Musical Interaction Design with the CUI32Stem: Wireless Options and the GROVE system for prototyping new interfaces

Dan Overholt
Department of Architecture, Design
and Media Technology
Aalborg University, Denmark
Niels Jernes Vej 14, 3-107
dano@create.aau.dk

ABSTRACT

The Create USB Interface is an open source microcontroller board that can be programmed in C, BASIC, or Arduino languages. The latest version is called the CUI32Stem, and it is designed to work 'hand-in-hand' with the GROVE prototyping system that includes a wide range of sensors and actuators. It utilizes a high-performance Microchip® PIC32 microcontroller unit to allow programmable user interfaces. Its development and typical uses are described, focusing on musical interaction design scenarios. Several options for wireless connectivity are described as well, enabling the CUI32Stem to pair with a smartphone and/or a normal computer. Finally, SeeedStudio's GROVE system is explained, which provides a prototyping system comprised of various elements that incorporate simple plugs, allowing the CUI32Stem to easily connect to the growing collection of open source GROVE transducers.

Keywords

Musical Interaction Design, NIME education, Microcontroller, Arduino language, StickOS BASIC, Open Sound Control, Microchip PIC32, Wireless, Zigflea, Wifi, 802.11g, Bluetooth, CUI32, CUI32Stem

1. INTRODUCTION & BACKGROUND

The CUI32Stem follows in the footsteps of the author's previous circuit board designs, such as the original Create USB Interface (CUI) [10] that utilized a PIC18F4553 microcontroller (an older 8-bit MCU), and the currently available CUI32 [2]. The CUI32 utilizes a PIC32MX440F512H, which is a modern 32-bit MCU (Microcontroller Unit) running at 80MHz. This allows the system to perform much faster, and the use of a free RTOS (Real-Time Operating System) has been implemented which includes an on-chip compiler for BASIC-language programs.

The RTOS is called StickOS [15], and was created by Rich Testardi. The CUI32 also includes a bootloader pre-installed. This allows advanced users to program it via either the Arduino language, or Microchip's free C-development environment and compiler (MPLAB X / C32) without having to purchase a separate programmer. The CUI32 circuit board was designed by the author as an improved version of the original CUI, and is sold by SparkFun electronics and other online retailers, as shown in figure 1 (lower right corner).

NIME'12, May 21-23, 2012, University of Michigan, Ann Arbor.

2. OBJECTIVES & INNOVATION

One of the main objectives in the development of the CUI32Stem is to pursue the author's own long-term research focused on the development of new electronically enhanced (augmented) musical instruments. The aim of this research is to explore the potentials that such instruments have in the context of new compositions and new methods of performance. The overall goal is to add new dimensions and expressive possibilities to the capabilities of traditional electronic and/or hybrid acoustic instruments, exploring these in contemporary music, or in fact any performance setting. The research can be seen as a process of discovery, investigating the extension of musical instruments' expressive and performative ranges. In addition, through research-based teaching, students are exposed to the common issues that may arise in the field, and are asked to build significant projects through their semester work.

A complete discussion of the larger research challenges and innovations is not included here. However, a recent example of an augmented musical instrument created by the author is the Overtone Fiddle [11], which utilizes some of the technology described herein. This document is intended to serve as collection of useful technical information for those interested in working with the CUI32Stem, the GROVE system, and their corresponding methodologies.

2.1 A New Board: CUI32Stem

The first prototype of the newest member of the CUI family, the CUI32Stem (Figure 1, lower left corner and Figure 2) incorporates changes to the board design to accommodate the GROVE system's 4-pin connectors for sensors and actuators, as well as incorporate a more powerful microcontroller along with an updated bootloader that supports the Arduino language.

The design details can see found on the SeeedStudio wiki[1] (distributor of the CUI32Stem and GROVE systems), including more information about functionality and availability. The CUI32Stem board can be connected to a wide range of GROVE elements. More data about each of the individual GROVE elements is also available online, including schematics and related files. All of the GROVE elements can be acquired individually – for a specific design – or experimentation kits will be available which will include an assortment of various GROVE elements together with the CUI32Stem. These bundles are provided simply as a convenient way of obtaining a collection of various sensors and actuators that support musical (and other) interaction design processes. It is hoped that the ease of use of such bundles will facilitate rapid prototyping of many ideas within physical interaction design, a process that is

[1] CUI32Stem and GROVE system on SeeedStudio wiki: http://www.seeedstudio.com/wiki/CUI32Stem

sometimes referred to as *sketching in hardware*. Within musical interaction design specifically, this may include performance oriented systems, composition interfaces, interactive installations, and other types of designs in the context of lab / experimental / educational use.

