AC 2012-3825: INTEGRATING ELECTRIC VEHICLES INTO SOFTWARE ENGINEERING PROJECT-BASED EDUCATION

Prof. James N. Long, Oregon Institute of Technology

James Long is a professor of computer systems engineering technology at Oregon Institute of Technology. His primary teaching and research interests are real-time embedded systems, control theory and implementation, computer networks, and operating systems. He has 12 years of teaching experience in higher education and industry, and 25 years of experience as a software engineer in flight test systems, telephony and high speed networking, Doppler RADAR data acquisition and control, and medical imaging systems. Long is actively involved in the Oregon Renewable Energy Center (OREC) in process, modeling, and controls as PI and Co-PI in three research grants awarded by Oregon Built Environment and Sustainable Technologies (Oregon BEST) and Oregon Transportation Research and Education Consortium (OTREC). Long is an advocate for project-based learning models, bringing students into real-world problem solving situations bridging higher education and industry involvement in a relationship beneficial to both.

Integrating Electric Vehicles into Software Engineering Project Based Education

Abstract

Vehicle Systems are becoming increasingly dependent on microcontrollers and integration of computer systems. The average car currently uses between 30 and 45 microcontroller based electronic control units (ECU)[1]. Projections show number of microcontrollers in an average automobile increasing to greater than 70 by 2020. As the number of microcontrollers increases, so does the complexity of the microcontroller, transitioning from 8 bit to 16 bit, and then from 16 bit to 32 bit. The multiple microprocessor supported subsystems also require increasingly complex communication infrastructure. Automotive computing is trending toward vehicle personalization, smart freeways, and higher efficiency standards[2]. This increase in raw computing power coupled with higher levels of software based logic abstraction is moving vehicle borne computer systems into the realm of software engineering. Software engineering in the automotive industry provides a strong platform for student exploration.

One key hurdle for integration of automobiles into a software engineering curriculum is that of access. Vehicles based on classic internal combustion (IC) engine power sources require special laboratory space, have harmful emissions to deal with and are hard to keep clean. In addition to space issues, it is difficult to build bench test systems if the power plant is an internal combustion engine. Electric vehicles (EVs), on the other hand, provide a unique opportunity as a platform for software engineering systems. EV mechanical systems can be easily scaled down to create small versions of vehicle operational systems. With the EV as a software systems deployment platform, several aspects of vehicle based software can be developed as software engineering projects, with the ultimate end result being a “drivable” software system.

This paper looks at the use of EV technology as a platform for a software engineering projects involving student teams. Different aspects of vehicle systems and application to software engineering projects will be discussed. The use of an EV platform in a three term project will be explained describing the different systems involved, how the project was supported, construction of bench test systems, and final deployment result. Student involvement and success in the project will be presented.

Program Overview

The Computer Systems Engineering Technology department at Oregon Institute of Technology offers a four year Bachelor of Science degree in Software Engineering Technology. Oregon Institute of Technology is accredited through the Northwest Association of Schools and Colleges (NWASC). The Software Engineering Technology degree is accredited through the Accrediting Board for Engineering and Technology (ABET).

The Software Engineering Technology program currently has approximately 100 students enrolled in the four years. The program typically draws students directly out of high school. Students have exposure to computer programming and a high level of interest in becoming "programmers". Students who gravitate toward the technical program offered at Oregon Institute of Technology over computer science degrees offered at larger state universities are interested in hands-on activities in a highly interactive environment. By the end of the second year, students are well versed in programming skills. The third year of the Software Engineering Technology program focuses on systems analysis, design, construction, deployment, testing, and quality assurance. The core of this activity is a three term course sequence encompassing team based construction of a real-world enterprise scale system. The projects are based on problems derived from on campus research projects or industry partners.

Junior Project Course Sequence Educational Objectives

The Junior Project course sequence is the first exposure for most software engineering technology students to large scale software engineering problems. Topics include software engineering process, scheduling, architecture, and teamwork. This is also the first time students work on a project spanning multiple quarters. In some instances, projects are a continuation from a previous year team effort. The junior project is a required sequence and must be completed in a single year. Once a project is started, the teams remain consistent throughout the year. The sequence is split up based on quarters with specific outcomes identified for each quarter.

First Quarter – Software Process Management

1. Describe and contrast different software engineering process models including the iterative process used in this class.
2. Understand the benefits and problems of teaming, describing qualities and processes of effective teams, and describing the role of teamwork in system design.
3. Describe and contrast potential project organizations and the team member roles within those organizations.
4. Create a Team Charter to articulate how the team will track, manage and communicate project progress, changes in scope, changes in design, and defects.
5. Assess risk, probability of the risk, triggers and formulate contingency plans.
6. Construct a statement of work with appropriate acceptance criteria.
7. Describe the relationship between Testing and Quality Assurance.
8. Describe the Quality assurance practices appropriate for each part of the development life cycle.
9. Create user based requirements and engineering requirements.
10. Describe traceability and be able to map a requirement through all project artifacts.
11. Describe different modeling techniques and where they apply.
12. Describe the different architectural views and assign them to parts of the life-cycle.
13. Asses risk and develop risk management plans.

Second Quarter – JP I

1. Use case modeling to refine requirements.
2. Create a use case specification including steps and scenarios.
3. Derive sequence models from use case scenarios.
4. Create initial class models from sequence diagrams, requirements and uses cases.
5. Describe the roles of team members including the Team Leader and Quality Assurance Manager.
6. Describe the role of functional analysis in use case analysis and requirements gathering and refinement.
7. Understand the role of the low fidelity User interface in the software development life-cycle.
8. Develop a project schedule including refining estimates and allocating resources.
9. Manage the project including collecting status, analyzing variances, planning and taking adaptive action and reporting status.

Third Quarter – JP II

1. Management of requirements and understand the importance of requirements in a large scale software project.
2. Use case analysis and understanding its place in the software development lifecycle.
3. Understanding of software development event sequences and how they fit in the software development lifecycle.
4. Application of class modeling and understand how it fits into the software development lifecycle.
5. Application of software system development task scheduling and development estimates.
6. Application of software development implementation (writing code) and understanding how it fits into the software development lifecycle.
7. Use of software system requirements and the mapping of those requirements into the software development lifecycle for effective impact analysis for change control.

