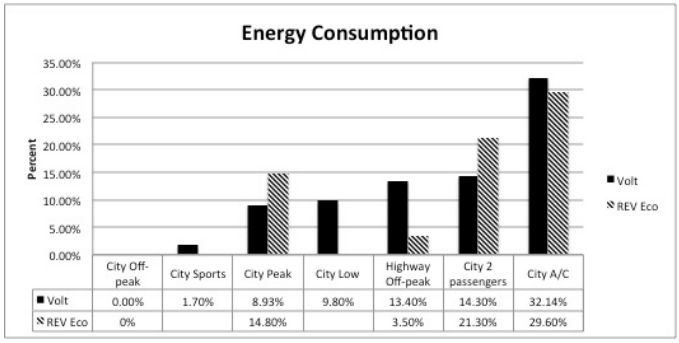

Fig. 4. Percentage view of energy consumption of the REV Eco vs. the GM Volt. Fig. 4 clearly shows the increased energy consumption using A/C and extra load.

III. Objectives

This project aims to improve the existing Human-Machine Interface (HMI) in the REV Eco and REV Racer. This is to be done predominantly by modifying the GUI in order to promote more efficient driving behaviours, hence extending the biggest issue for drivers using an Electric Vehicle (EV).

However, before improvements can be made to the GUI, there are a host of issues that need to be addressed. This included upgrades in software, hardware and bug fixes in the pre-existing GUI.

The tasks of this project are:

1. Compete redesign of GUI to enable easier usage and encourage efficient driving styles
2. Upgrade of operating system and upgrade of QT4 to QT5
3. Install of Hall-Effect Sensor to the dash to enable tachometer
4. Connection of REV Racer's CAN bus to Raspberry Pi
5. Design and install of IO hardware to connect from vehicle to Raspberry Pi
6. Re-implement a music player in the HMI
7. Re-implement a synthetic engine sound system

Through these changes, this project aims to improve the usability and functionality of the HMI inside of the two REV cars.

IV. Previous Design

A. *Pre 2013 – REV Eco*

Previously, the REV Eco used the Eyebot M6 embedded controller. The programs on the controller consisted of the main GUI, a logger, Global Positioning System (GPS), and Battery Management System (BMS) daemons. The BMS also used a TBS E-Xpert Pro [11] battery monitor that communicated to the Eyebot via a RS232 communication kit.

The interface was developed by Beau Trepp and allows users to view various details including battery condition, trip, money saved and a map of the current position. [12]

Figure 6. The Eyebot M6, the controller to control instrumentation in the REV Eco up until 2013.

B. *Pre 2013 – REV Racer*

The REV Racer previously contained a Dual Core Atom 1.6 GHz PC within a VoomPC-2 enclosure for work in automotive environments. This was connected to a Xenarc 700TSV touchscreen monitor [13] via USB interface.

The program was developed using the QT framework [14] by Thomas Walter in 2010 [15] and modified by Matthew Tyler in 2011 [33]. This program provided similar details to the REV Eco, with added functionality of being able to monitor individual battery cell condition. The BMS in this vehicle was developed by Ivan Neubronner to allow real-time monitoring of the battery cell voltage and to send this data to the PC.

Additionally, the program was designed to implement the Engine Audio Replicating System (E.A.R.S.). E.A.R.S. is a background thread that takes in the engine speed data from the GUI and opens a pre-recorded audio file of an engine for that corresponding speed rounded to the nearest 100 revolutions per minute (RPM).

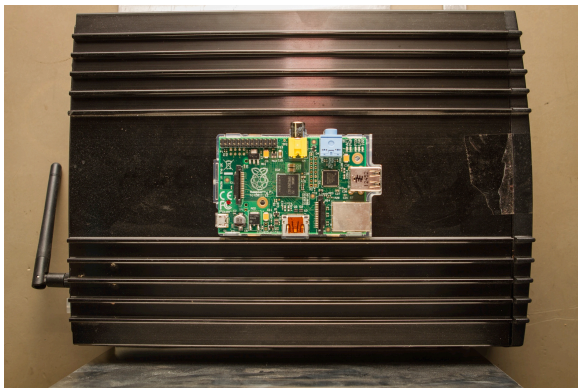

Fig. 7. A comparison between the old VoomPC, the system used to control the REV Racer, and the new system, the Raspberry Pi.

C. *Problems in the Pre 2013 Systems*

Though the old system was functional, there were many issues with the system overall. Firstly, the system was expensive and difficult to remove and modify. This led to increased maintenance costs, as the cars had to be brought into the workshop to get the hardware fixed.

The Eyebot had the only working copy of the software. This is a potential problem, as the controller did not shutdown safely, only turning off when the ignition power was cut. This action could possibly corrupt the image.

The PC inside the Lotus had a fan, which created more undesirable noise than the car itself. This in addition to being large and bulky made the system not ideal for automotive use.

D. *Post 2013 – Raspberry Pi*

Due to the host of problems of using the pre 2013 system, a hardware change was needed. This was proposed and implemented in 2013 by Gabriel Feng. [16]

The Raspberry Pi is now replacing the systems in both the REV Eco and REV Racer. The Raspberry Pi is a small and low power computer that is capable of doing many of the things a desktop PC can do, from word processing to playing video. The idea behind the production of the device was to create cheap computer in order to encourage students to explore and experiment without fear of damaging expensive hardware.

