Computer Science: Faculty Publications and Other Works

Faculty Publications and Other Works by Department

6-2011

Intelligent Systems Development in a Non Engineering Curriculum

Emily A. Brand
Loyola University Chicago

William L. Honig
Loyola University Chicago, whonig@luc.edu

Matthew Wojtowicz
Loyola University Chicago

Recommended Citation

Intelligent Systems Development in a Non Engineering Curriculum

Emily A. Brand
Loyola University Chicago
Department of Computer Science
Chicago, Illinois 60611 USA
+1.512.609.0338

eabrand@gmail.com

William L. Honig
Loyola University Chicago
Department of Computer Science
Chicago, Illinois 60611 USA
+1.312.915.7988

whonig@luc.edu

Matthew Wojtowicz
Loyola University Chicago
Department of Computer Science
Chicago, Illinois 60611 USA
+1.312.915.7989

mattu16@gmail.com

ABSTRACT
Much of computer system development today is programming in the large–systems of millions of lines of code distributed across servers and the web. At the same time, microcontrollers have also become pervasive in everyday products, economical to manufacture, and represent a different level of learning about system development. Real world systems at this level require integrated development of custom hardware and software.

How can academic institutions give students a view of this other extreme–programming on small microcontrollers with specialized hardware? Full scale system development including custom hardware and software is expensive, beyond the range of any but the larger engineering oriented universities, and hard to fit into a typical length course. The course described here is a solution using microcontroller programming in high level language, small hardware components, and the Arduino open source microcontroller. The results of the hands-on course show that student programmers with limited hardware knowledge are able to build custom devices, handle the complexity of basic hardware design, and learn to appreciate the differences between large and small scale programming.

Categories and Subject Descriptors
C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems – *microprocessor/microcomputer applications, real-time and embedded systems;* I.2.9 [**Artificial Intelligence**]: Robotics – *propelling mechanisms, sensors;* K.3.2 [**Computers and Education**]: Computer and Information Science Education – *curriculum.*

General Terms
Design, Economics, Experimentation.

Keywords
Arduino course, microcontroller course, embedded systems

1. INTRODUCTION
This paper presents the approach used in an experimental course to offer small scale microcontroller system development to computer science students. The results show that experienced student programmers (both advanced undergraduates and graduate students) are able to learn how to construct combined hardware and software systems. Further, the course successfully introduces smaller scale microcontroller development which they may not otherwise have an opportunity to learn.

This experimental course addressing small scale embedded programming fits in the ACM Computer Science Curriculum 2008 as the Intelligent System / Robotics knowledge area [4]. It was conceived to limit the amount and expense of customized hardware development but still allow students to gain exposure to advanced intelligent systems using sensors and robotics.

The original goal was to use a small microcontroller to provide students access to hardware control and software interactions in a participative and "tinkering" course (similar to [5,10]). Having experience with LEGO Mindstorms robots [13] which allow only a few simple plug-in sensors of fixed types, we sought a more open-ended and expandable platform. After an investigation of various microcontrollers, the Arduino microcontroller board was selected. Although there have been some earlier courses using the Arduino [2] many of these have focused on the small, flexible, wearable LilyPad variant of the Arduino controller [3,7].

2. MOTIVATION AND BACKGROUND

2.1 Small Embedded System Development
The Microsoft Windows XP operating system is 45 million lines of code [9]. A military operating system, for specialized surveillance computers, is 50 million lines of code [6]. Courses in Rapid Application Development using architectures and tools such as Service Oriented Architecture (SOA) or .NET allow students to quickly generate large complex systems with database management systems, network access, and web interfaces.

There is another kind of development: small, often real time systems never destined to run on a personal computer or the Internet. Instead of visible computers the software runs on microcontrollers; a microcontroller is a single semiconductor chip including a small 8 or 16-bit processor, timing circuitry, and volatile and static memory. Microcontrollers are inside other objects (automobiles, toasters, traffic monitors) and often a key

part of providing the user's features. In this world the software has two major differences. First, the application is more intimately tied to the physical world and hardware (sensors, controls, many kinds of analog or digital inputs). Second, the software running on a small microcontroller is fully in charge of the device without the need to timeshare with other applications for the user. Microcontrollers have limited memory and often much less processor speed than today's personal computers.

This smaller scale system development is becoming more visible and a focus in the popular press [11]. Computer Science students need to have opportunities to understand the differences from the large applications world, acquire skills for developing small intelligent systems, and be able to make informed decisions about their career directions.

