Embedded System Design for A Smart Rover

Sam Stokes
Microsoft Corporation
Yi Cheng, Kathleen Hayden, Saeed Monemi,
Rajan Chandra, Tim Lin, Zekeriya Aliyazicioglu
California State Polytechnic University, Pomona

1. Introduction

The Electrical and Computer Engineering Department (ECE) at California State Polytechnic University, Pomona (Cal Poly Pomona) offers Bachelor of Science degrees in both Electrical Engineering and Computer Engineering. Our Computer Engineering curriculum is composed of hardware and software courses, which emphasize the use of both technologies to produce optimized system designs.

In response to the NASA Research Announcement (2002), California State Polytechnic University, Pomona (Cal Poly Pomona) submitted a proposal for the "Partnership Awards for the Integration of Research into Undergraduate Education" (PAIR) program. The purpose of our proposal was to integrate the Jet Propulsion Laboratories (JPL) deep space exploration rover technology research into the undergraduate curriculum of the engineering, technology and computer science departments at Cal Poly Pomona. Cal Poly Pomona's proposal, "Deep Space Exploration using Smart Robotic Rovers", was selected for funding and began in September 2002.

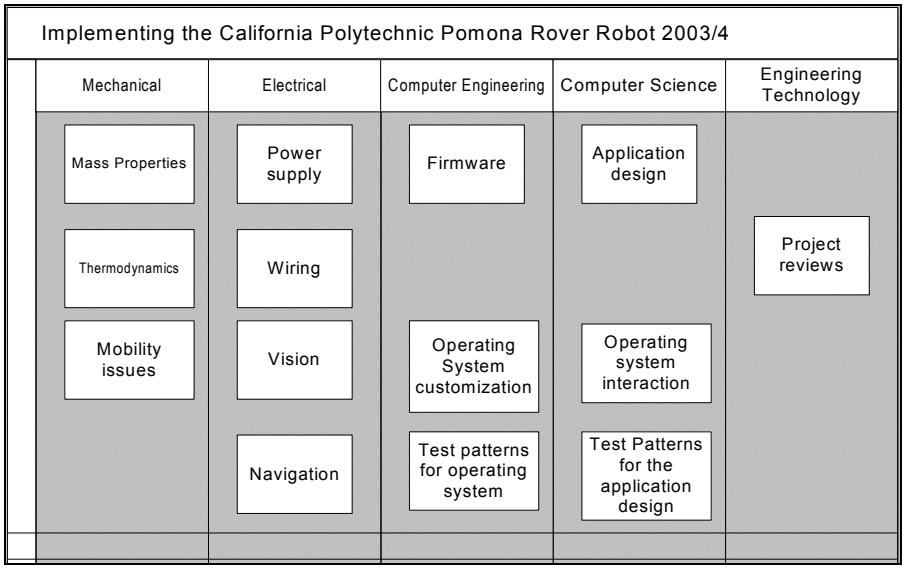

Figure 1. Work efforts by the students

Teams composed of faculty and students are engaged in the development of a robot, which will be capable of being deployed in an unfriendly environment, capable of autonomous navigation, scientific data collection and communication with a base station.

Through their participation and contributions towards the mission of this project, students receive degree credit, which can be used to fulfill their degree capstone requirement, Team Senior Project.

The project is an interdisciplinary project including students and faculty from the Electrical and Computer Engineering Department, the Computer Science Department, the Engineering Technology Department and the Mechanical Engineering Department. All faculty and students are divided among five sub-teams: hardware, firmware, software, digital signal processing and communications (See Figure 1).

A basic requirement for any Computer engineer is a fundamental understanding of operating systems. In today's marketplace sophisticated embedded microcontrollers are commonplace. Engineering applications employing these embedded controllers require sophisticated software, such as an operating system, to allocate resources and coordinate system activities.

Based on the complexity of the rover application, an operating system will be required to coordinate the various subsystems required for this design. After examining various operating systems, Windows CE .NET was selected as the most appropriate for this application.

