An Embedded Systems Laboratory to Support Rapid Prototyping of Robotics and the Internet of Things

James O. Hamblen, *Senior Member, IEEE*, and Gijsbert M. E. van Bekkum, *Member, IEEE*

***Abstract*—This paper describes a new approach for a course and laboratory designed to allow students to develop low-cost prototypes of robotic and other embedded devices that feature Internet connectivity, I/O, networking, a real-time operating system (RTOS), and object-oriented C/C++. The application programming interface (API) libraries provided permit students to work at a higher level of abstraction. A low-cost 32-bit SOC RISC microcontroller module with flash memory, numerous I/O interfaces, and on-chip networking hardware is used to build prototypes. A cloud-based C/C++ compiler is used for software development. All student files are stored on a server, and any Web browser can be used for software development. Breadboards are used in laboratory projects to rapidly build prototypes of robots and embedded devices using the microcontroller, networking, and other I/O subsystems on small breakout boards. The commercial breakout boards used provide a large assortment of modern sensors, drivers, display ICs, and external I/O connectors. Resources provided include eBooks, laboratory assignments, and extensive Wiki pages with schematics and sample microcontroller application code for each breakout board.**

***Index Terms*—Design project, embedded systems, mechatronics, microcontroller, microprocessor, networking, robotics, real-time operating system (RTOS).**

I. Introduction

FORECASTERS have predicted that the robotics industry will undergo exponential growth [1], becoming a $66 billion industry worldwide by 2025 [2] as a result of the rapid advances in the enabling technologies, which include computer hardware, artificial intelligence (AI), vision, energy storage, actuators, and sensors. For many robotics applications, networking is now critical. Over half of the new devices announced at the 2012 Consumer Electronics Show (CES) featured Internet connectivity. It is estimated that there are currently 9 billion connected devices, and that there will be 24 billion connected embedded devices by 2020 [3]. Embedded devices already accounted for over 98% of the world's processors in 2003 [4].

A course dedicated to embedded systems design is typically found at the junior or senior level in Electrical Engineering (EE), Computer Engineering (CmpE), and even in some systems-oriented Computer Science (CS) degree programs [5]–[11]. Such a course presents an early opportunity to introduce robotics and networking to students. A number of related topics are covered earlier throughout the undergraduate curriculum [12]. Some schools also utilize small robots in an earlier undergraduate laboratory course, based on MATLAB or LabVIEW, that also introduces microprocessors [13], [14]. Typically, programming, digital logic design and often a computer architecture course are prerequisites for the more advanced embedded systems or microprocessor design course that is the focus of this paper. For software development in the embedded systems industry, the C/C++ family of languages is still used in the large majority of new designs, according to annual industry surveys [15]. Many embedded systems, microcontroller, or microprocessor design courses started out with low-cost 8-bit processors with limited capabilities, but most of the development effort in industry has moved on to modern System on-a-Chip (SOC) 32-bit devices that contain a reduced instruction set computer (RISC) processor with volatile memory, nonvolatile flash memory, and a wide assortment of standard I/O interfaces, all on a single chip. According to annual industry surveys of embedded designers, 70% of new designs now utilize an operating system (OS), and 59% include networking [15]. The widespread development of these new embedded devices with networking has led to the highly touted concept of the "Internet of Things" [16], [17].

Now that a single-chip microcontroller already contains the processor, memory, and numerous I/O interfaces with built-in hardware controllers, it is appropriate to use a higher level of abstraction in such a course. An increased focus can be placed on robotics, networking, the use of existing C/C++ application programming interface (API) libraries to enhance productivity, basic operating system concepts, and rapid prototyping of devices. Less time can be spent on assembly language and lower-level hardware topics. This paper describes the experience gained developing a laboratory to support development of these devices; it will primarily focus on the new technologies used in the student instructional laboratories during the first three offerings of the new course.

II. Robotics and Internet of Things Laboratory

At the Georgia Institute of Technology (Georgia Tech), Atlanta, ECE 4180 Embedded Systems Design is a senior-level elective 3-h lecture course with a 1-h laboratory. Students have previously taken a Digital Logic Design class, Introduction to Computer Architecture, and C/C++ programming. A laboratory-based learning model is ideal for this course, and it is also an excellent place in the undergraduate curriculum to introduce

Manuscript received June 18, 2012; revised September 14, 2012; accepted October 11, 2012. Date of publication December 11, 2012; date of current version January 30, 2013.

J. O. Hamblen is with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: james.hamblen@ece.gatech.edu).

G. M. E. van Bekkum is with Google, Inc., Mountain View, CA 94043 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TE.2012.2227320

robots, networking, sensors, and team-based student design projects. An electronic textbook and numerous Web-based resources were developed for the course.

A new low-cost 32-bit microcontroller module with networking support is used in the student laboratory assignments for the first half of the semester. These early laboratory assignments are the focus of this paper. The remaining laboratory assignments in the course focus on higher-end embedded devices and are currently based on a low-power Intel Atom X86-based computer board that is similar to low-end PCs [18]. All materials for the laboratory assignments are provided on the Web and are updated each semester [19].