The Raspberry Pi model B, is fitted with a Broadcom BCM2835 chip, 512MB RAM, and a Videocore 4 GPU. It also has 2 USB ports, an Ethernet port, an HDMI port, RCA Video output, 3.5mm audio output, several GPIO pins and an SD card slot from which it stores all the physical memory required to run the system. [17]

Though the device was originally created for educational purposes, it proved to be ideal for use as a vehicular computer. Reasons for this included:

- Light and compact enabling easy mounting in a vehicle
- Removable memory (SD Card) enables easy modifications to the software without need of removing hardware.
- Low cost, which enabled easy replacement in case of hardware failure
- Low energy consumption, improving efficiency of the EV when in use

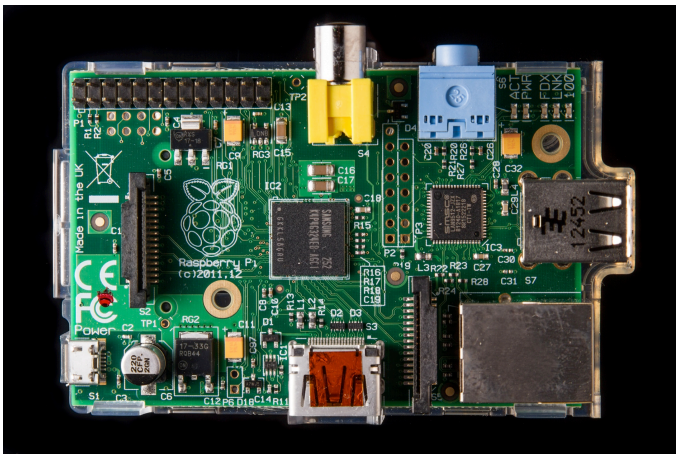

Fig. 8. The Raspberry Pi, which replaces the old controllers in both REV cars.

E. Post 2013 – Graphical User Interface

As a result of both vehicles using the same computer for their GUI, both REV vehicles now have similar GUIs. The user interfaces in both vehicles were given a makeover with aims to improve visibility of the information on the screen. The screen now incorporated more buttons, and a circular layout, in conjunction with the new Powermate Rotary Knob [18] and the Gizmo Daemon [19], which was installed in the REV Eco. Various improvements were made.

These included:

- Warning Icons
- Night Mode
- Larger Map display
- Trip display
- Engine Speed display/ gear change aid

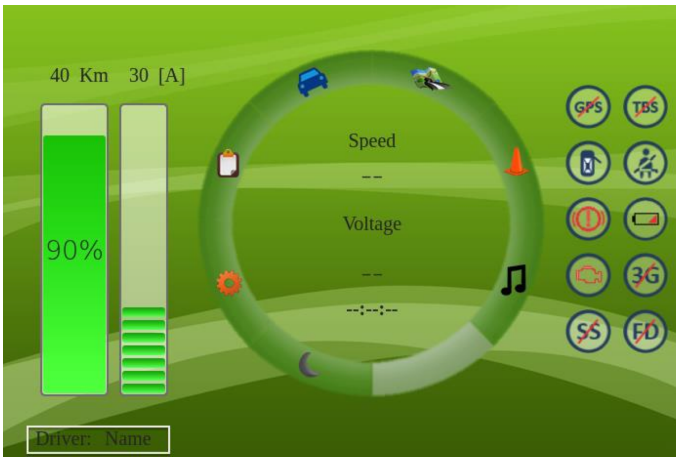

Fig. 9. The GUI designed by Gabriel Feng (2013)

F. Problems with the GUI

Though there were a lot of improvements made in terms of the hardware, there were still aspects of the vehicular instrumentation that needed updating. As with any major hardware change, much of the software had become obsolete. This caused a major loss of functionality in the user interfaces in the cars.

One of the major complaints about the GUI was the size of fonts and buttons. With the change to a circular button layout, the sizes of the buttons themselves were compromised. This made it difficult to navigate between pages and difficult to read test on the screen.

Additionally, there were some hardware problems with the instrumentation of the two cars. Firstly, the cluster tachometer was not working in either car, making it difficult to change gears. Secondly, though warning icons were created on the GUI and code had been written, the IO board to enable the Raspberry Pi to change the icons was not made. Hence the warning icons remained off regardless of the state of the car.

Also, the media player of the interface was not working either. And finally, due to the porting from a PC using Windows to a Raspberry Pi running Linux and the aforementioned fact that the IO module had not been made, E.A.R.S was not well implemented.

V. New GUI Design

The main focus of the new design was to cater for the target market for EVs, customers who are concerned about the environment. Hence the aim of the user interface is to enable an easy to use interface that encourages drivers to apply more efficient driving techniques.

A. Larger Fonts/Bigger Buttons/Simpler Design

One of the biggest complaints with the pre-existing GUI was to do with the size of buttons and the font size. The small buttons combined with the button layout caused difficulty navigating to different pages.

Small fonts also made it difficult to view data displayed on screen, requiring more time to view the data, taking attention away from the actual driving of the vehicle.

Additionally, the button layout was inefficient. The layout required users to have to return to the home page after navigating to other screens. This was not ideal as it again removed valuable time for drivers to concentrate on driving.

Hence a completely new scheme, which consisted of larger buttons, larger fonts and a simplified layout, was designed to combat these issues. Care has been taken to design the GUI so users focus on the more important pieces of information on screen, by being in larger and more bold fonts, whilst less important information is less focused on by being smaller and out of view.

One main focus for the GUI was for the GUI to be able to switch to other windows without the need to return back to the home screen. This is made possible through the addition of the buttons at the bottom of the window, which is present at all times, on all windows. Buttons are blackened out until focus is placed on the specific button, which highlights and gives that button colour, then pressed to get to that particular window.

Navigation of the buttons has not changed from the previous implementation. Navigation to different buttons is done through key presses due to the reliance of the REV Eco on the Griffin Powermate (rotating knob). Key presses are shown below.

