10-3-2011

Interactive Real-Time Embedded Systems Education Infused with Applied Internet Telephony

Kyle Persohn
Marquette University, kyle.persohn@marquette.edu

Dennis Brylow
Marquette University, dennis.brylow@marquette.edu

Accepted version *Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual,* (2011): 399-404. DOI:

Interactive Real-Time Embedded Systems Education Infused with Applied Internet Telephony

Kyle Persohn
Mathematics, Statistics and Computer Science, Marquette University
Milwaukee, WI
Dennis Brylow
Mathematics, Statistics and Computer Science, Marquette University
Milwaukee, WI

Abstract: The transition from traditional circuit-switched phone systems to modern packet-based Internet telephony networks demands tools to support Voice over Internet Protocol (VoIP) development. In this paper, we introduce the XinuPhone, an integrated hardware/software approach for educating users about VoIP technology on a real-time embedded platform. We propose modular course topics for design-oriented, hands-on laboratory exercises: filter design, timing, serial communications, interrupts and resource budgeting, network transmission, and system benchmarking. Our open-source software platform encourages development and testing of new CODECs alongside existing standards, unlike similar commercial solutions. Furthermore, the supporting hardware features inexpensive, readily available components designed specifically for educational and research users on a limited budget. The XinuPhone is especially good for experimenting with design trade-offs as well as interactions between real-time software and hardware components.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

SECTION I.

Introduction

It's a hardware problem. No, it's a software problem! What if we taught engineers and computer scientists to think, "Either way, it's *my* problem."? Hardware/software interfaces often require co-design in order to meet performance requirements, yet these concepts are frequently taught independently. This paper proposes laboratory exercises designed to introduce students to the technical challenges arising from real-time, resource-constrained systems.

Internet telephony is a rapidly evolving area that integrates cross-discipline curriculum such as circuit design, embedded systems, software development, and signal processing. The demand for knowledgeable engineers in these fields is increasing as service providers transition from traditional circuit-switched phone systems to packet-based Voice over Internet Protocol (VoIP) networks.[1] Moreover, VolP systems represent a unique confluence of both analog and digital design, signal processing, network protocols, and embedded system integration. This paper presents a platform designed to give students hands-on experience with hardware/software integration in a highly relevant application area. With tight budgets and diverse course prerequisite chains in mind, we set out to create a unique platform that satisfies the following goals:

- Requires only inexpensive, readily available parts,
- Features easy assembly with common tools, and
- Adapts readily to suit various educational backgrounds.

The XinuPhone project meets all of these goals, providing a flexible laboratory implementation of VolP for students using freely available tools and cheap consumer grade hardware. The components comprising this system can be deployed as an integrated set of hands-on laboratory exercises in an upper-division design elective, or as standalone modules threaded through a wide variety of core courses common to many computer science and/or computer engineering degree programs. Laboratory modules include

- Filter design,
- Timing and clock configuration,
- Serial communications,
- Interrupt design and resource budgeting,
- Network transmission, and
- System benchmarking.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

Some of these modules focus on more traditional computer engineering/hardware topics, while others are completely in the software realm of computer science. But the complex interactions between hardware and software in modern embedded systems is a topic worthy of study on its own; an individual instructor can filter out modules based on the background and preferences of their student population, and supplement with other module ideas presented later in this paper. The authors believe that the best experience comes out of challenging our students on either side of the computer engineer/scientist divide to reach out of their comfort zone toward a larger goal.

The following sections of this paper present related work, followed by a description of the laboratory infrastructure and hardware recommended to support the proposed learning experience. Next, we introduce modular laboratory exercises including learning objectives and sample solutions from our reference implementation. A cost analysis follows, highlighting the utility of this solution. Finally, we conclude with a summary of benefits and intended future enhancements.

SECTION II.

Related Work

Previous papers discuss implementations of streaming media systems with embedded hardware as well as the importance of embedded systems education. For example, Cuijuan et al. demonstrate a media gateway implemented on a Digital Signal Processing (DSP) chip.[2] Likewise, Zhang et al. leverage the Real-Time Transport Protocol (RTP) to stream media to an embedded Linux device.[3,4] Unfortunately, these authors have chosen not to publish their software and their choice of hardware makes it difficult to replicate these contributions in an educational setting. References[5,6] suggest a need for more emphasis on embedded systems education.