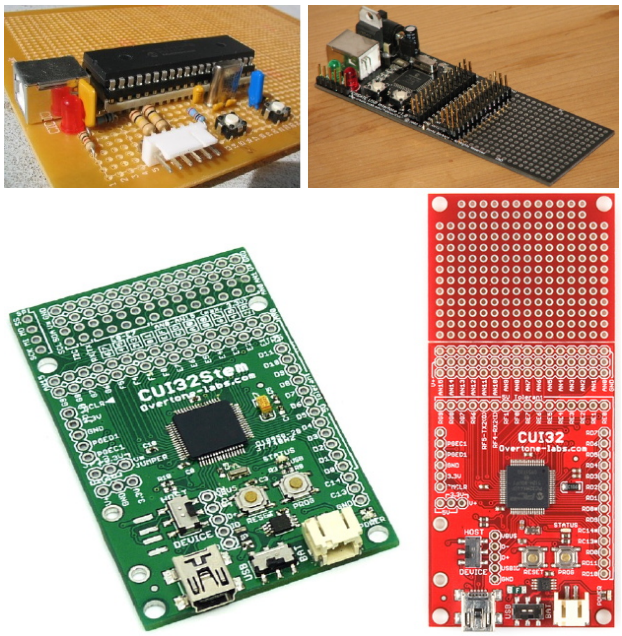

Figure 1. Historical progression of Create USB Interface (CUI) boards, clockwise from top left: The original CUI hand-built in 2004, the first surface-mount CUI in 2005 (black), the CUI32 as currently sold by SparkFun and other distributors (red), and a prototype CUI32Stem (green)

While the microcontroller (PIC32MX795F512H) on the new CUI32Stem is in the same family as the CUI32, and runs at the same clock speed (80MHz), it has more internal memory as well as an internal Ethernet controller (however, an add-on board would still necessary to plug in an Ethernet cable). A benchmark showing the performance of this microcontroller can be seen via its Coremark rating [3]. When compared to a well-known microcontroller in the research community, the CUI32Stem's Coremark is 203, while a standard Arduino has a Coremark of 18. This is with native C-code compiled with full optimization on both; clearly, some of the 'extra' speed of the CUI32Stem is lost when running BASIC code in StickOS – the tradeoff in this case is for ease of use (not needing to install an IDE on your laptop, etc.).

An example musical project worth noting in terms of the processing power of the PIC32, is Philip Burgess' open source project to build a self-contained polyphonic synthesizer [1]. Using piezo sensors as trigger inputs, this project was done with the ChipKIT MPIDE [8] (which allows the Arduino language to be used with PIC32-based boards such as the CUI32Stem). It implements a sample-playback polyphonic synthesizer including real-time effects. Such a project would be impossible on an Arduino, due to lack of sufficient internal memory (and adding a 'wave-shield' or similar to an Arduino in order to access external memory would make it difficult, if not impossible, to achieve the polyphony of overlapping musical notes/sounds, as demonstrated with Burgess' project).

2.2 Integration with the GROVE system

SeeedStudio, Inc. is the purveyor of the GROVE system for prototyping electronic interfaces. SeeedStudio is a Chinese 'open hardware facilitator' where design contributions are encouraged from around the world – via the internet – to add to their increasing collection of 50+ GROVE elements [5]. These elements include a wide range of sensors and actuators that may be useful in different fields of research. Those of primary interest within musical interaction design may be the type of sensors that capture human input (rather than environmental sensors, or others). One of the goals of the GROVE system is to provide accessibility to as many people around the world as possible, and as such the price points set for the GROVE elements (as well as the CUI32Stem) are rather aggressive (inexpensive).

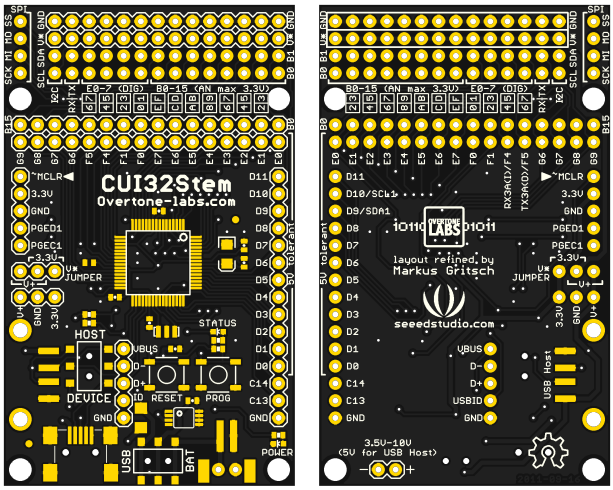

Figure 2. Top and bottom renderings of the latest CUI32Stem – it is very similar to the current CUI32, but swaps the generic prototyping area in favor of 4-pin connectors for SeeedStudio's GROVE system, allowing sensors and actuators to be attached - no need for solder

