DESIGN OF A FULL-FEATURED ROBOT CONTROLLER FOR USE IN A FIRST-YEAR ROBOTICS DESIGN PROJECT

Michael A. Vernier, Patrick M. Wensing, Craig E. Morin, Andrew H. Phillips, Brian A. Rice, Kevin R. Wegman, Chris P. Hartle, Paul A. Clingan, Krista M. Kecskemety, Richard J. Freuler
Engineering Education Innovation Center
The Ohio State University

Abstract

For the past nineteen years, the first-year engineering honors program at The Ohio State University has included a robotics design project as the cornerstone of its yearlong curriculum. Over these years, the MIT Handy Board has served as the controller for the autonomous robotic vehicles built by students. This paper details the design of a new, custom robotics controller that provides a modern update to the MIT Handy Board for use in a first-year robotics project. The unique features of this new controller are highlighted, along with a description of the prototype iterations and final design. The results of a pilot launch with this final design are also detailed which includes a performance comparison with the Handy Board. It is expected that this new controller will meet the needs of the program for years to come and could serve as a basis for other design courses that use similar devices.

Introduction

Since 1996, the first-year engineering honors program at The Ohio State University has included a robotics design project as the cornerstone of its year-long curriculum. This component of the program has helped to increase retention and has provided students with valuable teamwork skills. Over these years, the MIT Handy Board [1,2] has served as the controller for autonomous robotic vehicles built by students. It met the needs of the program for many years beyond the typical lifetime of such electronics, but the age meant that key components were no longer in production and thus, increasingly difficult to replace. Additionally, improvements in technology have allowed for increased sophistication in robots and competition courses which became limited by the Handy Board. This paper details the design of a new, custom robotics controller that provides a modern update to the MIT Handy Board for use in a first-year robotics project.

Background

In the 1990s as a part of the Gateway Engineering Education Coalition, Ohio State's College of Engineering replaced a traditional first-year engineering course sequence with a multiple-track program that retained the essential parts of those traditional courses but added hands-on laboratory experiences that led to design/build projects [3]. This approach has had a noticeable, positive effect on student retention [4]. The result of this effort was the development of a first-year engineering program with a track for honors students, the Fundamentals of Engineering for Honors (FEH) sequence [5], which was a tightly coupled three-course sequence offered in a quarter-based academic year. This three-course sequence under quarters was converted to a two-course sequence under semesters beginning in autumn 2012.

The first course in this sequence, named ENGR 1281H, presents an introduction to computer programming with C/C++ and MATLAB in order to teach engineering problem solving. The course has one dozen hands-on lab experiences designed to further explore the engineering disciplines. It also incorporates a short design project carried out by two-person teams over a one week period at the end of the academic term.

The second course in the sequence, ENGR 1282H, includes an introduction to engineering graphics and parametric solid modeling using SolidWorks. As a culminating experience for first-year engineering honors students, ENGR 1282H focuses primarily on the planning, execution, management, documentation, and presentation of an engineering design/build project [6]. In many respects, this first-year cornerstone design project course is comparable to a senior capstone design course in which a student might participate as part of the final requirements for a chosen engineering discipline. The focus on planning, management, documentation, and presentation in ENGR 1282H is in contrast to focusing on design alone, as done in many senior design projects.

Though students can pick from many different projects, the majority of students choose the robot design project where they create an autonomous robot to perform prescribed tasks within a specified time limit while operating over a specially constructed course. This project culminates in a head-to-head tournament in which a champion robot is determined. In designing and building the robots, the students make use of the graphics, the computer programming, the engineering problem solving, the hands-on labs, the physics, and the mathematics of the previous academic terms. Working in teams of three or four, the students are required to demonstrate and present the results of their efforts by submitting progress reports, participating in performance reviews, writing a formal project report, and making an oral presentation about their project.

Beginning in 1996, the Handy Board had been used for this robot design project. This board was developed in 1995 at the MIT Media Lab by Fred G. Martin [1,2]. Designed for experimental mobile robotics work, this once popular Motorola 68HC11-based controller has a variety of digital and analog inputs for interfacing with sensors and support for controlling four brushed DC motors and six servos. The Handy Board has 32 KB of battery-backed static RAM, and a simple connector system that allows sensors to be easily plugged into the board. It includes a small 32-character LCD screen, as well as a rechargeable battery pack. The Handy Board can be programmed in a Windows environment called Interactive C, which allows a subset of the C language to be used.

After more than twelve years of experience using the Handy Board, many issues arose due to the device's age that led the FEH program to the decision that an upgrade was needed.

Rationale for a New Robotics Controller

The Handy Board was an innovative design for its time that incorporated many features into a device that could be hand-assembled by students. The simplicity of the design kept the board small and easy to assemble, but those advantages meant it lacked physical protection and protective circuit elements resulting in a continuous need for repairs. From an operations standpoint, the cost and labor required to maintain a stock of Handy Boards increased every year as key components became increasingly difficult to replace.

The limitations of the Handy Board also negatively affected the student experience. While imposing design constraints is important in a design project, the limitations of this particular controller distracted from the primary focus of the course. For example, the Handy Board predates technologies such as large flash memory, USB, and high-capacity battery technologies, which are typically found in modern microcontrollers. Modern computers, especially laptops, do not have serial ports, requiring USB-to-Serial adapters to load programs. The battery capacity and charge times limited the number of design and testing iterations that were possible.

Evaluation of these and other limitations led to the development of a list of requirements and a list of desired features for a replacement controller:

Requirements—

- Reliability and robustness
- Support for both C/C++ and LabVIEW programming
- Sufficient digital and analog I/O ports
- Support for a broad range of motors
- Sufficient servo channels
- High-capacity battery with fast charging
- Easy application loading interface, such as USB
- Similar size and weight to the Handy Board
- Text display
- Interface buttons
- Sufficient processing power

Desired features—

- Graphic display
- Integrated wireless communication
- Replaceable battery modules
- Similar pricing to Handy Board
- Extendable design for future use
- Support for multiple programs chosen at runtime

Assessment of Available Replacement Options

The initial plan was not to design a new controller from scratch, but rather to identify an existing controller to be adopted by the FEH program. Many microcontrollers available at the time were reviewed. Table 1 contains a list of some candidates and their limitations. Each complete controller available on the market failed to meet one or more of the primary requirements, so the focus shifted to developing a custom solution. Since LabVIEW support was deemed critical to the FEH program, the team began development and testing around a Texas Instruments LM3S8962 evaluation board that could be used as a LabVIEW Embedded target. Initial work focused on expanding the board to include support for the required features, but unnecessary functionality in the evaluation module resulted in an excessively large system. Therefore, the project moved on to developing a completely new board based on the core components of the evaluation module.

Overview

After none of the available options met the needs of the FEH program, teaching assistants began working on a design for a new robotics controller in 2009. This device has come to be called Proteus. The FEH program introduced this new robotics controller as a replacement for the Handy Board in spring 2013. The Proteus robotics controller is a device similar in size to the Handy Board and features a top-mounted, touch-sensitive LCD color display. Its increased processing speed, memory, and expandability allow for more unique designs and solutions to the design competition challenges.