This project also receives inspiration from commercial products that do not quite meet the needs of an educational development platform. Cisco offers a voice gateway that provides a robust array of features.[7] However, the lack of programmability renders the devices useless for learning about design trade-offs and ways to improve existing standards. Freescale's VoIP development kit[8] provides the necessary flexibility; however, the price tag is unreasonable for most institutions. We have been inspired by the Arduino development team (www.arduino.cc) to create open, accessible hardware and software designs that others can easily customize to suit their own needs.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

Our contribution is unique in combining the following attributes:

- Affordable, with a modest lab fee,
- Flexible hardware/software configurations, and
- Available source code and design schematics.

In the remainder of this paper, we describe in depth how the XinuPhone design accomplishes these objectives.

SECTION III.

Laboratory

To support practical VoIP laboratory experiences, we rely upon the Project Nexos [9] framework to provide a basic layer of support. This is a multi-university effort aimed at providing a modern, low-cost strategy for implementing hands-on embedded systems laboratory modules that scale across a wide variety of courses within the traditional computer science and engineering undergraduate curricula.

At the core of a Nexos laboratory is a centralized pool of available embedded hardware platforms. Remote connection software allows users to automatically reserve a target device - called a *backend* machine in Nexos terminology - powercycle into a "known good" starting state, upload a user-generated embedded system image, and interact with the backend via one or more serial links.

While dedicated, experimental embedded systems laboratories are not unheard of in large, well-funded universities, the special appeal of the Nexos approach is that it relies on cheap, ubiquitous embedded devices, a modest quantity of off-the-shelf serial aggregators, and a bevy of open source software and hardware designs. Half a dozen schools have now invested a few thousand dollars to construct general purpose experimental embedded systems laboratories that can support multiple courses each term.

Our current laboratory configuration supports multiple MIPS-based wireless router platforms (Linksys WRT54GL, Cisco E2100L, Asus g550,) as well as Intel x86 backends and Motorola 68HC11 and 68HC12 development boards. This configuration allows us to share a small amount of embedded hardware among many students, and greatly speeds up the develop/compile/test cycle from a user perspective. Ultimately, the WRT54GL wireless router hardware with a Project Nexos laboratory pool provides for an easily programmed

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

embedded processor within a cheap, consumer level product that can readily handle the serial sound data and Ethernet packets that comprise the backbone of an Internet phone.

The smallest of embedded devices are normally programmed by hand, in assembly languages. Larger scale embedded systems are often programmed in higher-level, compiled languages such as C or Java. A student who is to be well-versed in embedded system development should be exposed to both ends of this spectrum, and our laboratory infrastructure allows this. Whenever students are to tackle larger problems that require complex device interactions, a process/thread abstraction, or resource management on par with an operating system, we find it extremely helpful to rely on the open source Embedded Xinu[10] operating system component of Project Nexos.

Embedded Xinu is a modern RISC implementation of the classic Xinu O/S design,[11] long used for experimental O/S courses. Embedded Xinu provides a basic process model, with priority scheduling, shared memory, interprocess communication, and a TCP/IP networking stack. For software oriented students, Embedded Xinu presents an elegant design for a small O/S that can be completely understood and extended in a single term. For others, it provides a starting point for basic embedded application development, while requiring minimal system resources and only a modest learning curve.

SECTION IV.

Hardware

The hardware selection for this platform provides low cost, easy assembly, and maximum versatility. The audio interface simply consists of voltage regulators, operational amplifiers, a serial level converter, and a digital signal controller (DSC) as well as various resistors, capacitors, and interconnects. All components are available in through-hole packages, eliminating the need for the resources and patience demanded by surface mounted devices. This also affords students the opportunity to easily customize and prototype their own designs on a breadboard before producing a printed circuit board (PCB).

