Author: Angelo Fraietta (angelo_f@bigpond.com)

Address: PO Box 859, Hamilton NSW, 2303

Submission Category: Paper – Abstract reviewed

Title: "Smart budgeting for a Smart Controller"

Keywords: Algorithmic Composition; MIDI Controllers; Recycled Hardware; Software

In 1998, I was involved in two collaborative projects -- the "Laser Harp[1]" and the "Virtual Drum Kit"[2] -- where performers controlled MIDI sound engines using MAX and analogue to MIDI converters. In both cases, analogue signals were converted into MIDI messages, manipulated with an algorithm through MAX, converted back to MIDI, and finally sent to an external sound engine. At least three separate pieces of equipment[3] were required for a performance; each introducing associated software and hardware costs, physical space requirements, and latency in performance. Space restrictions could be reduced by using alternative control voltage (CV) to MIDI converters, such as the I-Cube or some other custom device; however, these devices still require the use of an algorithmic or programmable logic control (PLC) software package on a host computer in order to execute all but the simplest of algorithms or patches. Although one could use a laptop computer, "weight and size isn't everything when it comes to stage (or touring) fitness. The plastic casing and the display on hinges sure don't encourage one to drop the PB [Power Book] from a table."[4] Ultimately, some sort of stand-alone device or "Smart

[1] Alex Cockburn. The Laser Harp is an instrument that is performed by cutting laser beams with the body.

[2] Guy Robinson. The Virtual Drum Kit entails a performer playing an invisible drum kit. Each of the drummers arms had two potentiometers that modified a control voltage, thus abstracting the polar coordinates of the drummers arms in space as two MIDI controller values. These coordinates are fed into an adjustable threshold matrix that quantizes each midi message into 1 to 4, and then selects one of sixteen outputs relative to the input coordinates. Each output of the matrix enables a gate that represents a drum.

[3] The Virtual Drum Kit actually had four pieces: SY-77 and SY-99 for analogue to MIDI conversion; Power PC for running MAX; and TG-300 for sound engine.

[4] Sukandar Kartadinata, "hardMax," http://members.xoom.com/Sukandar/hardMAX.html.

Controller" is required to replace the CV to MIDI converter and the laptop computer, thus allowing a simple interface of controller input to sound engine.

I proposed and commenced designing the "Smart Controller," as there were no products currently available that could satisfactorily perform the required functions. Sukandar Kartadinata proposed "hardMax [--] a stand-alone box to run MAX/msp patches;"[5] the project was suspended[6] before I commenced work on the Smart Controller; however, he states that he has resumed work on the project.[7] A release date has not been announced.

In designing the Smart Controller, I am not creating a sound generation unit, but rather an intelligent controller for analogue and MIDI instruments. It responds to input control voltage and MIDI messages, producing output control voltage and MIDI messages (depending upon the patch). The device can be programmed remotely through the use of a patch editor or Workbench, which is an independent computer application that simulates and communicates with the hardware; the workbench, however, is not required during performance. The resultant patches can also be sent to the Smart Controller hardware using MIDI system exclusive messages, which can therefore be saved in the standard MIDI file format. The Smart Controller is a stand alone device -- a powerful, reliable, and compact instrument -- capable of reducing the number of electronic modules required, particularly the requirement of a laptop computer, in a live performance.

Designing and building the Smart Controller may at first appear to require a substantial budget for software and hardware tools. The cost can be reduced significantly when one considers using recycled hardware and low cost software including educational licenses, freeware, and shareware. I decided early in the project to assess where the

---

[5] Ibid.