The main contribution of the Proteus is its integration of a wide set of features needed for robotics education into a compact, affordable device. Even today, years after the initial concept, there is not a similar, readily available

Table 1: Limitations of existing controllers.

Microcontroller	Source	Limitations
NXT	Lego	Very limited I/O, underpowered
Blackfin Handy Board	Dr. Fred Martin	Large, expensive
Arduino*	Arduino	Required significant expansion to meet requirements, no LabVIEW support
Qwerk and other Miniature Linux PC	Charmed Labs, Various	Large, heavy, significant power requirements, and required I/O peripherals
Flex Stack	Boston Engineering	Many modules required to meet specifications, expensive, large

* Note that Arduino was still in infancy at the time of this evaluation

commercial solution that would meet the program's needs. It was not a single innovation that has enabled the capabilities of the device, but rather a number of important design ideas that contributed to the device's success. The use of a powerful main single-core chip with flexibly configured I/O allows for highly reconfigurable inputs, while the use of a multi-core secondary chip allows responsive and precise motor control. Many smaller innovations, such as the use of a low-cost laser cut case, have come together to make the Proteus a unique, full-featured robot controller that is unparalleled by other currently available options.

The remainder of this paper is organized as follows. The following section describes the initial design and a first, small-scale pilot launch of the new controller. The next section provides an overview of the final design and details of its most innovative features. The results of a second, medium-scale pilot are then presented with student performance results compared to results with the Handy Board controller. A summary and description of future prospects for the Proteus controller is presented in the last section.

Proteus 1.0 and First Pilot

Based on the requirements identified above, a custom robot controller was developed. Prior to a full launch of the new controller, an initial design iteration was tested in a limited-size pilot course, ENG 694. This section describes the new controller as well as the course objectives, lab exercises, and lessons learned.

Initial Design Overview

The first iteration of the Proteus, shown in Figure 1, was based on an LM3S ARM Cortex-M3 processor [7] from Texas Instruments (TI) operating at 50 MHz. The main design features of the Proteus 1.0 are summarized in Table 2. For this board, the choice of the main processor was heavily influenced by the program's previous work to add custom expansion modules to a processor evaluation board provided by TI. This processor had 256 KB of flash and 64 KB of RAM. This main processor ran the students' control logic and interfaced directly with the analog input and digital input/output modules through the SPI serial bus. Other functionality was offloaded to two other supporting processors.

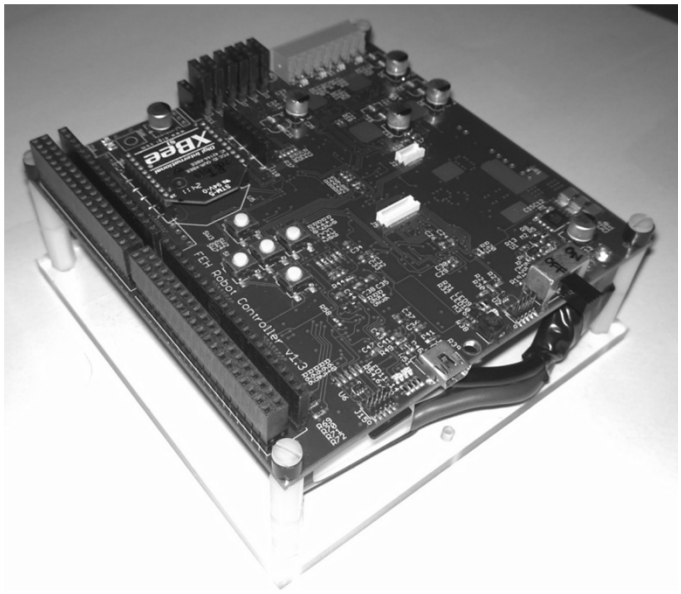


Figure 1: Proteus 1.0 controller.

Table 2: Overview of the Proteus 1.0 controller.

Processor	50 MHz ARM Cortex-M3
Storage	256 KB flash, 64 KB RAM, 2 GB microSD
Display	None
Wireless Communication	250 Kbps XBee, 2400 bps RF
Motors	4 brushed DC at 5A max, 8 hobby servo
I/O	20 x 10-bit analog in, 18 x digital in/out
Battery	2200 mAh lithium polymer
Battery Charger	Off-board (but internal) smart charger
Programming	GCC/Qt Creator & miniUSB

A second processor, the Parallax Propeller [8], was used to control four brushed DC motors, seven hobby servo motors, an XBee wireless module [9], and a 434 MHz wireless module. An ARM Cortex-M0 processor from NXP was to be used as a third processor to interface with an external LCD module which was not finished before the pilot course.

Proteus 1.0 was powered by a high capacity three-cell lithium polymer battery capable of handling the maximum power requirements of the output ports. An off-board smart charger safely charged the battery. For simplicity, this charger module was packaged inside an acrylic case and was always connected to the battery. For charging, only a connection to an external power supply was needed.

Students used the Qt Creator [10] integrated development environment to create programs for the device. Custom plugins were created to integrate an ARM GCC compiler into Qt Creator. These plugins also handled the processor programming procedure. Custom firmware libraries were created to provide access to the various input and output capabilities, a serial console for text output via USB, and a real-time operating system.

ENG 694 Course Objectives and Lab Exercises

ENG 694 was offered in autumn 2011 to upper-level students interested in designing, developing, and testing C/C++ software for the first design iteration of the Proteus. The students who enrolled in ENG 694 had completed the first year robot design project earlier in their academic career. As such, the emphasis of ENG 694 involved using the controller with small robotic vehicles, but in contrast with the freshman course, students were given a standardized pre-built robot rather than being required to construct one. This allowed focus on programming the Proteus. This atmosphere allowed for low-risk testing of the Proteus hardware under realistic conditions. As a byproduct, the course allowed the students already familiar with high level embedded programming a chance to gain experience with the tools necessary for lower level embedded design.

The course was divided into two halves: a structured set of laboratory exercises and an open ended design project. The laboratory exercises included:

- Introduction to development work flow
- Reading analog and digital sensors
- Logging data to an on-board microSD memory card
- Controlling brushed DC and servo motors
- Detecting random infrared frequencies
- Processing data from an on-board accelerometer
- Communication with two separate wireless systems
- Robot navigation based on line following and wirelessly transmitted position information

The design project portion lasted for the final four weeks of the term. During this time,

students worked on two or three person teams to complete a project of their choosing. Projects included:

- Wireless programming of the device
- Robot control via a serial interface from a LabVIEW application
- Image acquisition using an external camera
- Robot Laser Tag where robots were controlled remotely via mobile phones. Wireless communication between robots and a positioning system determined the success of hitting the opponent.
- Path planning and robot navigation via virtual potential fields

These independent projects pushed the hardware capabilities of the device and illuminated many successes and areas for improvement of the initial design. Many issues were resolved during the course through updates to the software libraries and programming environment.