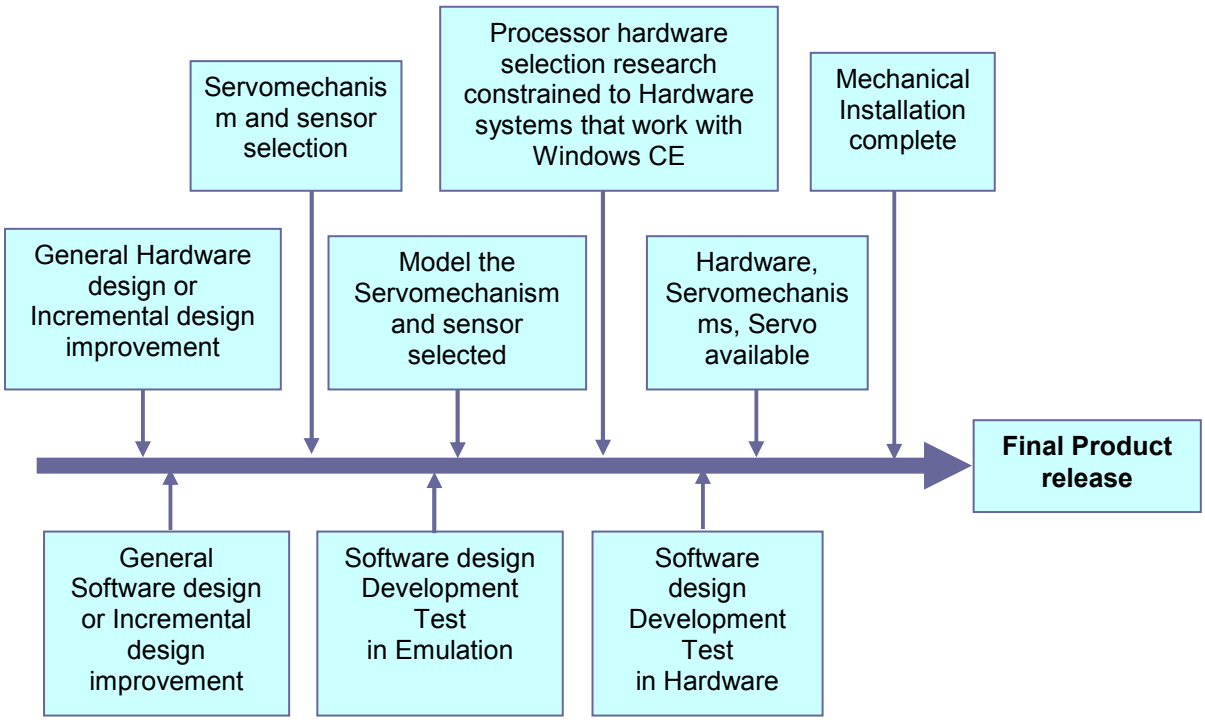

Figure 2. How the academic disciplines work together

Using the Windows CE .NET Platform Builder, the student team have been able to quickly perform the following tasks:

- Design a custom operating system using open or custom designed components.
- Modify existing software components and add them to the operating system

- Start software development and troubleshooting prior to receiving hardware using a high fidelity emulation tool
- Perform in-circuit trouble shooting without any additional test equipment such as an in-circuit emulator
- Gain insight on hardware development that uses a standard software platform; determination of necessary tools that must be designed to deploy customized hardware
- Use testware to debug software components and hardware
- Implementation of a graphical user interface design

The Windows CE .NET, Visual Studio.NET, Visio, SQL server and MS Project products are available to students and faculty through the Microsoft Developer Network Academic Alliance program. All of these design tools map to various industrial/academic products. For example, the Visio approach to UML maps to the approach used in Rational [2] from IBM, MS Project maps to project tools such as Primavera[3], and so forth. The choice of tools was made independently and these products have been successfully used in this project.

2. The Smart Rover Project

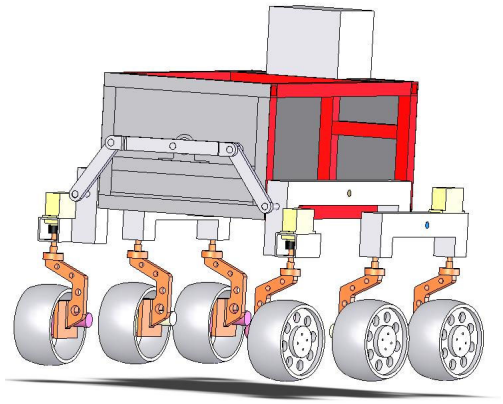

Figure 3. Smart Rover

System Specification

- The rover shall use a Real Time Operating System (RTOS).
  To use a commercial off-the-shelf RTOS such as the Windows CE .NET allows system designers to use many available device drivers. For this project, real-time controls will be initiated by the FPGA based hardware controller.

