BeagleBone Black for Embedded Measurement and Control Applications

Mr. Stephen A. Strom, Pennsylvania State University, Erie

Stephen Strom is a lecturer in the Electrical and Computer Engineering Technology department of Penn State Behrend, and holds a B.S. in electrical engineering from Carnegie Mellon University. His career includes over thirty years experience in designing and programming embedded systems and has multiple patents for both hardware designs and software algorithms

Prof. David R. Loker, Pennsylvania State University, Erie

David R. Loker received the M.S.E.E. degree from Syracuse University in 1986. In 1984, he joined General Electric (GE) Company, AESD, as a design engineer. In 1988, he joined the faculty at Penn State Erie, The Behrend College. In 2007, he became the Chair of the Electrical and Computer Engineering Technology Program. His research interests include wireless sensor networks, data acquisition systems, and communications systems.

BeagleBone Black for Embedded Measurement and Control Applications

Abstract

Lower-division courses in an Electrical and Computer Engineering Technology (ECET) program provide the background needed for introductory programming and embedded microprocessors. A C/C++ programming course emphasizes the software development process, language constructs, algorithms, and structured procedural design and possibly object oriented design. A microprocessor based course covers software design in C for input and output interfacing for various applications. For IoT applications, there is a need in upper-division courses to cover new, powerful, and inexpensive system-on-a-chip (SoC) devices that are capable of performing the responsibilities of a computer on a single chip and can be used in a variety of courses. One such device, the BeagleBone Black (BBB), is an open-source Linux platform that contains a variety of busses (SPI, I2C, CAN), general purchase I/O pins, serial ports, PWM outputs, and analog inputs.

The intent of this paper is to demonstrate the usage of the BBB in a variety of upper division courses, illustrating several applications. Some potential courses include measurements and instrumentation, wireless communications, control systems, and advanced microprocessors. Labs covered with the BBB include digital and analog I/O operations, UART interface, TCP/IP interface, touch screen display, and student chosen end-of-semester lab projects. Programming is achieved using C++ and Python. Several of these labs will be discussed in this paper, along with schematics, configurations, and results.

Introduction

In an Electrical/Computer Engineering Technology degree, there are many classes that use microprocessors/microcontrollers as part of their curriculum. The format for these classes are similar (in curriculum) in that their end goal is to teach 'C' or 'C++' programming, as well as embedded hardware and applications. For some universities, the engineering technology program includes multiple embedded systems courses, and in these situations, an instructor has the benefit to focus each course on a specific area of programming/hardware.

While educational philosophy and pedagogy will vary from one instructor to another, it could be said that most computer courses employ a large amount of hands-on lab material and selecting a proper embedded processor/operating system can greatly improve the outcome and success of the course. In general, the preference is to use a processor/operating system that has:

- Wide industry acceptance (usage after graduation). This allows the students to leverage their knowledge into better/more advanced job positions.
- Development tools that are quickly installed and are easy to use. There are always questions about compiling, downloading and debugging, and it is important to have local (and online) help tools that can provide solutions to common problems.
- Low cost of development tools, boards, kits. A strong suggestion is to have the students purchase their own components. Students that own their own board take better care of them,

and are more likely to use them in other projects.

In addition to these requirements, a microcontroller also needs to fulfill the outcomes required by the course. An advanced programming class would need to teach ‘C’ programming (and data structures), as well as software algorithms, microprocessor hardware/peripherals, and operating systems. To teach a wide range of subject matter, it is necessary to examine:

- The type of processor / operating system
- The amount of flash and ram memory (within the processor)
- Number of I/O pins (digital and analog)
- Internal circuits (digital, a-to-d, pwm, uart, spi, etc.)
- Timers and Interrupts

The goal of this paper is to show how the BeagleBone Black can be used successfully in an advanced embedded systems course [1].

Course Overview

The Computer Engineering Technology curriculum is set up so that programming and embedded circuit design is taught via a series of courses: (a) Introduction to ‘C’ programming, (b) Digital Design and Embedded Systems, (c) C++ and object orientated programming (d) an Intermediate Embedded Systems course and (e) an Advanced programming/operating systems course. After completing this series, the students should be thoroughly prepared for any job based on programming and embedded systems.