Project Motivation

The Software Engineering Technology program at has been working with the Mechanical Engineering (ME) program for over eight years with cross discipline projects involving modifications to different type of vehicle systems. Projects have ranged from conversion of a Volkswagen Jetta into a gas-electric hybrid to the control system of a human powered-electric velomobile.

The vehicle based projects have proven effective at emphasizing aspects of software engineering revolving around team communication, project and timeline management, and heterogeneous system integration. Students involved in the cross discipline projects have a much higher level of appreciation of the work level required to obtain successful project outcome. In exit exams, 80% of students involved in the cross discipline, automotive projects, showed higher proficiency in technical writing, project and task management, and oral communication skills.

In 2011, two grants were received from Oregon Built Environment and Sustainable Technologies (Oregon BEST) and Oregon Transportation Research and Educational Consortium (OTREC) allowing for exploration of an applied research area related to the control and monitoring of a gas-electric hybrid vehicle. One grant related directly to the advance of commercialization of technology developed for creation of a small three wheeled gas-electric, plug in hybrid commuter vehicle. The second grant was related to the testing and commissioning of the developed gas-electric hybrid technology for further refinement. The receipt of these grants provided an opportunity to involve undergraduate and graduate students in a cross discipline project, allowing specification of tools and platforms for support of automotive software systems engineering.

Automotive Software Engineering Problem Areas

Historically, software written for vehicles has been based on proprietary tools and software engineering processes[3]. The trend in the automobile industry is of increasing integration of software with digital and mechanical systems. Areas of impact range from vehicle power plant control to that of integrated entertainment systems complete with Internet access. The sheer bulk of code integrated in a typical vehicle is pushing the engineering of software systems out of the proprietary realm and into software engineering mainstream where the automotive industry must adapt best practices for control of software system reliability and complexity. The amount of development activity directly related to software engineering has risen, in some cases, to 40% of the overall engineering effort required to produce a new vehicle[4]. These trends emphasize the growing relevance vehicle systems as project platform for software engineering students. For the purposes of the software engineering technology program at Oregon Institute of Technology three term sequence, relevant vehicle specific problem areas were targeted such that:

1. There is a large software specific component.
2. The system required can be expanded into an enterprise software system.
3. Cross discipline or cross team activity is a required element of the project.
4. Hardware and mechanical systems can be simulated through both software and hardware based systems.

The ultimate goal is the integration of the three term sequence educational outcomes with a carrying project targeted at the automotive industry. Since the primary audience was students in the Software Engineering Technology program, the projects also had to be structured such that:

1. Any hardware specific development could be isolated and minimized.
2. System abstractions are easy to fit into a UML based Use Case, Class Model, and Component Model structure.
3. System interface points are clearly delineated for construction of shared testing and integration simulation environments.

4. The project construction, initial installation and final release can be done in a nine month time scale, allowing for a full cycle of the software engineering lifecycle.

To fulfill the above requirements, the electric vehicle component of the hybrid vehicle system was targeted as the operational platform to build the software system. The electric vehicle control system has the features of:

1. Electric motor control system isolation is easily be done through use of a standard analog throttle and off the shelf motor controller (Figure 1).
2. Programming could be done using standard C, C++, and other high level languages.
3. The vehicle operational system could be emulated on a small scale (Figure 2) and scaled up to a fully operational vehicle (Figure 3 and Figure 4).
4. System partitioning could be done allowing multiple teams to work on subsystems where any subsystem could be simulated by the dependent teams. This allowed a team to fail without adversely impacting other teams working on the system.

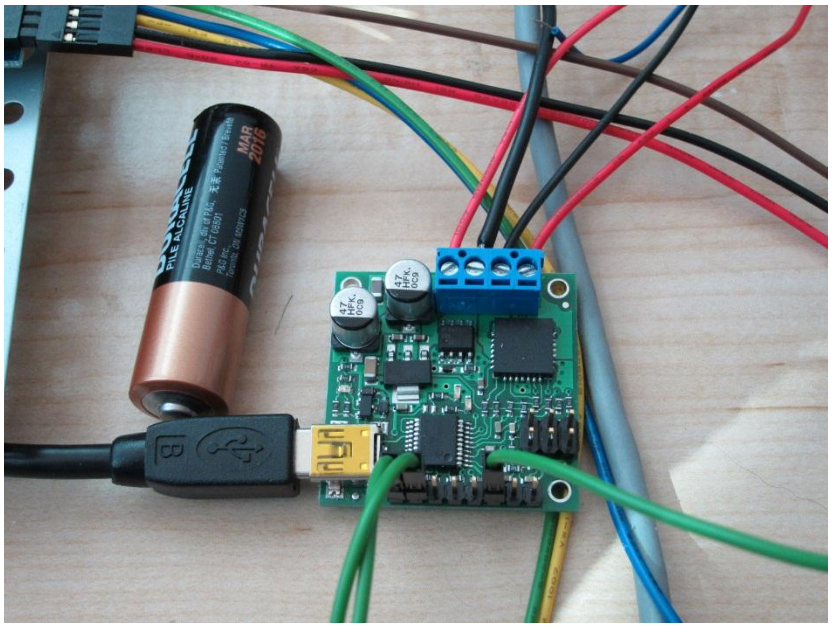

Figure 1 – The small scale electric motor controller for use in vehicle system emulation.