There are many concepts similar to this GROVE system. For example, Teenage Engineering's 'Oplab' [9], Teague's 'Teagueduino' [14] as well as Microchip's own 'Digilent Cerebot with P-MOD' (Peripheral-Modules) system [4] all allow non-soldering approaches to rapid physical interaction design prototyping. While the Opblab is specifically focused on musical interaction design, the Teagueduino and Microchip offereings are not. CUI32Stem and the GROVE system are also general-purpose interaction design toolkits, but the author's focus nonetheless lies primarily in musical interaction design research. Nonetheless, this generic approach facilitates teaching within many areas, and the need for a more general-purpose system arises commonly among educational programs.

3. WIRELESS OPTIONS

One of the major issues that always tends to arise when designing a NIME, is that of wireless connectivity to a laptop and/or other systems, such as an iPhone/iPad/Android device. The CUI32 has many options for wireless capabilities, of which three have been explored by the author, and are described here.

3.1 Bluetooth

The first wireless option is Bluetooth. Bluetooth can have a maximum of 7 devices paired to one master node, and latency is usually on the order of 10s of milliseconds, both of which are potential problems in musical scenarios. Also, re-connections (if range is inadvertently exceeded, etc.) can be problematic with Bluetooth. Nevertheless, three different Bluetooth modules have been tested with the CUI32 to various degrees of success. Each of them requires some configuration to set them

up correctly according to their datasheets (baud rate and other communication parameters).

The "BlueSMiRF Gold" from SparkFun Electronics has 100 meters range ('class-1' Bluetooth), and costs $65 USD. Another Bluetooth module is available that has only 10 meters wireless range ('class-2 Bluetooth), but is much cheaper at $6 USD[2]. This inexpensive module works acceptably under very constrained circumstances (specifically, unidirectional data transmission), but exhibits a serious problem with bi-directional communication. For example, transmitting data to a master Bluetooth controller (host computer or smartphone) can be achieved without problems, but when interleaving the transmission and reception of data streams, there is unacceptable latency introduced (a full second or more). This seems to be caused by the internal firmware of the cheap Bluetooth module, and is not present with the more expensive BlueSMiRF Gold. Both of these modules work in the same way as would a standard USB-cable connected between the CUI32Stem and a computer, as is explained on the CUI32 website[3] (with downloadable examples for PureData and MaxMSP), including the ability to log into StickOS via terminal program and re-program the CUI32Stem wirelessly if desired.

There is also a 'Serial Bluetooth' module[4] available that is a part of SeeedStudio's GROVE system. The price of this module falls between the other two Bluetooth options at $20 USD, and similarly, its performance lands midway between them. More specifically, it does not have the internal firmware issue limiting bidirectional communications, however it only provides 10 meters range. As part of the GROVE system, this module is the only one that can be connected to the CUI32Stem without soldering a some wires to the Bluetooth module. In addition to Bluetooth, the GROVE system includes other (only unidirectional) radio modules, however these will not be covered here.

3.2 ZigFlea

Another wireless option is ZigBee. The CUI32Stem and StickOS support a subset of ZigBee nicknamed "ZigFlea", which has less bandwidth than Bluetooth (only 250Kbps), but can actually achieve slightly lower latency (due to more efficient firmware stacks), and automatically re-connects when devices have lost contact for some reason. This implementation of ZigFlea utilizes an add-on board for the CUI32Stem designed by Øyvind Nyborg Hauback, which is based on a Freescale (Motorola) MC13201 wireless transceiver. It is programmed as described in the StickOS User's Guide[5].