Action	Keyboard Key	Powermate KeyPress
Left on navigation button	Q	Rotation left
Right on navigation button	W	Rotation right
Selection on navigation button	E	Press down
Navigation on buttons in windows	TAB	Rotation left/right
Selection of buttons in windows	SPACE	Press down
Exit a window	ESC	Hold down
Exit a menu	ESC	Hold down

Table 2. The navigation keys in the GUI of the REV Eco.

Fig. 10. The newly designed GUI used in the REV Eco.

B. Real-Time Current/Speed Graph

As part of the aims to encourage more efficient driving techniques, a graph that displayed the current and speed was created in the trip screen which enabled users to easily see the effects of their driving, in real-time.

The values of current and speed are taken from the TBS meter and GPS respectively and points are plotted to the screen every second. Additionally, as points are added, the scale and axes increase in size to accommodate the increases in the graph, capped to 3600 points, so as to not overextend any memory constraints.

The graph contains both readings on separate scales and with different colours on for each graph. This enables easy and quick readability for users.

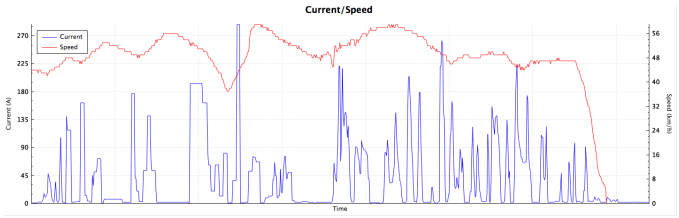

Fig. 11. A current/speed graph created by the GUI on a drive around UWA in the REV Eco. The main purpose of the graph is for drivers to view their driving habits in real time and attempt to reduce peaks in current.

C. Efficiency Energy/Distance Graph

In addition to the current/speed graph, design included another graph, the efficiency graph. This graph plots a point corresponding to the energy used and distance travelled in previous trips, in order to view any trends in driving behaviour and to further encourage drivers to employ more energy efficient driving techniques in the long term.

Data is taken from the TBS meter and GPS module to be stored in a text file. Users can choose to add data points upon pressing the 'Save Efficiency' button, which stored the values of Ah and distance travelled. This is then displayed upon the graphical image in the log tab so users can view their statistics of previous trips. One addition that is to be included is the display to the current trip and the average of previous trips below the graph to allow users to see how well they compare with respect of prior trips.

This graph enables driver to have a visual representation of the energy usage and to try and drive in ways as to reduce the energy used per trip. This is located in the log tab on the GUI.

D. *Efficiency Bar/Economy Label*

The final addition to the series of aids to push drivers to drive more efficiently is the efficiency bar and economy label displayed on the home screen.

The efficiency label increases dependent on the amount of energy that a user has consumed during the current trip. As a driver increases usage, the bar decrease and the percentage efficiency drops. The method to calculate efficiency is a simple integer addition scale. Depending on the amount of current draw, an integer is added to the usage integer. This is then divided with a separate counter that sums up the equivalent time. Finally to calculate the percentage, these numbers are divided to give a relative 100% for how efficient a driver is driving.

The efficiency label is created using data taken from the TBS meter and the GPS module. The label takes Amp-hour readings from the TBS meter and multiplies this with the nominal voltage of the car to get an estimate for the kWh used in the trip. The distance is obtained from the GPS module. These two pieces of data are then simply divided to get a reading for the current efficiency of the trip, which is updated every second.

This idea was taken from observation of various pre-existing EV vehicle instrumentation and actively promotes drivers to try and improve upon their efficiency.

Fig. 12. The efficiency bar, shown in red, and the economy label, shown in blue, on the home window of the new GUI.

E. *Media Player*

The pre-existing GUI did not have a working media player. This was due mainly to broken libraries in QT and was effectively disabled in the GUI.

The new media player allows users to do many of the things that commercial media players can do. The old media play format contained simple buttons for the play/pause, skip and stop buttons, which were not visually aesthetic. This has been replaced by icons in the new window.

Due to the difference in method to program the media player, discussed later, the format of the playlist table and the display of the current song had to be changed. The window now contains a table for the playlist, which only includes the paths of the songs that are to be played with the media player.

The current song is now displayed through the two text boxes rather than highlighted in the list due to a different method to run the player. The buttons on the side allow users to add and remove songs as desired to the current playlist, which is all done using file dialogs to enable easy additions without a text editor.

Again with the theme, all buttons are larger and fonts are bigger and bolder to enable easy viewing by the driver when in use.

Fig. 13. The music window of the GUI. Layout of the window has changed dramatically due to the change in implementation for the player.

F. *Updated Trip Window*

The trip tab also has had a new update. The biggest update to the tab is the inclusion of the current/speed graph. In addition to the graph, there have been a variety of changes to the data displayed on the screen.

Some of these include the moving of the some of the data points like average speed and current to the bottom left of the display as well as the inclusion of energy remaining and total energy use. The fonts used have again become larger to be easier on the eye when in use.

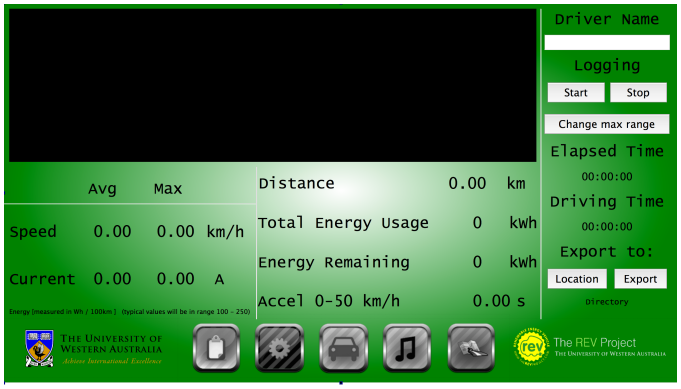

Fig. 14. The trip window of the GUI. Newly added features inclue the aforementioned current/speed graph, Total energy usage and Energy Remaining labels.