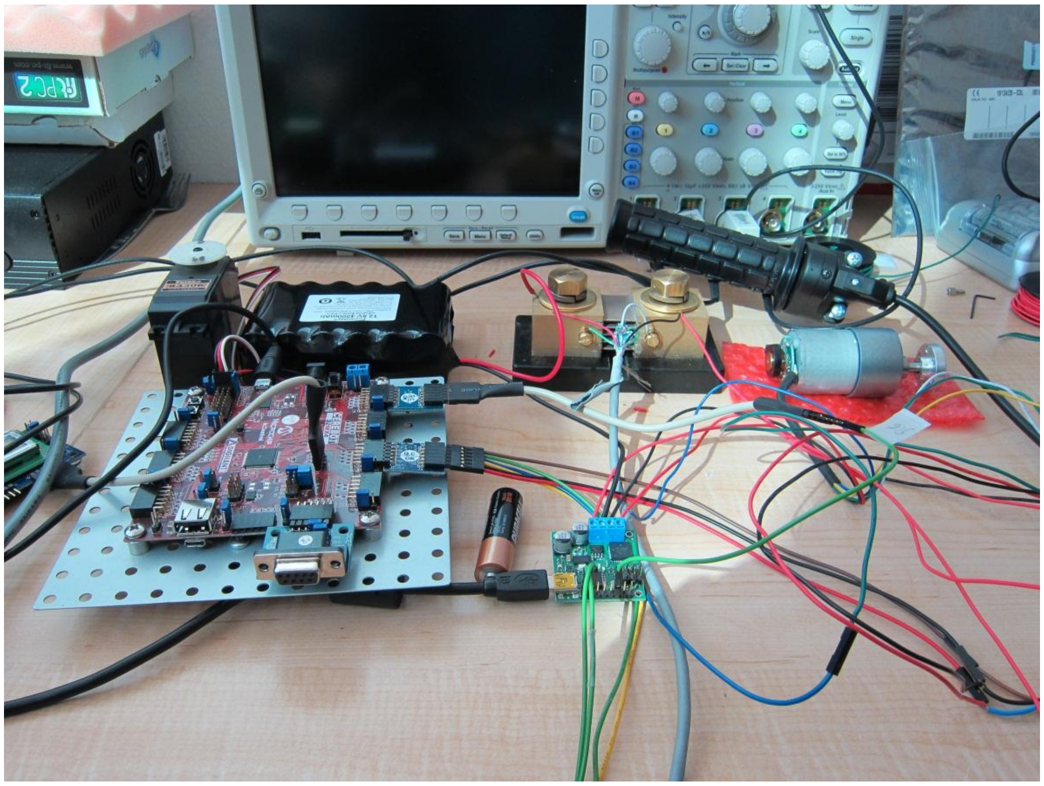

Figure 2 – The chaos of the software engineering student constructed vehicle emulation system.

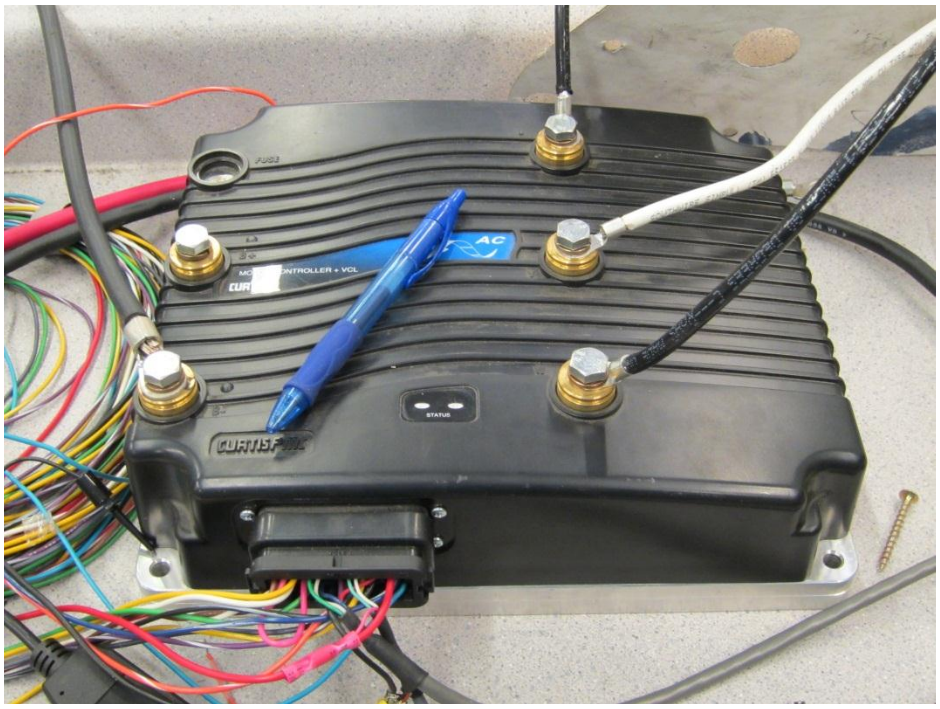

Figure 3 – The target electric motor controller.

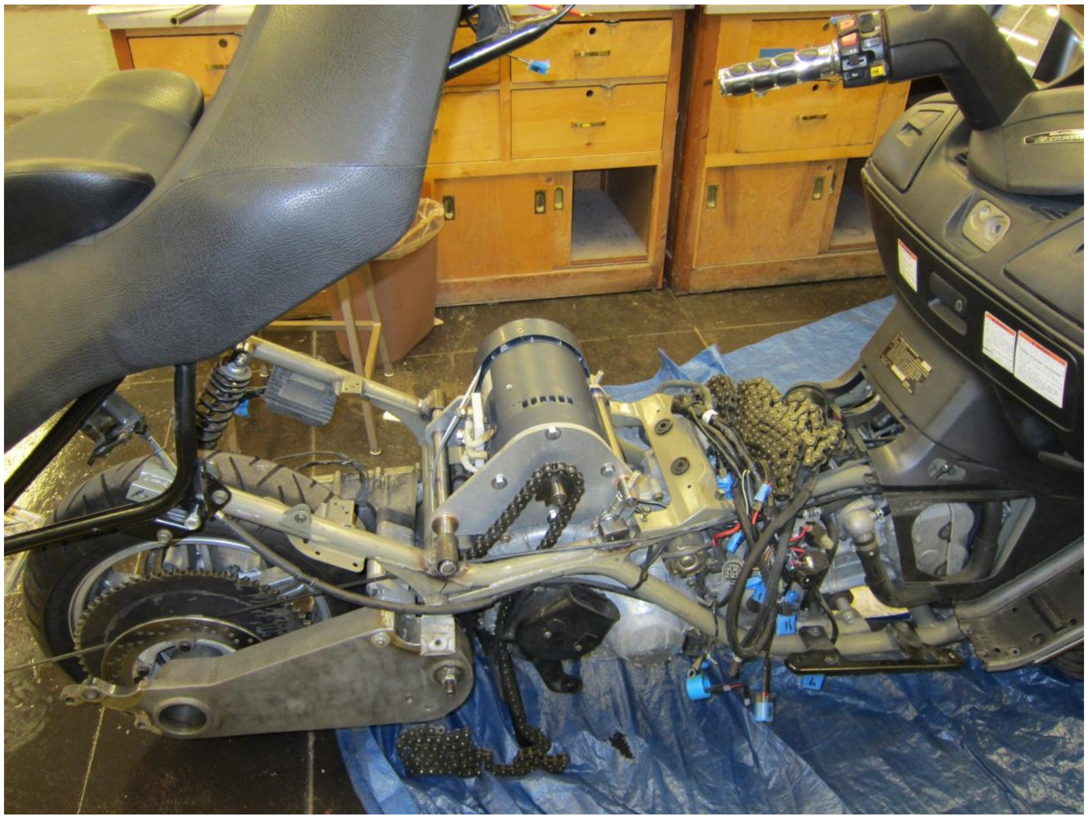

Figure 4 – The target platform vehicle.