Rapid advances in technology force instructors to frequently update embedded system courses. Selection of the hardware and software for a student laboratory is always a complex decision. It also requires significant curriculum development effort and funding to update the laboratory with new technology. By adopting the some of the recent approaches being used in industry, students should be more productive and more rapidly able to produce prototypes of robots and other complex embedded devices.

PIC and the Atmel AVR used in Arduino boards are popular in many existing embedded system, microcontroller, or microprocessor design courses for historical reasons [20], [21], but 32-bit ARM RISC processors are by far the most widely used processors in new designs for embedded devices [15]. Embedded designers report that 61% of new designs use 32-bit processors for the main processor. It has been estimated that around 80% of the 32-bit processors are ARM-based [15]; they are found in virtually every cellphone and many other battery-operated devices. Power consumption has also become a major design consideration and is one of the main reasons for the wide adoption of ARM processors.

Numerous options were considered for the laboratory computer boards. The popular 8-bit PIC and Arduino boards have limited memory, low clock rates, and lack on-chip support for networking. Newer Arduino shield boards or small system boards such as the Libelium Waspmote can add selected I/O features along with wired or wireless networking, but they also significantly increase cost. Shield-style boards are also a bit more difficult to use with breadboards when adding custom hardware for student laboratory projects.

New technology, small, low-cost 32-bit ARM-based boards are also now available in the same price range; these offer significantly increased memory, higher clock rates, and performance along with greatly expanded on-chip I/O hardware features including networking. In addition to hardware advantages, several of these newer boards also provide a complete ISO C/C++ software development environment, more comprehensive higher-level C/C++ I/O APIs, and optional support for a real-time operating system (RTOS).

A number of higher-power and higher-performance boards with ARM and graphics processors are also available, such as the Beagle board, Panda board, and the Raspberry Pi. Mini ITX boards with Intel X86 Atom processors are another option. This approach is too costly and overpowered for many low-end embedded device product applications that are now appropriate for new SOC microcontrollers; these boards are also a bit more difficult for students to use when breadboarding prototypes with additional custom hardware. They support more demanding embedded applications and would be more appropriate for use in later courses or laboratory projects that require more memory, processing power, a graphical user interface, and an operating system.

The software tools available and the quality of the documentation are often more important than the hardware when selecting a board for use in a student laboratory. After examining both the hardware and software ecosystems, the ARM-based mbed module was selected for its C/C++ API support, available documentation, software tools, low cost, and ease of use for students building custom prototypes on breadboards. An internal research project at ARM to make the embedded development process easier both in industry and at schools led to the development of the mbed module [22]–[25]. The small 40-pin mbed module contains an NXP LPC1768 SOC processor that plugs into a standard student solderless breadboard. The LPC1768 contains a 32-bit 96-MHz ARM Cortex M3 processor, 64 kB RAM memory, 512 kB Flash memory, and an Ethernet network controller. Additional on-chip I/O interfaces include a real-time clock, timers, the universal serial bus (USB), a serial peripheral interface (SPI), interintegrated circuit communication (I2C), general purpose I/O (GPIO), analog-to-digital conversion (ADC), digital-to-analog conversion (DAC), an RS-232 serial interface, a controller area network (CAN), and pulse width modulation (PWM). A USB cable provides power, or an external 4.5–9-V battery or dc supply can be used instead. The USB interface can download code and also function as a virtual com port allowing mbed C/C++ programs to perform standard I/O such as "printfs" or "scanfs" using any terminal application running on the PC. It contains a flash accelerator and code runs from flash with RAM being used for data. In production quantities, the LPC1768 SOC IC costs around $5. The price of the assembled mbed module with a USB cable and user account is about half the cost of most college textbooks.

III. Cloud-Based C/C++ Software Development

There are several commercial and open-source options for ARM C/C++ compilers. The most widely used and supported software development tool for mbed is the cloud-based C/C++ compiler [24]. It runs on any PC platform with a Web browser and is extremely easy for students to use.

A. Cloud-Based Compiler

The cloud-based compiler for the mbed module is based on the commercial Keil Tools C/C++ compiler. After logging onto an mbed user account, the cloud-based compiler can run in any Web browser as seen in Fig. 1.

The mbed module attaches just like a 2-MB USB flash drive, and a simple mouse click can save new executable files to program the device. When the reset pushbutton is depressed on the mbed module, or power is cycled, the bootloader runs the most recent executable file from flash. The mbed.org server hosts all student source files and documentation. This means that there is no software to install or maintain, and development can effortlessly move between any machines equipped with a Web browser. Instructors can get free passwords for students to setup

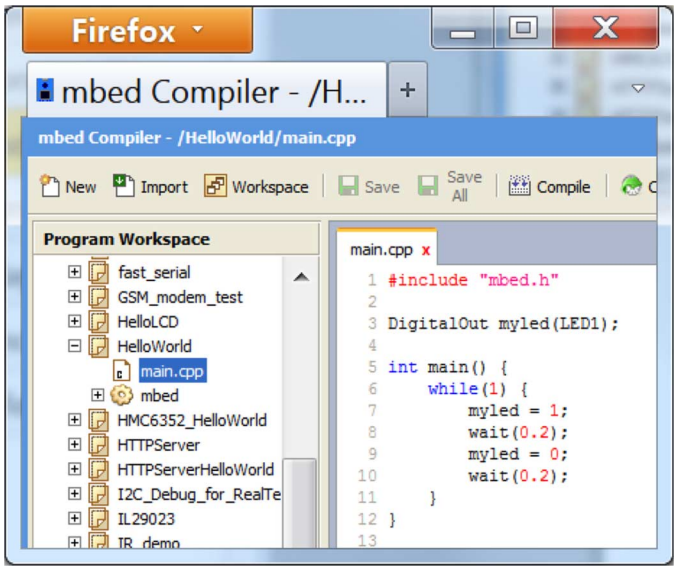

Fig. 1. The C/C++ cloud compiler for mbed runs in any Web browser, so development can move between computers at need. It is based on the industry standard Keil tools compiler.