This arrangement of courses is a bit unusual in that Technology departments do not always offer embedded systems designs to this depth; however, it is one that really prepares the student to succeed as an embedded programmer/designer. However, success in a curriculum does not always relate to the number of courses available, and often it is the content and delivery that meters the student’s progress.

The beginning (or entry level) computer course may use an “Arduino” style computer board to teach ‘C’ programming and embedded hardware. As this is an introductory course, all of the hardware interfaces are accomplished via API/library calls. Students learn ‘C’ programming, and with the available hardware libraries can ignore (for the most part) the low-level register bit manipulations that are actually occurring. A final lab project was used to incorporate the computer board plus motor and sensing circuits into an autonomous robot.

The intermediate embedded course is more in depth, where the students build their own computer (PCB) board, learn to surface mount components and design software that interfaces directly to the hardware. Students need to understand manufacturer’s datasheets and learn how to use ‘C’ to directly address the microcontroller’s internal status and control registers. Topics such as processor interrupts, (digital) input debouncing and A-to-D filtering are also covered. Each lab builds on the knowledge gained from the previous applications, and the final lab project is of their own choosing. Examples of final projects include:

- Outreach games such as Nim or Simon, all of which interact with the user via push

buttons, LED's as well as the LCD display. Some games also used the piezo to play notes or music.
- Rock/Paper/Scissor game that uses an XBEE so that the two players can interact without wires.
- GPS location finder to use for Geocaching.
- Traffic light controller (used one of the school's traffic light modules)
- Elevator control module (used one of the school's PLC elevator modules)

The last course (advanced programming) is in embedded Linux, where the students learn to use operating system calls, plus threads, fork, shared memory, semaphore, sockets, and pipes. In addition to the operating system aspects of applications, students get more depth to their 'C' programming studies in learning data structures and pointers. Class projects include algorithms such as queuing, sorting and binary search techniques. Similar to the other programming courses, the final lab project is one in which the student gets to choose. Examples of final projects include:

- Outreach games such as a "laser" target game
- Traverse a 2-D maze
- Motion sensor alarm
- Wave file playback

One of my teaching philosophies is to have a final lab project in which the student must come up with the end goals. In most of the prior labs, the development process turns into a simple recipe that is explained to the students prior to the lab. By having a unique and student driven project, the lab takes on a special role in that the student is utilizing design, analysis and communication (all critical thinking) skills. Thus, the assessment of the final project is an excellent measure of the course material.

Linux Board Selection

There are many embedded Linux development boards that can be used in a university setting. An abbreviated list is shown below:

Raspberry Pi [2] ($35)
Glomation Inc [3] (GESBC-9G20U) ($55)
BeagleBone Black [4] ($55)
BeagleBone Green [5] ($45)

All of these boards were examined and comparisons were made based on processor speed, memory, I/O capabilities, and compiler/development environments. The end result was that the BeagleBone Black was selected for use in the advanced programming course (see Table 1).

Table 1 – Linux PCB Comparison

Linux Board	**Pro's**	**Con's**
Raspberry Pi	Low cost	Few I/O Pins
		Needs an external compiler
		Needs H/W interface to run RS-232
Glomation	Large number of I/O pins	Needs an external compiler
	Has an RS-232 admin console	Reading I/O pins is very slow
		Need to rebuild kernel for A-to-D
BeagleBone Black	Large number of I/O pins	Slightly high cost ($55)
	Has female headers (easy to wire)	
BeagleBone Green	Large number of I/O pins	Uses "seed studio" peripherals
	Medium cost	

The hardware on the board is a 32-bit, 1 GHz Arm (Cortex-A8) processor, and includes 4 GB of flash, 512 MB of ram and has an on-board floating point accelerator. The cost of the board is reasonable ($55) for a student to purchase/own the board. It has a large assortment of I/O pins, supports digital I/O, A-to-D inputs, PWM outputs, RS-232, SPI and I2C serial protocols, has an Ethernet port, USB support and comes with its own ‘C’ compiler and development environment.

In an academic environment, this board is ideal as an embedded Linux teaching platform. The wide assortment of I/O pins allows a variety of lab projects, the cost is reasonable for a student to purchase the board and it comes with its own on-board ‘C’ compiler; and there is no problem of compatibility between a school’s Linux cross compiler and the target hardware.

