Transformation of an Introductory Computer Engineering Course Utilizing Microprocessors and a Focus on Hardware Limitations

Mr. Charles Carlson, Kansas State University

Mr. Charles Carlson is a Graduate Teaching and Research Assistant in Electrical and Computer Engineering at Kansas State University, and he is currently pursuing his Ph.D. in Electrical Engineering. He works in the biomedical lab, and is a teaching assistant for ECE 241: Introduction to Computer Engineering. He is interested in engineering education, biotechnology, and bioinstrumentation.

Dr. Dwight Day, Kansas State University

Dwight Day received his Bachelors, Masters and PhD from Oklahoma State University, in 1980, 1981 and 1987. Dwight was an engineer at Texas Instruments 1983-1985, and Boeing Military Airplanes from 1987 to 1990. Dwight is an Associate Professor at Kansas State University, where he has been teaching and doing research since 1990. Dwight has taught classes in digital design, microprocessor applications, numerical methods, digital image processing and digital signal processing. Dwight's research area have ranged from image processing for quality control to signal processing for road-profiling. Dwight has also done research for Sandia National Labs (High Performance Computing) and NASA (Low-Power Communications).

Transformation of an Introductory Computer Engineering Course: Utilizing Microprocessors and a Focus on Hardware Limitations

1. Introduction & Motivation

Keeping a class's curriculum current and fun is a difficult challenge within the ever-expanding field of computer engineering. This is especially true at the introductory level as first year courses are intended to provide an overview of the entire field. Prior to the Fall 2015 semester, the Introductory Computer Engineering course at Kansas State University was centered around digital design (logic gates, flip flops, K-maps, etc.), with the lab sessions requiring students to write HDL software with limited hands-on experience with hardware. For the Fall 2015 semester, the class was transformed to utilize microprocessors and focus on hardware limitations. The transformation was done for primarily two reasons. 1) To advance the course's curriculum. 2) Improve student retention.

Every day we interact with and are surrounded by embedded systems. From cars to microwaves, they have become an integral part of everyday life. It's no surprise then that the area of embedded system design has grown tremendously in the past few years [1]. More graduates are working with microprocessors as a result of the growing embedded systems field and would benefit from working with them and coding during their undergraduate coursework. Therefore, it was decided that the courses' new focus would be centered on embedded systems by having the labs utilize a microprocessor.

One of the motivations for improving student retention was that past students who had transferred out of the program expressed that they felt overwhelmed or intimidated as a reason for transferring. To address this problem, it was clear that a more hands-on experience with hardware and software with an easy to use, user friendly platform was needed. Thus, the Arduino microcontroller was chosen as the base point for the lab section of the class. Arduino offers several cheap, simple, microcontrollers with an intuitive Integrated Development Environment (IDE) that was originally developed for hobbyists and artists with little to no programing experience; this makes it ideal to use in an introductory, freshman level class where a large portion of the students have no prior coding experience. Arduino also has an extensive online presence, something few other platforms offer.

However, it is easy to lose track of hardware limitations when using a higher level programming language. As an example, consider the Arduino function, digitalWrite(). It is a built-in function in the Arduino IDE and makes it easy to control the output state of a digital pin. But how long does such a function take to execute? Is there another command that can execute faster? And what is the limiting factor controlling the speed of port manipulation? These are just a few of the questions posed to students during the lab portion of the class.

Our goal is to first have students become comfortable with the programming environment and hardware involved. Then, we can ask the more fundamental questions to shift the focus to the

hardware limitations. To measure the effectiveness of the transformation, the authors used first year student retention data, and a survey measuring confidence to quantify results. Only preliminary survey results are presented in this paper, since the transformation has only been implemented for the last three semesters. In the following section, the approach taken for the transformation is discussed. In section 3 the class structure is detailed along with how we implemented our approach. Section 4 outlines how we focus on hardware limitations. In section 5 preliminary survey numbers are presented. Finally, the next steps and what we have learned thus far are presented in section 6.