Two very short BASIC programs are shown in Figure 3, exemplifying how easy it is to use this ZigFlea implementation to set up two CUI32Stem boards to communicate wirelessly with each other. A potentiometer is connected to one of the analog input pins ('an0') on the first CUI32Stem, and this knob is used to control the brightness of an LED (via PWM) on an output pin ('rd0') of the second CUI32Stem via a "remote variable" as declared on line 20 of the BASIC code. As can be seen, remote variable wireless updates take place completely transparently to the user. As long as the remote variable has the same name on both of the CUI32Stems, nodeID 2 in this example has nothing to do in its main 'while-1-do loop', yet the brightness of the remote LED will still change corresponding to the rotation of the knob on the first CUI32Stem.

```
On a CUI32Stem set to nodeid 1, which has a knob connected to an0:

10 dim potentiometer as pin an0 for analog input
20 dim led as remote on nodeid 2
30 while 1 do
40   let led = potentiometer
50   sleep 100 ms
60 endwhile

On a CUI32Stem set to nodeid 2, which has an LED connected to rd0:

10 dim led as pin rd0 for analog output
20 while 1 do
30 endwhile
```

Figure 3. example BASIC programs for a simple ZigFlea wireless connection between two CUI32Stems - the computer is no longer needed after these BASIC programs are saved on the two CUI32Stem boards

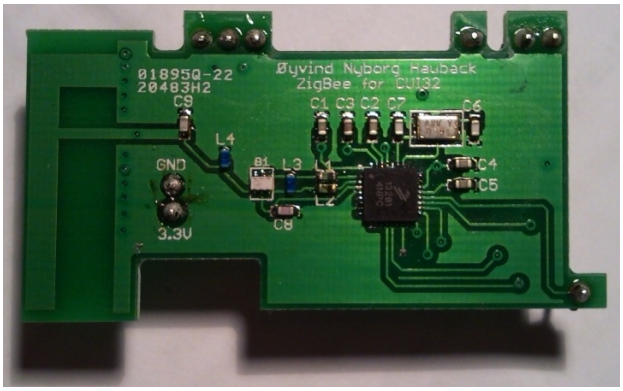

Figure 4. the ZigFlea addon for the CUI32, developed by Øyvind Nyborg Hauback at the University of Olso, Norway, soon to be sold by www.seeedstudio.com

The actual ZigFlea board design was designed in the form of an add-on that plugs onto the top of the CUI32Stem, as shown in figure 4. A 'CUI32 ZigFlea starter kit' software download is available at the Google Code website[6], and includes the necessary StickOS BASIC code as well as a MaxMSP patch for receiving data from a remote node when ZigFlea add-ons are attached to both CUI32Stems. It should be noted that the maximum data rate of ZigFlea in StickOS is somewhat limited currently [16], but that this will be optimized in future StickOS releases.

3.3 WiFi – 802.11b/g

The final option for wireless connectivity on the CUI32Stem is 802.11b/g – commonly known as WiFi. It uses a module called the WiFly (see Figure 5) made by Roving Networks [13] that can be configured to broadcast its own 'AdHoc' 802.11 base-station, which can then be chosen as the wireless network to join in the 'WiFi settings' of your computer, iOS, or Android device. This allows the CUI32Stem to send raw UDP and/or TCP-based OSC packets, and communicate easily with any software that supports Open Sound Control. One of the strengths of this approach is that the CUI32Stem can be used directly with any iOS or Android device, without having to use a laptop as a 'bridge'.

[2] Inexpensive Bluetooth modules are available on e-bay, etc…

[3] http://overtone-labs.ning.com/profiles/blogs/cui32-analog-inputs-in-1

[4] GROVE "Serial Bluetooth" module, http://www.seeedstudio.com/depot/grove-serial-bluetooth-p-795.html

[5] StickOS users's Guide: http://cpustick.com/stickos.htm

[6] ZigFlea starter kit for the CUI32Stem, http://code.google.com/p/cui32/downloads/list

The measurements from sensors attached to the CUI32Stem (GROVE or otherwise) can be used to control any parameters of real-time processes running on a mobile device, such as audio synthesis or effects algorithms in Pd (RjDj [11], LibPd [6]) or SuperCollider [7] (ports to iOS and Android exist). The mapping of such controls to real-time parameter updates is of course a major task given to a composer / performer / developer of the system, as well as any haptic feedback for the user that should be controlled by the CUI32Stem.

Figure 5. The WiFly 802.11b/g radio module.

A major advantage of using WiFi 802.11b/g over either ZigFlea or Bluetooth, is the much greater bandwidth and lower latency it offers. Two examples of sending data to an iOS device include one that sends raw UDP data to RjDj (Pd-vanilla running on iOS), and another that sends Open Sound Control (OSC) packets to SuperCollider running on the iOS device.