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

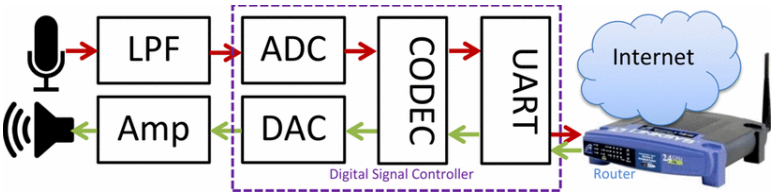

Figure 1. VoIP Development System Block Diagram

A Universal Asynchronous Receiver/Transmitter (UART) connects the audio interface to a network-enabled device, such as a modified Linksys WRT54GL router. The audio interface provides the analog filtering, analog to digital, and digital to analog conversions the router lacks. Figure 1 illustrates a block diagram of the system. A Microchip dsPIC33FJ64GP802 serves as the DSC for the reference implementation. This device has a wide selection of peripherals as well as royalty-free DSP library functions. Moreover, Microchip provides implementations of popular CODECs (coder-decoder) such as G.711,[12] G.726A,[13] and Speex.[14] Lastly, the popularity of Microchip's product line makes this an ideal DSC for users who might already have immediate access to dsPIC compatible development tools.

SECTION V.

Course Modules

Ideally, the following curriculum segments would form the basis for the laboratory component of an upper-level undergraduate design elective. Nevertheless, due to limits on course offerings and diverse student backgrounds we propose a flexible, modular content structure. These modules offer "drop-in" content for revitalizing dated material in existing lab courses and serving as projects for primarily theoretical courses that could benefit from the spice of an application. Furthermore, this modularity provides instructors with flexibility to cater directly to a variety of degree majors by substituting the most applicable content.

A. Filter Design

Anti-aliasing is one of the most practical applications of a low-pass analog filter. Before a continuous-time signal can be properly sampled, the high frequency components must be blocked; otherwise, the Shannon-Nyquist theorem may not be satisfied.[15] After sampling, the aliased frequencies are indistinguishable from the desired content, thus reinforcing the need for analog filtering even in a primarily digital world. In the this

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

module, students design a filter to pass the human vocal range and reject frequencies subject to aliasing at the industry standard sampling rate, $fs = 8$ kHz. Next, students realize their design on the audio board using the Sallen-Key topology.[16]

Our reference design uses a filter cutoff frequency $fc = 3.4$ kHz. This choice of f_c harnesses the upper spectrum of human speech, yet provides ample room for roll-off before f_c. A series of second order stages cascaded together create a 6th order Butterworth filter. For a detailed explanation of this process and supporting Matlab code, see.[17] Example firmware provides an interactive demonstration of aliasing. Without the low-pass filter, one can clearly observe the detrimental effects of undersampling in the output. Students must demonstrate that their filter prevents aliasing while incurring minimal attenuation in the pass-band.

B. Timing and Clock Configuration

When undergraduates learn about high-level programming, they rarely write software with timing constraints in mind. Conversely, real-time systems require strict awareness of deadlines. The timing module teaches students how to synchronize their software with a wall-clock. Students must configure their device to meet all the timing needs of various peripherals. For the VoIP hardware, these include the analog to digital converter (ADC), digital to analog converter (DAC), and serial baud-rate generator. Furthermore, scalars and divisors must ensure intermediate clock frequencies meet constraints specified by the DSC's datasheet. The dsPIC peripherals have the following requirements:

- 8,000 Hz timer interrupt for ADC
- 2.048 MHz clock to DAC (256x oversampling)
- 115,200 bps UART baud rate

Furthermore, the dsPIC imposes additional constraints on intermediate frequencies in the clocking subsystem:

- High speed primary oscillators operate above 10 MHz.
- Phase-Locked Loop (PLL) input must be 0.8–8.0 MHz.
- PLL output before post-scalar must be 100–200 MHz.
- The device operating frequency must be 12.5–80 MHz.

For an example configuration satisfying the dsPIC's requirements, see.[10] Based on the instruction clock, the designer must verify integer divisions precisely define the ADC timer period and the DAC clock. On the other hand the asynchronous nature of the UART allows it

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

to function with reasonably tolerant variations in baud rate. Although the math is basic, the process of juggling these parameters can easily overwhelm an entry-level engineer without prior exposure to proper clock configuration.