2. Approach to Transformation

As mentioned above, the previous version of the class focused on digital design, and it was decided that the focus should shift to microprocessor based designs. There are numerous microprocessors to choose from, so why did we decide to use the Arduino platform? Two platforms were given considerable thought, the Arduino and Raspberry Pi. Both are relatively inexpensive and have vast online resources. The Arduino platform has been utilized in several other institutions' introductory courses with positive student feedback [2-5]. In [4], the author discusses how the Raspberry Pi is being used for summer STEM enrichment programs at MIT. Ultimately, the Arduino was chosen since it is easier to investigate the hardware limitations of a microcontroller than a small computer (Raspberry Pi) and requires minimal background knowledge. Once the Arduino platform was chosen, the specific microcontroller needed to be selected. The Arduino Nano was chosen since it was smaller, and easier to integrate into projects (keeping in mind the classes' focus on embedded systems).

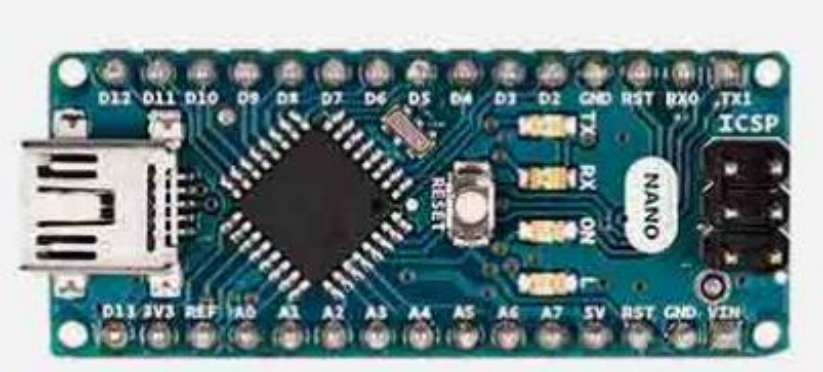

Figure 1. Arduino Nano [6].

We also wanted to make sure the class didn't lose track of hardware limitations. For this reason, the Digilent Analog Discovery was used along with the Arduino Nano. The Analog Discovery is a USB-powered, handheld device that can replace an entire lab bench of equipment. It includes a two-channel digital oscilloscope, two-channel arbitrary function generator, 16-channel digital logic analyzer, and a multitude of other features detailed on the Digilent webpage. The logic analyzer and oscilloscope were mainly used in this course. For example, in lab 2 students are able to measure the time it takes the digitalWrite() function to set a bit high then back low, and then compare that time, using the logic analyzer, to pulsing the bit using port manipulation. The

logic analyzer is also used to time serial communication (UART) in lab 3 and SPI communication in lab 8.

