2018 ASEE ANNUAL CONFERENCE & EXPOSITION
JUNE 24–27, 2018
SALT PALACE CONVENTION CENTER
SALT LAKE CITY, UT

Paper ID #21623

Single-board Computer Used for Network Streaming Audio Player TFT Touchscreen-based Application

Dr. David Border, Bowling Green State University

David A. Border, Ph.D., holds a principle research interest in electronic information systems. This field includes digital communication and networking and intelligent networked devices. His work includes wireless sensor networks. Prior research included work on signal bandwidth compression and signal specific data encoding techniques. His technology application interest includes networked systems. Typical teaching duties include junior- and senior-level courses in the Electronics and Computer Engineering Technology (ECET) program. Within this course set are the curriculum's networking and communication courses. As is true with his ECET faculty colleagues, Border supports the program with teaching assignments, as needed, in freshman- and sophomore-level courses offerings. Examples of these include the sophomore level electric circuits and digital electronics courses. Border teaches a digital communication graduate course within a Ph.D. Consortium Technology Management program, as well as other graduate level courses at BGSU.

Border served as interim department chair of the Engineering Technologies department. He served as chair of the university Faculty Senate curriculum and academic affairs committee. He is chair of the University Faculty Senate.

Single Board Computer used for Network Streaming Audio Player TFT Touchscreen-based Application.

New microcontrollers have become widely available that far exceed the capabilities of microcontrollers marketed a decade ago. Because of increased bus sizes that boost memory addressing and data transfer rates, the new microcontrollers are capable of hosting an OS system. Thus, they have transitioned from hosting dedicated applications to hosting many applications. Often these devices are referred to as single board computers.

The physical product created herein was a network streaming audio player on a single board computer. The hardware platform chosen was an Odroid-C1. The board features a multicore RISC architecture ARM processor. Like other single board computers of its class, it features much RAM, as well as eMMC memory which acts as "disk" memory. The base unit features an abundant array of connections to the outside world, including 40 GPIO pins, four USB ports for keyboard, mouse, WiFi USB device, plus separate RJ45 Ethernet jack, plus micro-USB and micro-HDMI ports. Odroid markets a platform compatible 3.2" TFT touchscreen display for use with the C1. In this project, the programmed touchscreen provides a user-friendly kiosk-like interface. The HMDI port acts as the streaming audio output port.

Project software development intentionally sought to leverage the strengths of a fast and lightweight Ubuntu Linux variant, Lubuntu. QtCreator is the software Integrated Development Environment (IDE) for programming, chosen for its Graphics User Interface (GUI) programming reliability. As a whole, the completed project integrates diverse software development tasks, from program scripting, to work with windowing systems, to event-driven software. The skill sets needed by the project are enumerated. The implementation of a network streaming audio player was not a strenuous task. The working product is reliable and user-friendly. Two different course sections received a lecture on the completed work. The students were surveyed and the survey results were analyzed.

I. Introduction

Faculty who must deliver upper-level microcontroller/microprocessor curriculum content have relied on producing component level skills and aptitudes in the students. It matches well with the subject matter and presentation of knowledge in textbooks. Reliance on developing electronics and computer component level knowledge to the exclusion of other knowledge has its critics [1].

Broadly said, this work addresses, in part, the program's electronics and computing faculty concern that the program must make room for "timely content" in our semester length microcontroller/microprocessor curriculum. It must reach for a theme that could not be reached by simply gluing component skills together. It must exercise “integrating skills” helpful to success in senior projects and internships. It must not consume too much class time or laboratory resources.

The general theme of the work is a kiosk-style music player. The player's "interface" is touchscreen-based; powered by a microcontroller-based single-board computer. Table 1 lists the desired attributes of a single board computer and its operating system.

Rugged Hardware	Low Cost
No Hardware Modifications Required	Rich Development Environment
	Rich Device Management

Table 1. Desired Machine Attributes

The hardware platform chosen was an Odroid-C1 [2]. It was a single-board computer and worked at room temperature with no heat sink or fan. Like competitors products [3, 4, 5, 6] in academic use, it was low cost, under $120 with touchscreen included. No hardware modifications were needed. Its four USB ports, one HDMI port, one mini-USB port and one RJ45 Network jack allowed ample I/O.

The audio theme seemed tailored to the Odroid as the C1 onhand had successfully been tested playing audio streams, (however it was less than a total success with video media players). The board features a multicore RISC architecture ARM "microcontroller family" processor.

Its Linux kernel and Lubuntu distribution proved a productive development environment and the device management (Linux device manager “udev”) handles all needed peripherals. From prior work, the Linux distribution exhibited good system “responsiveness” during web-browsing (providing video streaming was avoided!), and during use of desktop applications, such as office suite software.

The work of the project divides into four tasks, shown in Table 2. Subsequent sections examine each task.

Identification of Student Knowledge Set (Needed Knowledge)
Creation of Network Streaming Audio Device
Consideration of Placement of Work into existing Microcontroller/Microprocessor Course
Presentation of Work to Students

Table 2. Project Tasks

The goals of the work are straightforward. The project equipment should be found suitable for laboratory work. The skill sets needed by the project should be enumerated. The implementation of a network streaming audio player should not be a strenuous task. The working product should be reliable and user-friendly. The demonstrated product should appeal to students.

II. Needed Knowledge

What knowledge is needed to complete the project? What is likely known by students? What is likely unknown by students? These are questions Table 3 addresses. Remedies, where needed, are cited.