Looking at a typical modern vehicle, the digital systems are broken down into three main areas, shown in Figure 5.

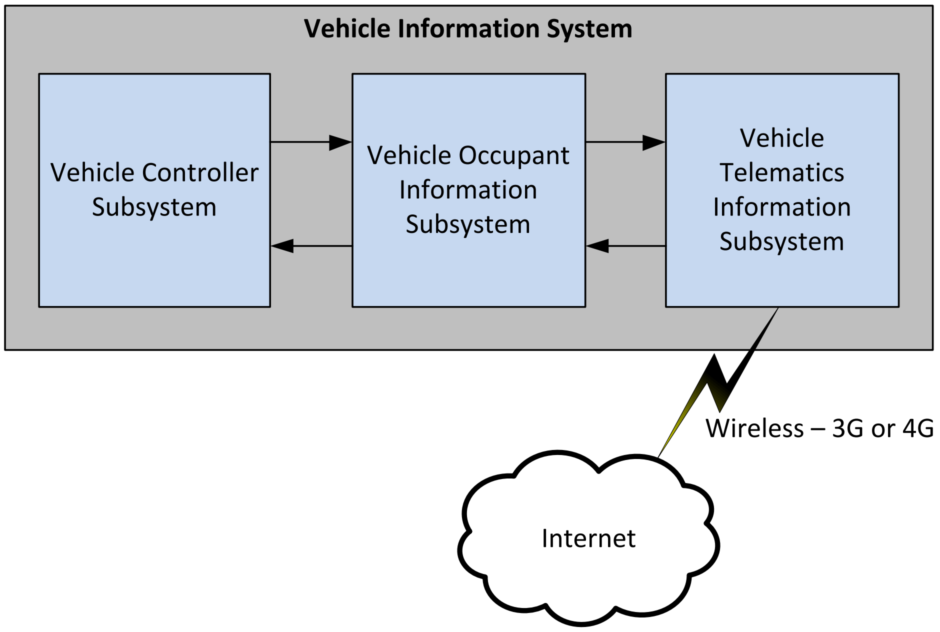
Vehicle Information System
Vehicle Controller Subsystem
Vehicle Occupant Information Subsystem
Vehicle Telematics Information Subsystem
Wireless – 3G or 4G
Internet

Figure 5 – Subsystem breakdown for vehicle based projects[5].

Vehicle Controller Subsystem:

In this problem area, projects related to control of the vehicle propulsion system and other vital electronics is approached. This subsystem lends itself to computer engineering and embedded systems engineering projects. Cross discipline projects fit nicely in here where hardware students design and build components to interface directly to vehicle electrical and mechanical systems. Embedded systems students integrate a microcontroller component for implementation of simple control strategies, data acquisition, and dissemination. This area involves standard Control Area Network (CAN) interfacing, RS232 interfacing, Analog to Digital, Digital to Analog, and Transport Control Protocol/Internet Protocol (TCP/IP) for data transmission. This subsystem should be implemented as a secure entity for vehicle operation protection.

Vehicle Occupant Information Subsystem:

This area encompasses projects related to human/computer interface allowing vehicle occupants to interact with vehicle digital operations. Software engineering projects are deployed in this area where standard off the shelf computer systems can be used to effect direct user control over the Vehicle Controller Subsystem as well as implementation of a data recording black-box. Effective deployment of software systems can be done on tablet PC platforms for use of touch technology. These projects can be done with TCP/IP as the standard communication protocol. This area involves a stronger Human Computer Interface (HCI) component as well as a focus on system security.

Vehicle Telematics Information Subsystem:

This area extends the physical boundaries of the vehicle into an Internet realm such that data acquired from the Vehicle Controller System through the Vehicle Occupant Information Subsystem can be streamed in real-time to an Internet based server. Projects in this area allow the integration of Web based technologies for data streaming, recording, analysis and historic playback. These projects are done based on layer 7 web protocols running on top of TCP/IP with the vehicle to Internet link implemented through a standard cellular phone service via a service provider data dongle.

These three areas of system development provide a wide range of possible software engineering projects applied directly to information systems targeted at vehicle borne deployment. The subsystem boundaries also allow a project team to decouple project outcome dependence on other team performance through definition and creation of system simulation and emulation modules. Because of the information dependences of systems, project teams must collaborate in definition of shared interface protocols for eventual integration of all systems in the test vehicle. The final system deployment depends on an operational hardware system capable of operating a test vehicle.

To allow for testing of systems in a closely controlled laboratory environment, vehicles based solely on internal combustion power sources are not practical due to emissions and required mechanical infrastructure. These limitations were overcome through adaptation of electric vehicle (EV) technologies. EV platforms provide real-world components that are clean, easy for software students to construct, and easy to scale from "toy" size platforms to full size vehicle deployments.

Electric Vehicle Integration

Electric vehicle technologies provide a clean platform for use in development of software based vehicle projects. With minimal cost, a simulation and/or emulation environment can be constructed allowing full development and deployment of an operational system at a small scale. The software based logic can then be adapted to scaled up hardware for operation of full size vehicles ranging from electric bicycles to full size electric cars. As the system scales from the micro to macro level, a greater amount of involvement is required from the mechanical engineering student teams. Figure 6 shows a block diagram of the different hardware components for use in an electric vehicle project.