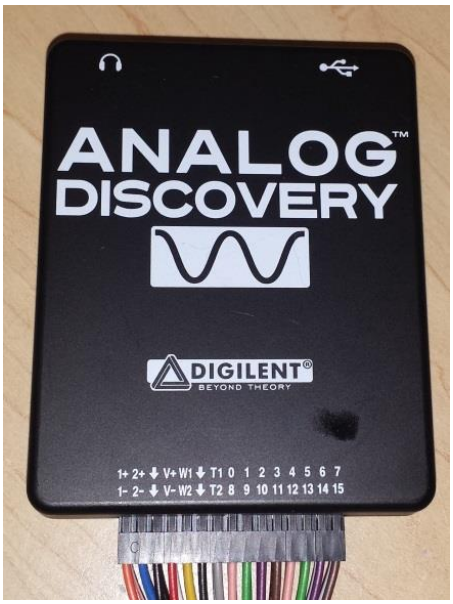

Figure 2. Digilent Analog Discovery

One of the methods used to gauge the effectiveness of the transformation is a survey designed to measure student confidence. The survey consists of 22 questions measuring confidence from two directions using a 5 point Likert scale ranging from either very confident to not at all confident for the direct method or very difficult or not at all difficult for the indirect method. The first eight questions directly measure confidence in the core topics covered in the introductory class. The next three questions are centered on topics covered in the follow-up microprocessor course in the computer engineering department. The same structure was used to indirectly measure confidence by asking students how difficult they found a task to be in the next 11 questions. Again, the first eight questions for the indirect method are centered on core topics to the introductory class while the last three questions dealt with topics in the follow-up microprocessor course. The survey was conducted in the follow-up microprocessor class at the end of the semester for the Spring 2016 and Fall 2016 semesters. For the Spring 2016 semester, most of the students had not taken the new form of the class (25 total students participated. 22 had taken the hardware (old) version of the class and 3 the micro (new) version. The survey results from the Spring 2016 semester provided a control group to compare future survey results with as most of the students had not taken the new version of the class. The last three questions of the survey were centered on topics only covered in the follow-up course with the hope that A) student responses to those questions would remain the same regardless of which version of the introductory class they had taken while B) students who had taken the new form of the class would have higher levels of confidence in the first eight questions. Thus showing that the transformation had a positive impact on student confidence in the introductory class's core topics. The survey can be found in Appendix B.

3. Program Structure/ Course Description

The introductory computer engineering course is a 3-credit hour class that meets 3 times a week for 50 minutes over 16 weeks. The course is meant to introduce the student to using a computer to interact with real world inputs and outputs, or simply, embedded systems. The name embedded systems comes from the idea that the computer is embedded within a device, becoming a subset of the device. With the class being an introductory level course, some basic topics are covered to help students understand the fundamentals of the field. Mainly:

- Number systems (binary and hexadecimal)
- ASCII
- Binary addition/subtraction
- Two's complement representation vs operation
- Masking

With the goal of students having more practical experience with software and hardware, the class is heavily centered on the labs with the core topics covered in lab being:

- Timers
- Serial communication
- Ports
- Interrupt servicing
- Masking

With the character of the class, topics are not covered in a purely sequential manner. Rather, topics are introduced and demonstrated as their need arises. A full list of the topics discussed in the class can be found in Appendix A. Since some of the students do not have experience with programming, the lectures also cover some fundamental programming topics needed for the lab. For instance, serial communication hardware interfacing is discussed before lab 3 where students set up serial communication between the Arduino Nano and a serial terminal (built-in to the Arduino IDE). The timing of bits for serial communication is also discussed, which students measure in the lab using the Digilent Analog Discovery (described in more detail in section 4.) Other programming topics covered in the lectures are:

- Logical operators
    - Bitwise vs logical
    - Logical equations
- Event driven programming
    - No delay function calls
- State machine design
    - Design software before writing code

The grade distribution is shown in Figure 3 and the lab schedule in

Figure 6. Almost half of the points are in the labs section, thus if a student were to miss two or more labs, they would most likely have a difficult time passing the course.

Grade Distribution:	Labs	225
	Final Project	50
	Three Exams	150
	Homework & Quizzes	70
	Total	500

Figure 3. Grade distribution.

In the lecture prior to the start of a new lab assignment, a demonstration is given detailing what the students will need to know for the upcoming week's lab assignment. Students are sold lab kits during the first week of class. The kit includes the Arduino Nano and a small prototyping board designed in house. The board (shown in Figure 4 and Figure 5) makes it easy to interface to other hardware including a liquid crystal display and an encoder to the Arduino. The header pins for the prototyping board are not soldered onto the board when the student receives the kit. This gives the students a chance to learn how to solder, an important skill that is often overlooked in introductory classes. The first three labs do not require the use of the prototyping board, so the students have ample time to put the boards together. There is also a "learn to solder" session available to the students in case they have no prior soldering experience.

Figure 4. Prototyping board designed in house.