an account on the cloud compiler with space for file storage via e-mail [25], so students can easily work on their projects in the lab, at home, or even any location with WiFi. Most students can have a "hello world" application running on mbed in under 5 min. Other more traditional offline tool chains are also supported, but they require more initial setup time for students.

B. C/C++ Object-Oriented I/O APIs

The C/C++ object-oriented I/O API support provided for the mbed module is also innovative. Network drivers, basic file system drivers, and easy-to-use APIs were developed for the NXP1768's on-chip I/O features. These add higher-level API support for networking, files, PWM, SPI, I2C, Analog I/O, timers, delays, and RS-232 serial ports using simple easy-to-use C++ object-oriented library calls.

Predefined C/C++ pin names for the APIs, *px*, are used to specify the use of individual I/O pins on the mbed module. A typical I/O pin has one of four different programmable I/O functions on the processor. The C++ object-oriented I/O APIs are able to automatically configure multifunction pins and hardware controllers based solely on the pin names (numbers) and the APIs used. Students are able to hook up new I/O device hardware without ever checking the detailed data sheet and user manuals for the processor or writing directly to bits in I/O control registers through the use of the object oriented C/C++ I/O APIs. I/O registers can still be accessed directly using C/C++ or assembly, if so desired.

The mbed Web site contains a number of helpful resources for students in addition to the cloud compiler. Each student account gets a "notebook" area where they can save documentation Web pages about projects. The "handbook" pages contain an online API reference manual for the mbed I/O and OS functions [26]. An extensive Wiki site called the "cookbook" contains documentation and code examples provided by users and a forum where students can post questions [27].

C. TCP/IP Protocol Suite

To support networking and rapid prototyping of the Internet of Things, a large assortment of network clients and servers are also available including a TCP/IP stack, HTTP Web servers, sockets, Web services, and new Websocket support for HTML5. Many of these are available as libraries that can be added with a few mouse clicks to user projects. The module's 2-MB flash can be used for file storage, and if necessary, external storage such as an SD card breakout board or a USB flash drive can be added. Since data memory is a bit limited, it is also likely that only one type of network protocol would be supported at a time on the embedded device, and the number of active connections is limited (unlike larger systems such as a PC).

D. RTOS Support

The development environment includes a basic RTOS based on ARM's new CMSIS 3.0 RTOS and I/O API standard [28]. CMSIS provides a standard interface to numerous other RTOS venders. The mbed RTOS provided free with the cloud compiler supports multiple threads with 1-ms time slice priority-based scheduling, mutexes, semaphores, signals, message queues, and timers. The term thread is used instead of process since the microcontroller does not include a memory management unit (MMU). It is possible to use the basic I/O APIs without calling or including RTOS functions in the main thread. The RTOS API interface is less complex and easier for students to understand than a traditional desktop OS.

IV. Return to Breadboarding

For a number of years, many schools moved away from using solderless breadboards for prototyping, as most new ICs were available only in small surface-mount packages, rather than the older 1/10-in DIP-style IC packages that plug directly into breadboards. Another problematic issue with breadboarding was that the number of wires and time required increased rapidly as system complexity grew. There is some educational value in having students actually build a circuit rather than always using a large preassembled board. A breadboard also allows students to easily add their own custom hardware, which is helpful when changing laboratory experiments and for design projects.

Recently, two factors have combined to once again make breadboards an attractive option to consider for use in student laboratory work. New SOC processors already have sufficient internal memory and I/O interfaces on-chip for many applications. Most external I/O subsystems for embedded devices (i.e., sensors, displays, drivers, and networks) now use one of the standard serial interfaces (i.e., SPI, I2C, RS-232, USB, or Ethernet) that require only a few wires for interfacing. The greatly increased level of hobbyist activity resulting from the current generation of inexpensive single-chip microcontroller kits such as Arduino has provided a vast array of low-cost external I/O sensors and devices preassembled on small printed circuit boards, called breakout boards, that contain newer surface-mount ICs. With breakout boards, no surface-mount soldering is needed, as they have header pins that will plug directly into a standard student breadboard.