Potential Knowledge Categories	Known by Students?	If Yes, Explain	If No, Cite Remedy
Connecting peripheral devices	Yes	General knowledge	
Installation of OS on Odroid, TFT Touchscreen Support, etc.	No		Pre-install OS on eMMC memories
Installation of Applications on OS	No		Pre-install applications on OS
Use of Linux configuration files	No		Provide examples and references for the few configuration files needed in the project
Use of Linux automatic program startup	No		Provide reference on how to cause auto program startup
Use of Linux scripting	No		Provide instruction in lecture
Knowledge of the Linux Command Line Interface	No		Provide information in class; information can be modest.
Knowledge of Linux Device Naming and File Systems	No		Provide information in class; information can be modest.
Knowledge of Media Players	Yes	Gen. Knowledge	
Knowledge of Window managers	No		Provide instruction in lecture
Use of GUI designers	No		Instruct students. Also make comparisons to Microsoft Visual Studio, National Instruments Circuit Designer, and Altera Quartus II Block Editor FPGA designers seen in prior courses.
Use of C programming inside GUI designer	Yes	Introductory C++ course / Microsoft Visual Studio	

Table 3. List of Knowledge of Potential Importance

As indicated in the table, a prerequisite for the existing microcontroller/microprocessor course is an introductory course on C++ programming. The course textbook used is Gladdis' "Starting Out With C++ from Control Structures to Objects." The students make use of Microsoft's Visual Studio development integrated development environment (IDE). This IDE-based work should be adequate preparation for students to write code in the GUI designer. However, the students have no prior course navigating the designer. Instead, they have general experience navigating the National Instruments Circuit Designer and Alteris Quartus II FPGA designer. Success using the GUI designer will require a modest amount of classroom student instruction.

Since Linux distributions include a user-friendly GUI desktop environment, the table 3 entries concerning the "Linux Command Line Interface" and "Device Naming and File Systems" represent somewhat limited knowledge bases. Linux configuration files, automatic program startup, and Linux scripting are contextually in proximity to one another. Each speaks to the "unobserved" way in which a Linux Developer gets useful work done. The number of configuration files discussed in this work is small.

In Table 3, two "knowledge-based" skills necessary for project completion are not class content material. Hidden from the student is specialized preparation work on the master eMMC memory modules (Figure 1 shows an eMMC module plugged into the Odroid). The module preparation is:

a) Install the Linux distribution. It is an "inflate and install" process.
b) Add specialized software applications needed for programming.
c) Add hardware drivers and configuration needed for specialized hardware.

Note: In practice two eMMC modules needed preparation. One module for code-development work (items a&b, above), the other for the actual "target" kiosk device (items a&c, above). The later eMMC module will have the 3.2" TFT touchscreen hardware drivers installed, and the OS configured to treat the screen as the "main" system monitor.

Figure 1. Linux Distribution eMMC Module Plugged in the Odroid Single Board Computer

III. Creation of Network Streaming Audio Device

QtCreator [7] is the GUI designer chosen for the project. It is a stable cross-platform application that is both versatile and mature. The design was laid out rapidly to establish the inputs and outputs for the front-end graphic as it is the operator's interface for the network streaming audio application. Shown in Figure 2 is the design layout. Placing the "pushbutton" components design buttons onto the window is a drag and drop operation. The layout uses seven "pushbuttons."

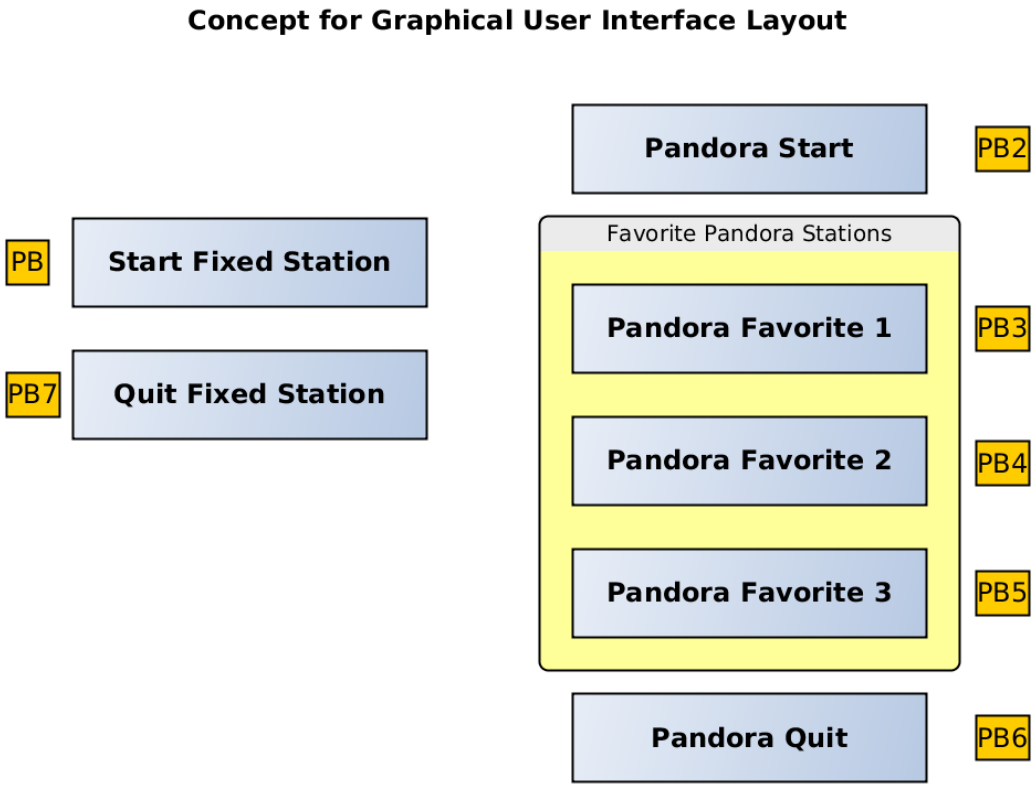

Figure 2. GUI design

The QtCreator "Widgets" design application builds the various files for the project. Figure 3 shows the components.