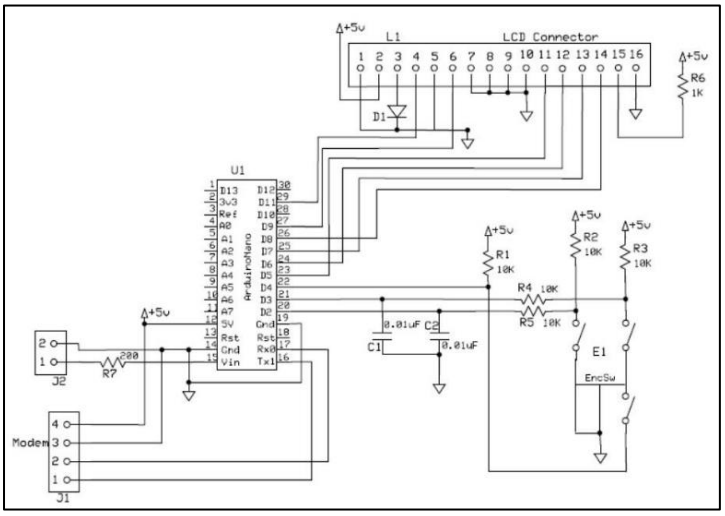

Figure 5. Prototyping board schematic.

Week 1)	August 22-26 (No Lab)
Week 2)	August 29 - Sept 2 (Lab 1) Arduino and their Integrated Development Environment (IDE) Objective: Familiarization with the Arduino, its software environment, and accessing outputs.
Week 3)	September 6 - September 12 (Lab 2) (Shift for Monday, due to Labor Day holiday) Logic Analyzers and Monitoring Digital Signals. Objective: Familiarization with the Digilent Logic Analyzer.
Week 4)	September 13 - September 19 (Lab 3) Serial Port Communications. Objective: Understanding ASCII characters and sending data over a serial port.
Week 5)	September 19 - Exam I September 20 - September 26 (Lab 4) LCD Displays Objective: Setting up the library software and using it to control the LCD.
Week 6)	September 27 - October 3 (Lab 5) Reading a Simple Switch. Objective: Using timing to consistently read a switch/button.
Week 7)	October 4 - October 10 (Lab 6) Communicating with Other Devices Objective: Using communication hardware to communicate with other devices.
Week 8)	October 11 - October 17 (Lab 7) Reading the Encoder Inputs Using Interrupts. Objective: Using interrupts to monitor external hardware.
Week 9)	October 18 - October 24 (Lab 8) Start of Programming Assignment.
Week 10)	Exam II October 31 October 25 - October 31 (Lab 8) Programming Assignment.
Week 11)	November 1 - November 7 (Lab 9) Analog Read and Write. Objective: Reading analog sensors.
Week 12)	November 8 - November 14 (Lab 10) Application of PWM as an DAC Objective: Creating an analog output voltage.
Week 13)	November 15 - November 18 (Project Preparation)
November 21-25 University Holiday.	
Week 14)	November 28 - December 2 (Project Preparation)
Week 15)	December 5 - December 9 (Project Check off)

Figure 6. Fall 2016 lab schedule.

Students complete a lab report for each lab that is due the following week. For the reports, they are to include the code used for each section. Good commenting, indentation, and overall code organization are skills that are emphasized in the class, so a significant portion of the lab report grade is placed on the structure and formatting of their code. Along with a lab report, students complete a simple prelab assignment where they write the code needed for the first part of the lab. For the final project, student's work in teams of two and select one of the following projects:

- Motor Controller
- Thermostat
- Solar Tracker
- Dispenser

Students are provided with the objective along with the design requirements for the project they select. For example, the thermostat project description, which is by far the most popular, is shown below