  It was decided to implement all motor control functions and sensors detection in an FPGA to meet the stringent real-time requirements of digital control systems, including sampling frequencies of 200 Hz or higher. Selection of an FPGA to accomplish these tasks provides maximum design flexibility. The rover will include four DC motors to propel the rover, four servomotors to adjust the wheel position and an additional four servomotors to control the robotic arm movement.

- The rover shall be capable of navigating through obstacles from point A to point B with minimal human interaction.
  The smart rover shall have autonomous navigation capabilities because future space explorations to Mars and beyond will make remote control very difficult due to the extremely long delays in the communication channels.

- The rover shall be capable of communicating via wireless using IEEE 802.11b/g standards.

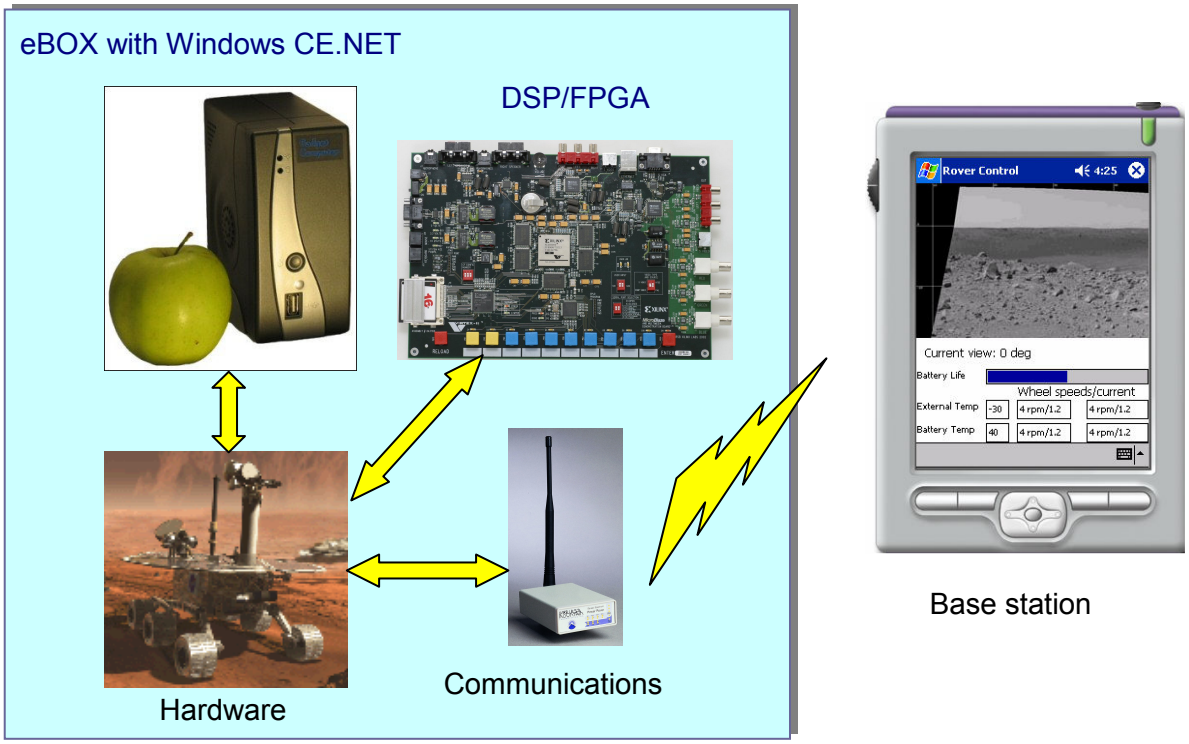

Figure 4. Integrated Smart Rover Systems

The wireless communication system is used to emulate communications between the rover and the base station. A human controller can observe the images transmitted from the rover, and send commands, if necessary, to the rover.