This was the first year that the BeagleBone board was used, and both the students and the instructor had to learn and use the development tools, and understand the board’s capabilities. Over the 15 weeks of the course, the students had to complete labs that used digital I/O, analog input, PWM outputs, RS-232, and TCP/IP. Throughout that time, the students were mastering data structures, pointers, shared memory, forks, threads, and semaphores. The final lab was one in which the students could individually pick an end goal and design an application (both hardware and software) to achieve that goal.

To assist the students, a set of class notes (developed prior to the class) were printed and passed out at the beginning of the semester. The class notes included operation of the board, techniques to access the board’s I/O pins, the operation of the development environment, compiling and general application information regarding semaphores, shared memory, forks, threads and pipes.

BeagleBone Board / Development Environment

The BeagleBone is composed of three microprocessors (see Figure 1). The main processor (32-bit Arm Cortex) runs the Linux operating and can be used to compile and run applications. There are two other co-processors or PRU's (programmable real-time unit) that can also interface to the shared memory and most of the peripheral I/O pins. The purpose of the PRU's is to run dedicated applications that can process I/O data in real-time and then pass the results over to the main Arm Cortex processor. It was not the intention to have the advanced programming course cover the mechanism for compiling and downloading to the PRU sub-system.

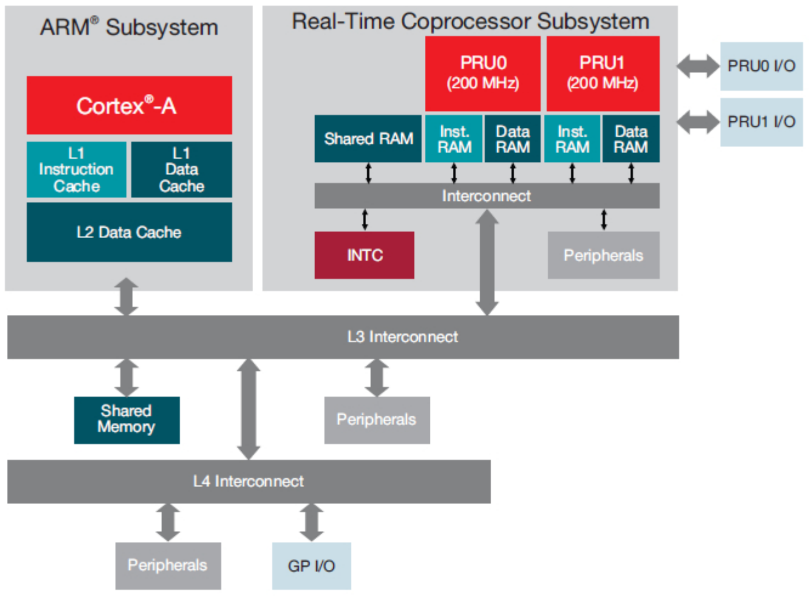


Figure 1 – BeagleBone Processor Architecture [6]

The embedded Linux operating system supports 'C' development along with Java, Python and Linux shell scripts. There are several ways in which to access the I/O pins:
- Python
- Java
- BoneScript (an extension of Java Script)
- C Application (direct hardware access)
- C Application (operating system/file access calls)

A decision was made to focus the course in accessing the I/O pins via a 'C' program, and specifically, using the file system of the BeagleBone.

Once the BeagleBone drivers have been installed, the BeagleBone can be connected directly into the PC using a standard USB cable. For the most part, the board can run off the power from the PC (see Figure 2).