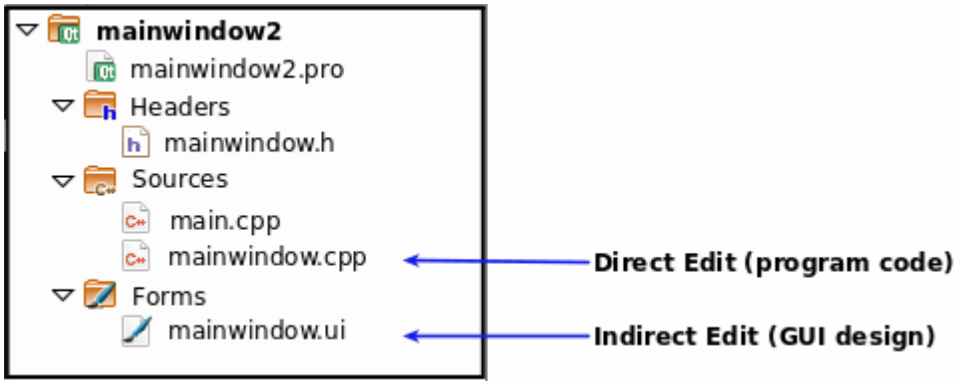

Figure 3. Project Files within a Widgets-based QtCreator Project

The "mainwindow.pro" file, the "mainwindow.h" file, the "main.cpp" should not be hand-edited by the programmer. The mainwindow.pro file tells the compiler which files are source files, headers, and so forth. The mainwindow.h file references each container's graphic-objects (so-called "public slots"). All are "pushbuttons" for this work. The mainwindow.ui file is coded indirectly by the programmer. Its contents result from the designer's "drag and drop" work. The programmer edits the mainwindow.cpp program file. It has the handles for the pushbutton events, such as "button press."

The mainwindow.cpp content customizes the GUI. A series of flowcharts were drawn to direct the program coding. Figure 4 shows the flowchart for the coding of Pushbutton “PB” (refer to Figure 2, for the location of PB).