G. Updated Maps Window

There has not been much change with the maps tab. The main changes include removal of some of the unused function in the map tab, allowing the existing buttons to be enlarged and hence become easier to use.

Additionally the hide buttons button does not fully disappear upon pressing like the previous implementation of the GUI. The button now changes to 'show control' upon pressing, hiding all the buttons except the 'show control' button, allowing users to once again reopen the control panel.

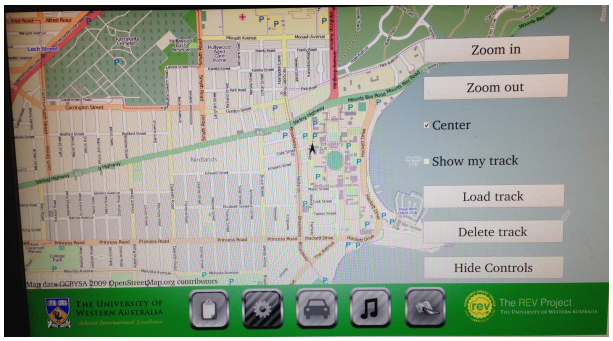

Fig 15. The maps window of the GUI. Larger fonts and buttons are employed for easier usage.

VI. Implementation

Though the Raspberry Pi has been implemented in 2013, not all the hardware was complete for a fully functional GUI. Additionally there were a lot of software problems that keep the user interface from being as flowing, as it should be. There was a fair amount of functionality that was not working in the GUI like the media player, which caused the GUI to be unstable during use.

This meant that a lot of the pre-existing code needed to be altered and changed before any improvements could be added to the GUI.

A. Upgrade from QT4 to QT5

Before beginning on the project, it was found that the software used in both cars was outdated and needed an upgrade. The old system used QT4 as the base program that ran the GUI. Though this worked and ran on the system, QT5 is a much better candidate due to greater functionality, more efficient libraries, easier design and more visually aesthetic GUIs.

QT4 was released seven years ago and hence is quite old in terms of software. QT5 offer quite benefits such as better graphic capabilities and performances, higher developer productivity and flexibility, simpler cross-platform portability and open development. [20]

This however, required a full reformat of the old system, which caused considerable problems due to the lack of documentation for QT5 on Raspberry Pi. Additionally, many of the methods had to be rewritten and background programs had to be individually reinstalled on the image.

However, once this was done, the improvements in the GUI were evident.

B. QCustomPlot

QCustomPlot is a third party library that enables plotting and visualization for QT. It is a widget that has no further dependencies and creates 'good-looking, publication quality 2D plots, graphs and charts'. [21]

QCustomPlot was the base library used to create both graphs used in the GUI. The graphs incorporated bright colours, easy to read labels and shapes in order for users to quickly obtain drive information.

Code used to create the graphs can be viewed in Appendix II.

C. Media Player

The media player in the two REV cars uses an external program mplayer [22]. Mplayer is a command line media player that runs on linux operating and provides efficient streaming of media without using large amounts of RAM.

Mplayer is run in a background thread using QProcess [23] to start and give commands on the main program. Users are able to play/pause, stop, and navigate through their user created playlist. This is done through the slave argument passed to the program on initialization. This enables users to control the player from an external program. However, as such, this means that there are limitations to the functionality available to control the media player from within the HMI.

Additionally, the Meta data from the media in the playlist cannot be accessed prior to the playing of the song. This means that the HMI is unable to obtain Song Name, Artist Name and Album data and this is reflected in the playlist.

This playlist is saved to the desktop and loaded automatically on startup of the GUI and can be played simply by pressing start and then play on the Music tab of the GUI.

Playlists can be modified by users adding or removing songs to their wishes using the simple buttons on the side of the playlist.

D. In Out (I/O) Module

A major obstacle that had to be overcome for fully functionality of the system was the I/O Module. The I/O module was required to allow the Raspberry Pi to sense the outside world and display the data on screen.

The I/O Module is intricately important to the running of the GUI for a variety of reasons. The GUI needed to be able to sense various inputs from the car for the warning icons and the engine sound replication system, as well as provide outputs, a PWM in particular, to generate a signal for the REV Racer tachometer. Additionally, the board needed to be cost efficient as well as compact to be suitable for an automotive use.

1) PiFace Digital

PiFace is a simple, compact and cheap I/O expansion board for the Raspberry Pi. It provides a quick and easy way to connect the Raspberry Pi to the real world. [24]

PiFace connects directly onto the Raspberry Pi GPIO socket, has 2 changeover relays, 4 tactile switches, 8 digital inputs, 8 open-collector outputs and 8 LED indicators. PiFace Digital provides exactly what is needed to connect the Raspberry Pi to the two REV vehicles.

To use the PiFace Digital, commands are made thorough a background thread using QProcess. This calls a secondary Python script, which returns a message to the command line only when the PiFace senses a change in the state of the input. This message is then picked up by the HMI and the Boolean state of the icons is triggered resulting in displaying the icon on the GUI.

This board also allows for reading in a PWM signal, like the one obtained from the Hall effect sensor in the REV Eco. Hence, it is possible to implement an accurate gear changing aid for the REV vehicles.

Finally, the two relays provide an easy method to output a PWM signal to drive the tachometer in the REV Racer.