- The rover shall be capable of transmitting at least one image per second from each on-board camera

Tasks currently being investigated and resolved include:

Digital Camera

- USB interfacing
- Digital image compression
- Object recognition for vision-based navigation

Rover Control System

- Sensor interfacing
- Actuator interfacing
- Control calculation

- Navigation using GPS or other RF devices

Wireless Communication

- Network Interfacing
- Digital image transmission
- Rover command transmission

Software Hardware Partition

The tasks that can be handled by existing software modules, such as USB interfacing and network interfacing, are assigned to the X86 processor in the eBOX-II. The real-time tasks and the hardware sensor interfacing that requires many I/O pins are assigned to the hardware on FPGA Board. The X86 processor uses a serial communication link to send and receive information from the FPGA hardware. The block diagram of the embedded controller for the rover is shown in Figure 5.

2.1 Hardware

The primary processing component for the rover project will be provided by quasi-PC, the eBOX-II. The single chipset, Vortex86 family, provides a high performance and low cost SoC (System on Chip) solution by integrating an x86 compatible processor, high performance North Bridge, advanced hardware GUI engine and a Super-South bridge. In addition, the Vortex86 family complies with Easy PC Initiative that supports Instantly Available/OnNow PC technology, USB, Legacy Removal, CIR, Memory Stick, Smart Card and Slotless Design for a variety of IA (Information Appliance) applications. Vortex86 family integrates a high-performance processor that supports x86 instruction set with 3 integer units, 3-way superscalar architecture, and a fully pipelined floating point unit. In addition, the processor is a power-efficient design that optimizes the power consumption for information appliance applications. It has a main memory of 128 MB of DRAM, and a flash memory of 256 MB. Also included are one parallel port, one serial port, three USB ports, and one Ethernet port. It supports Windows CE .NET, Windows XP Embedded, Widows 98 and LINUX.

A Xilinx Spartan 2E FPGA Board was chosen to implement the hardware controller because of its exceptional flexibility in implementing hardware functions and interfaces. The FPGA has 154 I/O ports and 200,000 logic gates. Thus, it can control and interface with many devices and generate many signals. This capability is important because it reduces the burden to the processor (eBOX-II) thereby lessening the potential degradation of the system's performance when handling these hardware control tasks. Furthermore, the FPGA's ability to be reconfigured is very valuable for this experimental robotic design. The FPGA design can be easily adjusted to accommodate modifications to the rover design including the number and types of motors required by the rover's propulsion system and robotic arm.