C. Serial Communications

An embedded device is frequently one sub-component of a larger system; consequently, communication with other hardware is a necessity. The DSC communicates with the network device over a standard 8 data bit, no parity, 1 stop bit (8N1) serial connection. Audio samples flow to the network in a periodic stream as dictated by the sample rate. Conversely, the network traffic travels in packets, arriving back to the DSC in bursty blocks. In this module, students combat the challenges imposed by simultaneously handling various data flow patterns.

Normally, a designer may employ a simple first-in first-out (FIFO) buffering scheme to smooth out the fluctuations in data arrival. However, this particular situation is complicated by the intermediate processing stage of the CODEC. The data stream leaving the DSC has large gaps between samples (Figure 2). In order to minimize delay, the DSC compresses each sample individually during the idle time on the serial bus. On the other hand, samples arriving from the network come in blocks; there is not enough time to expand an audio sample before the next byte arrives. The UART FIFO is small relative to the network packet payload so the buffer saturates quickly and data is lost.

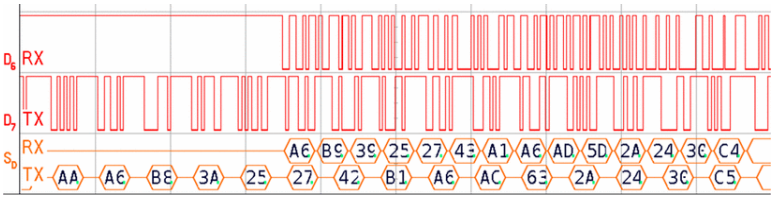

Figure 2. UART transmit data is evenly spaced. Conversely, received bytes arrive in contiguous blocks grouping the bus idle time together.

The reference implementation mitigates data loss using a revolving double buffer. The DSC stores enough incoming samples up to the size of the network payload then processes them as a single block. Meanwhile, a second buffer captures the serial data arriving before the expansion process finishes. These buffers continuously flip-flop; as a result, the DAC always has data to read and there is always a place to store incoming

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

samples. Furthermore, Direct Memory Access (DMA) handles data transfer to the DAC, which frees the CPU to continuously process data.

There is a trade-off between increasing the buffers to handle larger packets and aiming to minimize delay. As a starting point, we defer to the Internet Society's recommendation for streaming media payload sizes.[18] Braun et al. provide competing reference points for ideal delay in.[19]

D. Interrupts and Resource Budgeting

Embedded systems often arbitrate between various tasks using interrupts. The DSC must handle multiple roles:

- Starting ADC conversion at regular intervals (f_s),
- Compressing and transmitting samples,
- Buffering data received by the serial port,
- Expanding incoming samples, and
- Updating the DAC.

On the dsPIC, the first step is handled internally by the timing subsystem. The second task is handled by an interrupt triggered when an ADC conversion completes. Immediately, the sample is compressed by this interrupt service routine (ISR) and sent to the UART for transmission.

Reconstructing speech data from the network is slightly more complex. One ISR, triggered via UART data reception, buffers incoming serial data. When the buffer reaches capacity, the ISR begins filling an alternate memory location and sets a global flag. This indicates to the CPU there is a block of data ready for expanding. The main routine polls this flag and runs the CODEC when there is data available and the CPU is not servicing an interrupt. Meanwhile, the DAC generates periodic interrupts when it needs to request a new sample to reconstruct. This interrupt is different from the others; it does not activate an ISR for the CPU to process. Instead, the DAC generates a DMA request; thus, the DAC updates without CPU intervention.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

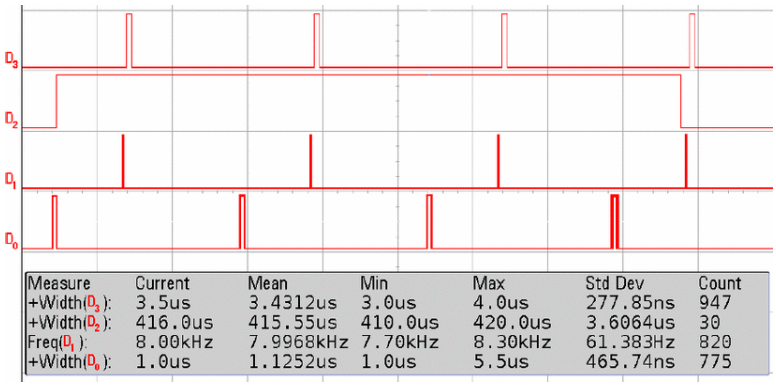

Figure 3. GPIO pins are useful for gathering approximate data on the worst, average, and best case execution time of various subroutines. Shown are UART RX (D_0), f_s Timer (Dl), G.711 μ-Iaw CODEC expansion (D_2), and ADC conversion complete (D_3).