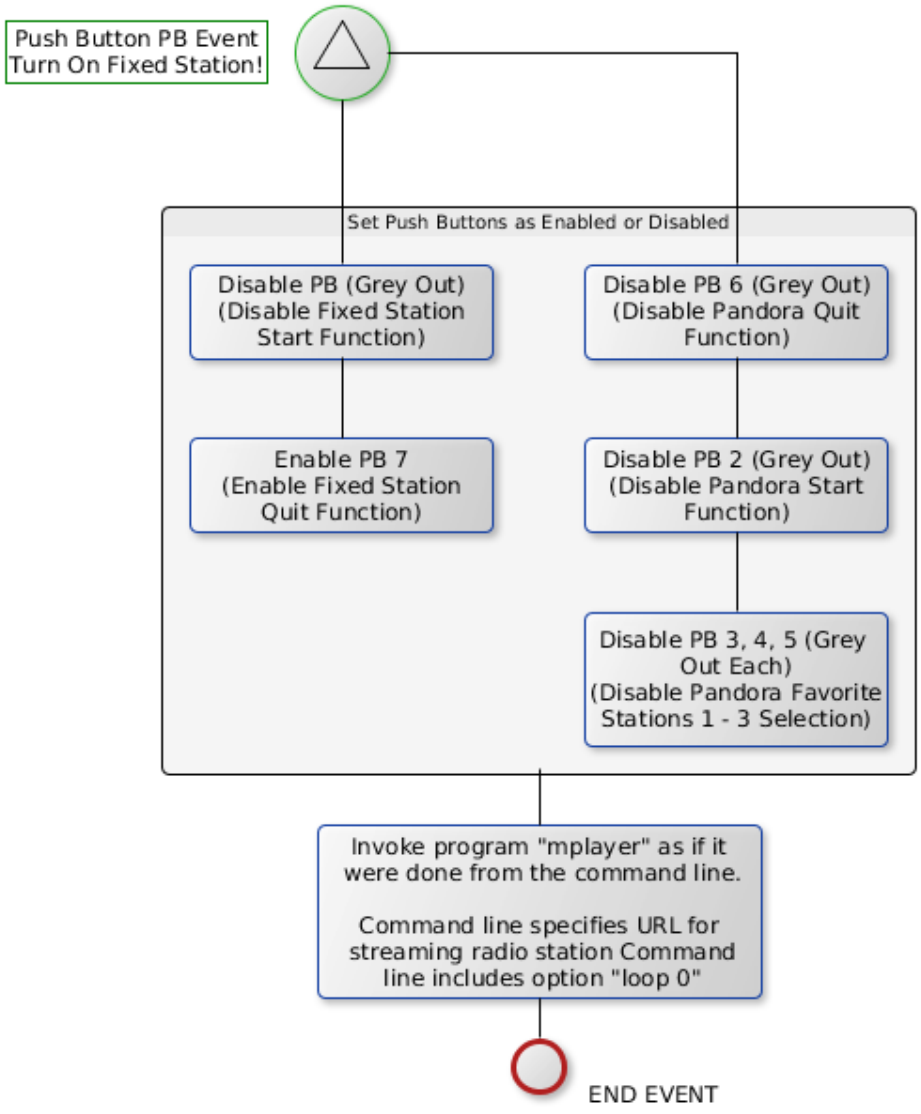

Figure 4. Flowchart for Pushbutton PB

Each flowchart block needed implementation in C. A few of the flowchart blocks use the "system" function call (found in "stdlib.h"). Since the work implements a pushbutton-navigated user interface, there are coding commonalities throughout the work. Table 4 shows sample C language instructions used.

Command	Function
ui→ pushButton →setEnabled(false);	Turn off pushbutton PB (disables and dims).
ui → pushButton_7 → setEnabled(true);	Turn on pushbutton PB7 (enables and brightens).
sleep(1);	One second delay. Programmatic delays are sometimes necessary when interacting with a user-operator.
system(“sudo -u odroid mplayer -loop 0 -playlist http:// (etc., etc.)”);	Start mplayer. Play audio stream from given URL. Loop on “reconnect” if connection is lost.
system(“killall -9 mplayer&”);	Kill any instance of a mplayer
system(“sudo – u odroid pianobar&”);	Start pandora application (run in background)
system(“sudo -u odroid echo -n ‘s’ >/home/odroid/.config/pianobar/ctl&”);	Send command to pandora control file to force change of station dialog.
system(“sudo -u odroid echo ‘0’ >/home/odroid/.config/pianobar/ctl&”);	Send command to pandora control file to send a zero (in the context of selecting station “0”).

Table 4. Sample C Language Commands used to Implement Streaming Logic

Pianobar, the Pandora executable program, was tested separately before writing the QtCreator design. The pianobar configuration file (i.e., /home/odroid/.config/pianobar/config) was edited to contain the proper connection information including the username, password, and the web client security string that must match the security string at the Pandora server. Also, the operation of the fixed FM station was tested separately using mplayer. Tests used the Linux command line interface.

QtCreator program coding is written into mainwindow.cpp to handle object events. The application determines the required initial conditions. For this work (1) the fixed FM station button and the Pandora Service button are initially enabled, and (2) remaining buttons are initially disabled. See Figure 6. After program start, the event-handling code determines work dynamics. The event handling actions for all pushbutton events were coded. Chief features used are pushbutton enable/disable (brighten/dim). The project was debugged and tested. The coding work was successful.

Figure 6. Starting Point for Operation, 5 Pushbuttons “Not Enabled”

Work included making the player available via a "clickable" desktop text file. The network streaming audio application was then successfully started on a "mouse button click." That same desktop text file was copied to the"/.config/autostart" system directory. Application desktop text files in that directory automatically start after boot. The system was rebooted and tested. The network streaming audio player's operation from boot worked well.

IV. Consideration for Placement of Work into existing Microcontroller/Microprocessor Course

It is likely that most microcontroller/microcontroller courses teach the Serial Peripheral Interface (SPI). Teaching the Odroid's 3.2" TFT touchscreen in depth means teaching SPI. The LCD touchscreen module uses the well known Iliek driver chip. The Iliek chip uses SPI [8]. Figure 8 shows interface specifications. For classroom illustration, captures and dumps of SPI data traffic going from the touchscreen to the OS can be shown using a Linux command line program.

SPI Interface TFT. Pinout for ILITEK, ILI9341 TFT LCD Single Chip Driver

Pin #	Symbol	Description
1, 17	3.3V	Power positive (3.3V power input)
2, 4	5V	Power positive (5V Power input)
3, 5, 7, 8, 10, 22	NC	NC
6,9,14,20,25	GND	Ground
11	TP_IRQ	Touch Panel interrupt, low level while the Touch Panel detects touching
12	KEY1	Key
13	RST	Reset
15	LCD_RS	LCD instruction control, Instruction/Data Register selection
16	KEY2	Key
18	KEY3	Key
19	LCD_SI / TP_SI	SPI data input of LCD/Touch Panel
21	TP_SO	SPI data output of Touch Panel
23	LCD_SCK / TP_SCK	Serial clock of LCD/Touch Panel
24	LCD_CS	LCD chip selection, low active
26	TP_CS	Touch Panel chip selection, low active

Figure 8. SPI Touchscreen/Driver Interface to Odroid GPIO Lines

Based on the hardware used, student instruction on small consumer electronics communication peripherals is possible. For example, the network streaming audio device uses an HDMI monitor as the audio output device. A consumer electronics HDMI video/audio splitter module splits project output audio to, (1) a speaker set or, (2) a Bluetooth transmitter. Instruction concerning the design of the HDMI/splitter would be a good course topic; many microcontroller/microprocessor courses include the digital to analog converter (DAC) (the heart of the HDMI audio splitting work) in their course outlines. If a Bluetooth module transmitter is used to transmit audio, course instruction in the basics of Bluetooth might also have value.