TABLE I
POPULAR BREAKOUT BOARDS FOR USE IN STUDENT PROJECTS

Purpose	Sensor or Device	Interface, IC, and comments
Add networking	MagJack RJ45 connector	Use on-chip Ethernet hardware
Wireless data transmission	Zigbee/XBee/WiFi	Serial – XBee XBPxx, WiFly
PWM to drive DC motors	H-bridge driver	PWM – STMicro VNH3SP30
Measure Distance	IR reflection	Analog - Sharp GP2xx
Ultrasonic Range	Sonar	Analog - MaxSonar XL EZx
Motion or orientation	MEMS Gyro/Accelerometer	I2C – ITG322 & ADXL345
Direction	Electronic Compass	I2C - Honeywell HMC6352
Cell phone network for data	Cell Phone modem	RS-232 Serial with SIM - various
Location and speed	GPS receiver	RS-232 Serial NEMA - various
Add flash file system	Micro SD card socket	SPI – various
Add USB thumb drive	USB A connector	USB – various
Display ASCII text	B&W Text LCD Display	Parallel TTL - HD44780
Display text and graphics	Color LCD display	SPI – Nokia 6100 128x128
VGA display on a monitor	VGA controller	TTL Serial – PICASO-SGC
Capture Still Images	JPEG Color Camera	TTL Serial - LinkSprite - LSY201
Mouse and/or keyboard	PS/2 connector	PS/2 serial protocol
Digital audio output	MP3/AAC/WMA decoder	SPI - VLSI VS1033
Touch switch or keypad	Touchpad controller	I2C – Freescale MPR121
Environmental data	Temperature & Humidity	TTL Serial - Sensirion SHT15/21
Display 2^{30} colors on LED	RGB LED & driver	SPI - Allegro A6281 Shiftbrite
Connect to a serial port	RS-232 level convertor	RS-232 Serial - various

To support student design projects, an extensive list of over 100 commercial breakout boards with sensors, displays, drivers, and I/O connectors was developed at Georgia Tech and posted on the mbed Wiki site. Table I lists examples of some of the most popular breakout boards with Wiki C/C++ code that have been used by students in design projects. All have interface wiring details and driver and code examples posted on the mbed cookbook Wiki [27], [29].

A. Breadboarding Using Breakout Boards

A sample student breadboard-based Internet clock project built with breakout boards is seen in Fig. 2. The network cable on the right connects to the Internet, and the USB download and power cable is on the left. Four jumper wires were used to add an Ethernet jack breakout board, and eight for the LCD text display. The program uses the network connection to automatically set the clock using a network time protocol (NTP) server. For wireless networking, a small WiFi breakout board can be used.

B. Breadboarding Low-Cost Robotics Platforms

The small size, power requirements, I/O features and APIs, and ease of use of the mbed module make it ideal for mechatronics and robotics applications. Wiki code examples include motor speed control using H-bridge drivers with PWM, adding PID feedback control using encoders, and numerous sensors that are used in most robotics applications [27]. WiFi network communication links are often used in mobile robots. The majority of the student robotics projects have included wireless networking breakout boards, and many have been controlled or have transferred data using the Internet.

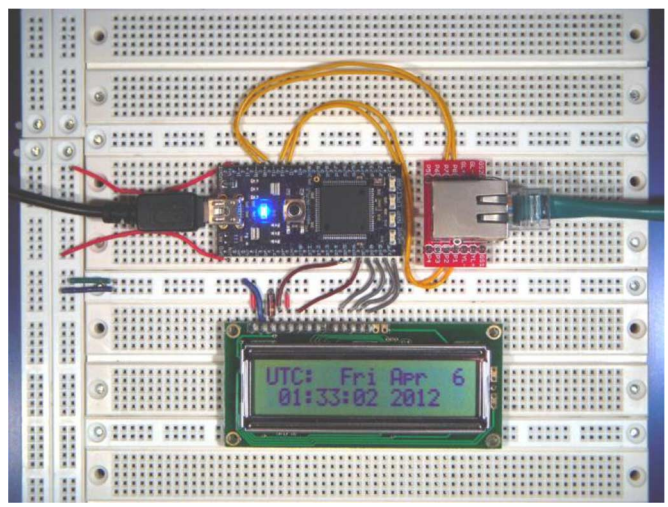

Fig. 2. Internet clock student project built using the mbed 40-pin DIP module (center left) with a text LCD (bottom) and an Ethernet connector breakout board (center right) on a breadboard. The time on the LCD is automatically synchronized using a NTP time server via the Internet [27].

A number of low-cost robotics platforms are available in the laboratory for use in final design projects. Several student robotics projects from the course are seen in Fig. 3. Documentation, YouTube videos, and code examples for each of these robotics projects are posted in the Wiki [27].

V. WIKI-BASED LABORATORY ASSIGNMENTS AND DESIGN PROJECTS

At Georgia Tech, this senior-level class contained a mixture of EE and CmpE undergraduates along with several CS graduate