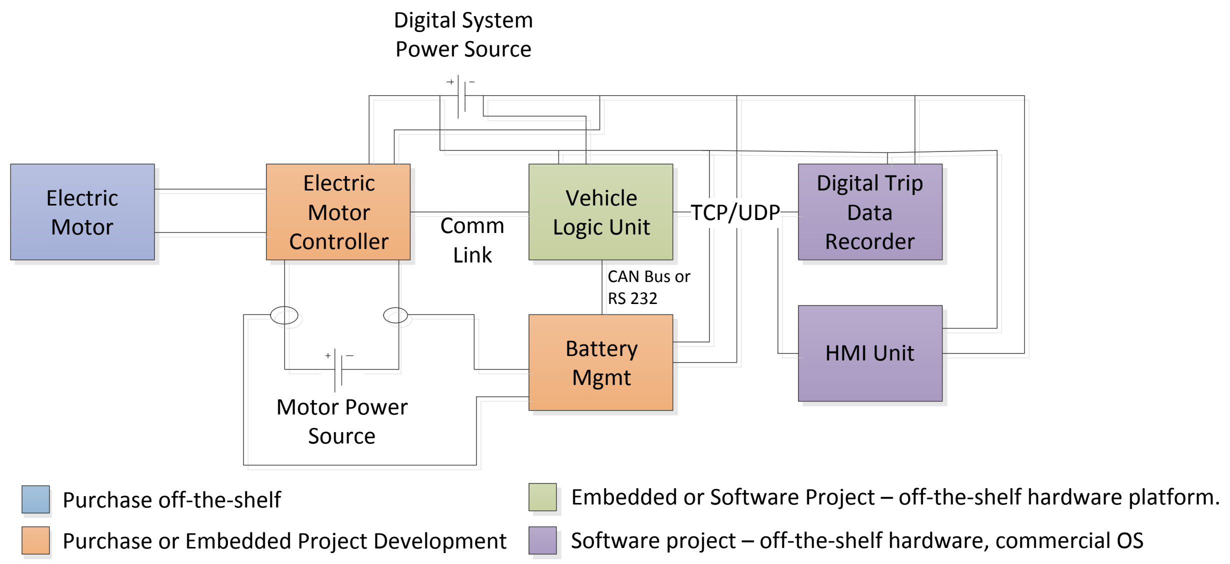
Digital System Power Source
Electric Motor
Electric Motor Controller
Comm Link
Vehicle Logic Unit
TCP/UDP
Digital Trip Data Recorder
CAN Bus or RS 232
Battery Mgmt
HMI Unit
Motor Power Source
Purchase off-the-shelf
Purchase or Embedded Project Development
Embedded or Software Project – off-the-shelf hardware platform.
Software project – off-the-shelf hardware, commercial OS

Figure 6 – Electric Vehicle Project Hardware Components

Electric Motor: The electric motor can easily be purchases as a hobby motor allowing the project to start out with a small motor where current draw and power will not be of concern. From a software perspective, there is no difference between interfacing with a small or large electric motor, yet the motor and motor controller pair can be scaled to support a vehicle capable of carrying multiple passengers.

Electric Motor Controller: The electric motor controller can be purchased "off-the-shelf", typically configured via a host computer through a USB connection. It could alternatively be built as an electrical engineering/embedded systems project. Purchasing a controller capable of operating from a 0-5v analog signal allows the system to be tested using a potentiometer and later scaled up through the deployment of a larger controller capable of control through an analog signal. An example of such a larger is the line of Curtis line of controllers, commonly used in electric vehicle applications.

Battery Management: The battery management system has several dimensions. Depending on battery type, it can be done very simply through an analog input and monitoring of battery voltage level (this works fine for simple lead-acid battery configurations), or as a more complicated scenario, LiFePO4 battery banks must be monitored using specialized IC configurations. This component can be purchased for full size vehicles off-the-shelf at a price ranging anywhere from $200.00 to $6000.00. Electrical engineering and embedded systems students can build this component as a project for very little in parts cost.

Vehicle Logic Unit: The vehicle logic unit is responsible for acquiring data from any attached sensors, processing the acquired data, and then sending control signals to the electric motor controller as well as streaming acquired data out on a Transport Control Protocol (TCP) or User Datagram Protocol (UDP) connection. For the purpose of a student team project, input into the vehicle control unit may contain input throttle signal from the vehicle pilot, state of battery charge, battery bank temperature, motor temperature, motor rpm, vehicle acceleration, and throttle aggressive index. From this input, the software system can produce an appropriate throttle signal to the electric motor controller and estimate how much distance is remaining until the batteries hit critical discharge. Adding a Geographic Positioning System (GPS) unit to the vehicle logic unit allows the gathered data to be correlated in time and location for logging and use in vehicle telematics. Depending on the level of I/O and data processing, this component can be built by embedded systems students, cross discipline embedded systems and software engineering, or just software engineering students.

Digital Trip Data Recorder: This component is the "black box" for the vehicle employing a small database for the purpose of time stamping and archiving data as the vehicle operates. The digital trip data recorder is also the unit that will communicate with and Internet based server allowing remote control and monitoring of vehicle behavior. A GPS can also be added in this component for the purpose of data stream to location correlation. A small PC form factor can be used for this unit with all programming being done in a commercial operating system environment. The combination of the digital trip data recorder and the web site to be used for data reduction and analysis makes for a good software engineering project.

Human Machine Interface (HMI) Unit: The HMI component provides an opportunity for the integration of tablet and smart phone technologies into the operations and control of vehicle. This area also allows for integration of social media platforms and mapping Application Programmer Interfaces (APIs) into the overall vehicle system. The simplest incarnation of the HMI project is for providing a simple dashboard for the vehicle along with simple control scenarios such as remote start and horn honk for vehicle location. A more complicated project involves the mapping of charging stations and looking at vehicle to grid interfacing for utilization of electric vehicles as temporary power storage to soften power generation/demand spikes on the utility grid.

Any one of these components can be developed as a stand-alone project. This allows project teams to control tightly the domain of their project functions. The necessary integrated nature of the overall system for complete deployment as an electric vehicle forces project teams to formally define interface boundaries for use as current collaborations or as starting points for future projects.

Electric Vehicle Project Definitions for the 2011-2012 Academic Year

For the 2011-2012 academic year, three projects were approached. These projects contained components from each of the three subsystems, Vehicle Controller Subsystem, Vehicle Occupant Subsystem, and Vehicle Telematics Information Subsystem. The projects were defined based on a gas-electric hybrid vehicle as defined in a commercialization grant from Oregon BEST[6] and a follow on grant from OTREC[7]. The portion of the gas-electric hybrid project scaled from hobby level to fully operational vehicle was the electric drive system. Each different project will be discussed detailing hardware and software platforms adapted, interface elements required for definition, bench testing and system scaling. Preliminary results of each project will also be presented.