With all tasks accounted for, one must assign priorities to each interrupt (task) in such a way that all deadlines are met. Most importantly, the ADC must start conversions and the DAC must update on a precise schedule. In the reference implementation, we configure the DSC in such a way that these operations do not rely on the CPU; therefore, they cannot be interrupted. Data flows out of the DSC at the highest priority. The packet-sized receive buffer allows more elasticity; consequently, it yields to transmission. Moreover, when the network bestows an entire block of data upon the DSC at once (see Section V-C), transmission must take priority or else data loss and artifacts occur. Finally, CODEC expansion assumes least priority. It is not critical when the CPU performs this operation so long as it finishes an entire block of data sometime before the next block arrives completely.

For the hands-on portion of this module, students design their own priority scheme to handle voice processing and communication. There are many opportunities for different choices with corresponding justifications. Specifically, students should identify where data "leaks out" of the system when deadlines are missed and be prepared to defend decisions. As a visual aid for this process, we recommend students toggle general purpose input/output (GPIO) pins on the DSC at the entry and exit points of different tasks. Using a digital logic analyzer students may observe task arbitration in real-time. Figure 3 illustrates an example where the ADC and UART routines are correctly taking priority over the CODEC expansion process with ample overhead.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

E. Network Transmission

Of course, all of this effort to sample and reconstruct audio signals is of little use without networking functionality to transport the speech data. In the networking module, students employ Embedded Xinu[9] as a translator between the UART and the Internet. Xinu provides a basic shell, serial driver, threads, and network stack enabling communication between two XinuPhones over an Ethernet network. The current network stack supports Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) devices, which grants an opportunity to learn about flow-control and the best-effort service model.[20]

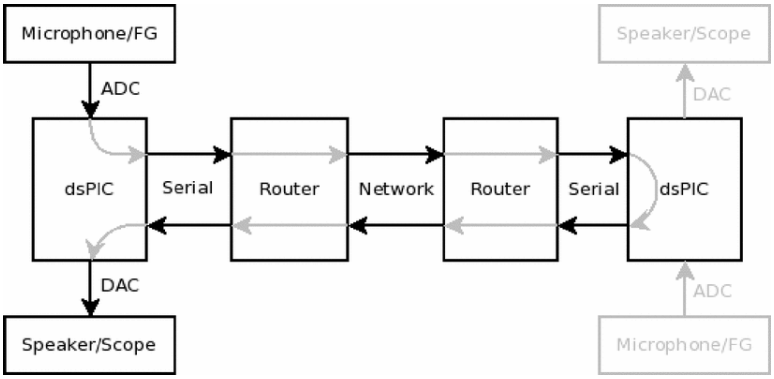

Figure 4. Test Configuration for Roundtrip Measurements

From a design standpoint, students can create their own lightweight protocol for transmitting the data, and measure its performance versus straight UDP and TCP. Protocol enhancements can include timing data, varying degrees of reliability, and support for multiple channels. This, in turn, can be compared with a full-fledged RTP + RTCP (RTP Control Protocol) implementation.

F. System Benchmarking

The final module focuses on evaluating performance of the overall system. There are several standard metrics for VolP Quality of Service (QoS) assessment. Examples include One-Way Delay (OWD) at the network level, blocking probability at the call level, and perceptual quality evaluation at the user level.[19] In this module, students evaluate their implementation using the Perceptual Evaluation of Speech Quality (PESQ)[22] Mean Opinion Score (MOS) and various latency measurements.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

The PESQ-MOS is useful for benchmarking performance in terms of the subjective opinion of a group of listeners without needing to conduct an extensive study. Students play a reference signal into their system and record the output. Then, the International Telecommunication Union (ITU) reference implementation[22] assigns the degraded signal a score from 1 (Bad) to 5 (Excellent). Additionally, students record latencies between all communication interfaces in the system using loopbacks when applicable.[21] Evaluating subsets of the overall system is important for isolating problematic areas. For example, one might analyze:

- External test equipment (normalization reference),
- ADC-DAC passthrough (with and without CODEC),
- Audio modules directly linked via UART, or
- End - to-end system including the network (Figure 4).