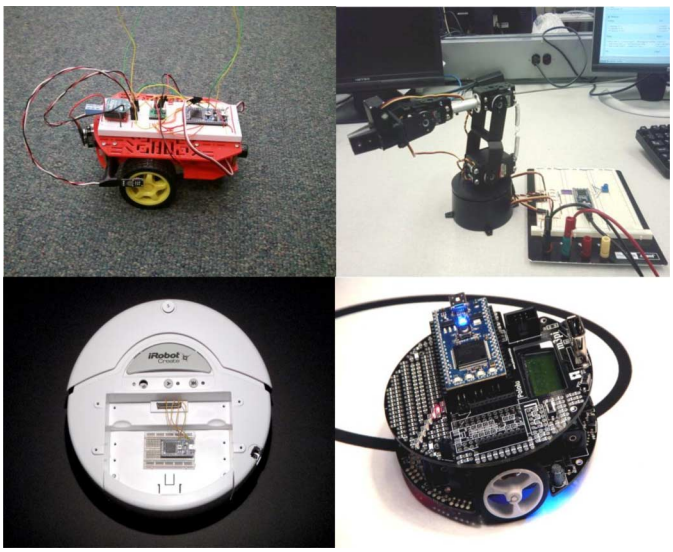

Fig. 3. Robot prototypes built by students using the mbed module. (Clockwise from upper left) Small robot built using a low-cost Sparkfun Magician robot base kit with a breadboard with breakout boards for a dual H-bridge motor driver, IR encoders, Sonar, and wireless networking; a Lynxmotion robot arm kit using five low-cost R/C servos controlled by mbed's PWM outputs that picks up colored blocks using USB end effector position commands sent by a PC; an iRobot Create robot interfaced to the mbed module using a small breadboard; and Pololu's small m3pi robot base with a top level mbed and protoboard area [27].

students. It is a 3-h lecture course with a 1-h lab credit. Enrollment is typically around 50 students, and the course has now been taught for three semesters While students have had previous experience in C/C++ programming and digital hardware, they have had minimal experience from prior coursework in robotics, hardware interfacing, I/O programming, networking, and operating systems. Students work in teams of two on normal laboratory assignments, and design project teams range from two to four.

A. Development of Laboratory Assignments Using the Wiki

The mbed online handbook API reference and cookbook wiki are valuable resources for course instructors developing new laboratory assignments. Hyperlinks to I/O APIs in the handbook and I/O device examples from the cookbook can be added to student assignments [26]. When sensors or I/O devices are selected from the cookbook area, the Wiki page typically provides hardware interface and wiring details, photographs, YouTube videos, links to datasheets, working C/C++ code examples, and commercial sources for the device and breakout boards [27]. These can be used by instructors as source materials to develop laboratory assignments or provided as hyperlinks in short online student laboratory assignments. The Wiki page typically provides students with enough information to incorporate the I/O device in more complex projects containing a mixture of different devices. Coverage of sensors and devices has greatly expanded since 2011, as users from industry and academia worldwide have contributed to the Wiki. Students using new devices are encouraged to post Wiki pages. Wiki code examples are set up using the MIT permissive use license.

B. Course Laboratory Assignments

The mbed module is used for the first two laboratory assignments in the course and is followed by a three-week design project. The course starts out with a basic introductory lab where students use mbed for digital I/O with pushbuttons, use hardware PWM to dim an LED, add an I/O port expander, and use power management to reduce power levels by adapting the C/C++ examples from the mbed Wiki pages. For extra credit, they can add a watchdog timer, or go back and use ARM assembly language instead of C/C++ for the basic digital I/O LED blink demo. Assembly language development is supported using standard $*.s$ files in the cloud compiler [27]. In the second laboratory experiment, students built prototypes on a breadboard using the breakout boards available in the laboratory with a number of different interfaces for robotic sensors and devices by adapting several of the C/C++ cookbook Wiki code examples including RS-232, I2C, SPI, Analog input and output, USB, Ethernet, LCDs, dc motors with H-bridge drivers and servo control using PWM, and an HTTP Web page server.

Each group then demonstrated to the teaching assistant (TA) that each different item was operating correctly on their breadboarded prototype. The mbed's higher-level C/C++ I/O APIs and the Wiki code examples greatly increased student productivity. During this period, the in-class lectures were covering different I/O interface standards, sensors, and drivers. The laboratory assignments reinforced, and expanded on, information presented in the lectures. Several short hardware tutorials and YouTube videos were developed at Georgia Tech to support these laboratory assignments and added to the mbed Wiki pages [27]. With all reference materials being Web-based, hardcopy is rarely used.

C. Student Design Projects

The two introductory labs using mbed were followed by a team-based design project that gave students the freedom to pick the project idea, subject to instructor and TA approval and guidance. Students posted their project documentation using the mbed Web site's notebook feature and were encouraged to include photographs and project demo video clips on YouTube in addition to oral presentations. Several examples of these student design projects built using only commercially available breakout boards and jumper wires are seen in Fig. 4. Additional design projects from the course are documented on the mbed site's student project Wiki pages [27].

VI. Pedagogical Results

To assess the success of the new curriculum and the updated technology for the course, several techniques were used. To date, approximately 150 students have completed the updated course and laboratory during three successive semesters. First, the traditional campus-wide course assessment tool was used in three sections of the course. In all categories, improvements in the Web-based anonymous student survey ratings were demonstrated when compared to the previous version of the course taught in prior years. The previous version of the course had