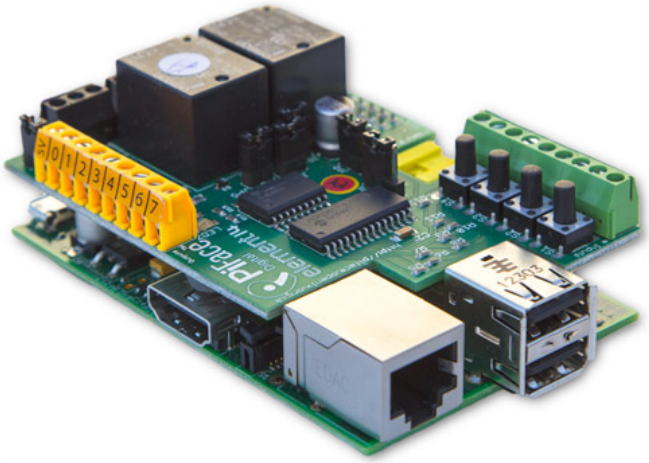

Fig. 16. The PiFace Digital. Used as part of the I/O module used in the two REV cars. The board enables easy connection to sensor available in the vehicles as well as two relays that are capable of creating PWM signals.

2) *Voltage Regulator Circuit*

The main problem with using the PiFace Digital is in that board can only support up to 5V inputs. This is a problem for the cars due to the output signals are 12V and will burn out the Raspberry Pi or the PiFace Digital without dropping the voltage down to 5V.

Hence a voltage regulator circuit is used to maintain a steady 5V input to the PiFace Digital. The Voltage regulator circuit consists of a linear regulator [25] with 2 decoupling capacitors and a bilateral switch [26].

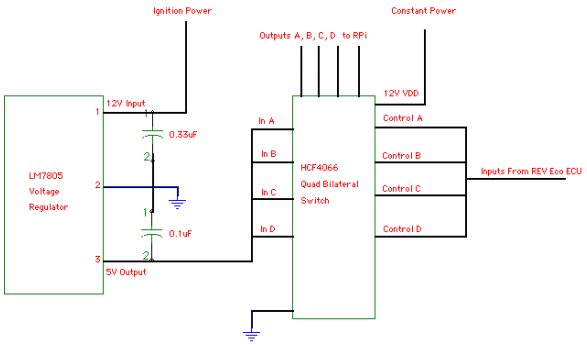

Fig. 17. The Volatge regulator circuit used to drop the voltage to 5V for the PiFace Digital. Care is taken to ensure low passive power consumption.

E. *Install of Hall Effect Sensor*

One easy way to increasing the ability to reduce energy use is to change gears. This however was not such an easy task for the two REV cars initially because the tachometers in both cars were not working.

The REV Eco employed a Hall effect sensor in the motor to enable readings for motor speed. This was then hooked up to the dash after analysis of the wave generated by the sensor.

On direct connection, the readings for the tachometer are double of the actual values expected and hence a simple D-Flip Flop [27] is used to get the tachometer to create a simple frequency divider circuit. [28].

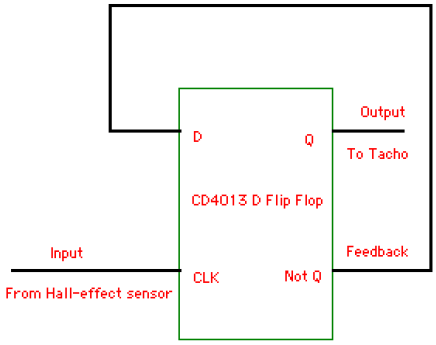

Fig. 18. The Frequency divider circuit used to halve the frequency of the signal to the REV Eco tachometer to match the correct motor speed.

F. *Modification for REV Racer*

The main work on the GUI focused predominantly on the REV Eco. As such, the GUI needed some slight modifications to be suitable for the REV Racer.

Although attempts to make the GUI uniform over both REV cars, some changes needed to be made. Main changes include the addition of the battery tab and small modifications the logging tab. The battery tab is able to function due to the BMS module created by Ivan Neubronner. This allows the HMI to read the state of individual batteries and hence is able to be displayed on the GUI.

Fig. 19. The Home window of the GUI used in the REV Racer. Note the coloured buttons, due to touchscreen functionality, and the extra battery button.

Fig. 20. The battery window used in the REV Racer, heavily using the BMS system created by Ivan Neubronner to display individual battery status.

VII. Future Work

A. *Tachometer in REV Racer*

One of the main goals for this project was to re-obtain functionality of the tachometer in the REV Racer. This unfortunately could not be done in time due to the difficulties in connecting the CAN Bus signals from the UQM motor to the Raspberry Pi. The major problem was due to the difference in operating system. The software for the UQM motor that allows readings for various sensors, including engine speed, are only available for Windows based machines. [29]

The Raspberry Pi can only support a linux-based operating system and as such, it is difficult to obtain the required data from the UQM Powerphase and read it on the Raspberry Pi.

However, with more time, the raw data signals from the UQM can be deciphered and possibly be translated into signals that the Raspberry Pi can read and then send out the PWM signal, through the PiFace Digital, into the tachometer for the REV Racer.

B. *E.A.R.S*

As the IO module for the cars was not completed early enough, E.A.R.S could not be implemented. On the addition of the IO module, the ability to read in the tachometer signal and implement a working version of E.A.R.S is possible.

C. *Safe On/Off for Raspberry Pi*

Currently, the Raspberry Pi turns on and off via power from the ignition. This means that whenever the car is turned off, the program abruptly turns off and with repeated use, this could corrupt the data on the SD card image.