Lessons Learned

Several areas for improvement were identified during ENG 694. Most prominent was a need to simplify the overall hardware design. The primary processor did not provide enough on-board functionality, thus introducing complexity throughout the design. For example, a third processor was required to provide sufficient I/O pins for the desired LCD. All input pins used by the student were handled by separate input expansion chips. Wireless functionality was managed by the Propeller processor and transmitted back to the primary processor introducing data losses at high transmission rates. Removing these complexities would reduce the overall system cost through reduced cost of components and shorter assembly time while reducing the required circuit board footprint.

Another crucial change related to the battery charger. The off-the-shelf charging module was not designed to be continuously connected to the battery because it contained a cell balancing circuit. This circuit would over discharge the battery pack making it unusable. Due to the high cost of new batteries, a new charging method would have to be developed by obtaining either a different charging module or developing a custom circuit.

Qt Creator was well accepted and proved to be an effective development environment. The primary software issue was found to be the deployment of updates to the libraries and development environment. Different versions were manually distributed and it was difficult to ensure the students were using the most recent version. This showed the need for a system that would obtain and apply software updates with minimal student or instructor intervention.

Proteus 2.0 Design

From the successes and failures identified in ENG 694, the hardware was significantly redesigned. This new design was centered on a different main processor which led to a significant decrease in the size and cost of the Proteus. The following two sections outline the hardware specifications and the software environment used in this final design iteration. Followed by fabrication and packaging details. Then a high-level comparison with the Handy Board is presented. Finally, the testing and troubleshooting procedures used for verifying the device functionality after production assembly are described.

Hardware Design

The redesigned Proteus 2.0 uses a Freescale Kinetis K60 processor [11] which has an ARM Cortex-M4 core running at 96 MHz. This chip was chosen because it provided ample I/O features allowing for direct control of the XBee wireless module and the LCD. There were also enough pins remaining to put all digital and analog inputs on the chip without the need for serial expansion modules. Using these pins directly also enabled advanced communication protocols such as I2C, SPI, UART, and CAN for future expansion. The processor change also doubled the amount of available flash and RAM for user programs, doubled the processor speed,

and added on-board DSP and USB functionality. Through additional software development, the processor would also be able to support programming with LabVIEW.

Figure 2 shows a block diagram of the major components inside the Proteus. These components were laid out on a two-sided circuit board as shown in Figure 3. The main processor is connected to only one secondary processor in this design. This secondary processor, the Parallax Propeller, remained in the design to control four brushed DC motors via individual high current drivers and eight hobby servo motors. The motor drivers provide a current feedback output which is monitored by the Propeller via a serial analog to digital converter. A small piezo buzzer is also attached to the Propeller to provide the user with audible feedback if necessary.

Instead of using separate banks of analog and digital inputs, the sensor ports provide both

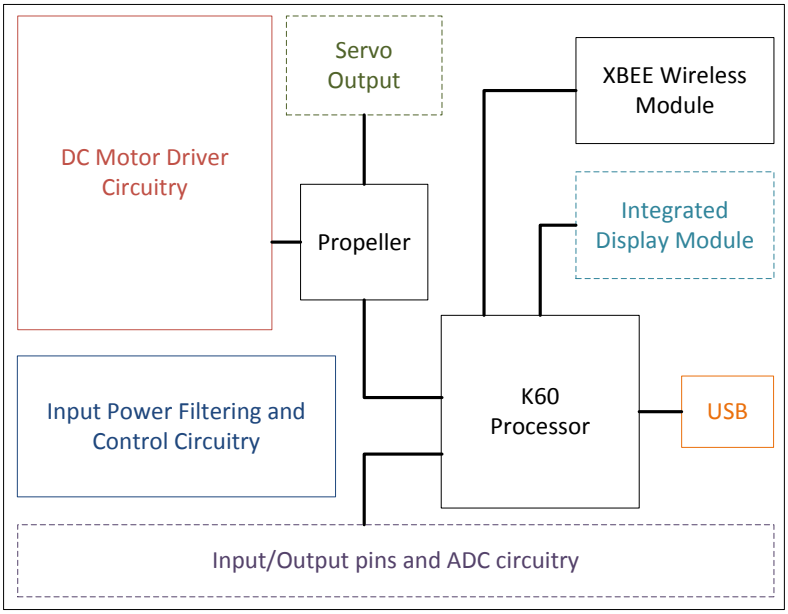
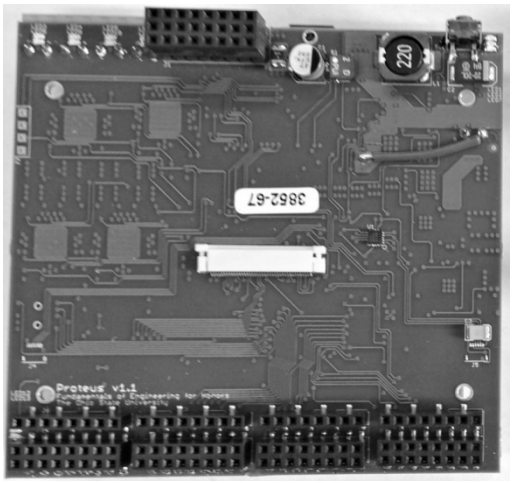


Figure 2: Proteus 2.0 controller diagram with main components. Solid outlined components are mounted to the back side of the PCB in Figure 3. Dashed components are mounted to the front.

(a)



(b)

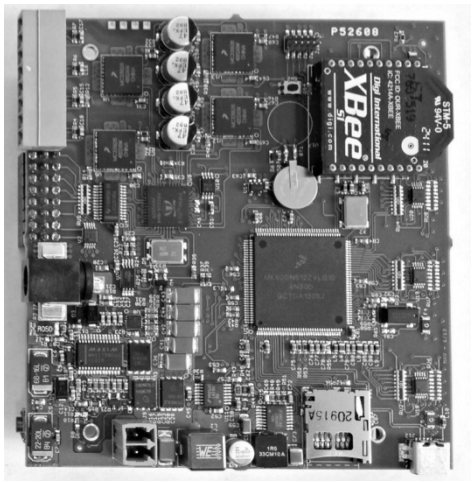


Figure 3: (a) Front side of Proteus PCB, and (b) Back side of Proteus PCB.

digital and analog functionality. This method provides a digital output, or an analog input without loss of performance. Additionally, these pins are connected directly to the K60 processor allowing additional multiplexed functionality such as serial communication to more complex sensors. These thirty-two pins are divided into four banks of eight pins for ease of identification. These sensor input/output pins, as well as all the other peripheral inputs, have easy-to-interface square pin connectors located on the outside of the case, as shown in Figure 4.

A 3.5 inch touchscreen LCD is used as the primary output for user feedback. The selected display is 320 x 240 pixels with 18-bit color and was purchased as an integrated display module. This module contains circuitry to handle the low level, timing critical control of the display and provides a higher-level parallel command interface consisting of five control lines and eighteen data lines. This interface allows refreshing of individual pixels or large regions as needed. In addition to the using LCD to print standard debug messages, the high screen resolution and fast refresh rates have allowed allows all thirty-two pins to be input, users to implement new functionalities such as the real-time graphing of sensor measurements and other more complex graphical feedback.

The integrated display module provides a serial interface for processing the touch point on the touch screen. Due to sourcing issues, not all the modules currently in use have a touchscreen, so this functionality has not yet been fully explored. To provide sufficient user interaction capabilities, a small three-button module was designed to connect to one of the four sensor banks.