Vehicle Controller Subsystem

The Vehicle Controller Subsystem is tasked with taking in a single throttle input, parceling out power request between the internal combustion engine and the electric motor. This project involves three professors and four students. The professors are from the mechanical engineering, and embedded systems engineering, and computer systems engineering departments, each professor responsible for a part of the system most closely related to their discipline. The student team is made up of a mechanical engineering graduate student, an embedded systems engineering student, and two software engineering students. The mechanical engineering graduate student is in charge of the design and fabrication of the hybrid drive system and mechanical interface. The embedded systems engineering student is responsible for the low level software/hardware interface and the general structure of the system continuous executive. The second computer systems student is responsible for hardware structure and packaging. The third computer systems student is responsible for communication protocol and a simple user interface.

This subsystem required the highest level of engineering work outside the direct discipline of software engineering thus funds from the supplied grants were used to pay for student time outside the standard OIT school year schedule. To make the vehicle controller subsystem more accessible to software engineering students, the embedded systems engineering technology and computer systems technology students created the hardware design and wrote a hardware abstraction layer giving software engineering technology students access to low level code through standard C programming and the Microchip MPLab Integrated Development Environment (IDE).

The Vehicle Controller Subsystem (shown in Figure 7), has been constructed with off-the-shelf components allowing a focus on software systems as a primary problem to solve. As a first phase, the hardware used for small scale deployment and testing of the vehicle control scheme is:

- Primary Control Platform: Digilent Cerebot 32MX4[8] - This single board computer provides a rich set of I/O as well as servo ports. The contained CPU is a Microchip PIC32, programmable with Microchip C. MPLab, the IDE that is delivered with the board allows full symbolic debugging out of the box. Cost: $79.00
- Analog to Digital and Digital to Analog devices: Digilent Pmod units. 1 A/D and 1 D/A. Cost: $64.98
- Network interface controller: Digilent Pmod unit PmodWiFi. Cost: $59.99
- Small Scale Electric Motor Controller: Pololu Jrk 21v3 USB Motor Controller with Feedback (Fully Assembled)[9] – This low cost hobby electric motor controller is used in early prototyping of the control system allowing full implementation of the control logic to be tested without necessity of the larger vehicle mounted controller. Cost: $51.95
- Small Scale Electric Motor: 19:1 Metal Gearmotor 37Dx52L mm with 64 CPR Encoder[10] - The use of this small scale motor allowed students to get an idea of how the control algorithm implemented on the PIC32 was being realized in a physical deployment. Cost: $39.95
- Battery State of Charge (SOC) Measurement unit: Manzanita Micro SOC Head – This unit is used to give battery charge as a "fuel gauge" measurement. This was used on the small scale platform since it was targeted at final vehicle deployment. (Not required for small scale simulation)
- Battery Bank: LiFePO4 12.8V Pack – This small battery pack is a scaled down version of the 5.65 KWh battery pack used on the vehicle. Cost: $65.00
- Servo: Futaba 1438 High Torque Servo – This servo is capable of pulling the internal combustion engine throttle, allowing direct manipulation of the internal combustion engine power output. The servo is integrated into simulation through manipulation of a rheostat connected to a unit that manipulated the gas tank fuel level signal passed back into the Cerebot. Cost: $50.00

Overall Vehicle Controller Cost: $410.87

The use of the above system components allowed software engineering students construct a working controller with very little direct manipulation of hardware. This hardware structure could easily be scaled up to a larger vehicle through changing of the hobby sized electric motor controller with a full scale vehicle sized electric motor controller. The control hardware, control signal, and control algorithm remain the same. This was done with an electric bike, then scaled to the working hybrid electric vehicle.