With the addition of the IO module, the Raspberry Pi can sense when the car is turned off and as such, it is possible to create a function to safely turn off the Raspberry Pi and turn the Raspberry Pi back on.

However, currently there is no remote method to turn the Raspberry Pi back on without manually unplugging the device.

D. *Cleaner Implementation of Music Player*

Though the there is a working implementation in the GUI, this was done using a third party program over command line using QProcess. The latest version of a media in the two REV cars worked by using a QT4 library called Phonon. However, since the update to QT5, the phonon library is no longer supported and has been updated to the multimedia library. This enables easy and smoother implementations of media players in a GUI environment.

Initial plans were to use QMediaPlayer and QMediaPlaylist to create a media player in the GUI. [30]

However, there are problems with regards to linking between GStreamer [31], the framework used in linux for handling media, and QT5 when running on the Raspberry Pi. There isn't currently a solution, but hopefully a solution may present itself in the near future.

This will allow for a much cleaner and functional media player.

VIII. Conclusion

Over the year, work has been done on both vehicles, predominantly the REV Eco to improve upon the pre-existing instrumentation created by previous students. Improvements have been made to both software and hardware used in the two REV cars.

Major updates have focused on aids for drivers, to employ more efficient driving techniques. Updates include, graphs to display short-term and long-term efficiencies and meters to gauge the efficiency of the current drive.

Hardware improvements include the creation of the I/O module to be used in the cars and connection of the hall-effect sensor to the REV Eco cluster, enabling a working tachometer. The I/O module also provides a platform for future students to add extra sensors to the system increasing the functionality of the system.

Major software updates have been done, including updates of the operating system and a major update on the QT version.

Unfortunately, due to system incompatibilities and delays in fabrication of the I/O module, the tachometer in the REV Racer and E.A.R.S could not be completed.

Most objectives, however, have been met and overall the system has been well received by users. More work still needs to be done to create an ideal GUI for users of the two REV cars.

APPENDICES

APPENDIX I

Fig. A. GUI for REV Eco by Daksh Varma (2009) [32]

Fig. B. GUI for REV Eco by Beau Trepp (2011) [11]

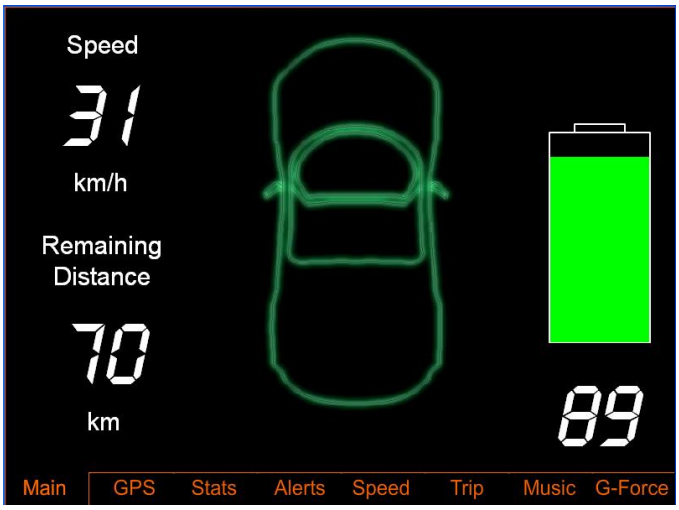

Fig. C. GUI for REV Racer by Daksh Varma (2009) [32]

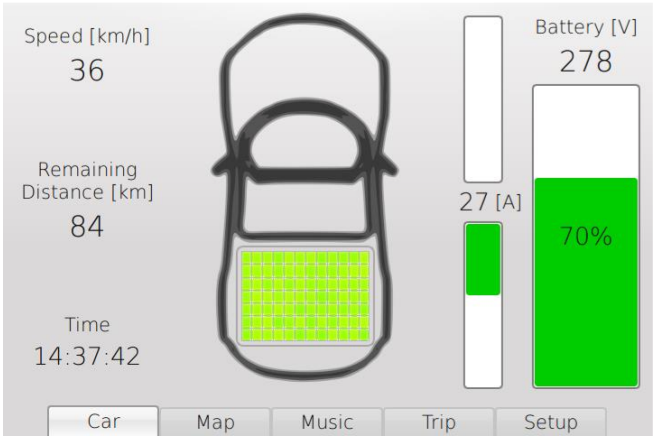

Fig. D. GUI for REV Racer by Thomas Walter (2010) [14]

Fig. E. GUI for REV Racer by Matthew Tyler (2011) [33]

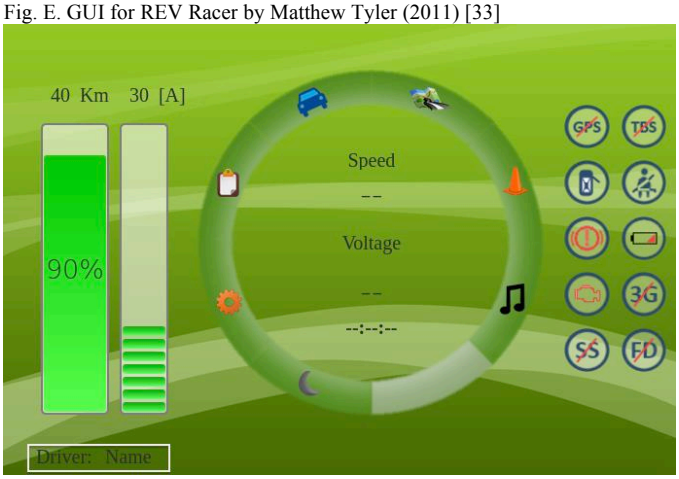

Fig. F. GUI for both REV cars by Gabriel Feng (2013) [15]