Hardware/Software topic mix would also be of interest. Typical students are unaware of events in the past decade that have resulted in microcontrollers evolving from machines running code to machines running an operating system. With microcontrollers crossing from 16-bit architecture to 32-bit architect, the address space, and so forth, became appropriate to host Linux distributions. Instruction on Linux topics has value. The Internet of Things [9] and the Internet of Everything is making more room for Linux powered concepts.

The software topic alone is an interesting topic to explore. While Linux can host the spectrum of computer language tools, spanning scripting, interpretive, and compiled languages, in this work coding centered on the use of the system function call. It was used to start and kill applications (Table 4). However, widening the scope of the project work widens the depth of student software work.

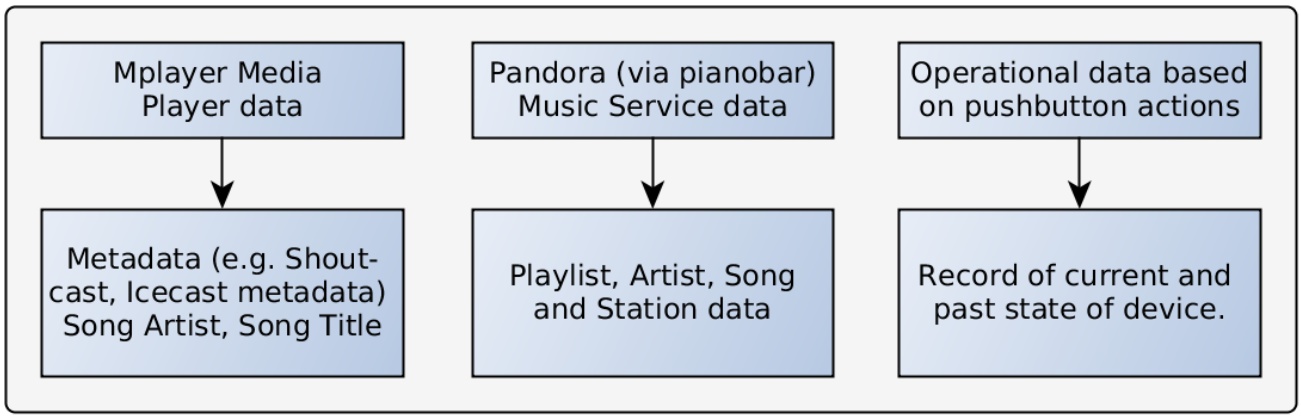

Figure 9. Data Collection and Analysis Opportunities

Consider the data collection (data logging) and analysis opportunities in the project (Figure 9). An active music feature (e.g., mplayer) produces text data that contain pertinent system information. Mplayer returns metadata about the current music by song artist and song title. Pandora (pianobar) returns, playlist, artist, song and station system information. Pushbutton event action handling in the GUI code provides the opportunity to record the current state of the device, such as “mplayer in use” or “Pandora in use.” Given that the network streaming audio program is named “XYZ,” invoking the capture of mplayer and pianobar output is done with a simple output redirect in the program startup command:

```
XYZ &>> ./music_log &
```

All mplayer and Pandora information is in the log file. The data can be programmatically analyzed. A possible companion task is to code a GUI program to display parsed data as current and recent song information.

V. Presentation of Work to Students

Two different course sections received a lecture on the completed work. The first was a junior level course section in microprocessors. The second was a freshman level “survey” course section in

electrical/electronic systems. The presentation included a slide-based lecture, a short demonstration of the QtCreator GUI designer tool and operation of the network streaming audio player.

The lecture divided into four themes, with themes shown in Table 5. Figure 10 shows the hardware for the demonstration. The hardware used was the Odroid C1 with WiFi, HDMI to analog audio converter module and external speakers. A remote desktop protocol (RDP) connection between the classroom computer/projection equipment allowed the Linux desktop environment to be displayed. The demonstration included QtCreator. Table 6 shows the Survey tool for the guest lecture. The tool used a five-point Likert Scale; see Figure 11. The student surveys outcomes are in Figure 12 and Figure 13. Surveys are analyzed the next section titled "Conclusion."

Themes	Work Goals
	Comparison of current laboratory hardware and project work hardware
	Capabilities of project hardware/software
	Developing GUIs with QtCreator