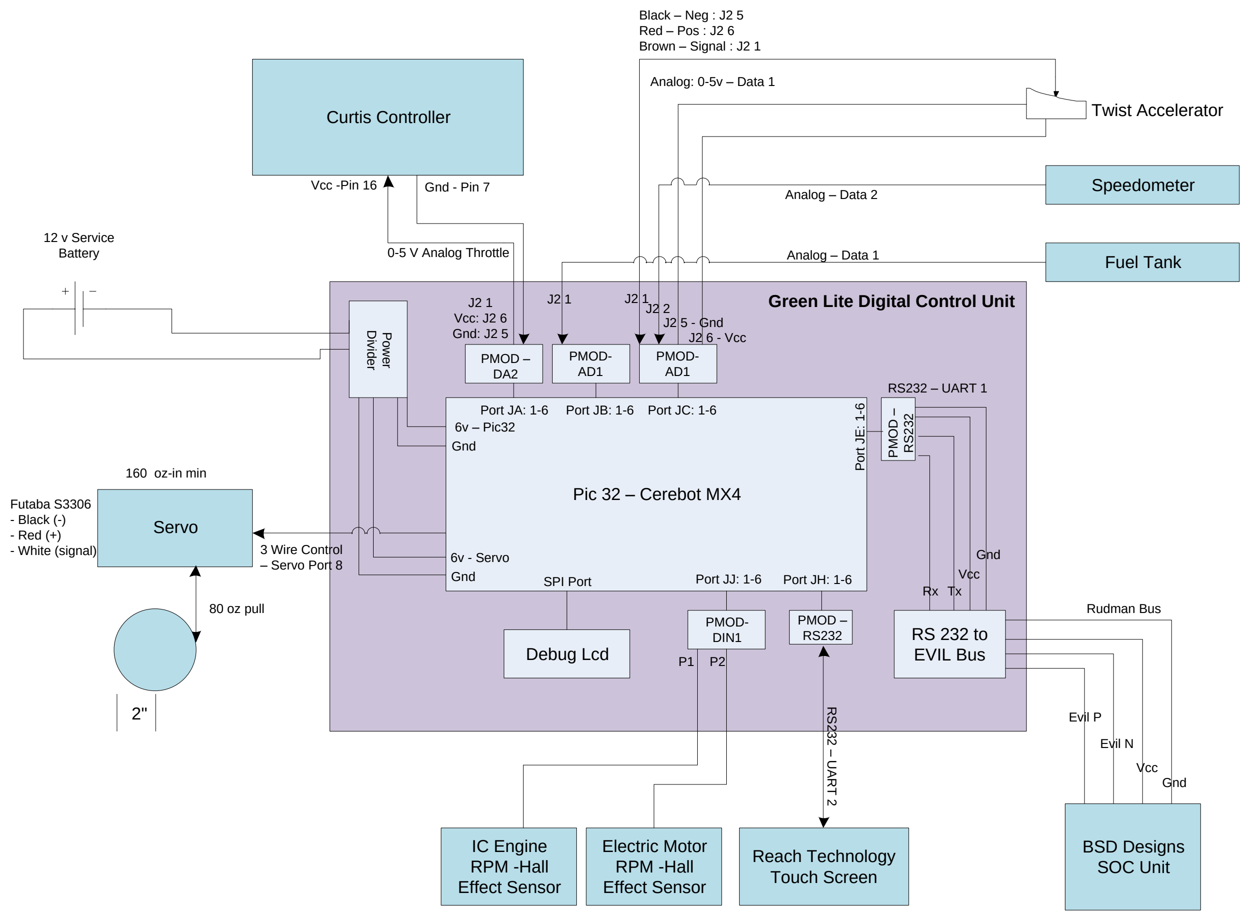

Figure 7 – Structure of the Vehicle Controller Subsystem

The second phase of the project for gas-electric hybrid vehicle control was full deployment of the system in a full size commuter vehicle. The control project was scaled up to full size with integration of components designed by a mechanical engineering student team.

- Platform Vehicle: Suzuki Bergman 650 – This two person scooter is an ideal platform for conversion to a gas-electric hybrid drive system. The smaller size of the vehicle allows testing to be done on a dynamometer designed for light four wheeled vehicles. This vehicle was modified to contain an electric motor to form a gas-electric hybrid commuter vehicle platform.
- Vehicle Deployed Electric Motor Controller: Curtis Instruments 1238 – This controller supports the target electric motor with a peak power draw of 550 Amps. The control mechanism is the same as the small scale controller allowing the Curtis 1238 to be directly

substituted for the smaller unit when the controller subsystem is deployed in the vehicle. Future goals include the integration of regenerative braking for extended mileage on a single vehicle charge.

- Vehicle Deployed Electric Motor: AC15 Electric Motor – This motor provides 8 Hp nominal and 46 Hp peak drawing a peak of 550 Amps.
- Battery Bank: 6 strands of 22 LiFePO4 Headway cells – The battery pack provides 5.5 KW of energy for use in operation of the hybrid vehicle. The Manzanita Micro SOC Head logic developed for the smaller battery bank is directly adaptable to the larger battery bank.

This part of the project subsystems required the largest cross-discipline effort combining integration mechanical, electrical, and software subsystems. The development of the emulation and simulation platforms also proved to be the most valuable as the entire system could be seamlessly moved from a safe, laboratory based platform to a highway ready vehicle without modification of software systems.

Vehicle Occupant Information Subsystem

The Vehicle Occupant Information Subsystem allows vehicle monitoring as well as basic vehicle control operations. Monitoring of the battery charging is an important aspect of electric vehicle operations and is provided so the Vehicle Occupant Inform Subsystem device can be removed from the vehicle for remote monitoring while the vehicle charges. The system also supports functions such as remote start and vehicle location and integration of charging station geo-location.

A team of four software engineering students worked on this project. The project was broken into a communication subsystem and a HMI subsystem. The platform used is an Apple iPad to allow use of multi-touch technology for the HMI. Communication with the Vehicle Controller is via an 802.11 connection, allowing the iPad device to be removed from the vehicle while maintaining remote access and monitoring capability. The point of interface between this subsystem and the Vehicle Controller subsystem is through definition of the communication protocol and data packet structure.

Students worked out a common data transmission format and collaborated on building a vehicle simulation system. The simulation system allows any of the three subsystems to produce common commands and generate or consume data for purposes of white box and black box testing.

Vehicle Telematics Information Subsystem

The Vehicle Telematics Information Subsystem receives real-time data from the vehicle controller subsystem, archives this data into a small internal database, then transmits real-time data through a 3G or 4G cell phone connection to a web server where the data will be stored for later retrieval, reduction, and analysis. This is a passive device from the perspective of the other two subsystems; however, students have proposed this subsystem act as a firewall device and message filter providing a physical separation between the internal vehicle communication systems and Internet exposed systems.

Required hardware for this subsystem is minimal. Although the hardware specified has requirements of space and power consumption, a standard PC discarded from lab use is used for initial system development keeping start-up cost low. The digital data recorder is being targeted for final deployment on a small PC, the Fit PC2[11] running a variant of Linux. Telematics support will be provided through a 3G dongle and a data plan arranged through a cell phone service provider. The web server side will run a typical Linux, Apache, MySQL, Php (LAMP) install.

A team of four software engineering students is worked on this project. The system requires UDP and TCP based communication defined for interface with the Vehicle Controller Subsystem as well as providing system decoupling. The system is partitioned into two unique subsystems. There is a subsystem installed directly in the vehicle acting as the vehicle digital data recorder and a web based system acting as a telemetry data recorder. The team has further broken down into two students working on the black box and two students working on the web presence. The web presence students were required to interface with the mechanical engineering team for definition of supported vehicle test scenarios performed during drive testing of the vehicle once all systems are integrated and deployed. The web system integrates vehicle telemetry monitoring and real-time drive test monitoring web pages into the end system deployment.

Classroom Integration Details

Each of the three projects was taken on by a team of students in the junior project sequence. This involved three separate teams required to:

1. Perform full requirements gathering and analysis for the specific subsystem chosen.
2. Perform full use case analysis for the specific subsystem.
3. Produce a sequence analysis of the defined use case scenarios.
4. Produce a full logical static class design for the specific subsystem.
5. Develop all operational code for the specific subsystem.
6. Develop testing and installation environments.
7. Test the system on the small scale test bench.
8. Install and test the finished system on the full scale vehicle.
9. Plan and manage tasks and resources throughout the 9 month project.
10. QA all deliverable project artifacts and development process states.