APPENDIX II

```
// === GRAPHS ============

/**
 * @brief REV_HMI::graphEfficiency Graphs the efficiency of previous
trips
 * @param customPlot
 */
void REV_HMI::graphEfficiency(QCustomPlot *customPlot) {

    customPlot->legend->setVisible(true);
    customPlot->axisRect()->insetLayout()->setInsetAlignment(0,
Qt::AlignTop|Qt::AlignLeft);

    customPlot->addGraph(customPlot->xAxis, customPlot->yAxis);
    customPlot->graph(0)->setPen(QPen(Qt::blue)); // line color blue
for first graph
    customPlot->graph(0)->setName("Distance");

    customPlot->addGraph(customPlot->xAxis, customPlot->yAxis2);
    customPlot->graph(1)->setPen(QPen(Qt::red)); // line color red
for second graph
    customPlot->graph(1)->setName("Energy Use");

    customPlot->plotLayout()->insertRow(0);
    customPlot->plotLayout()->addElement(0, 0, new
QCPPlotTitle(customPlot, "Distance/Energy Use"));

    customPlot->graph(0)-
>setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssCircle, 5));
    customPlot->graph(1)-
>setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssSquare, 5));

    //configure axes
    customPlot->xAxis->setVisible(true);
    customPlot->xAxis->setTicks(false);
    customPlot->xAxis ->setLabel("Trip");
    customPlot->yAxis->setVisible(true);
    customPlot->yAxis->setLabel("Distance (km)");
    customPlot->yAxis->setTickLabels(true);
    customPlot->yAxis2->setVisible(true);
    customPlot->yAxis2->setLabel("Energy Use (kWh)");
    customPlot->yAxis2->setTickLabels(true);

    QStringList data = readEfficiency();

    int i=0;

    if(!data.isEmpty()) {
    while(i<data.size()) {
        QString temp = data.at(i);

        QStringList splitted = temp.split(",");
        ui->efficiencyPlot->graph(0)->addData(i,
splitted.at(1).toFloat());
        ui->efficiencyPlot->graph(1)->addData(i,
splitted.at(0).toFloat());
        i++;
    }
    }

}

/**
 * @brief REV_HMI::graphIV Sets up the current/speed graph
 * @param customPlot
 */
void REV_HMI::graphIV(QCustomPlot *customPlot) {

    customPlot->legend->setVisible(true);
    customPlot->axisRect()->insetLayout()->setInsetAlignment(0,
Qt::AlignTop|Qt::AlignLeft);

    customPlot->addGraph(customPlot->xAxis, customPlot->yAxis);
    customPlot->graph(0)->setPen(QPen(Qt::blue)); // line color blue
for first graph
    customPlot->graph(0)->setName("Current");

    customPlot->addGraph(customPlot->xAxis, customPlot->yAxis2);
    customPlot->graph(1)->setPen(QPen(Qt::red)); // line color red
for second graph
    customPlot->graph(1)->setName("Speed");

    customPlot->plotLayout()->insertRow(0);
    customPlot->plotLayout()->addElement(0, 0, new
QCPPlotTitle(customPlot, "Current/Speed"));

    //configure axes
    customPlot->xAxis->setVisible(true);
    customPlot->xAxis->setTicks(false);
    customPlot->xAxis ->setLabel("Time");
    customPlot->yAxis->setVisible(true);
    customPlot->yAxis->setLabel("Current (A)");
    customPlot->yAxis->setTickLabels(true);
    customPlot->yAxis2->setVisible(true);
    customPlot->yAxis2->setLabel("Speed (km/h)");
    customPlot->yAxis2->setTickLabels(true);


    // make left and bottom axes always transfer their ranges to
right and top axes:
    connect(customPlot->xAxis, SIGNAL(rangeChanged(QCPRange)),
customPlot->xAxis, SLOT(setRange(QCPRange)));
    connect(customPlot->yAxis, SIGNAL(rangeChanged(QCPRange)),
customPlot->yAxis, SLOT(setRange(QCPRange)));
    connect(customPlot->yAxis2, SIGNAL(rangeChanged(QCPRange)),
customPlot->yAxis2, SLOT(setRange(QCPRange)));

    connect(&bms, SIGNAL(updated(BatteryData)), this,
SLOT(graphIVSlot()));
}

/**
 * @brief REV_HMI::graphIVSlot Uses data to graph current/speed graph
 */
void REV_HMI::graphIVSlot() {

    double key =
QDateTime::currentDateTime().toMSecsSinceEpoch()/1000.0;

    if(!currentList.isEmpty()) {
        ui->IVPlot->graph(0)->addData(key, currentList.last());
    }

    if(!speedList.isEmpty()) {
            ui->IVPlot->graph(1)->addData(key,speedList.last());
    }

    // let the ranges scale themselves so graph 0 fits perfectly in
the visible area:
    ui->IVPlot->graph(0)->rescaleAxes();
    ui->IVPlot->graph(1)->rescaleValueAxis();
    ui->IVPlot->replot();
}
```

Code used to create the graphs on the GUI

ACKNOWLEDGMENT

Firstly, I would like to thank Prof. Thomas Bräunl for the continual guidance and suggestions as to the look and functionality of the GUI, as well as allowing me to undertake the project.

Additionally, I would like to thank the Hexacopter team, Thomas Smith, Garrick Paskos and Gabriel Feng for giving assistance when needed throughout the project.

Thanks are also extended to Ivan Neubronner and Stuart Speidel for their work on the cars while proceeding with the project.

Finally, I would like to thank my friends and family who have supported me throughout the year.