Table 5. Presentation Themes

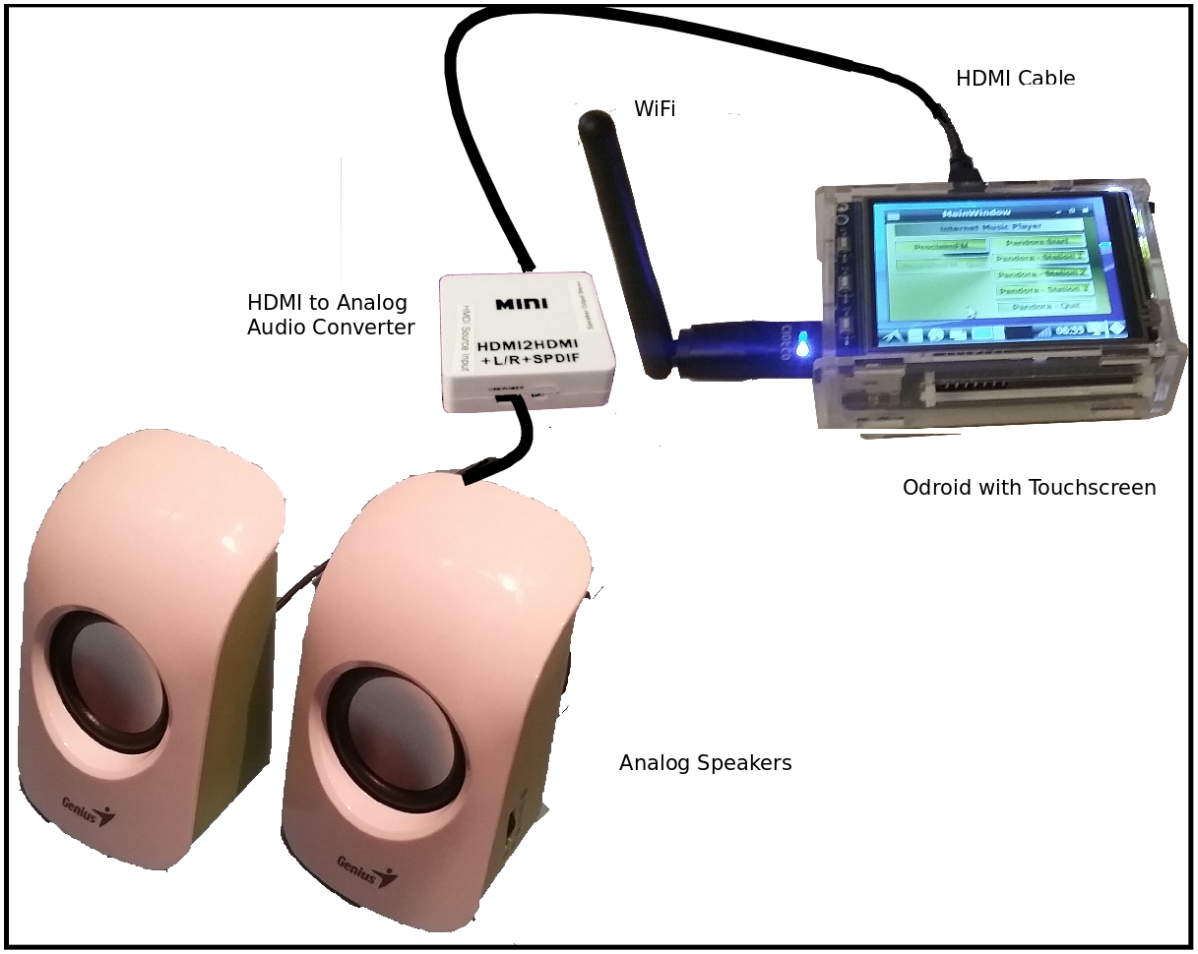

Figure 10. Demonstration Hardware

Figure 11. Likert Scale used in Surveys

Q1. Prior to this lecture I was aware of “single board computers” such as the Raspberry Pi and Odroid.
Q2. Prior to this lecture I was hoping to see lecture and lab content on “single board computer” in the program.
Q3. Based on this lecture lab content on “single board computers” should be in the program’s courses.
Q4. I learned from this lecture.

Table 6. Student Survey Tool

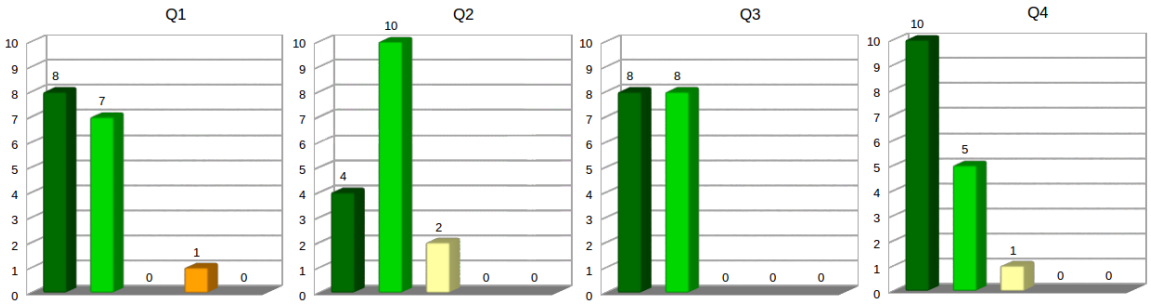

Figure 12. Survey Results, Audience: Junior Level Microprocessor Class Section

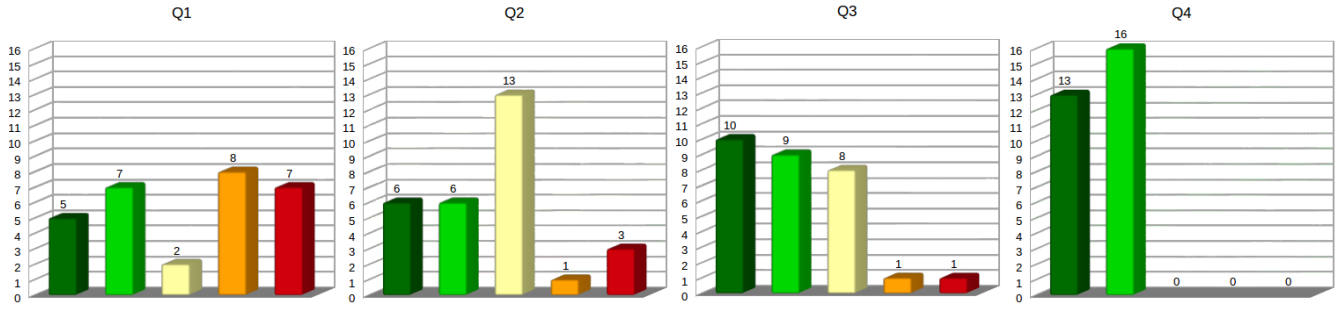

Figure 13. Survey Results, Audience: Freshman Level Electrical-Electronics Systems Class Section