2.2 Open Source Microcontroller

The technical heart of the experimental course is the Arduino microcontroller board. The Arduino Duemilanove model used in class [1, 12] is a 6.8 by 5.3cm printed circuit board and includes an ATmega microprocessor, connecting pins for digital and analog input and output, several powering options, a boot loader, and 32K bytes of memory. Thus, it is a ready-made piece part for small projects including a variety of inputs and outputs. Students are able to use the board without needing to learn and build the lowest level of hardware including timing and power supply regulation.

The Arduino hardware and software are both covered with open source licenses. The hardware design is available to interested users under a Creative Commons Attribution Share-Alike license. Although not important to the class described here, the hardware design may be freely modified and incorporated into products and shared with the same license.

3. COURSE STRUCTURE

3.1 Student Preparation

The course was designed for experienced programmers but did not require specific electronics or hardware preparation. Students had previously completed a minimum of three software development courses, with emphasis on object oriented development in Java. Some students had considerably more experience in distributed systems, server based software, and mobile application development. Students included both advanced undergraduates and masters degree seeking students.

3.2 Instruction Topics

The course covered a combination of hardware and systems / software topics with the goal of preparing students to undertake an individually designed project in the latter part of the 15 week semester. Topics included:

- Introduction to Arduino microcontroller hardware and system development (hands-on implementation of first circuit and software)

- Electronics tutorial (amps, volts, ohms, and circuit diagrams), building basic breadboard circuits, and dangers of Electro Static Discharge (ESD)

- Physical world input and output (light and temperature sensor, LED, speaker, and motor control)

- Real time software strategy without an operating system

- Designing interactive real time systems using Structured Analysis and Design Technique (SADT) [8].

Figure 1 is an example early project used to expose students to the basics steps of hardware and system design. The Arduino board at the rear of the figure is connected to a circuit of multiple lights on a solderless breadboard (front of figure). The project explores real time performance by increasing the rate of blinking each light and detecting when all processor time is consumed.
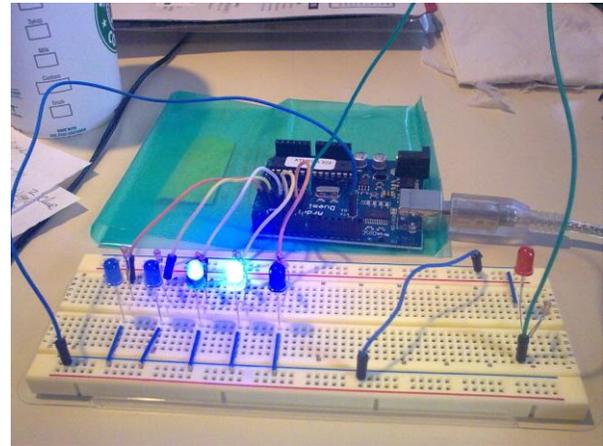

Figure 1. Example real time performance project.

Students completed this lab with multiple blinking LEDs with and without using the microcontroller's delay() function for real time control. Then they moved onto adding buzzers and other outside sensors to create a more complicated system with light and temperature sensors, other input buttons, sound output, and small motors and servo controls.

The course was structured as a seminar class with emphasis on student prototyping or "tinkering". There were no formal examinations. Student's grades were partially based on peer evaluation of their projects and class contributions by students.

3.3 Course Equipment

The course was new and the university had not previously taught similar classes. The physical meeting space for the course was a departmental research and project laboratory with limited space. The department acquired Arduino controller boards, wireless breadboards, electronics parts, switches, joysticks, sensors, and robotics kits for the class and from which the student's built their final class projects.

The course equipment cost US$2500 of which US$500 was for hand tools, soldering equipment, and storage cases for parts (supporting 12 students, the limit imposed by lab space). A single Arduino Duemilanove microcontroller board, fully assembled, costs less than US$40. Much of the equipment survived the course unharmed and will be used in future course offerings.

3.4 Student Projects

After working on initial simple systems with a few LED lights or sensors, students spent about half of the course developing their projects (some individually, others on two person teams). Student projects included:

- Airplane glider control to maintain a heading (Figure 2)

- Tracked robot rover with wireless interface

- Memory testing game similar to SIMON

- Guitar sound modification system

- Music / speech sound generation system

- Wearable environmental sensing clothing

Figure 2. Microcomputer autonomous glider

4. COURSE FINDINGS AND LEARNING OUTCOMES

The course was conducted as an experiment to determine both the feasibility of teaching more hardware intensive courses and to see if students would learn the differences between developing large and small scale computer based systems.