Up to this point, students explore different design options, document trade-offs, and justify implementation choices. With evaluation tools and performance data in hand, we encourage students to assess which areas of the system need most improvement. Now, it is time to revisit some of the critical choices made earlier and see if quantifiable enhancements are possible.

Table I Audio Module Approx. Costs

Component	*Cost*
dsPIC33FJ64GP802 DSC	$5.88
LD1117–3.3 LDO Regulator	$0.68
LM7805 Linear Regulator	$0.58
SP3232 RS-232 Level Converter	$1.88
MCP6024 Quad Op-Amp	$2.08
LM386N Audio Op-Amp	$0.95
Capacitors	$3.42
Resistors	$1.97
Connectors	$4.88
Total	$22.32

G. Alternate Tracks

Internet telephony hardware and software combines knowledge from many courses in various degree majors. We further propose these additional possibilities as customizations to make the project more applicable to existing courses.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

1. *Circuits and Electronics:* Printed circuit board design is an important skill for any electrical engineer. We provide Eagle CAD files as a starting point for generating new designs based on our hardware.[10] For example, consider adapting the audio interface to support an analog telephone handset. This opens up an interesting opportunity for software dual-tone multi-frequency (DTMF) decoding.
2. *Computer Networks:* Networking in the current release is very simple. Additional enhancements to include QoS features are readily implementable in Embedded Xinu.[23,19] Furthermore, students can use network emulation coupled with previously introduced performance metrics for assessing the effects of various load conditions on different CODECs and transport protocols. Lastly, students with operating system design experience can provide in-band signaling to the audio module via interprocess communication.
3. *Signal Processing:* The dsPIC is an ideal platform for implementing a wide variety of classic DSP algorithms. Namely, acoustic echo and noise cancellation are highly applicable to VoIP. In the regular modules, the DAC is treated as a black box. Alternatively, students can explore up/downsampling, interpolation, decimation, and intermediate filtering methods for emulating the built - in features of the DAC.[15] As an application, the dsPIC pulse-width modulation (PWM) module, with an appropriate analog reconstruction filter, creates the output signal.[24]

SECTION VI.

Cost Analysis

Cost-effectiveness is a primary goal for making this project a success. So far, we have demonstrated how to maximize the utility of our device in various courses. Next, we present approximate costs of the audio module components (Table I). All of these parts (or equivalent substitutes) are readily available from the usual suppliers.

At the time of writing, one can purchase a Linksys WRT54GL for under 50. $The aforementioned Cisco voice gateway retails for around 70$. Thus, we conclude that even with the added cost of the audio interface, our proposal remains competitive with a commercial offering. Further more, an open and flexible platform allows for interactive experimentation, where as the off-the-shelf device does not. As another reference point, the Freescale VoIP development kit retails for approximately $800 per unit. With a reasonable investment in parts and equipment, it is possible to offer a priceless hands-on laboratory experience.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

SECTION VII.

Future Work

The integration of standard signaling and streaming protocols into the Embedded Xinu operating system is the current focus of our next efforts. The addition of RTP and Session Initiation Protocol (SIP) support will enable interoperability with other devices and greatly enhance the network related curriculum. Additionally, the audio interface requires some hardware enhancements before it can operate as a fully functional analog telephone adapter for legacy handsets. Lastly, we hope to collect empirical data on the effectiveness of our work in the classroom.

SECTION VIII.

Conclusion

In this paper we present novel curriculum for educating students on real-time embedded systems. While the content centers around an Internet telephony application, the skills obtained in the proposed modules are relevant to many industries and core courses. Our focus on design encourages students to optimize trade-offs for a particular goal, as opposed to seeking single answers without context. Further more, modularity in the content structure provides adopters with the flexibility to customize and integrate into existing course objectives. Our solution is inexpensive, using cheap commodity embedded platforms with simple modifications that are an order of magnitude less expensive than our closest educational competitor. The resulting system cost is comparable to an existing commercial system, albeit one that is completely unsuitable as an open, experimental platform.