The larger project was primed with summer work funds. This allowed the engineering and construction of a dumbed-down controller for use as a data source and system emulation environment. The smaller system was constructed of lesser components (Cerebot MX4, Pololu Jrk 21v3, 19:1 gear motor, SOC Head, and 12.8V battery pack). This is a bench size system with all elements of the larger hybrid electric vehicle. The software engineering hybrid vehicle controller project hardware set-up is shown in Figure 2. With these lower cost components and the up-front work done, software engineering students were able to build software in the controller without involving hardware engineering. The electric vehicle component integration allowed students to see physical results of the system they created.

The course focus is on team based software development and supporting processes. The simple controller creates an environment where the project can be done with no knowledge of control algorithms and no special implementation other than direct pass-through of driver power requests. With the Cerebot framework in place, augmentation of the control system algorithm is simple for software students to perform, opening an interesting and exciting area of investigation.

The 14 students working on the different subsystems were first instructed to develop a unified communication standard. This initial meeting of the different involved parties drove an early consensus in system structure and created a common core of understanding for all involved students. In exit interviews, students stated this early collaboration was one of the most important pieces of the three term educational experience. Each team then used the standard to construct simulated UDP communication, taking a member from the individual teams to create a fourth test bench team. The test bench proved an important aspect of entire project, galvanizing the individual teams around a single deliverable system element.

Over the course of the year, projects were worked on as separate entities, each team responsible for their specific subsystem. An integration demonstration was performed at the end of the second term and a final installation and fully operational vehicle demonstration is scheduled to be performed at the start of June, 2012. In the first demonstration, one team failed to meet operational requirements of the system and threatened to cause system failure of a second team. Because of the work done on early simulation systems, the other two teams were able to demonstrate complete operation of their systems independent of the failed subsystem.

Assessment

At the end of the second term, students were assessed for retention of ideas and development of skill specifically related to:

1. Requirements gathering and analysis
2. Use case analysis and scenario definition
3. Sequence diagram analysis of the defined use case scenarios
4. Logical UML static class modeling
5. Testing derivation and system correlation
6. Task and resource management
7. QA roles and responsibilities

This was done through administration of an exam involving individual effort based on a small problem statement. Students derived and explained all of the above mentioned elements. There were a total of 29 students in two section of the junior project sequence. The results showed no difference in student performance between students in the vehicle based projects and the non-vehicle based projects.

Students were also given a questionnaire related to their appreciation of the course in general, their feelings of the subject of their project, their feeling of project outcome relevance to their expected professional work after graduation, and their perception of the overall team experience. Input was taken as a scale from 1 to 5, 5 being high. The results of the questionnaire are shown in Table 1:

Subject	**Average Non-Vehicle Project**	**Average Vehicle Project**
Course Appreciation	3.4	3.0
Project Subject Appreciation	3.8	4.3
Project Relevance	3.2	4.1
Team Experience	2.5	4.2

Table 1 – Project Relevance Questionnaire Outcome

The results show the students involved in the vehicle based projects had a higher level of satisfaction related to the overall content of their project subject and a better feeling that the project they had worked on was relevant in society. They also had a better team experience than the other teams in the course. This is in contrast to an overall appreciation of the course content and the sequencing of course subjects. The “real-world” application of the vehicle based projects most likely contributes to the feeling of project relevance and subject appreciation. Direct manipulation of a tangible object, such as a larger vehicle or even electric bike, gives students immediate feedback related to impacts they can have on a typical person’s life. The team experience is definitely enhanced through cross discipline collaboration tightly coupled with a well partitioned problem. The unexpected outcome was students taking on the vehicle based projects tended to be those with a lesser appreciation of software engineering process and related artifacts and therefore a lower opinion of the three term course sequence.

Conclusions

The Vehicle Information System has proven to provide a rich environment for the purposes of teaching software engineering tools and techniques to students working on cross discipline team based projects. The overall system is easily partitioned such that individual project groups can succeed or fail without having great impact on dependent teams. The nature of the subsystem partitioning also requires student groups to collaborate to produce a successful system.

Adapting electric vehicle technologies proved to introduce a platform that was inexpensive as a small scale system; however, software systems developed for the small scale system were directly adaptable to the full scale vehicle as deployed for eventual commercialization as defined in the two grants received from Oregon BEST and OTREC to move deployment of the vehicle forward and educate students in technologies integrated into electric vehicles. The low level coding interface is easily handled by students of embedded system engineering. Tool support of these low level areas has evolved enough to make development of embedded systems related platforms accessible to students with software engineering background showing interest in aspects of low level coding, kernel implementation, and device driver development.

There was no apparent impact from the electric vehicle based project on the learning and understanding of software engineering principles. The most pronounced benefit to the students was in the team experience of the project and the feeling of project relevance. The structure of the vehicle system forced early collaboration. This activity created a strong bond between teams required to cooperate to achieve a common goal. In the end, the greatest reward was the excitement of the involved students around the deployment of their software system on something they could ride around the parking lot.

[1] "Automotive Technology: Greener Vehicles, Changing Skills – Electronics, Software, and Controls Report", Center for Automotive Research, May 2011 (http://drivingworkforcechange.org/reports/electronics.pdf)
[2] Ibid.
[3] "Challenges in Automotive Software Engineering", Manfred Broy, Proceedings of the 28th international conference on Software engineering, 2006
[4] Ibid.
[5] "In-vehicle infotainment software architecture: Genivi and beyond - Part 1", David Kleidermacher, Green Hills Software, Matt Jones, Genivi Alliance, EE Times Europe Automotive, Nov. 2011
[6] Oregon Built Environment and Sustainable Technologies , Retrieved March 2012 from http://oregonbest.org/
[7] Oregon Transportation Research and Education Consortium, Retrieved March 2012 from http://www.otrec.us/
[8] Cerebot MX4 Specification. (2010). Retrieved Sept. 11, 2011. from http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,396,754&Prod=CEREBOT32MX4
[9] Pololu Jrk 21v12 USB Motor Controller with Feedback Specification. Retrieved August 1, 2011. from http://www.pololu.com/catalog/product/1394
[10] 19:1 Metal Gearmotor 37Dx52L mm with 64 CPR Encoder Specification. Retrieved August 1, 2011. from http://www.pololu.com/catalog/product/1442/specs
[11] Fit PC2 Specification. (2010). Retrieved June 10, 201. from http://www.fit-pc.com/web/fit-pc2/fit-pc2-specifications/