4.1 Learning Small Microcontroller Programming

The Arduino development environment forced students to confront the differences of large and small scale programming. First, there is no operating system beyond a basic loader and a suggested division of the software into initialization code and a repeated main processing loop. The standard Arduino delay() function simply loops the processor for a number of cycles to use up time. Student software needs to decide what to do with all of the processor time and how to divide time between different parts of the system.

Second, unlike most programs, a controller system usually runs forever (or until a reset button is pressed or power is removed). While running, the microcontroller code needs to accommodate the differences between internal processor time and the connected real world components. For example, it does not work to test the state of a push button switch every millisecond (possible with the Arduino's processor speed of 16Mhz). Too rapid checking of the switch state (typically as a current flow across an Arduino pin) can result in many false inputs during the time the button is being pressed and the contacts begin to conduct electricity. Students learned to do "debouncing" to compare and time inputs to determine when to act upon them.

As a result, students were forced to consider the key differences between large and small scale programming. Although none of the students had developed small systems in the past, the course end survey (Figure 3) showed they believed they understood this key distinction.
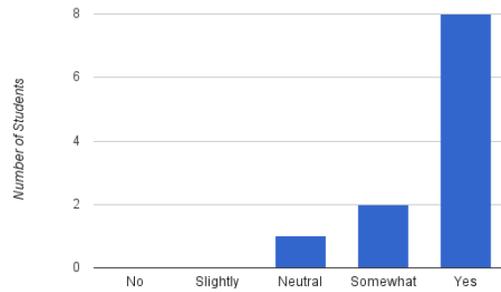

Figure 3. Do you understand the difference between programming microcontrollers and higher level programming(i.e., Java)?

The instructor concurred and saw further evidence in student's projects. For example, several projects dealt with large differences in real time demands between different parts of their systems (e.g. checking for inputs and driving output displays and generating sound).

4.2 Using SADT for Design and Communication

Microcontroller based systems are different from the computers students normally use in classes and projects. Instead of a keyboard, mouse, display and network connections, the microcontroller can connect to a number of specialized input and outputs depending on its intended function.

This combination of hardware and software thinking was a common problem students had to overcome. Once projects began to increase in complexity, students had trouble describing their projects to the class. The essence of the problem was clearly separating their software logic and hardware logic. Students looking at a peer's project had trouble understanding the system just from looking at the hardware and the source code.

The solution to these problems was using a high level analysis technique, Structured Analysis and Design Technique (SADT) diagrams [8]. SADT gave students a common diagramming paradigm that had the capabilities to describe both the system's hardware and software design on multiple levels. Design techniques such as UML and use case diagrams, due to their software focus, failed in comparison to SADT diagrams.

Figure 4 is a high level template SADT for student projects. The course used SADT diagrams to clearly define the actions of the software in response to physical world inputs. In this approach actions are the main components (boxes) with inputs coming into the action from the left and outputs leaving to the right. Data not manipulated by the system (i.e., state setting, control bits) are depicted as arrows coming into the top of the action. Software logic (debouncing, data manipulation, routing, real time control) is represented within the action or by decomposing it into another diagram. Arrows to the bottom of an action box are the mechanisms or tools used by the action.
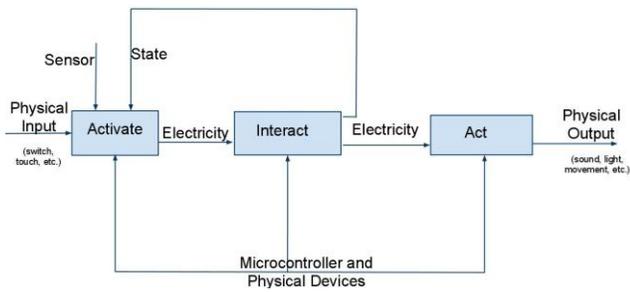
Figure 4. Generic SADT diagram for project analysis

All of the inward arrows coming together cause the action to occur and the output to be created. Students' systems typically consisted of a sensor listener to initiate actions, cause a physical manifestation using other devices, and possibly cause other actions to take place immediately or after some time.