VI. Conclusion

An Odroid-C1 had been on hand in the academic department office since the summer of 2015. It was used to run a “welcome center” display. It did that job nicely over a semester, executing a slide show in continuous mode. So, the device was not a total unknown, but close to it. However, working with the device in the laboratory proved enjoyable. The device has been in continuous use for two semesters and is extremely reliable. In code development work with QtCreator, the design application has never crashed. In development work testing media player applications, neither the application or machine crashes. When running the network streaming audio player, the only “crash” is at the network-side beyond the Odroid. Even then, the “-loop 0” switch, used when invoking mplayer, causes mplayer to patiently attempt a reconnection to the “external source” media server. The Lubuntu OS proved to be very stable.

During operation of the player, the touchscreen hardware and driver operated flawlessly. The limited screen-size (3.2”) was no hindrance for the project. There was ample space to place the seven pushbuttons. Highly user-friendly. For everyday general desktop operation the touchscreen is too small to be user-friendly. For general work, the Odroid-C1, with USB keyboard and mouse, connects to a monitor through the HDMI port. For the demonstration of the Odroid “look and feel” to an audience a remote desktop connection was used to provide a comfortable user terminal. To make the RDP-based connection, all that is needed is knowledge of the username, password, and Odroid’s IP address; a small GUI program was built to return the IP address of the Odroid-C1 with just a point and click.

The program development went quite well. QtCreator builds a very nice GUI form; dropping “widgets” into the workspace is easy. Working out the needed program logic when processing an event, such as a pushbutton action, is an interesting exercise. Table 4 illustrated program code examples used this application. Nothing insurmountable. The code making use of the “system call” is testable at the Linux command line interface (CLI). The project itself lends itself to an “incremental,” or iterative development methodology. The Linux command line tested all the media players before writing hard code. The GUI programming can initially implement just a few features; troubleshoot features; then recode for greater functionality. At no point in the project development was there scrapping of hardware, software, or OS distribution, or contacting vendors, because the project had hit a block.

The demonstration of the project to students was without difficulties. As expected, both demonstrations had no equipment failures. Both lectures had good attendance. Lecture material was identical for both presentations. As stated earlier, one presentation was to a freshman level class; the other was to a junior level class. Here are some relevant facts about the audiences:

Students were in mandatory program courses. Students in the freshman course include non-majors. Students in the junior level class should have already completed one semester-length internship.

Both classes indicate they learned from the demonstration/lecture (Q4). However, the classes show differences in their Q1, Q2, and Q3, responses. The junior level class response indicates nearly all students were aware (Q1) of single-board computers. The freshman class response indicates just over half the class were not aware (Q1) of single-board computers. Question Q2 rates a “program expectations” statement about single-board computers; a strong majority of the junior level class

indicates that share the “given” expectation. Whereas, the ambiance of Q2 for many in the freshman class is very evident.

Assuming the students read the question correctly, Question Q3 measures the persuasiveness of the demonstration. It, Q3, can be read in conjunction with Q2; using Q2 as the "before" case and "Q3" as the “after” case (see Figure 14). Based on the lecture contents, over half of the students in freshman class agreed that single-board computer content should be in the program. The junior class also contains the "Q2 to Q3" shift. All junior level students agreed that single-board computer content should be in the program.