One drawback of the integrated display module was the provided connection method consisting of two banks of twenty pins at 0.1 inch spacing. Connectors of this type are rather large and would have taken up significant board real estate. The board did however provide a single bank of surface mount pads that were also connected to the control lines. To utilize this connection, a small flex-cable was designed to be soldered to the integrated display module and provide an easier connection to a zero-insertion-force (ZIF) connector as shown in Figure 5.

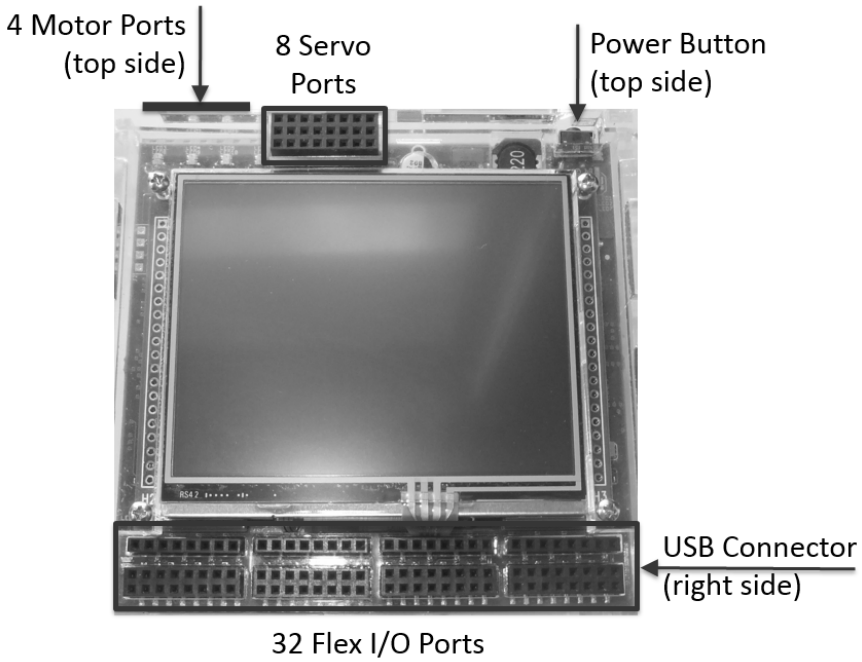


Figure 4: Diagram showing student interface with Proteus 2.0 hardware.

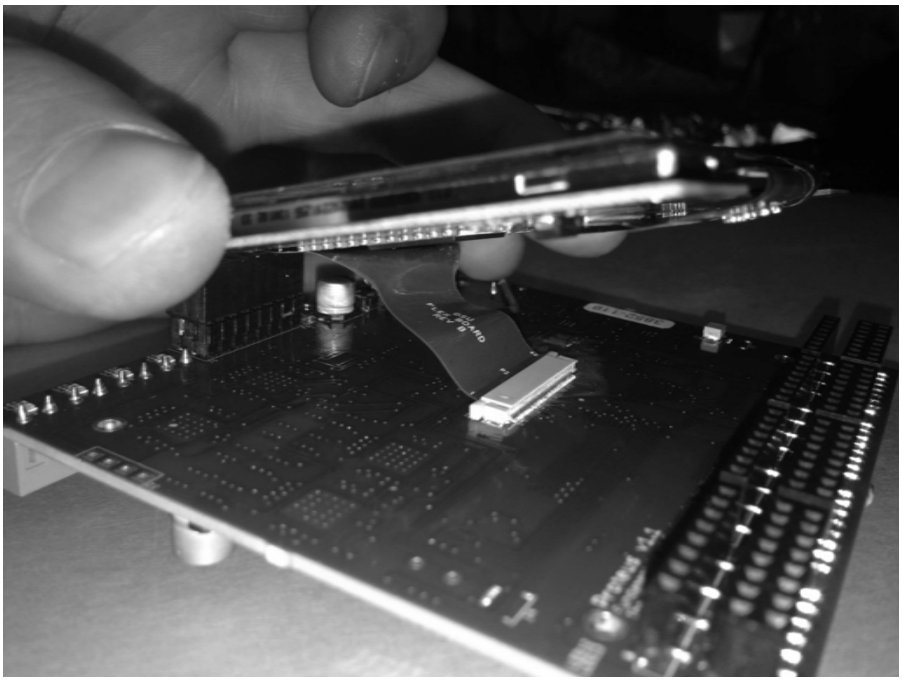


Figure 5: Flex cable connection for the Proteus 2.0 LCD module.

In comparison to the Proteus 1.0, a smaller three-cell lithium ion battery was chosen to reduce the overall device size. The battery module contains additional safety circuitry over the previous battery that would protect from short circuit, over charge, and over discharge. All charging circuitry was integrated into the primary circuit board of the Proteus. This new configuration provides a 3.5 hour maximum charge time from full discharge. Due to the battery's high capacity and student usage practices, typical charge times are much shorter than with the Handy Board.

Two switching power supplies were designed to regulate the necessary voltages for both the Proteus circuitry and any connected motors and sensors. Switching supplies were chosen over cheaper linear regulators because of their higher power efficiency and lower operating temperatures. A 5.0V supply was designed to supply a maximum of 13A. Cascaded from this, a 3.3V supply was designed to source a maximum of 4A. Almost all of the components used by the Proteus operate at 3.3V including the sensor inputs. Brushed DC motors are powered unregulated from the 11.1V nominal battery voltage while hobby servos are powered via the regulated 5.0V supply.

A push button controller was included to provide push-on, push-off power button operation as well as additional safe guards for ensuring that the battery was not discharged below its minimum capacity. This device requires intervention from the primary processor after the button is pressed to initiate a change in the power state.

The Proteus is packaged inside a multi-piece acrylic case. The clear acrylic allows students to see the inner workings of the device while still protecting the sensitive components from abuse. This sandwich-like design consists of several pieces of 0.25 and 0.125 inch acrylic laser cut and layered upon one another in an interlocking fashion. These pieces are held together using eight screws and four standoffs that traverse the entire thickness of the Proteus. Figure 6 shows an exploded view of the case.

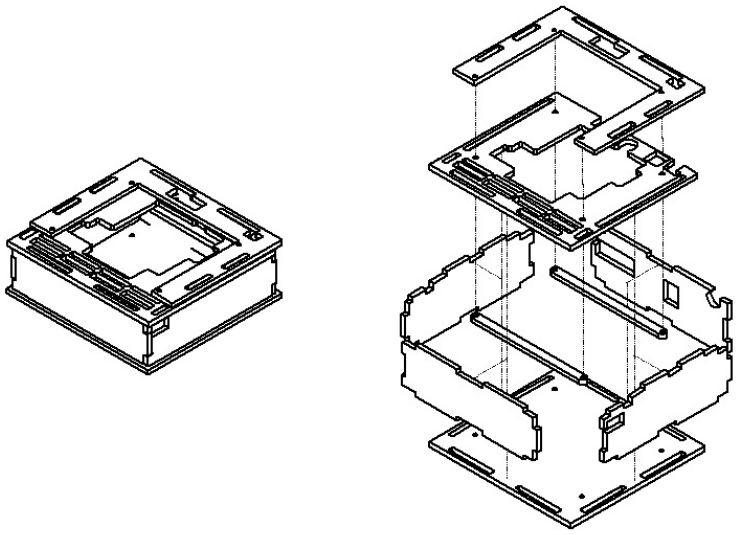


Figure 6: Exploded view of acrylic case.