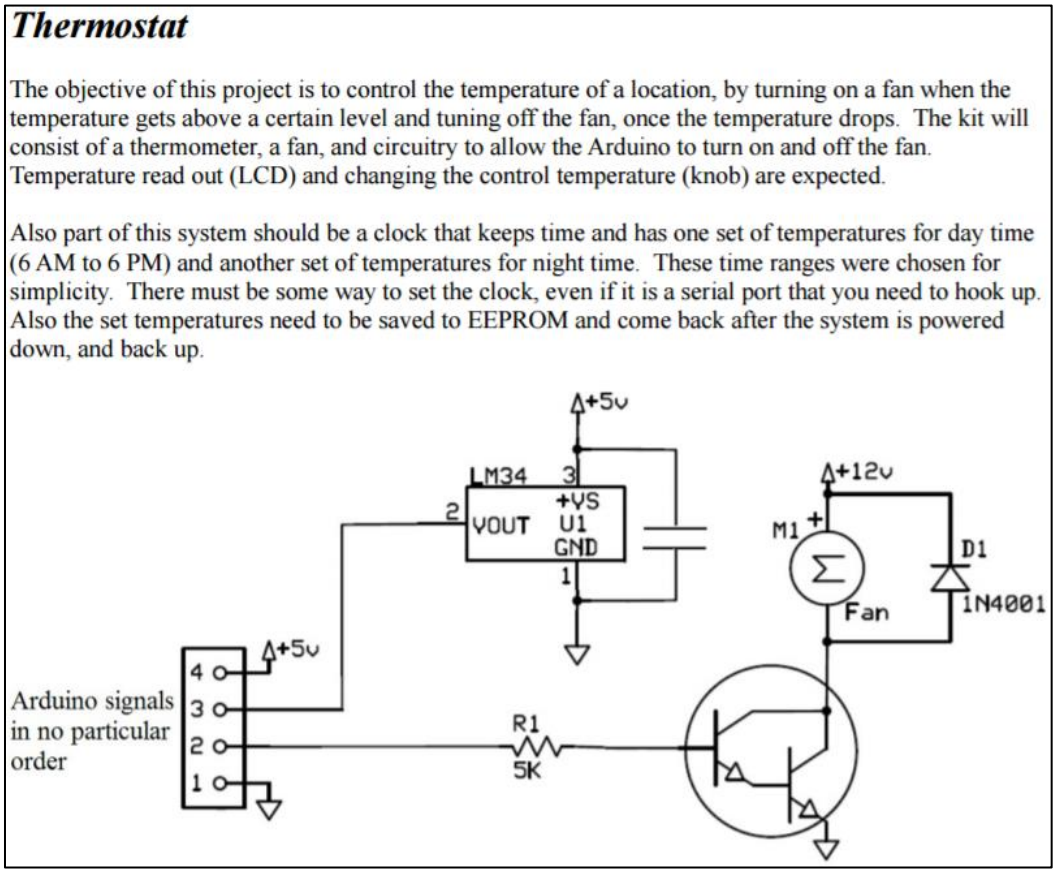

Figure 7. Thermostat project description.

Thermostat design requirements:

1. Operates with two temperature settings based on time.
2. Ability to turn fan off and on based on temperature settings.
3. Temperature thresholds stored in EEPROM.
4. Ability to set clock using encoder.
5. Current temperature and time is displayed on LCD.

4. Focus on Hardware Limitations

The following section details the labs that utilize the Arduino and the Analog Discovery to show how hardware limitations are explored, but does not detail every lab. The objectives of every lab can be found in

Figure 6. With the use of Analog Discovery and the Arduino, students not only gain insight into the timing of the Arduino itself, but also the timing of serial communication. In lab 3, students investigate ASCII characters and how data is sent over a serial port. Within the Arduino structure, sending data over a serial interface is straightforward. Only a few lines of code are required. For example, the following line would send the character 'a' over the serial interface.

Serial.print('a');

But what is going on in the background? How is this character being transmitted? With the use of the Analog Discovery's digital logic analyzer, students are able to watch the bit stream and measure the amount of time required to send each bit.

Lab 8 introduces serial peripheral interface (SPI) communication to receive data from an external thermocouple temperature sensor. Again, using the Analog Discovery's log analyzer, students can monitor the communication signals to better understand how SPI communication is different from UART. At the top level, this difference is much more difficult to understand. Lab 9 introduces the concept of analog to digital converters (ADCs). To understand the Arduinos ADC limitation, an external voltage is applied to one of the analog inputs. Students write the code needed to calculate the voltage from the analogRead() function. They also connect the external voltage to the oscilloscope of the Analog Discovery. They are then able to compare the external voltage value to the oscilloscope's measured voltage as well as the voltage calculated from their code.