SADT diagrams allowed students to coherently present and critique other students' projects. With the ability to communicate their projects, all students were able to receive quick and useful feedback. The diagrams also allowed students to pen and paper prototype before going through all the hardware set-up, allowing for instructors to catch problems early and prevent later frustration.

4.3 Running a Seminar Class with Tinkering

The course was structured as a hands-on seminar with laboratory workshops. This informal setting with only a few formal lectures allowed students to fully understand how microcontrollers work; it also enabled class discussions about the labs since every student was working on the lab at the same time. The students with more background were able to refresh and solidify their understanding of software or electricity and share it quickly with others. Less experienced students could delve in with a safety net since the professor, teaching assistants, and peers were all able to be of assistance when a problem arose.

The most interesting aspect of small intelligent systems is the hands-on capability and the ability to make mistakes without major consequences. Because of the relatively inexpensive cost of the equipment and no concern about affecting the wider network and servers, students were able to tinker and play with their creations. Sometimes students would simply try different circuits and make wiring changes to see what happened and try to explain the results. Outside of class time, students took their projects with them and worked on them as they wished.

The authors believe that forcing a more structured class could drain the students' ambition, interest in microcontrollers, and low-level programming. Keeping class lectures and exams to a minimum allows students to take advantage of and challenge their creativity and current skill-set. Pushing students to alter the labs to use their own desired inputs and outputs encourages creativity and discussion, two key features of this course.

As part of the course structure, students were asked to evaluate other students' work. Figure 5 summarizes the peer evaluation results for the final projects at the end of the class. While students do rate each other highly (above 7 on a 1 to 10 scale) they did show a reasonable distribution between the best and the worst work.
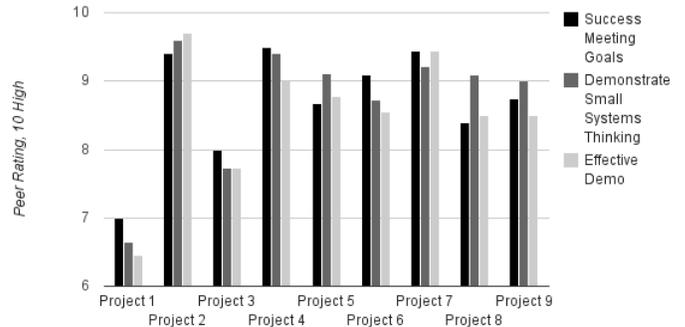

Figure 5. Average Peer Rankings of Final Projects

4.4 Other Findings

In addition, the student course end survey and feedback produced some other points of interest to those planning similar courses:

- Students enjoyed and appreciated the opportunity to learn and practice soldering of electrical parts; 100% rated soldering "useful" in the course end survey. Not all construction was possible using the solderless breadboard and jumper wires.

- Students indicated they would be willing to pay a laboratory fee for the class. Such a fee could be used to replace and expand the hardware components and tools used in the class.

- Students who attempted sound generation projects such as synthesizers had difficulties and were unsuccessful multiple times. These topics require more preparation and more sophisticated hardware components.

5. NEXT STEPS AND FURTHER COURSES

Future courses in similar topics can benefit from using a similar approach and considering several possible improvements.

5.1 Speeding up the Basics

One of the most difficult aspects for the students to grasp at the beginning of the course was electrical knowledge and understanding. Theoretical exploration of the topic in two lectures proved to cause more confusion than clarification. The best approach was to lead the students through a series of hands-on workshops that demonstrated resistance, electrical flow, and other relevant aspects. Students at the end of the course requested that next time there be more of these workshops in order to solidify their foundations in electrical know how.

In order to provide students with a solid base to begin their own project the authors suggest taking time to walk through the following labs for students without prior electronics training:

- A light on/off switch to introduce basic electric principles (using meters, not the microcontroller).
- A button on/off switch to teach debouncing and introduce microcontroller sensor interaction.
- An incremental on off switch that steps up LED brightness with each press of a button. This exercise will introduce topics such as state in a microcontroller

and analog output using Pulse Width Modulation (PWM).

5.2 Interest in Machine Learning and Robotics

When asked what topic they would most wish to continue in future classes, the students were split between an advanced microcontrollers class and a machine learning class. A possible solution, while still using resources economically, is a robotics class. This would allow for both groups to continue in their areas of interest without the need for two classes.