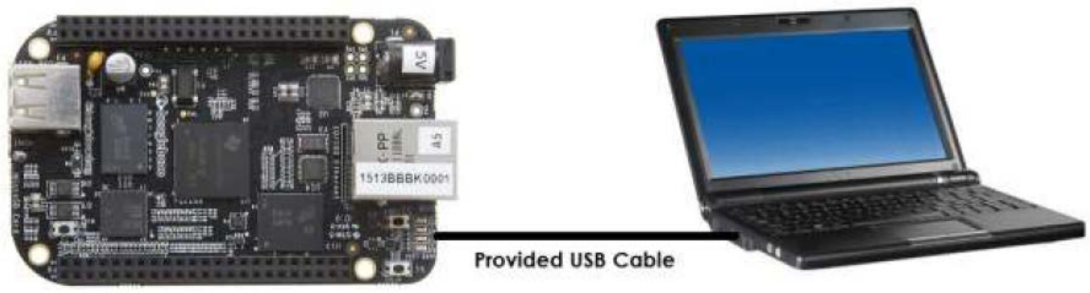
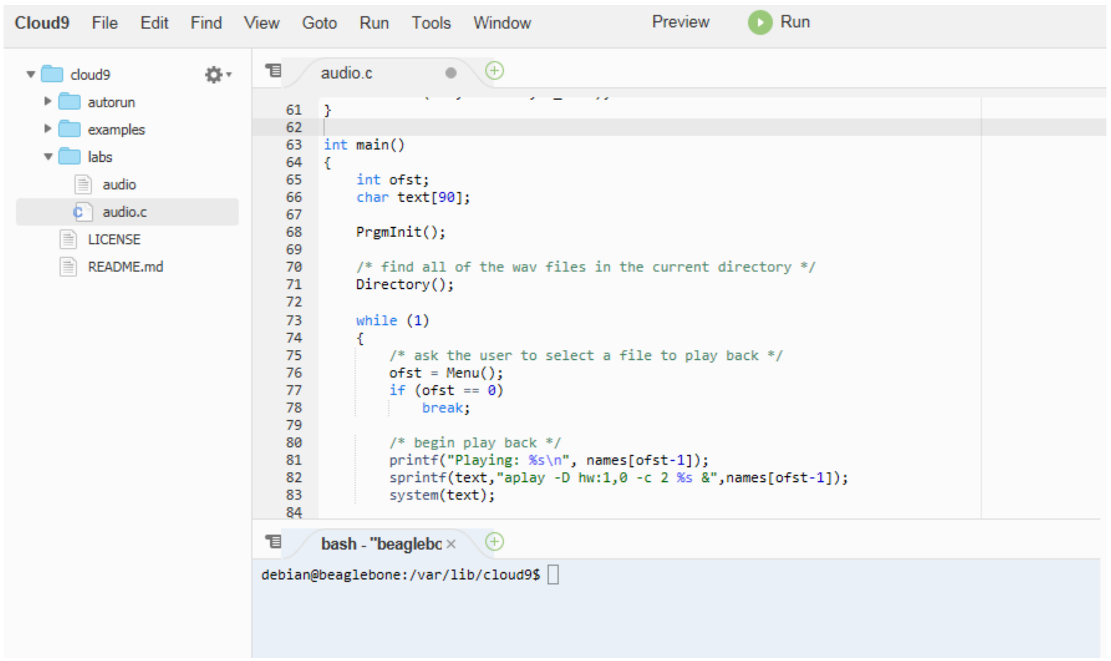


Figure 2 – BeagleBone to PC (USB) Interface Connection [7]

The simplest method of developing applications that run under the BeagleBone's Linux operating system is to edit and compile on the board itself. The development environment is Cloud9 (which interfaces to the PC via a web browser (such as Firefox or Chrome). By accessing the web address: http://192.168.7.2:3000 the web browser connects to the BeagleBone board via the USB cable and brings up a development environment that allows the student to scan through the on-board directories/folders, open/create as well as edit files (similar to that of Visual Studio) and also provides a command line console for compiling and executing programs (see Figure 3).



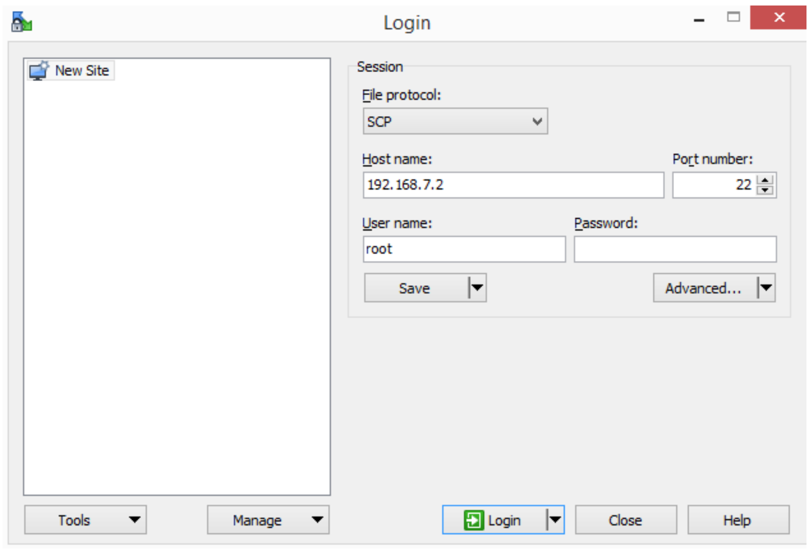
```
}

int main()
{
    int ofst;
    char text[90];

    PrgmInit();

    /* find all of the wav files in the current directory */
    Directory();

    while (1)
    {
        /* ask the user to select a file to play back */
        ofst = Menu();
        if (ofst == 0)
            break;

        /* begin play back */
        printf("Playing: %s\n", names[ofst-1]);
        sprintf(text,"aplay -D hw:1,0 -c 2 %s &",names[ofst-1]);
        system(text);
```

```
debian@beaglebone:/var/lib/cloud9$
```