[6] Sukandar Kartadinata , Cycling '74 Discussions ->MAX-MSP -> Controllers -> Smart Controller, 7 Sep 2000, http://www.synthesisters.com/cgi-bin/WebX?13@174.1y06asTPaIY^11@.ee6bdbe/9

[7] Sukandar Kartadinata, email to Angelo Fraietta, 5 June 2001.

greatest risks in the project were, and to develop the project focusing upon the software architecture instead of the hardware implementation.[8] Sukandar Kartadinata stated regarding hardMax:

> development cost: is obviously the biggest stumbling block. Even if a suitable DSP board could be found that would serve as the basic building block and thus minimize hardware development, the software part would still be immense. In any case without source code access to MAX/msp (or at least to some implementation details) it's next to impossible as everything would have to be re-written from scratch and would still suffer from compatibility problems.[9]

It was obvious from this perspective that the viability of the software would have to be resolved before attempting any sort of hardware implementation. "Software is the most expensive thing in the universe….It's cheaper to reuse code. It's smarter to buy, beg, or borrow a module than to code it ourselves."[10] I already had source code for the PLC from Algorithmic Composer.[11] I had intentionally written the code so the PLC engine was not dependent upon the graphical environment, and had low coupling with the operating system primitives. I did, however, have to find a Real Time Operating System (RTOS) to interface the engine with the target hardware. I decided that I would choose the hardware based upon the RTOS features rather than the other way round.

Cameron Shortland introduced me to RTEMS when I was working as a software engineer at Hunter Watertech. He explained to me that RTEMS was both open source

---

[8] Craig Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*, (New Jersey: Prentice Hall, 1998), 434.

[9] Sukandar Kartadinata, "hardMax."

[10] Jack G. Ganssle, "Open Source vs. Proprietry," *Embedded Systems Programming* 13 (Dec 2000), 189. Ironically, his argument was against the use of open source software.

[11] Angelo Fraietta, "Algorithmic Composer", Honours thesis, University of Western Sydney, 1998.

and free of cost; the GNU development tools used to build and debug RTEMS were free; the development tools ran on Linux, which was also free. Additionally, at that period of our employment, we were working on telemetry firmware based upon the i386ex CPU. We were using the ATI Nucleus RTOS and simulating the telemetry hardware an old 386 PC before we had any real hardware to work with. RTEMS supported both the i386ex and the PC386. I decided to consider RTEMS as my operating system and the i386ex as my target hardware device because I would be able to work on the software and simulate the hardware with a PC.

My first step was to set up a suitable host network at home where I could run Windows and Linux concurrently, which meant that I required at least two computers. I received a broken Pentium 75 computer from a friend, John Turon, as payment for fixing his other computer[12] and used the parts to build another computer. I obtained a Kingston Turbochip 400 for $237 over the Internet and a second-hand motherboard from the computer fair for $50. This enabled me to use the 64MB of RAM that I already had in a 400 MHz computer. I installed Red Hat Linux 6.2 (free) on my original computer, a Pentium 133, connected it to my Windows Machine using Ethernet, and used SAMBA[13] (also free) to copy files between the two computers. I was also able to control the Linux machine using telnet from the Windows machine, thus removing the requirement for a monitor on the Linux machine. This setup enabled me to build the RTEMS source on the Linux machine while concurrently working at the Windows machine. I later received a broken DX 386 computer from a friend, David Gibbins, who said that I could use it if I could fix it – he also insisted that he pay me to fix it. The hard drive was intermittent and so I told him it was not worth his money to fix it -- so he let me keep it (I provided him with a CDROM of the data on the drive). I replaced the hard drive with the one from the Pentium 75, installed Linux, and use a free ftp client, WS_FTP, to transfer the Smart Controller binaries that are built for RTEMS to the PC 386. The computer now uses a multi-boot menu to boot into Linux or any of the RTEMS binaries.

[12] All I did to fix his computer was to run Norton's Disk Doctor.

[13] SAMBA comes with Linux Red Hat 6.2.

I found out through the RTEMS newsgroup that I should boot the computer into RTEMS using GRUB.[14] After creating the GRUB boot floppies, I successfully booted the Hello World program on my PC. I removed or re-defined all the Windows specific calls in the Algorithmic Composer source (now called the Smart Controller engine source), and built it for RTEMS using the GNU development tools. I booted the computer with GRUB and successfully created objects via the computer keyboard using a simple menu interface.