Software Environment

The Qt Creator integrated development environment was customized for Proteus application development just as it was during ENG 694. To minimize the student learning curve, an installer was provided to automate the process of installing and configuring the necessary software components in this customized development environment. The environment allows a new programming project to be created, written, and deployed through a simple graphical interface without using the command line. This streamlined development environment allows new users to quickly become productive. Qt Creator also provides other features to accelerate development for new users. For instance, Qt Creator provides context-specific code completion, which lowers the learning curve of working with new libraries. By speeding up the software design process, students spend less time in development and have more time to test and validate their robot designs.

A custom, object-oriented application programming interface (API) was developed to allow students to interact with the hardware more intuitively than was previously possible in Interactive C. With these new libraries, students are able to concentrate on high-level program logic instead of low-level detail, which encourages them to develop more robust, elegant code.

In ENG 694, significant time was spent distributing frequent API updates to the users. This time consuming process was automated for Proteus 2.0 using the Git version control system [12]. A publicly accessible Git repository was created to host the API and other supporting files. During code compilation in Qt Creator, a fresh, up-to-date set of libraries is automatically retrieved and included in the current software build. This process enables any bug fixes to be rapidly and seamlessly deployed to all users.

The Qt Creator environment also automates the process of deploying compiled binaries to the device. To enable this functionality, a small bootloader program resides on the Proteus, which allows the device to be recognized as a USB mass storage device by the host computer. To deploy their application, the students place the device into a bootloader mode, and then click a single button in Qt Creator to compile their code and deploy the binary to the Proteus via USB. After a device reset, the recently loaded program is executed.

Final Fabrication and Packaging

A single four-layer printed circuit board was designed in house using Eagle PCB [13]. Initial layout was contracted to an electronic facility at

The Ohio State University, but final layout was performed in-house allowing for closer inspection of the component interconnections before the boards were manufactured. The layout of this board was optimized down to 3.6 by 3.8 inches in order to maintain a small footprint for the student robot designs.

Circuit boards were fabricated by an off-site vendor in panels of four. A local PCB assembly house was used to populate the boards with components. The proximity of the shop allowed the Proteus developers to work more rapidly to solve problems and exchange parts than with other, more distant assembly houses that may have provided cheaper services.

Initially, the Proteus was going to be packaged in a custom injection-molded case. Though the individual part cost of the case was inexpensive at approximately $10 per case, the tooling investment of approximately $15,000 was too large to make case design iterations feasible. Rapid prototyping through 3D printing was explored. This method provided low-cost design iterations but did not decrease in price as more cases were produced. Long-term cost for this method proved to be just as high as injection molding. Laser cutting the case provided much lower costs and decreased production time as a laser cutter was available on-site, but the material limitations of flat sheets of acrylic complicated the final design for a three-dimensional case.

Feature and Cost Overview of the Proteus 2.0

Table 3 shows the capabilities of the Handy Board versus the Proteus 2.0. The new Proteus design significantly outperforms the Handy Board in all categories, due largely to improved electronics and processor technology since the initial release of the Handy Board. Still, the Proteus design maintains the small footprint required for the robot design project and has a total, "home-built" cost around $255. This price is less than the Handy Board, which costs over $300 commercially.

A more detailed unit cost breakdown is provided in Figure 7. The PCB fabrication and

Table 3: Handy Board versus Proteus 2.0 Feature Comparison.

	Proteus 2.0	**Handy Board with Expansion Board**
Main Processor	Kinetis K60 ARM Cortex M4, 96 MHz	Motorola 68HC11 microprocessor, 2 MHz
	128 KB RAM	32 KB of battery-backed RAM
	512 KB Flash	
Propeller Processor	8 cores, 100 MHz, 32 KB RAM, 64 KB Flash	
I/O Ports	32 Flex I/O, individually programmable for analog input, digital input, or digital output	7 analog and 9 digital inputs. Expansion board provides additional: 8 digital outputs, 10 analog inputs.
Motors	4 bidirectional DC motor ports run on battery voltage, 5A max current per motor, 7A total system power limit.	4 bidirectional DC motor ports run on battery voltage, 1A max per motor.
Battery	11.1V, 2600mAh Lithium ion	9.6V, 850mAh NiCad
Servos	8 servo outputs. Runs at 5V DC, can double as 5V general-purpose output.	1 servo output. Expansion board allows for 6 total servo outputs.
LCD Screen	320 x 240 pixel, 18 bit color, graphic LCD with resistive touch screen.	16 x 2 character LCD screen.
Programming languages	C, C++	Interactive C

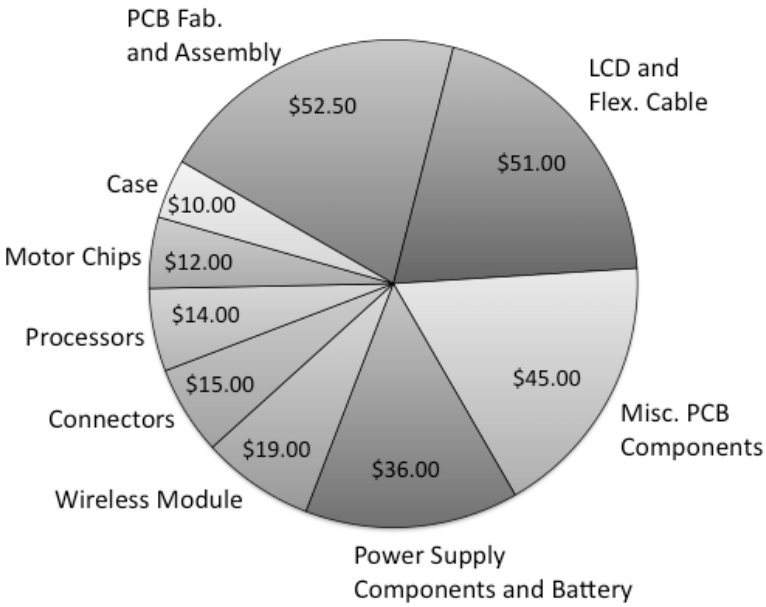


Figure 7: Cost breakdown for the Proteus.

out-of-house assembly make up the largest portion of the total cost. The large LCD and custom flex cable are the next most expensive components. Main components such as the ARM and Propeller processor chips, motor chips, and the case are less expensive.

Testing and Troubleshooting Methods

An in-house test procedure was developed to verify that each Proteus worked properly before delivery to students. Following a one-time initialization process, a custom test program was downloaded to the Proteus via the bootloader. This application provided debug information through the LCD in order to test each DC motor port, servo motor port, and all operation modes for the flexible I/O sensor ports. Displaying this information also verified that the integrated display module was properly connected inside the device.