Lab 10 then introduces the concept of digital to analog converters (DACs). The Arduino Nano does not have a built-in DAC, however, it does have a built-in timer and math library that can be used to generate a sinusoidal function with a relatively consistent period. The generated pulse modulated waveform (PWM) is then filtered using a simple RC lowpass filter to create a waveform that appears more sinusoidal. Again, students use the oscilloscope on the Analog Discovery to monitor the created waveform.

5. Preliminary Results

Survey data has been collected in two semesters thus far (spring and fall of 2016). The table below shows the number of students who participated in the survey each semester and the number of students who had taken the hardware or microcontroller (micro) versions of the class.

Semester	S16	F16
Total Participants	25	20
Hardware (Old)	22	4
Micro (New)	3	16

The average measurement of confidence for the students who had taken the micro version of the class was higher in seven out of the first eight questions, and their average measurement of difficulty was lower in the first eight indirect questions. The Questions for the survey can be found in Appendix B. These preliminary results look promising, but are currently not statistically significant to draw any concrete conclusions.

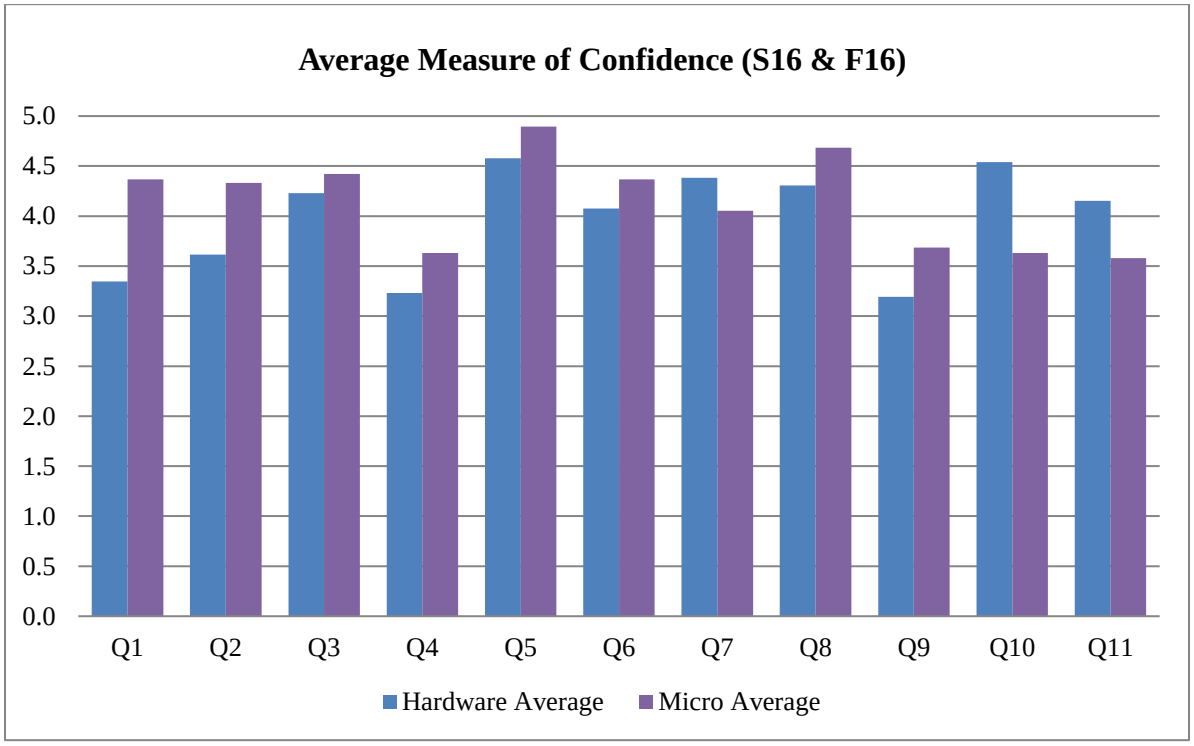

Figure 8. Average measure of confidence of combined data from the 2016 spring and fall semesters.