Figure 3 – Cloud9 Development IDE

Once the application has been debugged, the code can be brought back to the PC using WinSCP [9] (secure copy protocol). This is an open source free file transfer program that can communicate using FTP (file transfer protocol) as well as SCP and SFTP.

Initializing WinSCPis fairly straight forward as the BeagleBone listens on 192.168.7.2, port 22 and the user can simply log in as "root" (see Figure 4).



File protocol: SCP

Host name: 192.168.7.2

Port number: 22

User name: root

Password:

Figure 4 – WinSCP Startup

Once logged in, the student can click/drag files from the PC to the Linux board, or from the Linux board to the Windows PC.

Lab Projects:

The advanced programming course was structured to teach topics in ‘C’ programming as well as the Linux operating system. A lab was 1 or 2 weeks, and it included both a hardware component and a ‘C’ programming topic. For example, the first lab used the digital I/O pins along with disk file accessing. A second lab was to use analog inputs along with threads. A third lab used PWM and data structures. This continued with RS-232, TCP/IP along with fork, shared memory, semaphores, etc.

One question that instructors may have is for a design project (one that is beyond the material for the class), will the students have the knowledge base (and motivation) to go out and get additional information to complete an application? To test this, the final lab project of the course is one in which the students needed to implement something of their own choosing. As an incentive, “complexity” points were assigned as a means of evaluating if the student actually did take the time to learn new material.

The end goal for this paper was to assess the BeagleBone as a basis for the advanced programming course. Two methods in which to accomplish this are:

(1) Let the students come up with their own final lab project and then evaluate the results for complexity, interaction, and hardware components.

(2) Provide a written survey at the completion of the course and have the students compare the tools, hardware and applications to other microcontroller courses that they have taken.

Both of these techniques were used and sampled of the final lab projects are included below.

The students were asked to write up a description of the project, as well as include a hardware schematic and software flowchart. Along with a sample of their project, the ABET / ETAC outcomes are also described [10].

Embedded Systems Design – Audio Playback

In this lab, the student's goal was to build a functional music player. The concept of playing back audio was not covered in class, nor is there an audio output jack or codec present on the BeagleBone board. The mechanism to play back an audio file had to be researched and implemented.

This demonstrated the exact purpose of the final lab. A student used their existing knowledge base (from class) to further their abilities in the course material. Sadly, one of the problems with the BeagleBone is that there is no one depository of knowledge, and in that regard, it mirrors that of the Raspberry Pi. The student had to look at the boot-up configuration files, research the alsa-sound libraries and then had to merge this information with their own application.

The hardware implementation is fairly simple, using a pair of Logitech S-150 speakers, just plug this into the USB port on the BeagleBone (see Figure 5). The student then chose to modify the uEnv.txt (boot-up) configuration file to disable the HDMI interface and enable the USB speakers.