These tests were performed on the bare circuit boards without any casing. If all of the tests were passed, the device was assembled in a final case with a permanent battery and display module. After assembly, the board was powered on to verify proper connections between modules and that no other problems had been introduced during this process.

Results and Discussion

The Proteus 2.0 design was successfully piloted in ENGR 1282H during spring 2013. This section describes the results of this medium-scale launch and highlights lessons learned that have led to minor modifications to the 2.0 design. Additionally, this second pilot allowed the student performance with the Proteus to be directly compared to the Handy Board. While the test group is too small to draw statistically significant conclusions, the results encourage further use of the Proteus controllers.

Proteus 2.0 Pilot

During spring 2013, the Proteus 2.0 controller was deployed to teams competing in the robot design project in ENGR 1282H. Each team of four students was given nine weeks to complete the design project before a final robot competition. Due to production delays with the Proteus, all teams began the project using the Handy Board. After the fifth week, all teams were given the option to enter a lottery to swap their Handy Board for a Proteus. Students were cautioned that this was a pilot program and that there would be limited support, due to the instructional staff's lack of experience with the Proteus. The students were provided the

following potential benefits and disadvantages of switching to the Proteus:

Benefits—
- Increased battery life and faster charging
- Better display
- Increased motor power
- More memory
- Better servo performance
- More sensor ports

Disadvantages—
- New and less tested
- Instructional staff less familiar with the controller
- Requires a different user programming environment
- Need to modify source code
- Windows-only for Spring 2013
- Different size than Handy Board

If a team received a Proteus, they were required to transition completely away from the Handy Board within 2.5 weeks, by the end of the 8th week. Student reactions were varied. Some decided to switch due to frustration with the Handy Board and a desire for more testing time provided by the increased battery capacity. Other teams were happy with their robot's performance as controlled by the Handy Board and did not want to add a new variable into their design. Despite these limitations and challenges, 59 of the 86 robot teams selected to enter the lottery, and in the middle of the 6th week of the term, 44 teams received the new controller. The other half of the teams continued to use the Handy Board, which allowed for a direct performance comparison.

Student Reception

Students rapidly became acquainted with the Proteus syntax, an improvement over the more limited Interactive C language. Moreover, their object-oriented programs were easier to troubleshoot from both the student and teaching staff perspectives. Students reported being able to test and implement changes many times faster with the Proteus as compared to the Handy Board.

For comparison, Table 4 shows a simple navigation function for a robotic vehicle using Interactive C syntax for the Handy Board alongside the object-oriented code for the Proteus. Both implementations command the robot's motors to drive a certain distance forward as measured by wheel shaft encoders. The logic to accomplish this function has been greatly simplified, in comparison to other more robust implementations, to keep the code manageable in length and to focus on the differences between APIs. Through the incorporation of object-oriented programming, the code for the Proteus is more readable than the Handy Board. In addition, the Interactive C code includes additional type-casts, since Interactive C does not include automatic type casting. The use of a full C/C++ compiler for the Proteus programming provides automatic type casting and simplifies the code.

Despite the intuitive interface to the Proteus APIs, the 2.5-week code transition period was still a rapid change for the students that switched from the Handy Board to the Proteus controller. To assist with this transition, Proteus API documentation was provided through an internally-hosted wiki. This web application provided a page that could be updated by the instructional staff to provide API specifications and examples of use as well as answers to frequently asked questions. Specific code examples were provided that demonstrated equivalences between the Handy Board and Proteus APIs. A forum was also available for students to develop collaborative solutions to any challenges they encountered. The students and staff found this dedicated and fluid source for Proteus API documentation very helpful.

Table 4: Handy Board versus Proteus Code Comparison.

Handy Board Code for Drive Function

```
/* Shaft encoder libraries*/
#use fencdr4.icb
#use fencdr6.icb

/* Function definition */
void GoStraight( int distance, int power )
{
  /* Declare variables */
  int maxcounts;

  /* Set thresholds for shaft encoders */
  encoder4_low_threshold = 30; /*left*/
  encoder4_high_threshold = 60;
  encoder6_low_threshold = 30; /*right*/
  encoder6_high_threshold = 60;


  /* Calculate necessary counts */
  maxcounts = (int)((float)distance * 45.0
      / 3.14 / 2.75 );

  /* Reset shaft encoder counts */
  encoder4_counts = 0;
  encoder6_counts = 0;

  /* Turn on motors */
  motor( 0, power );
  motor( 2, power );

  /* Loop until maxcounts is reached */
  while( encoder4_counts < maxcounts ||
         encoder6_counts < maxcounts )
  {
      msleep( 100L );
  }
  /* Stop motors */
  motor( 0, 0 );
  motor( 2, 0 );
}
```

Proteus Code for Drive Function

```
// Motor and shaft encoder libraries
#include <Motor.h>
#include <IO.h>

// Create motor and encoder objects
Motor left_motor( Motor::Motor0 );
Motor right_motor( Motor::Motor1 );
Encoder left_encoder( IO::P1_0 );
Encoder right_encoder( IO::P1_1 );

// Function definition
void GoStraight( int distance, int power )
{
  // Set thresholds for shaft encoders
  left_encoder.SetThresholds( 2.5, 3.1 );
  right_encoder.SetThresholds( 2.5, 3.1 );

  // Calculate necessary counts
  int maxcounts = distance * 45.0 / 3.14 /
       2.75;

  // Reset shaft encoder counts
  left_encoder.ResetCounts();
  right_encoder.ResetCounts();

  // Turn on motors
  left_motor.SetPower( power );
  right_motor.SetPower( power );

  // Loop until maxcounts is reached
  while( left_encoder.Counts() < maxcounts ||
         right_encoder.Counts() < maxcounts )
  {
      Sleep( 100 );
  }
  // Stop motors
  left_motor.SetPower( 0 );
  right_motor.SetPower( 0 );
}
```

Second Pilot Final Competition Results

While the online documentation assisted students, teams with the Proteus controller were still at a disadvantage in terms of instructional support. In addition, the limited time frame to switch devices remained a challenge for some teams. However, despite these challenges the Proteus teams performed as well as the Handy Board teams in terms of the scores they received during their final competition. The robot competition for ENGR 1282H had two sets of scores used to examine the success of teams. The first score, the grade score, was based on the completion of basic tasks and was used to calculate the team's grade from the competition. The second score, the total competition score, included bonus points for completing extra challenge tasks beyond what was required for the grade score. The grade and total competition score distributions for the two controllers are shown in Figure 8. Because the distributions are non-normal, a Wilcoxon rank-sum test [14] was used to determine if the score differences were statistically significant. The averages, standard deviations, and p-values are given in Table 5. While the averages are slightly higher for the Proteus, the p-values indicate that these differences cannot be considered statistically significant [14].