REFERENCES

[1] Bellis, M. "History of Electric Vehicles" [Online]. Available: http://inventors.about.com/od/estartinventions/a/History-Of-Electric-Vehicles.htm

[2] Environmental Leader. (June 2012). "Tesla Model S Verified as Longest-Range Electric Vehicle, at 265 Miles" [Online]. Available: http://www.environmentalleader.com/2012/06/22/tesla-model-s-verified-as-longest-range-electric-vehicle-at-265-miles/.

[3] U. Sandberg (2012) "Adding noise to quiet electric and hybrid vehicles: An electric issue". Acoustics Australia.

[4] T. Braunl. (2014) "The REV Project", UWA [Online]. Available: http://therevproject.com

[5] Hyundai, "Hyundai Getz Shop Manual", 2001.

[6] Lotus Cars Ltd, "Service Notes Elise 2001 Model Year Onwards", DOT HS 811 092, 2001

[7] NHTSA, "Fuel Economy and Driver Interfaces: Design Range and Driver Opinion", Springfield, Virginia, August 2009.

[8] Chevrolet, "2015 Volt: Electric Cars – Hybrid Cars" (2014) [Online]. Available: http://www.chevrolet.com/volt-electric-car.html

[9] J. Oakley & T. Braunl, "Performance Evaluation of Battery-Electric Vehicles under Varying Road Conditions and Settings", UWA, Crawley, WA. March 2013.

[10] R. K . Pandey, M. Pham, J. Whale, M. P. McHenry, T. Braunl, "Comparative road performance testing of the Chevrolet Volt PHEV", UWA & MU, WA. 2014.

[11] TBS Electronics (2014), "E-Xpert Pro" [Online]. Available: www.tbs-electronics.nl/products_batmons_acc.htm

[12] B. Trepp, "Design of an embedded data acquisition and display system using a modular approach", M.S. Thesis, School of Electronic Engineering & Comp. Sci., UWA, Crawley, WA, 2011.

[13] Xenarc Technologies. (2014), "Xenarc Technologies – 700TSV – 7" Touchscreen TFT LCD Monitor with VGA and AV inputs [Online]. Available: http://www.xenarc.com/product/700ts.html

[14] Qt Project (2014). "Qt Project" [Online]. Available: http://qt-project.org

[15] T. Walter. "Development of a User Interface for Electric Cars", M.S. Thesis, Dept. Engineering Cybernetics, Univ. of Stuttgart, Stuttgart, BW, 2010.

[16] G. Feng, "Renewable Energy Vehicles' User Interface: Implementation of a Raspberry Pi & Engine Audio Replication System", M.S. Thesis, School of Mech. & Chem. Engineering, UWA, Crawley, WA, 2013.

[17] Raspberry Pi Foundation (2014), "Raspberry Pi" [Online]. Available: http://www.raspberrypi.org

[18] Griffin (2014), "Powermate USB Controller" [Online]. Available: http://store.griffintechnology.com/powermate

[19] Gizmo Daemon (2009), "Gizmo Daemon" [Online]. Available: http://gizmod.sourceforge.net

[20] Qt Project (2014). "Introducing Qt 5" [Online].Available: http://qt-project.org/qt5

[21] QCustomPlot, "QCustomPlot" [Online]. Available: http://www.qcustomplot.com

[22] Mplayer (2014), "MPlayer" [Online]. Available: https://www.mplayerhq.hu/design7/dload.html

[23] Qt Project (2014), "QProcess Class" [Online]. Available: http://qt-project.org/doc/qt-5/qprocess.html

[24] PiFace (2013), "PiFace Digital" [Online]. Available: http://www.piface.org.uk/products/piface_digital/

[25] Fairchild Semiconductor, "LM28XX/LM78XX A 3 Terminal 1A Positive Voltage Regulator", (2006)

[26] SGS-Thomson Microelectronics, "HCC/HCF4066B Quad Bilateral Switch", (1989)

[27] Texas Instruments, "CMOS Dual 'D'-Type Flip-Flop", (2003)

[28] Electronics Tutorials (2014), "Frequency Division using Divide-by-2 Flip Flops", [Online]. Available: http://www.electronics-tutorials.ws/counter/count_1.html

[29] UQM Technologies, "CAM Communication Summary", January 2011.

[30] Qt Project (2014), "QMediaPlayer Class", [Online]. Available: http://qt-project.org/doc/qt-5/qmediaplayer.html

[31] GStreamer (2014), "GStreamer: open source multimedia framework" [Online]. Available: http://gstreamer.freedesktop.org

[32] D. Varma, "Renewable Energy Vehicle Instrumentation: Graphical User Interface and Black Box", M.S. Thesis, School of Elec. Engineering, UWA, Crawley, WA, 2009.

[33] M. Tyler, "REV Performance Vehicle Instrumentation", M.S. Thesis, School of Elec. Engineering, UWA, Crawley, WA, 2011.

[34] Qt Project (2014), "Beginner's guide to cross-complile Qt5 on Raspberry Pi"[Online]. Available: http://qt-project.org/wiki/RaspberryPi_Beginners_guide

[35] IBEX, "Auto Running Programs – Raspberry Pi Projects" [Online]. Available: http://www.raspberry-projects.com/pi/pi-operating-systems/raspbian/auto-running-programs

[36] Raspberry Pi Foundation, "Raspberry Pi Documentation, config.txt" [Online]. Available: http://www.raspberrypi.org/documentation/configuration/config-txt.md

[37] Hyundai-Forums, "Dashboard Getz 2005" [Online]. Available: http://www.hyundai-forums.com/getz-forum/141869-dashboard-getz-2005-a-2.html