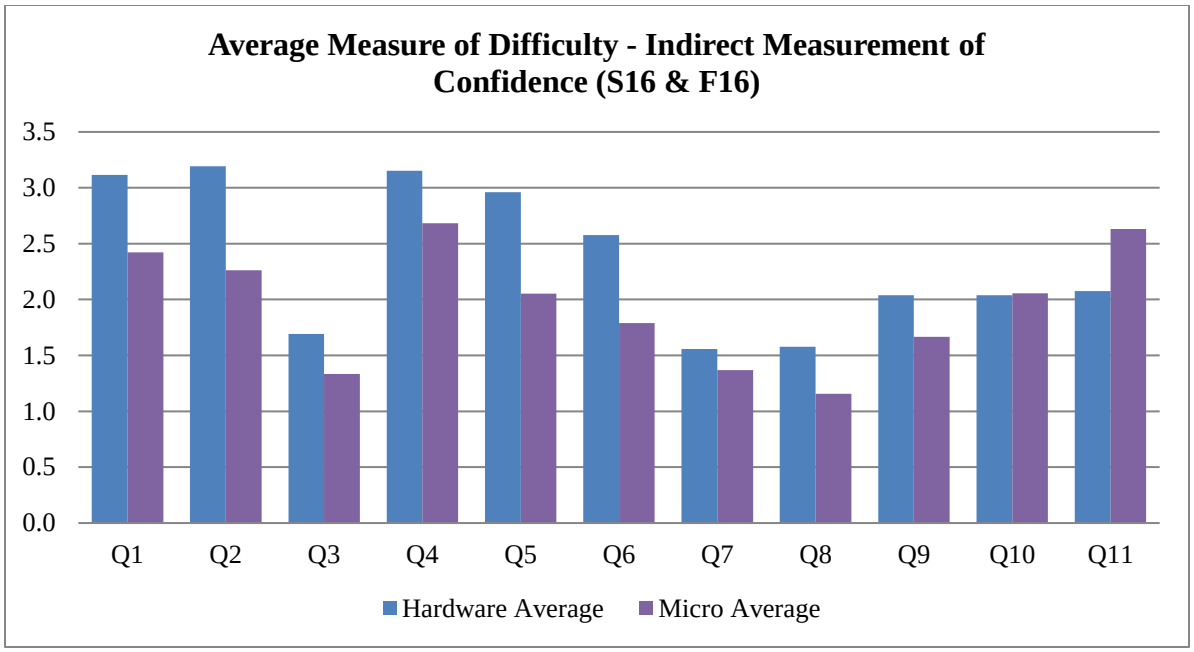

Figure 9. Average measure of difficulty (indirect measure of confidence) of combined data from the 2016 spring and fall semesters.

6. Discussion/ Conclusion

More survey data will be collected in upcoming semesters in order to better understand the effect the transformation has had on student confidence. Currently only 1 first year retention data is available, but will be used in the future to see how first year retention has been influenced. We learned that although the Arduino IDE is user-friendly and simple, students with little to no programming experience struggled more in the first labs. To address this, students simply program the Arduino Nano using the provided code for the first part of lab 1. Then for part 2, only pseudo code was provided. Previously, only pseudo code was provided for both parts. So far, we have received positive student feedback from this update. For students wanting additional help with programming, we are considering adding evening sessions to go over programming basics. Updating a course's curriculum is a challenging yet exciting process and in the field of computer engineering, tends to happen more often to keep up with the ever-expanding field. At Kansas State University the introductory computer engineering course was transformed to focus on microcontroller based design, keeping a focus on hardware limitations. Thus far, student feedback has been positive, and preliminary survey results have shown that student confidence might have increased from the new courses approach to teaching its core topics.

References

[1] D. R. S.-M. Dr. Afsaneh Minaie. "Capstone Projects in a Computer Engineering Program Using Arduino," *2016 ASEE Annual Conference & Exposition*. New Orleans, 2016, pp.