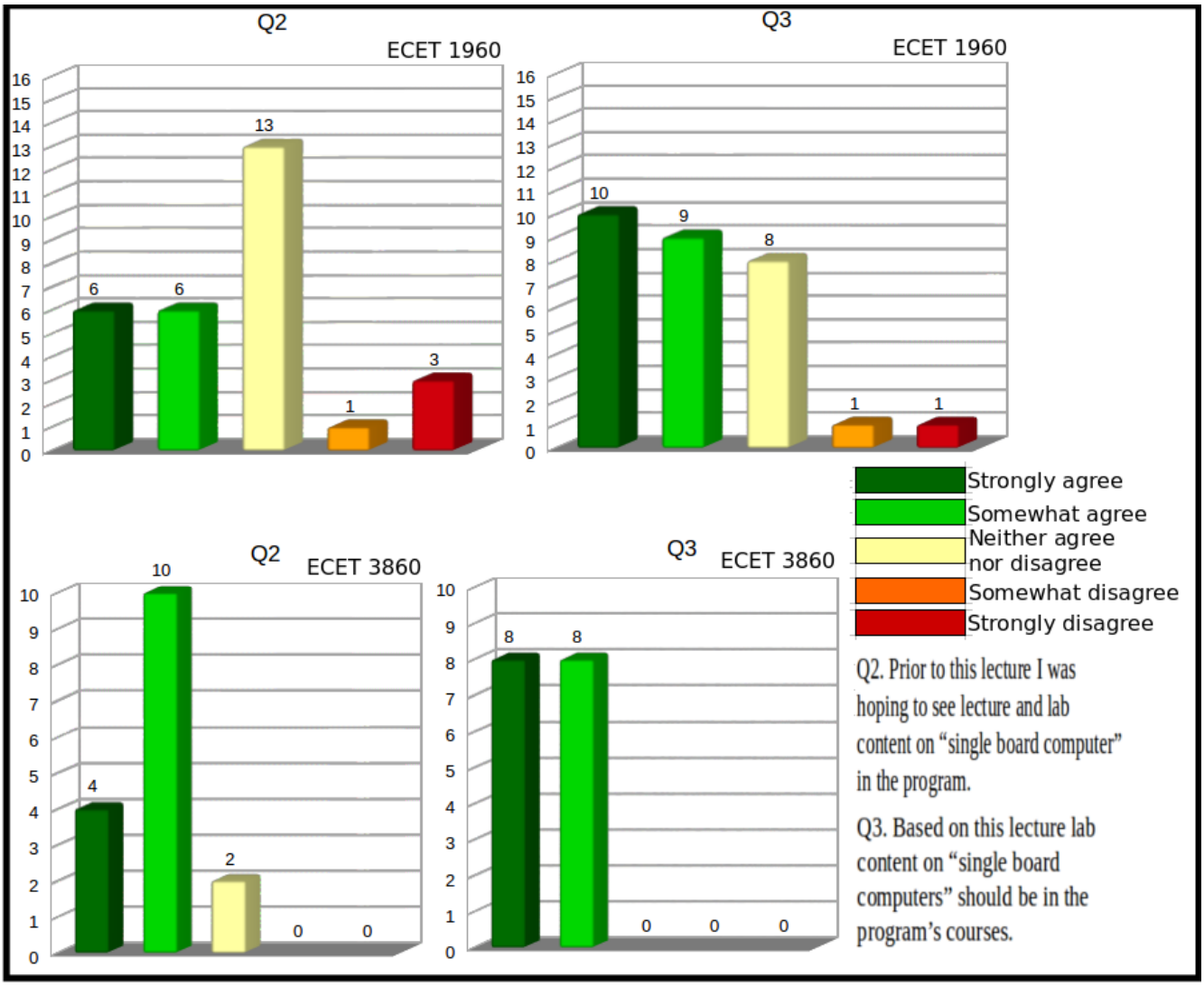

Figure 14. Student Audience Response for Q2/Q3 Question Pair

As a result of this work, we have all the necessary material for a regular one-class lecture in the freshman and sophomore courses mentioned. The next step will be to create a lab class experiences for each course. This project work is appropriate junior laboratory material. By subtracting the audio work and transposing the GUI interface into a “pushbutton” exercise, the work can be fitted easily into the freshman level course. The metadata and record of user pushbutton actions, mentioned in Figure 9, provides a real opportunity to have students code data collection and data analysis programs. Since the

Odroid can support web servers (e.g., Apache), any data synthesis or operational analysis work performed (and stored) could be made accessible to remote client devices. By extending the work and-or modifying the premise of the “network streaming audio” player work, a student could invent a new work proposal suitable for a senior project.

References

1. Chu, P. P., & Yu, C., & Hamlen, K. R. (2017, June), *Board # 23 : Integrating Computer Engineering Lab Using Spiral Model* Paper presented at 2017 ASEE Annual Conference & Exposition, Columbus, Ohio. https://peer.asee.org/27810

2. Roy, R., & Bommakanti, V., Odroid-C1 User Manual. Odroid Magazine, August 2015. https://magazine.odroid.com/wp-content/uploads/odroid-c1-user-manual.pdf

3. Husseini, N. S., & Kaszubski, I. (2017, June), *Incorporating the Raspberry Pi into laboratory experiments in an introductory MATLAB course* Paper presented at 2017 ASEE Annual Conference & Exposition, Columbus, Ohio. https://peer.asee.org/28514

4. Gilreath, J. W., & Bou-Saba, C. (2015, June), *An Advanced Streaming Internet Radio Player with Raspberry Pi* Paper presented at 2015 ASEE Annual Conference & Exposition, Seattle, Washington. 10.18260/p.23509

5. Sobota, D., & Karlovits, S. W., & Khan, A. S. (2017, June), *Senior Project Design: A Smart Pantry System* Paper presented at 2017 ASEE Annual Conference & Exposition, Columbus, Ohio. https://peer.asee.org/28820

6. Basu, D., & Purviance, J. S. G., & Maczka, D. K., & Brogan, D. S., & Lohani, V. K. (2015, June), *Work-in-Progress: High-Frequency Environmental Monitoring Using a Raspberry Pi-Based System* Paper presented at 2015 ASEE Annual Conference & Exposition, Seattle, Washington. 10.18260/p.25103

7. Vernier, M. A., & Wensing, P. M., & Morin, C. E., & Phillips, A., & Rice, B., & Wegman, K. R., & Hartle, C., & Clingan, P. A., & Kecskemety, K. M., & Freuler, R. J. (2014, June), *Design of a Full-Featured Robot Controller for Use in a First-Year Robotics Design Project* Paper presented at 2014 ASEE Annual Conference & Exposition, Indianapolis, Indiana. https://peer.asee.org/20260

8. Chu, P. P. (2017, June), *Integrating Computer Engineering Labs with "Video Theme"* Paper presented at 2017 ASEE Annual Conference & Exposition, Columbus, Ohio. https://peer.asee.org/28549

9. Mullett, G. J. (2016, June), *Teaching the Internet of Things (IoT) Using Universally Available Raspberry Pi and Arduino Platforms* Paper presented at 2016 ASEE Annual Conference & Exposition, New Orleans, Louisiana. 10.18260/p.26053