A suitable robotics class can expand on the real time and control knowledge and also allow more focus on learning and decision making algorithms. One of the first topics that can be addressed is communicating with other microcontrollers that control separate, multiple motors. Another aspect of small scale programming that robotics emphasizes is the importance of real time. Students will be pushed to handle real time events and program for responsiveness possibly with many inputs at once. Students would have to handle failures due to time constraints and learn how to minimize the loss such an error causes. Both of these topics could prove valuable for future students.

5.3 Possible Assembly Language Option

The Arduino microcontroller provides an excellent tool for students to get into smaller scale programming. The class used the open source Arduino development environment and the C programming language. Specific processor bits and flags can be accessed and manipulated from the C language directly (e.g. the processor library defines hardware timer number one's data as TCCR1B and makes it available to the C program as a variable).

However, the C language still comes between the student and direct control of the machine. The Arduino environment allows linkage to assembly language programs or inline assembly language instructions for the ATmega processor of the microcontroller.

For further and more precise planning and control of real time response or greater understanding of the performance limits of the microcontroller, it is reasonable to add assembly language programming. This lower level of programming may be usefully applied to a small device interface via the Arduino input and output pins or to better control time delays. Future versions of the course, or follow-on courses, will develop small projects in these areas.

6. SUMMARY

The course was successful in accomplishing its major goals. Institutions such as ours that have focused on purely software courses and are without major engineering facilities should not hesitate to bring more hardware based courses into the computing curriculum. Open source hardware and software such as the Arduino microcontroller make such a course both economical and practical; it possible to effectively teach a microcontroller course without a heavy financial cost. Students no longer need to think that all software runs on a personal computer and a web server.

7. REFERENCES

[1] Banzi, M. 2008. *Getting Started with Arduino*. Maker Media, San Rafael, CA.

[2] Brock, J. D., Bruce, R. F., and Reiser, S. L. 2009. Using Arduino for introductory programming courses. Tutorial Presentation. *Journal of Computing Sciences in Colleges*. 25,2 (Dec. 2009), 129-130.

[3] Buechley, L., Eisenberg, M. Catchen, J. and Crockett, A. 2008. The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems* (CHI '08), ACM, New York, NY, 423-432. DOI= http://doi.acm.org/10.1145/1357054.1357123.

[4] CS Review Task Force 2008. Computer Science Curriculum 2008: An Interim Revision of CS 2001. Association of Computing Machinery, IEEE Computer Society. http://www.acm.org//education/curricula/ComputerScience2008.pdf Retrieved Jan. 10, 2011.

[5] Gehringer, E. F., and Miller, Carolyn S. 2009. Student-generated active-learning exercises. In *Proceedings of the 40th ACM technical symposium on computer science education* (SIGCSE '09), ACM, New York, NY, 81-85. DOI= http://doi.acm.org/10.1145/1508865.1508897

[6] Hersh, S.M. 2010. The online threat: should we be worried about a cyber war? *The New Yorker* (Nov. 2010).

[7] Lau, W. W. Y., Ngai, G., Chan, S. C. F., and Cheung, J. C. Y. 2009. Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students. In *Proceedings of the 40th ACM technical symposium on Computer science education* (SIGCSE '09), ACM, New York, NY, 504-508. DOI= http://doi.acm.org/10.1145/1539024.1509041.

[8] Marca, D. A. and McGowan, C. L. 1987. *SADT: Structured Analysis and Design Technique*. McGraw-Hill, New York, NY.

[9] Microsoft Corp. 2010. A history of Windows. http://windows.microsoft.com/en-US/windows/history Retrieved Jan. 10, 2011.

[10] Qian, K., Liu, J., and Tao, L. 2009. Teach real-time embedded system online with real hands-on labs. Poster. In *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education* (ITiCSE '09), ACM, New York, NY, 367-367. DOI= http://doi.acm.org/10.1145/1595496.1563009.

[11] Siegele, L. 2010. It's a smart world, a special report on smart systems. *The Economist* (6 Nov. 2010), 3-18.

[12] Russell, D. 2010. *Introduction to Embedded Systems: Using ANSI C and the Arduino Development Environment*. Morgan and Claypool, Sebastopol, CA.

[13] Williams, A.B. 2003. The qualitative impact of using LEGO MINDSTORMS robots to teach computer engineering. *IEEE Transactions on Education.* 46,1 (Feb. 2003), 206. DOI= http://10.1109/TE.2002.808260.