[2] D. S. G. N. Dr. Jose Antonio Riofrio. "Teaching Undergraduate Introductory Course to Mechatronics in the Mechanical Engineering Curriculum Using Arduino," *120th ASEE Annual Conference & Exposition*. Atlanta, GA, 2013, pp.

[3] D. Y. E. Dr. Warren Rosen, Mr. M. Eric Carr. "An Autonomous Arduino-based Racecar for First-Year Engineering Technology Students," *121st ASEE Annual Conference & Exposition*. Indianapolis, IN, 2014, pp.

[4] D. J. D. Steinmeyer. "Project-Based Learning with Single-Board Computers," *122nd ASEE Annual Conference & Exposition*. Seattle, WA, 2015, pp.

[5] D. J. E. Tian. "A design approach in an Introduction to Engineering course," *121st ASEE Annual Conference & Exposition*. Indianapolis, IN, pp.

[6] A. Nano. "https://www.arduino.cc/en/Main/ArduinoBoardNano,"

Appendix A

List of various topics covered throughout the semester.

1) Number Systems
   - Radix Based Numbers
     Decimal
     Binary
     Hexadecimal
   - Number operations
     addition
     signed / unsigned
     Finite length numbers
   - Arbitrary
     ASCII, representing letters on a computer.
     Converting Numbers to ASCII and vice versa
2) Programming
   - Data Types (int, signed/unsigned ...)
   - Operations
     Addition
     Logical (and, or, xor)
   - Flow Control
     if -
     for -
     while -
   - Arduino Structure
     Setup
     Loop
3) Hardware
   - Input/Output (IO) pins
     Port Manipulation
   - Memory structure
     arrays
     stack and heap
   - Communications
     Serial Asynchronous
     Serial Synchronous
   - Interrupts
     Interrupt Service Routines (ISR)

Appendix B

How confident are you that you will be able to:

	Very confident				Not at all confident
1) Implement an Interrupt Service Routine to call a function.	O	O	O	O	O
2) Send data through Serial Communication.	O	O	O	O	O
3) Access i/o pins using port registers.	O	O	O	O	O
4) Implement critical timing events using built-in Timer hardware.	O	O	O	O	O
5) Define a variable as an integer.	O	O	O	O	O
6) Create finite state machines to model problems.	O	O	O	O	O
7) Use Masking for bitwise operation.	O	O	O	O	O
8) Set up variables using constant definitions.	O	O	O	O	O
9) Use a control register to configure a clock signal.	O	O	O	O	O
10) Explain the difference between RAM and ROM.	O	O	O	O	O
11) Explain how a program counter works, particularly in sub routines.	O	O	O	O	O

How difficult would you find the following tasks

	Very difficult				Not at all difficult
1) Setup an Interrupt Service Routine to be called at a specified interval.	O	O	O	O	O
2) Send data over a Serial Peripheral Interface (SPI).	O	O	O	O	O
3) Setup a pin as an output using port register calls.	O	O	O	O	O
4) Create a 50% duty cycle waveform using built-in Timer hardware.	O	O	O	O	O
5) Grab data from a circular buffer.	O	O	O	O	O
6) Design a finite state machine to make an LED blink at 1 Hz.	O	O	O	O	O
7) Modify a single bit of an 8-bit variable using masking.	O	O	O	O	O
8) Create a program constant.	O	O	O	O	O
9) Grab data from a look up table.	O	O	O	O	O
10) Divide a clock signal by 2 using prescalars.	O	O	O	O	O
11) Name the main components of a microcontroller's architecture.	O	O	O	O	O

How challenging did you find the labs to be? (5 being extremely difficult)	1	2	3	4	5
How clear were the instructions for the labs? (5 being extremely unclear)	1	2	3	4	5

Are there any additional topics that you would have liked to have seen in the lab?

What was your most/least favorite lab?

What version of 241 have you taken? (circle one) micro hardware

micro(Arduino based) vs hardware(previous to Fall 2015)