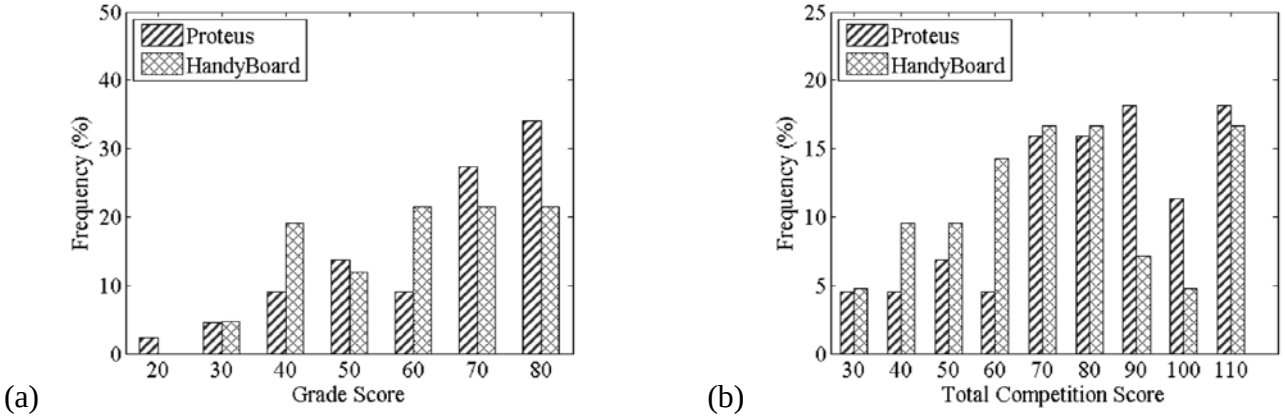


Figure 8: Comparison of robot competition scores for the two controllers: (a) Grade Scores, and (b) Total Competition Scores.

Table 5: Handy Board versus Proteus Score Comparison.

	Handy Board Average	Handy Board Standard Deviation	Proteus Average	Proteus Standard Deviation	p-value Handy Board vs. Proteus
Max Grade Score (out of 80)	63.05	14.16	66.73	14.74	0.1451
Max Competition Score (out of 110)	76.95	22.91	84.96	21.47	0.0926

While the obvious goal is to have a controller that outperforms the Handy Board, the limitations imposed upon the groups that were using the Proteus likely limited their performance. As a result, it was promising that the Proteus teams did not underperform compared to the Handy Board teams. Additionally, some of the advantages available in the Proteus software, including increases in memory and the ability to program in C++, could not truly have been taken advantage of by many of the teams that made the switch. Because those teams started out with the more limited Handy Board and then, transferred their programs to the Proteus, the programs were designed to accommodate these limitations. Also, the reduced experience by the instructional staff with the Proteus impacted these teams. The Handy Board had been used for this project for over fifteen years, so all teaching assistants and instructors were familiar with the Handy Board, but only a few teaching assistants that had worked on Proteus development had significant exposure to the new device. Therefore, the Proteus teams were forced to be more independent.

Lessons Learned

Throughout the Proteus 2.0 pilot, there were numerous lessons learned which led to small improvements for the Proteus. A number of these improvements have been released for the spring 2014 term. In the spring 2013 pilot, many issues arose from the board not being resilient enough to student use. There were several instances where the power button or USB connector broke off due to students being less cautious under the stress of the competition. For these components, the areas on the acrylic case around these components were modified to better secure them. UV-cure epoxy was also added to better affix the button and USB connector to the printed circuit board. Additionally, case modifications were made in order to allow for partial and fast disassembly, which allowed for more rapid repairs. Figure 9

shows an exploded view of the updated case design.

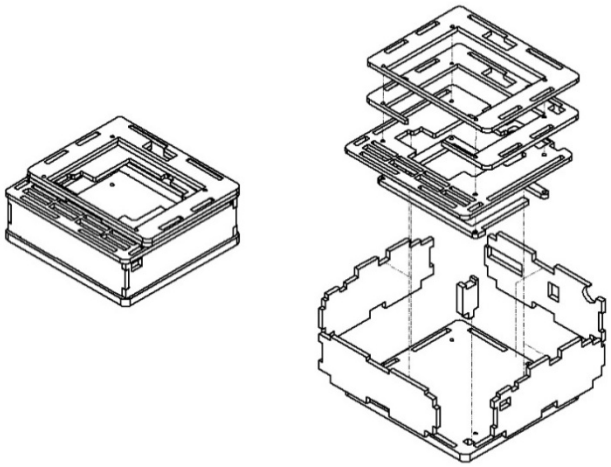


Figure 9: Exploded view of case modifications.

The repeated full disassembly of the Proteus also had negative side effects on the LCD module. There were instances of the custom flex cable attached to the LCD breaking loose after several assembly/disassembly cycles. To remedy this, the flex cable was redesigned to provide a better solder connection to the integrated display module. UV-cure epoxy was also added between the cable and the module to further strengthen the joint.

Only one major issue with the circuit design was found during the Proteus 2.0 Pilot. Several analog sensors were observed to have decreased dynamic range from that measured with the Handy Board. After investigation, this issue was attributed to the on-board pull-up resistors attached to the sensor input lines. The executed solution was to remove these resistors in favor of having the students wire an appropriately valued pull-up resistor to their sensor when needed.

Due to the ability to provide software updates in-situ, minor software changes were implemented during the pilot. The embedded environment was much less forgiving than standard PC application development so an inefficient algorithm or small memory leaks in the students' code could cause serious issues for the robots. The resulting difficulties encountered by students, often falsely attributed to errors in the Proteus API itself, were the primary complaint seen during the term.

The current programming method, using a USB mass storage device in a user-entered bootloader mode, is an area for improvement. Issues arose with the device not always entering the proper mode or with the incomplete program flashing if the students were to remove the USB connection too early. A new bootloader and downloading process is in development to introduce a more fault tolerant design that should resolve these issues.

Summary and Conclusions

This paper detailed the design of the Proteus robotics controller, a controller designed to be a modern replacement for the MIT Handy Board. The pilot of an initial design of this device illuminated important practical drawbacks of the design that were addressed in the subsequent revision. The updated design, Proteus 2.0, was based around an ARM Cortex M4 processor which allowed for the majority of the peripherals to be directly connected rather than handled by more complex interconnections. The design also features a secondary processor, an 8-core Parallax Propeller, which interfaces with all the motor driver circuitry. The other main design features including a large touch screen LCD, modern battery charging circuitry, and custom acrylic case have also been detailed. These components provide a capable, integrated, and compact robotics controller that was pilot-tested by some students as part of ENGR 1282H in spring 2013 and rolled out to all students in spring 2014.

In a direct comparison, the students using the Proteus in ENGR 1282H performed slightly better than the students with the Handy Board despite a number of challenges that limited their performance including reduced working time and limited instructional support. Although this initial release of the Proteus 2.0 has led to a few small updates, it is anticipated that this controller will serve the FEH program for many years to come. Following more thorough

assessment in-house, this controller may be suitable as a basis for other design courses that use similar devices. The Proteus may also be beneficial to new design courses beyond the first year, as no commercially available device exists with similar capabilities at the current time.

References

1. Martin, F.G., *Robotic Explorations – A Hands-On Introduction to Engineering*, Prentice Hall, New Jersey, 2001.

2. Martin, F.G., "The Handy Board", Internet: http://handyboard.com/hb/, (Accessed December 2013).

3. Freuler, R.J., A.W. Fentiman, J.T. Demel, R.J. Gustafson, and J.A. Merrill, "Developing and Implementing Hands-on Laboratory Exercises and Design Projects for First Year Engineering Students", *Proceedings of the 2001 American Society for Engineering Education Annual Conference*, Albuquerque, New Mexico, June 2001.