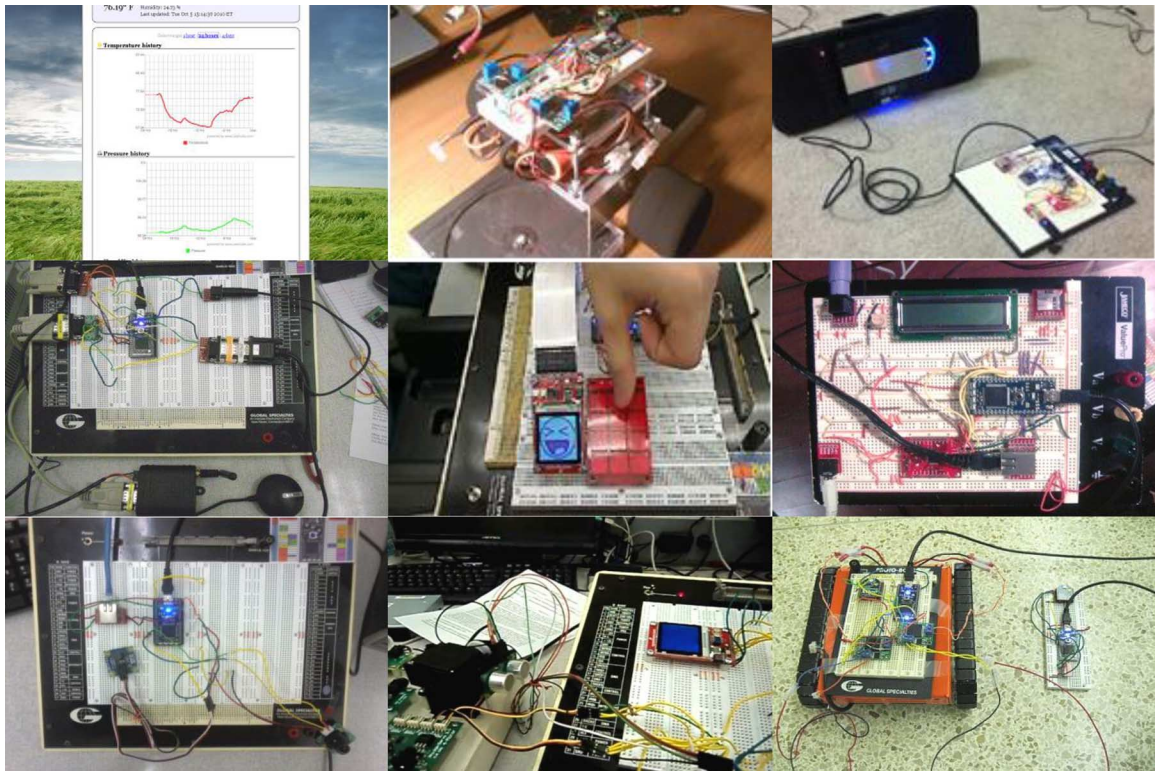

Fig. 4. Student design projects built on breadboards. (Top row, left to right) Environmental sensor readings provided by Web page server; self-balancing two-wheeled robot using MEMS accelerometers and gyros; an Internet radio. (Middle row, left to right) SMS text messages reporting GPS location coordinates using a cell phone modem; a digital photo frame with touch control; universal speech translation using Google's Internet Speech APIs. (Bottom row, left to right) Live camera images from the Web page server; sonar images of nearby objects on an LCD; wireless-controlled treaded robot [27].

TABLE II
AVERAGE STUDENT RESPONSES TO COURSE SPECIFIC SURVEY STATEMENTS

Average	Course Specific Survey Statements and Instructions
	For these questions: 1-strongly disagree … 3-Neutral … 5-strongly agree
4.4	I think the experience with labs that used breadboards with breakout boards was worthwhile.
1.87	I would prefer labs where everything was already connected on a circuit board even though I might have somewhat less flexibility to do different things on projects.
4	I would prefer an mbed design project rather than a third traditional lab assignment using mbed.
3.6	I would prefer the "cloud compiler" web browser approach versus a more traditional development tool that was only available for use on the laboratory PCs. (Assuming they have the same features)
3.8	I would prefer electronic copies of course materials versus traditional printed course materials and printed textbooks. (Assuming content and cost are about the same)
4.87	I prefer a team-based design project with oral presentations over a more traditional final exam.
	For these questions: 1-less time … 3-about right … 5-more time
3.4	How should the amount of time spent on the ARM mbed labs 1 and 2 be altered
3.07	How should the amount of time spent on the ARM mbed design project be altered

overall course ratings ranging from 3.9 to 4.1 out of 5.0. After updating the laboratory portion of the course, the overall course ratings increased to 4.2, 4.3, and 4.44 over the first three terms.

In addition to the traditional campus-wide assessment tool, another anonymous student survey was given with detailed statements tailored to the course materials, laboratory assignments, design project, and the new technologies used in the course. The results are summarized in Table II. Perhaps the most interesting feedback from the course-specific survey questions in Table II was that an overwhelming majority of students prefer using the breadboard approach to hookup devices, even though it requires a bit more time and effort than a preassembled board. A small majority preferred the cloud compiler approach to that of using locally installed software tools. A slightly larger majority preferred textbooks and other course materials to be distributed in an electronic format rather than as hardcopy. An overwhelming number of students were in favor of a team-based design project with oral presentations during the final exam period as opposed to a more traditional written final exam. Finally, average responses from students indicated

that the amount of time spent on laboratory assignments and the design project was appropriate.