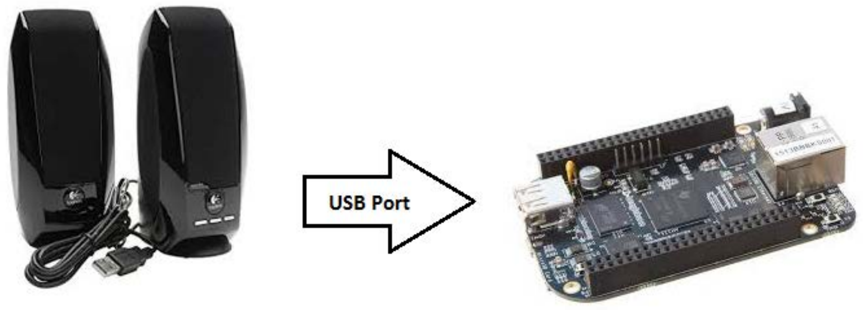


Figure 5 – BeagleBone to Logitech Speakers [7,8]

The software application that the student wrote used alsa-utilities (**alsamixer** and **aplay)** to set the volume and begin the playback. The utilities were spawned from a 'C' application, thus simplifying the playback process. The user would interact with the program via the console and could initiate playback / advance to the next song in a pre-determined list.

Embedded Systems Design – Maze Game

In this lab, the student's goal was to design an interactive game, where a "maze" will be drawn on an external LCD display and the maze will be traversed using a joystick. The end goal would be to reach the exit of the maze. The student selected a 4x20 character display that interfaced to the BeagleBone over RS-232. The maze walls were drawn using simple line/dash symbols (see Figure 6) and a joystick was used to "point" in the direction to move.

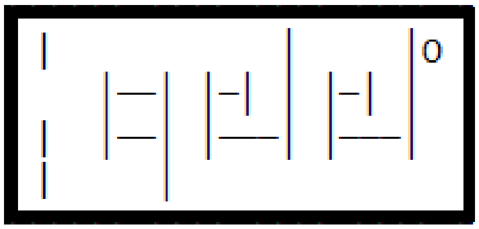


Figure 6 – Maze Game LCD Display Example

The hardware implementation used RS-232 asynchronous communications (to the display), analog input (from the joystick) and a single digital input (from a "start" button). All of the basic I/O elements were covered in class, and while the lab is not overly aggressive (in terms of complexity), it does represent a summation of the course.

Embedded Systems Design – Accelerometer Interface

In this lab, the student's goal was to interface the BeagleBone to an ADXL345 (3-axis) accelerometer using I^2C. The application was take the acceleration data and create a motion sensor that will output an alarm (plus log an entry with a time/date stamp) when the board was moved, and to clear the alarm once the board had stopped moving.

Similar to the first lab, the student had to perform some research to complete the lab. The I^2C interface was not covered in class and the student needed to learn how to design the hardware and software to accomplish his goals. The schematic (see Figure 7) included the ADXL345 accelerometer, a tri-color LED and a 4x20 LCD display.

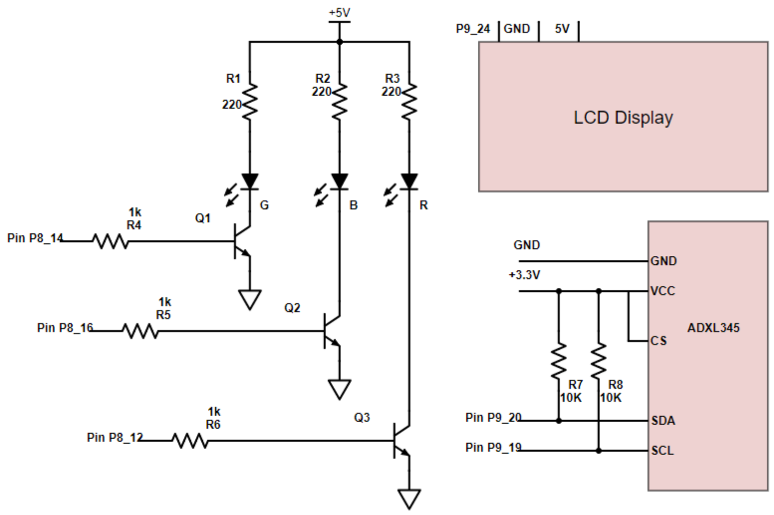


Figure 7 – Schematic for the motion sensor application

The complexity of this application centered on the I^2C interface, as this was the focal point of "new" knowledge to the student. While the actual algorithm was not overly difficult, the lab demonstrated the fact that the student could perform research and acquire the knowledge needed to complete the requirements of the application.