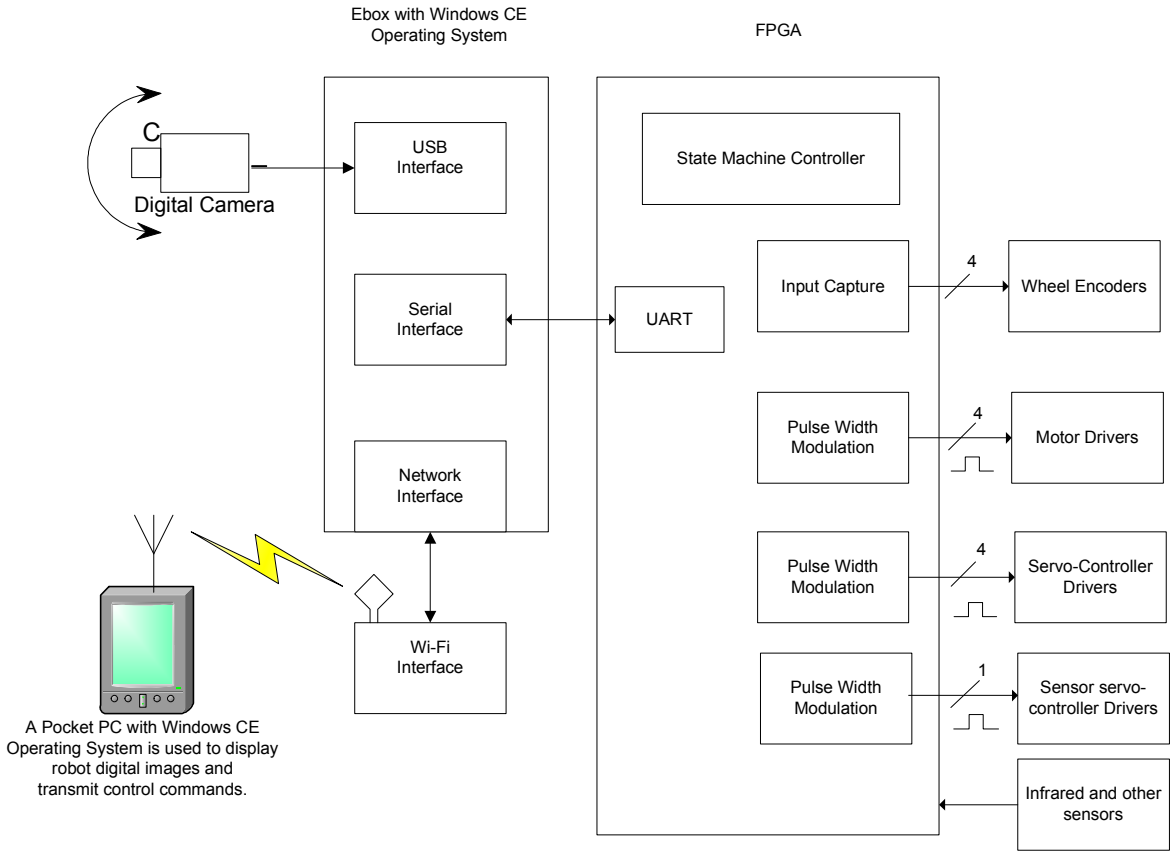

Figure 5. Firmware subgroup

The FPGA board has two built-in hardware interfaces to an eBOX-II: a serial and a parallel port. The data rate requirement for communication between the eBOX-II and the rover is quiet low, therefore a standard serial communication rate of 9600 BAUD has been selected. Furthermore, our students are already familiar with the design of a Universal Asynchronous Receiver Transmitter (UART) for the FPGA Board and the serial communication software for the eBOX-II. Commands are transmitted serially to the FPGA and sensory information collected by the FPGA is returned to the eBOX-II via this same serial interface. We plan to add a USB interface in the future to the FPGA Board to handle the anticipated higher data rates.

A State Machine Controller has been designed for wheel and servo control, and the generation of the pulse width modulated signals required by the DC Motors and servomotors. Each output is determined from the input CMD (command). The original design consisted of 5 basic commands which included stop, forward, reverse, rotate right and rotate left. These commands are represented by the main states with the same name as shown in Figure 6. However, it was later determined that additional states were required to verify the state of the servo motors controlling the angle of the wheels prior to rotating the wheels.