I was successfully able to build the RTEMS source with the Smart Controller engine source on the Linux machine and run it on a PC within two months of starting the project. This was primarily due to the fantastic support of the RTEMS user group, which was truly shown when I was testing the soft-floating point capabilities of the RTEMS tool-sets. The use of the DX 386 was significant as a simulator as it did not have a math co-processor, which enabled me to test the floating point libraries that would be used in the final target. I was initially unable to run the binaries on the 386 (I had been testing them on a Pentium machine) and posted a message through the mailing list.[15] I was able to determine (after being notified through the mailing list that the problem was a co-processor error) that I could stop the fault by simple placing a co-processor into the 386 computer (fortunately, I had had a co-processor in an old IBM 386 that was no longer serviceable). What eventuated over a period of thirty-six days was a combined effort from the RTEMS users to find and fix the problem, which resulted in thirty-eight email messages. Ralf Corsepius found the problem, but the staggering fact is that most of the people working on the RTEMS project are volunteers.[16] OAR Corporation, the distributors of RTEMS, also provides paid support -- if this was the level of the free support, I find it hard to imagine how good the paid support must be. RTEMS was

[14] Mahesh revuri@seeyes.co.in , Re: hello_world_c, http://www.oarcorp.com/rtems/maillistArchives/rtems-users/2000/august/msg00022.html

[15] Angelo Fraietta, Soft Float for PC386, 23 Jan 2001. (Mailing list archive is missing)

[16] Joel Sherrill, Re: Soft Float for PC386, 28 Feb 2001. (Mailing list archive is missing)

originally developed for the US Army, but was released for non-military uses later.[17] When asked why RTEMS was given away instead of being sold, Mark Johannes of OAR stated “It was a natural fit to release RTEMS as open source to leverage the environment that promotes the further development and utilization of the product.”[18]

During the development, I have used many other software packages that were obtained either as freeware, shareware or with an academic license. A listing of the software has been provided at the end of this paper. One shareware package, “Hex Workshop Hex Editor,”[19] was so good that I actually decided to pay for it. Hex Workshop enabled me to find and edit the binary file that was crashing the 386. I found the hex values of the floating point instructions by disassembling the binary using “objdump” (free with Linux) to find the binary file position that had the floating point instructions, and then replaced the floating point instructions with NOP instructions[20] using Hex Workshop.

Some software tools have a shareware version with limited functionality, enabling the user to try (or get hooked on) the product. When the user finds that he or she requires the full version, they are forced to pay for the full version. I decided that I would use a computer-aided software engineering (CASE) tool to generate code from class diagrams, which in turn would facilitate concurrent code and documentation synchronization. I requested and received a copy of Rhapsody 2.3 Modeler from I-Logix through their Web page http://www.ilogix.com. I read about this free modeler from an advertisement in

---

[17] Joel Sherrill, Re: History of RTEMS, http://www.oarcorp.com/rtems/maillistArchives/rtems-users/2001/march/msg00033.html

[18] Mark Johannes, Re: History of RTEMS, http://www.oarcorp.com/rtems/maillistArchives/rtems-users/2001/march/msg00032.html

[19] http://www.bpsoft.com/ordering/.

[20] Barry Brey, *Embedded Controllers: 80186, 80188, and 80386EX* (New Jersey: Prentice Hall, 1998), 575.

Embedded Systems Programming Magazine[21]. The product is in direct competition with Rational Rose, and so I-Logix provides it for free in order to take some of the market from Rational. I already had a copy of Rational Rose Student Edition[22], so the issue was not so much the price, but which product better suited my needs. Although a complete comparison between the products is beyond the scope of this paper, I chose Rational Rose mainly because it enabled me to edit code both from within and outside of the CASE tool. Additionally, *Visual Modeling with Rational Rose and UML,*[23] gave an in-depth description of how to use the CASE tool effectively, with a section on code generation and reverse engineering.