ABET ETAC Outcomes

Once completed, each of the final lab projects could be used as instruments for a variety of ABET ETAC [10] student outcomes, five that are suggested include:

Student Outcome-B
Apply knowledge of mathematics, science, engineering, and technology to engineering technology problems that require the application of principles and applied procedures or methodologies.

Student Outcome-D
Design systems, components, or processes for broadly-defined engineering technology problems appropriate to program educational objectives.

Student Outcome-F
An ability to identify, analyze, and solve broadly defined engineering technology problems activities.

Student Outcome-H
An understanding of the need for and an ability to engage in self-directed continuing professional development.

Program Specific (CMPET and EET) Student Outcome-A
Application of circuit analysis and design, computer programming, associated software applications, analog and digital electronics, microcomputers, operating systems, local area networks, and engineering standards to the building, testing, operation, and maintenance of electrical, computer and software systems.

Assessment

A survey was provided out to the students upon completing their final lab project. Comparing the BeagleBone (and Linux) to their previous programming course (PIC microcontroller), the students were asked to compare and rate their experience. There were four students that completed the survey. The scale was 1 to 7 on each question (see Table 2):

Table 2 – Survey Results

Questions	**Results**	
1. Compared to the Pic Microcontroller, how easy was the BeagleBone hardware to learn and understand?	6.0	1:difficult, 7:easy
2. Compared to MPLAB, how easy was the Cloud9 development IDE to bring up and understand?	6.25	1:difficult, 7:easy
3. Compared to the Pic hardware, was the BeagleBone board reliable?	5.5	1:not reliable, 7:reliable
4. Compared to the PIC 18F2520 micro-controller, was it easy to write an application using the Linux operating system?	6.25	1:difficult, 7:easy
5. Was the BeagleBoneCProgramming.pdf document helpful to the course?	5.25	1:not helpful, 7:very helpful
6. How many additional hours were spent (on an average) to complete each lab?	5.75 hours	
7. Do you have an interest getting a job/career that uses Embedded Linux?	6.25	1:none, 7:a lot
8. To successfully complete this course, what pre-requisites (from CMPET 355) should be stressed?	Add data structures to the CMPET 355 course, More work on bitwise operators and hexadecimal arithmetic	
9. Any comments or suggestions?	Add Python, Add Java, Add an audio component to the labs	

Summary

The BeagleBone Black is excellent for an academic setting, it is low cost (approx. $55), has a large assortment of I/O pins, supports digital I/O, A-to-D inputs, PWM outputs, RS-232, SPI and I2C serial protocols, has an Ethernet port, USB support and comes with its own ‘C’ compiler and development environment. The hardware on the board is a 32-bit, 1 GHz Arm (Cortex-A8) processor, and includes 4 GB of flash, 512 MB of ram and has an on-board floating point accelerator.

In an academic environment, this board is ideal as an embedded Linux teaching platform. The wide assortment of I/O pins allows a variety of lab projects, the cost is reasonable for a student to purchase the board and it comes with its own on-board 'C' compiler; the student does not have the problem of compatibility between a school's Linux cross compiler and the target hardware. Using the class notes that were made available to the students, they were able to quickly learn the development environment, create their schematics and debug the code.

From the assessment (survey), the main change to the course would be to improve the class notes that were provided to the students. Sections that will be added include:

- How to access the manual pages that are present on the BeagleBone
- How to configure and operate the USB based audio speakers
- How to download and load the latest Debian Linux image
- How to create additional users

Additional hardware (audio speakers) will also be provided and integrated into the labs.

Bibliography

[1] Web Site: https://beagleboard.org/black/
[2] Web Site: https://www.raspberrypi.org/products/
[3] Web Site: http://www.glomationinc.com/products/index.php?n=SBC.GESBC-9G20u
[4] Web Site: https://www.sparkfun.com/products/12857
[5] Web Site: https://beagleboard.org/green
[6] Image downloaded from: http://linuxgizmos.com/beaglebone-cape-eases-access-to-the-sitara-socs-pru
[7] Image from BeagleBone Black System Reference Manual: https://cdn.sparkfun.com/datasheets/Dev/Beagle/BBB_SRM_C.pdf
[8] Image downloaded from: www.amazon.com
[9] Web Site: https://winscp.net/eng/docs/introduction
[10] Web Site: http://www.abet.org/accreditation/accreditation-criteria/