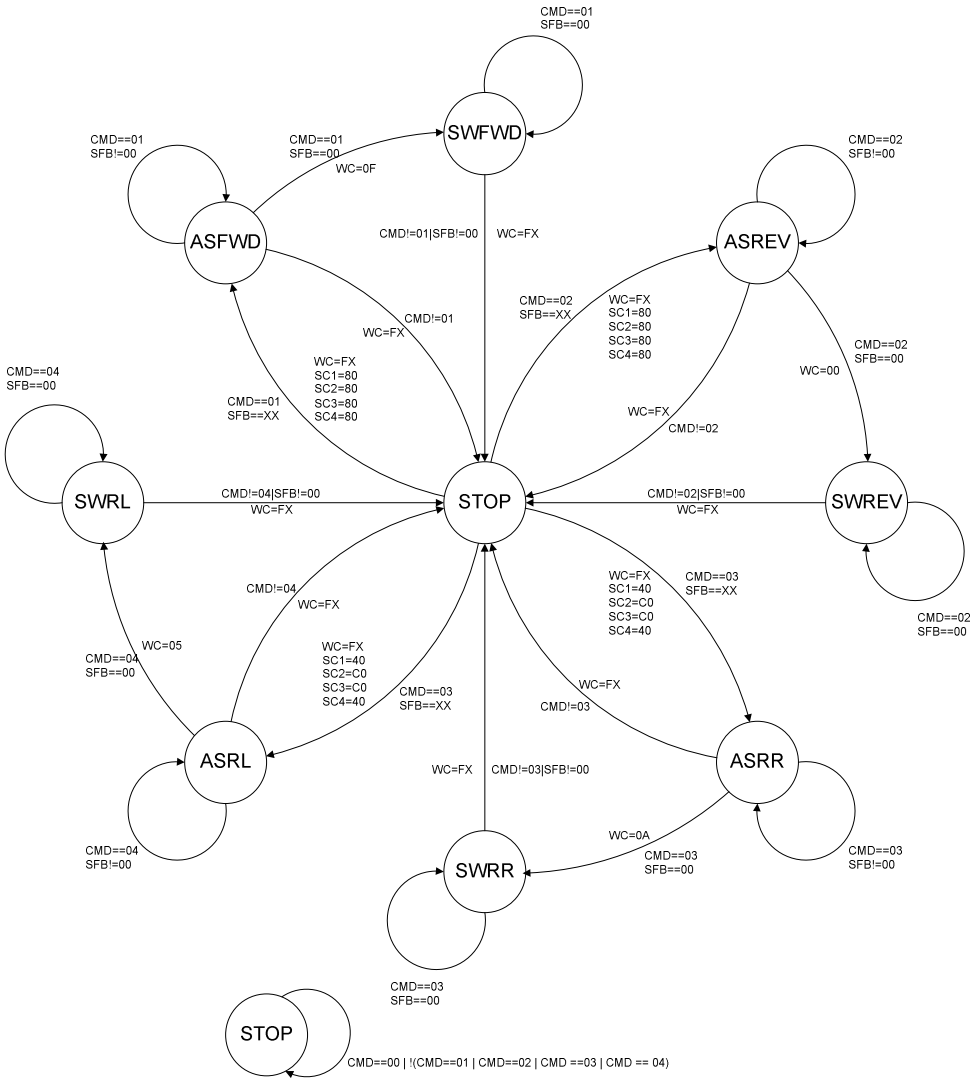

Figure 6. Finite State Machine Block Diagram

Verilog Hardware Description Language was used for implementing the FPGA design. All the hardware modules were coded using behavior modeling. Logic simulations were performed using ModelSim. Design verification tests have been successfully conducted using the prototype vehicle.

A classical Proportional-Integral-Derivative (**PID**) controller is used to control the velocity and displacement of the rover. Short distance sensors, including scanning infrared sensors, provide distance information to nearby obstacles. The rover will turn right or left to avoid the obstacle. A vision-based navigation is under investigation.

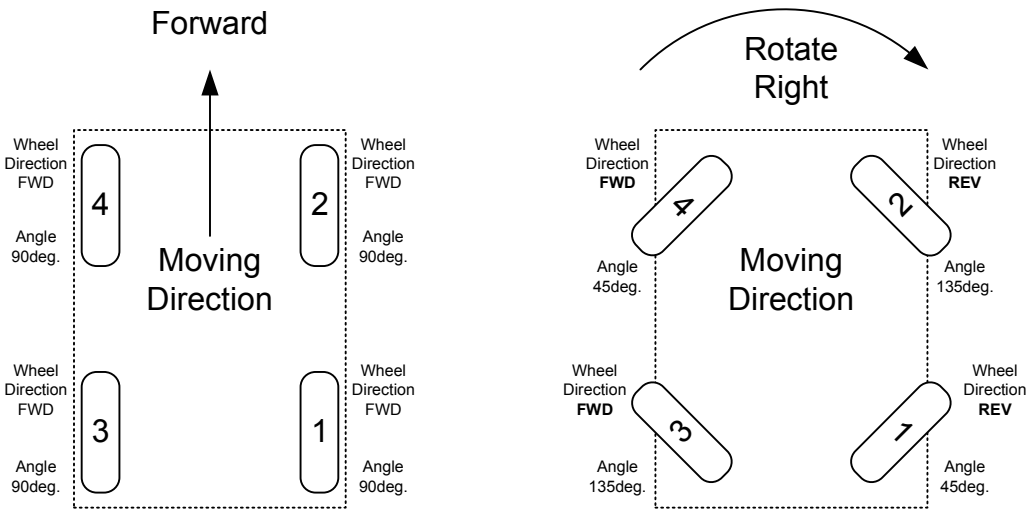

Figure 7. Wheel angle and direction of the Forward and Rotate Right commands

A **Fuzzy Logic** Controller will be used to control the five servomotors on the robotic arm. The rover will be given a command to pick up an object, and to move it closer for a camera to take high-resolution images.