In addition to the student-based assessment tools, the course's instructors noted other improvements. The ease of use of the software tools and breadboards enabled inclusion of a greatly expanded number of I/O interface standards, devices, and sensors in the assigned laboratory experiments. This increased student interest in the laboratory projects, increased course enrollments, and enabled the course's lectures and two written tests to cover a wider range of topics and more current design issues. Over 95% of the students were able to complete the breadboard laboratory assignments and build a functional design project in each of the first three offerings of the course. After hands-on experience with a wide range of I/O standards and devices, students typically selected new sensors or devices for use in projects that were not previously used in the assigned laboratory experiments and were able to successfully incorporate them in their design project. Virtually every student project has included several features that would be difficult to develop without the networking and API support provided, given the time constraints of doing this within a single course. Design projects now routinely utilize robotics, the Internet, and custom I/O devices with a variety of sensors. Using the RTOS, multithreading and synchronization can also be used when appropriate. The new technology with breadboarding, I/O breakout boards, and the C/C++ object-oriented I/O APIs, along with Wiki-based documentation and code examples, allowed students to develop significantly more complex laboratory and design projects than were produced in the previous version of the course. After taking the course, a number of students went on to use the robots, I/O sensors, and the mbed module again in their team-based semester-long senior design projects.

VII. Conclusion

The Wiki proved to be extremely useful in providing documentation and code examples for laboratory experiments. Students felt comfortable with this approach, and it worked well to disseminate the information needed for both traditional laboratory experiments and student design projects.

Using the cloud compiler for software development was easier for students and required less support effort than the traditional approach of using tools that run locally. No computer support was required other than initially handing out student passwords, enabling network access for the mbed modules, and installing the virtual com port driver. Availability of the cloud-based compiler and server has been excellent, but schools with extremely slow and unreliable Internet connections would probably want to use one of the offline compilation options.

Initially, the virtual com port driver used for serial communication from mbed to the PC with C/C++ stdio functions such as printf(), locked to each individual device's serial number. When students moved the module to a different computer, the driver had to be reinstalled. A recent registry change can now fix this problem.

Breakpoints are not currently supported for debugging in the cloud compiler; this triggered some initial concerns. The majority of student problems were actually a result of errors in wiring up breadboards rather than coding errors. It is also now possible to compile, set breakpoints, and debug code via the USB cable using ARM's Keil Tools traditional offline compiler or by using software emulation.

Using breadboarding, a wide range of interesting robots and embedded devices were successfully prototyped for the design projects. The low-cost robot kits were popular with students, and a large majority of the projects used the Internet or wireless networking. Students need to select a design project idea early in the term to allow ample time for any custom parts to arrive. Supporting diverse design projects also requires a larger assortment of robots and breakout boards. Fortunately, all of the robots and breakout boards can be reused.

Coverage of RTOS topics and the new debugging tools is being expanded this term. Currently, the mbed module is being incorporated earlier in the ECE curriculum for several in-class active learning demonstrations [10], [30] in the C/C++ Programming and Introductory Computer Architecture classes.

References

[1] Next Big Future, San Francisco, CA, "The robotics industry is now on an exponential growth path," Oct. 2011 [Online]. Available: http://nextbigfuture.com/2011/10/robotics-industry-is-now-on-exponential.html

[2] Japan Robotics Association, Tokyo, Japan, "The state of the industry-worldwide robotics market growth," Jun. 2011.

[3] GSMA, London, U.K., "The connected life," Feb. 2012 [Online]. Available: http://connectedlife.gsma.com/wp-content/uploads/2012/02/conn_lif_pospaper_web_01_11-13.pdf

[4] J. Turley, "The two percent solution," *Embed. Syst. Program.*, vol. 16, no. 1, p. 29, Jan. 2003.

[5] J. W. Bruce, J. C. Harden, and R. B. Reese, "Cooperative and progressive design experience for embedded systems," *IEEE Trans. Educ.*, vol. 47, no. 1, pp. 83–91, Feb. 2004.

[6] D. T. Rover, R. A. Mercado, Z. Zhang, M. C. Shelley, and D. S. Helvick, "Reflections on teaching and learning in an advanced undergraduate course in embedded systems," *IEEE Trans. Educ.*, vol. 51, no. 3, pp. 400–412, Aug. 2008.

[7] C. Lee, J. Su, K. Lin, J. Chang, and G. Lin, "A project-based laboratory for learning embedded system design with industry support," *IEEE Trans. Educ.*, vol. 53, no. 2, pp. 173–181, May 2010.

[8] Z. Ye and C. Hua, "An innovative method of teaching electronic system design with PSoC," *IEEE Trans. Educ.*, vol. 55, no. 3, pp. 418–424, Aug. 2012.

[9] K. Hwang, W. Hsiao, G. Shing, and K. Chen, "Rapid prototyping platform for robotics applications," *IEEE Trans. Educ.*, vol. 54, no. 2, pp. 236–246, May 2011.

[10] A. Carpeno, J. Arriaga, J. Corredor, and J. Hernandez, "The key factors of an active learning method in a microprocessors course," *IEEE Trans. Educ.*, vol. 54, no. 2, pp. 229–235, May 2011.

[11] J. Kim, "An ill-structured PBL-based microprocessor course without formal laboratory," *IEEE Trans. Educ.*, vol. 55, no. 1, pp. 145–153, Feb. 2012.