The XinuPhone tools are built upon a proven experimental embedded systems laboratory, already in use at multiple institutions for a variety of courses. Many of the underlying components have already been extensively tested in student hands. A chief contribution of this paper is the addition of the design component of the curriculum, an aspect that is sorely lacking in much undergraduate coursework. In summary, the XinuPhone provides an accessible, inexpensive, highly flexible infrastructure suitable for a sequence of modules in an upper-division design elective, or as standalone laboratory experiences. The authors hope the resources contributed by this project assist others in developing applied, hands-on educational experiences.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

The XinuPhone tools are freely available from the Project Nexos/Embedded Xinu portal.[10]

Acknowledgment

The authors would like to thank Zachary Lund for the prototype design as described in his thesis.

References

[1]B. Douskalis, Putting VoIP to Work, Prentice Hall, 2002.

[2]G. Cuijuan, M. Changyun, W. Zhigang, L. Jie, "Design and implementation of simplified MGCP stack based on DSP", *ICFN '10: Second Int'l Conf on Future Networks*, pp. 381-384, Jan 2010.

[3]R. Zhang, J. Liu, Y. Gao, J. Qiao, "A realized embedded streaming media system", *IET Int'l Conf on Wireless Mobile and Multimedia Networks*, pp. 1-4, Nov 2006.

[4]H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", *Internet Engineering Task Force RFC 3550*, Jul 2003.

[5]W. Stapleton, "Microcomputer fundamentals for embedded systems education", *FIE*, pp. 6-10, Oct 2006.

[6]B. Benson, A. Arfaee, C. Kim, R. Kastner, R. K. Gupta, "Integrating embedded computing systems into high school and early undergraduate education", *Education IEEE Transactions on*, no. 99, pp. 1-6, Sep 2010.

[7]*SPA3102 Phone Adapter with Router Cisco Systems*, 2008.

[8]MCF532x/7x Embedded VoIP Solution, Freescale, 2007.

[9]D. Brylow, B. Ramamurthy, "Nexos: A next generation embedded systems education", *SIGBED Review*, vol. 6, no. 1, Jan 2009.

[10]D. Brylow, "Embedded Xinu", [online] Available: http://xinu.mscs.mu.edu/.

[11]D. E. Comer, Operating System Design: The XINU Approach, Prentice Hall, 1984.

[12]"Pulse Code Modulation (PCM) of Voice Frequencies", *ITU Recommendation G.711*, Nov 1988.

[13]"Adaptive Differential Pulse Code Modulation (ADPCM)", *ITU Recommendation G.726*, Dec 1990.

[14]J.-M. Valin, "The Speex Codec Manual", May 2007, [online] Available: Valin/Xiph.org.

[15]A. Ambardar, "Digital Signal Processing: A Modern Introduction", *Thomson Learning*, 2007.

[16]R. Schaumann, M. E. V. Valkenburg, Design of Analog Filters, Oxford University Press, 2001.

[17]B. Lathi, Linear Systems and Signals, Oxford University Press, 2005.

[18]H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP Profile for Audio and Video Conferences with Minimal Control", *The Internet Society*, Jul 2003.

[19]T. Braun, M. Diaz, J. E. Gabeiras, T. Staub, End-to-End Quality of Service Over Heterogeneous Networks, Springer Verlag, 2008.

[20]L. L. Peterson, B. S. Davie, Computer Networks: A Systems Approach, Morgan Kaufmann, 2007.

[21]Z. D. Lund, "A VoIP Implementation on an Embedded Platform", *Master's thesis*, 2010.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.

[22]"Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs", *ITU Recommendation*, pp. 862, Feb 2001.

[23]D. Minoli, E. Minoli, Delivering Voice over IP Networks, Wiley Publishing, 2002.

[24]M. Mitchell, "Using PWM Timer_B as a DAC", *Texas Instruments Application Report SLAA116*, Dec 2000.

2009 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC, 2009), (October 3, 2011): pg. 199-204. DOI.