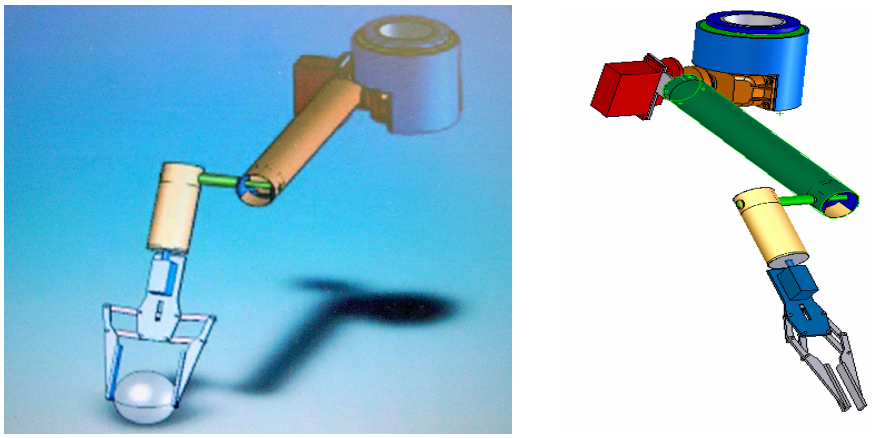

Figure 8. Rover Robotic Arm

2.2 Software

The software requirements for Smart Rover are as follows:

1. To use a real-time operating system such as the Windows CE .NET,
2. To use Universal Serial Bus (USB) interface to transfer image files form a digital camera,
3. To transmit digital images and to receive Rover commands from the network interfaces,
4. Send commands to the FPGA and receive status from the FPGA via a serial communication interface.

The process of building a custom operating system using Windows CE .NET was very straightforward. The students used the Windows CE .NET Platform Wizard to build a customized operating system for the Smart Robotic Rover. They used the Board Support Package for the X86 Emulator, and selected the Custom Configuration option. They included all device drivers required by the rover, including serial communication, local area network, wireless network and USB interface. Embedded Visual C++ [5] was used to compile the interface programs to work with the operating system.

2.3 Digital Signal Processing

In the first year, the digital signal processing (DSP) group concentrated on developing lossless still image compression algorithms. For lossless compression, the original image can be reconstructed from the compressed file without error. The algorithms selected for this task utilize recent advances in wavelet theory, especially integer wavelet transforms, and coding theory to maximize the compression ratio. An uncompressed color or black-and-white image in bit map (BMP) or tagged image file format (TIFF) is transformed into a text file consisting of a matrix of pixel values. This text file is filtered (transformed) using various integer wavelet transforms. The integer wavelet transforms are applied as many times as needed. This process reduces the redundancy of the original picture as much as possible. The integer wavelet transforms such as (1,1), (2,2), (2,4), (2+2,2), (3,1), (4,2), (4,4), (6,2), 9-7, D4, S+P are used in the transform and the compression ratios are compared for various input images. The transform that provides the best compression ratio is selected for the given image. The transformed image is segmented in 8 by 8, 16 by 16, 32 by 32, 64 by 64, or 128 by 128, and each segment is encoded using elementary Golomb codes or binary arithmetic codes. The size of the segment that provides the best compression ratio is selected for the given image.