[12] K. Ricks, D. Jackson, and W. Stapleton, "An embedded systems curriculum based on the IEEE/ACM model curriculum," *IEEE Trans Educ.*, vol. 51, no. 2, pp. 262–270, May 2008.

[13] A. Behrens, L. Atorf, R. Schwann, B. Neumann, R. Schnitzler, J. Balle, T. Herold, A. Telle, T. Noll, K. Hameyer, and T. Aach, "MATLAB meets LEGO Mindstorms—A freshman introduction course into practical engineering," *IEEE Trans Educ.*, vol. 53, no. 2, pp. 306–317, May 2010.

[14] J. Gómez-de-Gabriel, A. Mandow, J. Fernández-Lozano, and A. García-Cerezo, "Using LEGO NXT mobile robots with LabVIEW for undergraduate courses on mechatronics," *IEEE Trans Educ.*, vol. 54, no. 1, pp. 41–47, Feb. 2011.

[15] EE Times Group, New York, NY, "2010 embedded market study," Apr. 19, 2010 [Online]. Available: http://www.eetimes.com/electrical-engineers/education-training/webinars/4006580/2010-Embedded-Market-Study

[16] N. Gershenfeld, R. Kirikorian, and D. Cohen, "The Internet of Things," *Sci. Amer.* Oct. 2004 [Online]. Available: http://www.scientificamerican.com/article.cfm?id=the-internet-of-things

[17] A. Vance, "You too can join the Internet of Things," *New York Times* Sep. 20, 2010 [Online]. Available: http://bits.blogs.nytimes.com/2010/09/20/you-too-can-join-the-internet-of-things/
[18] J. O. Hamblen, "Using a low-cost SoC computer and a commercial RTOS in an embedded systems design course," *IEEE Trans Educ.*, vol. 51, no. 3, pp. 356–363, Aug. 2008.
[19] J. O. Hamblen and G. M. E. Van Bekkum, "Using a Web 2.0 approach for embedded microcontroller systems," in *Proc. FECS*, Las Vegas, NV., Jul. 2011, pp. 277–281.
[20] R. Reese and B. Jones, "Improving the effectiveness of microcontroller education," in *Proc. IEEE SoutheastCon*, Mar. 2010, pp. 172–175.
[21] P. Jamieson, "Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat?," in *Proc. FECS*, Las Vegas, NV., Jul. 2011, pp. 289–294.
[22] S. Ford, "Rapid prototyping for microcontrollers," Arm Holdings, Cambridge, U.K., Mar. 2012 [Online]. Available: http://www.embeddeddeveloper.com/corp/flex/mbed-IQ28.pdf
[23] Arm Holdings, Cambridge, U.K., "ARM university program," Mar. 2012 [Online]. Available: http://www.arm.com/support/university/
[24] J. Bungo, "Embedded systems programming in the cloud: A novel approach for academia," *IEEE Potentials*, vol. 30, no. 1, pp. 17–23, Jan.–Feb. 2011.
[25] C. Styles, "mbed educational program," Arm Holdings, Cambridge, U.K., Mar. 2012 [Online]. Available: http://mbed.org/handbook/Education
[26] Arm Holdings, Cambridge, U.K., "mbed API handbook," Mar. 2012 [Online]. Available: http://mbed.org/handbook/Homepage
[27] Arm Holdings, Cambridge, U.K., "mbed cookbook wiki," Mar. 2012 [Online]. Available: http://mbed.org/cookbook/Homepage
[28] Arm Holdings, Cambridge, U.K., "Cortex microcontroller software interface standard version 3.0, February 2012," Mar. 2012 [Online]. Available: http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php
[29] Arm Holdings, Cambridge, U.K., "IC sensor and driver breakout boards," Mar. 2012 [Online]. Available: http://mbed.org/cookbook/IC-Sensor-and-Driver-Breakout-Boards
[30] K. E. Holbert and G. G. Karady, "Strategies, challenges and prospects for active learning in the computer-based classroom," *IEEE Trans. Educ.*, vol. 52, no. 3, pp. 356–363, Aug. 2009.

James O. Hamblen (S'73–M'76–SM'89) received the B. S. degree from the Georgia Institute of Technology (Georgia Tech), Atlanta, in 1974, the M. S. degree from Purdue University, West Lafayette, IN, in 1976, and the Ph.D. degree from Georgia Tech in 1984, all in electrical engineering.

He is currently a Professor in electrical and computer engineering with Georgia Tech. Prior to earning the Ph.D. degree, he worked as a Systems Analyst with Texas Instruments, Austin, TX, and as a Senior Engineer with Martin Marietta, Denver, CO.

Prof. Hamblen received the ECE Outstanding Teacher Award in 2004 and the 2006 W. Roane Beard Outstanding Teacher Award at Georgia Tech.

Gijsbert M. E. van Bekkum (S'09–M'12) received the B.S. and M.S. degrees in electrical engineering from the Georgia Institute of Technology (Georgia Tech), Atlanta, in 2009 and 2011, respectively.

In 2010 and Fall 2011, he worked as a Graduate Teaching Assistant with Georgia Tech. He currently works as a Software Engineer for Google, Mountain View, CA.