Designing hardware simulators that run on desktop computers – such as Macintosh or PC -- allow the software to be tested before committing to a target hardware device. I have been able to run and debug all of the non platform specific code on my PC at home using the compilers that I am already familiar with (Borland BC 5.02). After building and running the code on the PC, I build the code for RTEMS and run the binary on the 386. This has had a twofold effect: the code is compiled using completely separate compilers, thus facilitating platform independent code; and I can always run the code and know where it is not faulty in the embedded target. One area where this approach has been particularly successful was during the development of the workbench and editor communication protocol. I managed to develop and test the interface in two stages: implement a GUI and perform all engine queries and manipulations through a single function call; and perform the manipulations through the communications ports on the computers. In the first instance, I created a console driven menu to create and delete objects through the upper layers of the interface protocol. After succeeding, I modified the Algorithmic Composer GUI code and hooked it into the interface. The success of this

[21] *Embedded Systems Programming* 13 no 8 (2000), 41. The advertisement states "There is nothing Rational about paying for UML Modeling."

[22] http://www.rational.com/products/rose/tryit/stud_edition.jtmpl

[23] Terry Quatrani, *Visual Modeling with Rational Rose and UML* (Reading: Addison-Wesley, 1998).

stage proved that I could interface between the GUI and the engine by passing parameters between the two sections using a character buffer. This meant that interfacing the two though a serial cable was possible by implementing a lower layer, which could be done separate from the application. I designed the lower layer and tested it with a driver application through the Windows communications ports. I then implemented the lower layers into the Smart Controller code. I was able to then control the engine running on one computer with a GUI running on a different computer through a serial cable. This has proved that the workbench on a desktop computer can control the engine remotely. This step was vital before attempting to embed the engine software because I know that an inability to communicate with the embedded device is not due to the interface protocol.

Developing the Smart Controller is more than just developing a target device – it is developing a methodology whereby I do not have to always put my hand into my pocket to develop software systems. When I do, however, I know I am getting the best value for my (Australian) dollar.

Software Packages Used

CVS -- Concurrent Versions System. Management system for controlling and tracking source code modifications. http://www.gnu.org/software/cvs/. Cost: free.

Cygwin. Provides a Unix/Linux environment when developing in Windows. http://www.redhat.com/products/support/cygwin/. Cost: free.

GNU Emacs. A powerful text editor available for most platforms. http://www.gnu.org/software/emacs/. Cost: free.

GNU GRUB. Multi-boot boot loader. GNU GRUB is derived from GRUB (GRand Unified Bootloader), which was originally designed and implemented by Erich Stefan Boleyn. http://www.gnu.org/software/grub/. Cost:free

Hex Workshop Hex Editor. Hex editor for modifying binary files. http://www.bpsoft.com/ordering/. Cost: $US24.95 academic.

Rational Rose Student Edition – CASE tool limited to thirty classes and three packages. http://www.rational.com/products/rose/tryit/stud_edition.jtmpl. Cost Free.

Red Hat Linux 6.2. Linux OS obtained from Ray Kiefel at Hunter Watertech. http://www.redhat.com/products/software/linux/. Cost: free under the GNU GENERAL PUBLIC LICENSE.

RTEMS – Real Time Executive for Multiprocessor Systems. GNU tool-sets are also available. http://www.gnatrtems.com/RTEMS/rtems.html. Cost: free

WS_FTP Limited Edition – File transfer client. http://www.ipswitch.com/. Cost: free for academic use.

Bibliography

Brey, Barry B. *Embedded Controllers: 80186, 80188, and 80386EX*. New Jersey: Prentice Hall, 1998.

Embedded Systems Programming 13, no 8 (2000): 41.

Fraietta, Angelo. "Algorithmic Composer." Honours thesis, University of Western Sydney, 1998.

Ganssle, Jack G. "Open Source vs. Proprietry." *Embedded Systems Programming* 13, no. 13 (2000): 189.

Kartadinata, Sukandar. "hardMax." http://members.xoom.com/Sukandar/hardMAX.html.

Kartadinata, Sukandar. Cycling '74 Discussions ->MAX-MSP -> Controllers -> Smart Controller, 7 Sep 2000. http://www.synthesisters.com/cgi-bin/WebX?13@174.1y06asTPaIY^11@.ee6bdbe/9

Larman, Craig. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*. New Jersey: Prentice Hall, 1998.

Quatrani, Terry. *Visual Modeling with Rational Rose and UML*. Reading: Addison-Wesley, 1998.

RTEMS Mailing Archive. http://www.oarcorp.com/rtems/maillistArchives