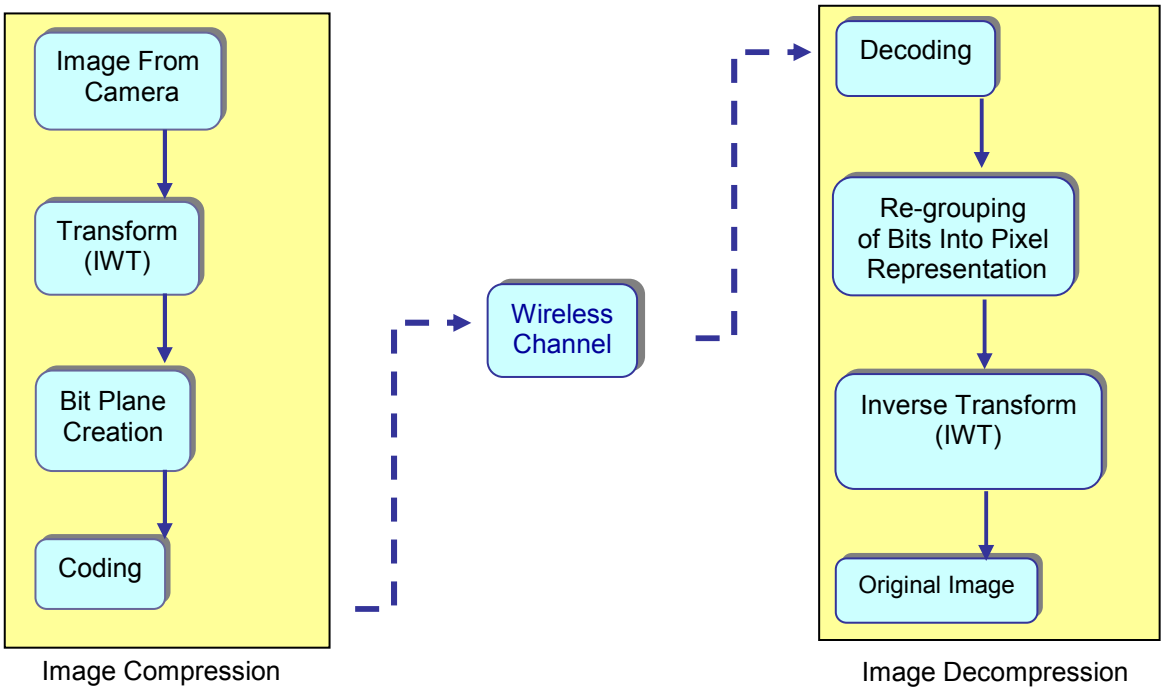

Figure 9. Flow Diagram of Image Compression

The compressed binary data from each segment are combined into one file consisting of header and data. The header contains information on all the parameters that are necessary to decode at the receiver. The procedure for compression and explosion is diagrammed in Figure 9.
Digital cameras will be employed to capture scientific images to be transmitted to the base station. These images will compressed using Integer Wavelet Transform techniques and sent to the base station.

A handheld or portable device will be used as a base station. The digital images form the rover cameras will be displayed. The status of the robber, including camera viewing angle, battery life, external temperature, wheel rotational speed, current, etc. will also be displayed. A human operator can also use this device to send commands to the rover.

Figure 10. Hand Held Base Station

3. Conclusions

1. Students garner real-world experience in the design of an operating system and high-level application programs for a smart rover robot application.
2. Students learn to use sophisticated software development tools, which will enhance their education and may lead to future job opportunities.
3. Students gain a greater understanding of tasks required to undertake and solve complex engineering problems.
4. Students appreciate the superiority of integrated systems, which employ both hardware and software.
5. Students appreciate the synergy derived from working in a multi-disciplinary, inter-department team.

6. Students learn to coordinate the tasks required to ensure a successful conclusion to this engineering effort.

References:

[1] Wulf, W. A. 2000. The standards for technological literacy: A national academies perspective. The Technology Teacher, 59, p.10-12.
[2] http://www-306.ibm.com/software/rational/
[3] http://www.primavera.com/solutions/software.html
[4] http://www.msdnaa.com
[5] http://msdn.microsoft.com/vstudio/device/embedded/evcandcenet.aspx
[6] http://www.microsoft.com/windows/Embedded/ce.NET/evaluation/hardware/hcl.asp
[7] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcepb40/html/pbconplatformdebuggingtesting.asp; "Microsoft Windows CE .NET 4.2 Platform Debugging and Testing"
[8] John Sharp, Jon Jagger, Microsoft Visual C#.NET, Microsoft Press, 2003.
[9] Tom Archer, Andrew Whitechapel, Inside C#, Microsoft Press, 2002.
[10] Doglas Boling, Programming Microsoft Windows CE.NET, Microsoft Press, 2003
[11] Harvey M. Deitel, Paul J. Deitel, Visual C++.NET: How to Program, Prentice Hall, 2004.