4. Demel, J.T., R.J. Gustafson, A.W. Fentiman, R.J. Freuler, and J.A. Merrill, "Bringing About Marked Increases in Freshman Engineering Retention", *Proceedings of the 2002 American Society for Engineering Education Annual Conference*, Montreal, Canada, June 2002.

5. Demel, J.T., R.J. Freuler, and A.W. Fentiman, "Building a Successful Fundamentals of Engineering for Honors Program", *Proceedings of the 2004 American Society for Engineering Education Annual Conference*, Salt Lake City, Utah, June 2004.

6. Freuler, R.J., M.J. Hoffmann, T.P. Pavlic, J.M. Beams, J.P. Radigan, P.K. Dutta, J.T. Demel, and E.D. Justen, "Experiences with a Comprehensive Freshman Hands-On Course – Designing, Building, and Testing Small Autonomous Robots", *Proceedings of the 2003 American Society for Engineering Education Annual Conference*, Nashville, Tennessee, June 2003.

7. Texas Instruments, "C Series for Connected MCUs – LM3S Cortex-M3 Series – LM3S8962 – TI.com", Internet: http://www.ti.com /product/lm3s8962, (Accessed January 2014).

8. Parallax Semiconductor, "Propeller | Parallax Inc.", Internet: http://www.parallax.com /microcontrollers/propeller, (Accessed January 2014).

9. Digi International Inc., "Digi XBee Wireless RF Modules", Internet: http://www.digi.com /xbee, (Accessed January 2014).

10. Qt Project Hosting, "Qt Creator", Internet: http://qt-project.org/wiki/Category:Tools: QtCreator, (Accessed January 2014).

11. Freescale, "Kinetis MCUs based on ARM Technology", Internet: http://www.freescale. com/kinetis, (Accessed January 2014).

12. Anon., "Git", Internet: http://git-scm.com, (Accessed January 2014).

13. Newark, "CadSoft EAGLE PCB Design Software", Internet: http://www.cadsoftusa. com/eagle-pcb-design-software, (Accessed January 2014).

14. Gibbons, J.D., and S. Chakraborti, *Nonparametric Statistical Inference*, 4th Ed., Marcel Deeker Inc., New York, 2003.

Biographical Information

Michael A. Vernier is a Graduate Research Associate at The Ohio State University. He received his B.S. degree in Electrical and Computer Engineering from The Ohio State University in 2007, and his M.S. degree from The Ohio State University in 2010. Currently, he has been working toward a Ph.D. in Electrical and Computer Engineering at Ohio State with a focus on robot autonomy and intelligent transportation. Formerly, a Graduate Teaching Associate, he was heavily involved in teaching and content development with the Fundamentals of Engineering for Honors (FEH) Program.

Patrick M. Wensing is an NSF Graduate Research Fellow and Graduate Teaching Associate at The Ohio State University. He received his B.S. degree in Electrical and Computer Engineering from The Ohio State University in 2009. He received his Ph.D. in Electrical and Computer Engineering at Ohio State in August 2014. He currently teaches

and develops content for the laboratory portion of the Fundamentals for Engineering for Honors Program and is actively involved in humanoid locomotion research.

Craig E. Morin is the Engineering Manager at MindWare Technologies in Gahanna, Ohio where he has worked since 2008. He received a B.S. in Electrical and Computer Engineering and an M.S. in Biomedical Engineering at The Ohio State University in Columbus, Ohio. Prior to his current role, he was a Design Engineer with MindWare Technologies and a Graduate Teaching Associate with the Fundamentals of Engineering for Honors Program at The Ohio State University. Beyond product design with an emphasis on electrical hardware, his interests include home automation, 3D printing, and ceramics.

Andrew H. Phillips is an Electrical and Computer Engineering (ECE) student at The Ohio State University and an Undergraduate Teaching Assistant for the OSU Fundamentals of Engineering for Honors program. He is also a member of the inaugural class of the OSU Eminence Fellows full scholarship program. He will graduate with his B.S. in Electrical and Computer Engineering from The Ohio State University in May 2016.

Brian A. Rice is a sophomore in the Department of Mechanical and Aerospace Engineering at The Ohio State University. He is also an Undergraduate Teaching Assistant for the OSU Fundamentals of Engineering for Honors Program for the two-semester FEH engineering course sequence. He will graduate with his B.S. in Mechanical Engineering in May 2016.

Kevin R. Wegman is a third year Chemical Engineering undergraduate student at The Ohio State University who is a teaching assistant for the FEH program in the Engineering Education Innovation Center (EEIC). His roles in the EEIC include lead Undergraduate Teaching Assistant and head of robot course construction. He plans to begin graduate school in Nuclear Engineering at The Ohio State University.

Chris P. Hartle is a Graduate Teaching Associate for the Fundamentals of Engineering for Honors Program at The Ohio State University. Mr. Hartle received his B.S. degree in Electrical Engineering from The Ohio State University in 2014. He is currently pursuing a M.S. degree in Electrical Engineering with specializations in computer systems and control. His interests include small scale positioning systems, used within the Fundamentals of Engineering for Honors program, and flight control systems. He has spent his previous three summers working on helicopter vibration control and brushless DC motor drive applications for military aircraft development.

Paul A. Clingan is a senior lecturer in the Engineering Education Innovation Center (EEIC) at The Ohio State University where he has taught in the FEH Program since 2002. He received both his B.S. (1986) and M.S. (1988) in Chemical Engineering from Bucknell University. Prior to coming to the EEIC, he worked at UOP – Monirex Systems in Des Plaines, IL developing software for the Sorbex family of control systems. In addition, he has taught in the London Honors Study Abroad program, and currently serves as a project team advisor in the Senior Multidisciplinary Capstone program, and as an advisor to students in the Second-year Transformational Experience Program (STEP).

Krista M. Kecskemety is a senior lecturer in the Engineering Education Innovation Center at The Ohio State University. She received her B.S. in Aerospace Engineering at The Ohio State University in 2006 and received her M.S. from Ohio State in 2007. In 2012, she completed her Ph.D. in Aerospace Engineering also at Ohio State. Her engineering education research interests include investigating first-year engineering student experiences, faculty experiences, and the connection between the two.

Richard J. Freuler is the Director of the Fundamentals of Engineering for Honors Program in the OSU Engineering Education Innovation Center. He teaches the two-semester FEH engineering course sequence and is active in engineering education research. He is also a Professor of Practice in the Mechanical and Aerospace Engineering Department and conducts scale model investigations of gas turbine installations for jet engine test cells and for marine and industrial applications of gas turbines at the Aeronautical and Astronautical Research Laboratories at Ohio State. He earned his Bachelor of Aeronautical and Astronautical Engineering (1974), his B.S. in Computer and Information Science (1974), his M.S. in Aeronautical Engineering (1974), and his Ph.D. in Aeronautical and Astronautical Engineering (1991) all from The Ohio State University.