As mentioned, it is possible to send raw UDP data to RjDj, which receives it via the [netreceive] object in Pd. While this is specific to RjDj (as it uses PureData internally), the same functionality can be achieved with other iOS or Android applications that are capable of receiving Open Sound Control messages. For example, SuperCollider requires the use of OSC-format strings in order to receive UDP data. Tests have shown that is possible to add the proper OSC syntax (string identifiers and 4-byte boundaries) to a BASIC program on the CUI32Stem to do this. Only 4 wires are necessary to connect the CUI32Stem to the WiFly: V+/GND/TX/RX. These connections will all be made via a single GROVE plug soon (no soldering necessary), as a simple board is being designed to turn this WiFi module into a GROVE element.

In order to use the sensor data from the CUI32Stem with most interactive music programming environments, the data must be sent either via OSC, serially via USB, or use another format that the host application understands (MIDI, HID, etc.). I have focused here on wireless connectivity, but the CUI32Stem website provides many USB examples and corresponding MaxMSP and Pd patches that capture sensor values on the 16 analog input pins on the CUI32Stem. There are also many different types of analog sensors that can be used with CUI32Stem, in addition to those in the GROVE system. The author has online documentation of further examples that incorporate the control of LEDs, small motors, or other actuators for user feedback. See www.overtone-labs.com for more information.

4. CONCLUSION

It is the hope of the author that the system of electronics for new interface development described herein – with a focus on *sketching in hardware* for musical interaction design – is useful to others in the discourse and teaching of NIME-related programs. It is shown in the examples provided, that the use of a high-performance microcontroller and a free RTOS bring about an ease-of-use that can be good for fluid development processes (as well as end users), including students and researchers who do not wish to get 'lost in the details'.

The methodology is intended to encourage us to transcend some of the nuts and bolts of the technological issues encountered when building NIMEs, thereby leaving more time and effort to concentrate on advanced concepts for interaction design, and actual musical practices. While the CUI32Stem and GROVE system are of course general-purpose toolkits, this paper has focused specifically on musical interaction design scenarios for their use. Further explorations of this will be undertaken during the workshop offered at this year's NIME conference in Ann Arbor, Michigan, USA.

5. REFERENCES

[1] Burgess, Phillip. chipKIT Sketch: Mini Polyphonic Sampling Synth. http://hackaday.com/2011/06/08/chipkit-sketch-mini-polyphonic-sampling-synth/, accessed February 7, 2012.

[2] CUI32, as sold by SparkFun Electronics, http://www.sparkfun.com/products/9645 See also: open source firmware at http://code.google.com/p/cui32/ Both accessed February 7, 2012.

[3] Coremark benchmarks, online at http://www.coremark.org/benchmark/, accessed February 7, 2012.

[4] Digilent Cerebot and P-MODS system, http://www.microchipdirect.com/searchparts.aspx?q=cerebot&resperpage=10, accessed February 7, 2012.

[5] GROVE system, from SeeedStudio Inc., http://www.seeedstudio.com/wiki/GROVE_System#Grove_elements accessed February 7, 2012.

[6] LibPd, http://gitorious.org/pdlib/ accessed February 7, 2012.

[7] McCartney, J. Rethinking the Computer Music Language: SuperCollider. *Computer Music Journal, 26(4), 61-68.* 2002

[8] MPIDE, from ChipKIT, http://www.chipkit.cc/wiki/index.php?title=MPIDE_Installation accessed February 7, 2012.

[9] Oplab, from Teenage Engineering, http://www.teenageengineering.com/products/oplab/ accessed February 7, 2012.

[10] Overholt, D. Musical interaction design with the CREATE USB Interface: Teaching HCI with CUIs instead of GUIs. *Proc. of the 2006 International Computer Music Conference*, New Orleans, 2006.

[11] Overholt, D. The Overtone Fiddle: an Actuated Acoustic Instrument. *Proc. of the 2011 New Intefaces for Musical Expression conference*, Oslo, Norway, 2011.

[12] RjDj, http://www.rjdj.me/ accessed January 29, 2012.

[13] Roving Networks (WiFly GSX module), http://rovingnetworks.com/ accessed January 29, 2012.

[14] Teagueduino, from Teague Labs, http://teagueduino.org/, accessed January 29, 2012

[15] Testardi, R. (StickOS), http://cpustick.com/stickos.htm accessed January 29, 2011.

[16] Tørresen, J., Hauback, Ø.N., Overholt, D., and Jensenius, A.R., Development and Evaluation of a ZigFlea-based Wireless Transceiver Board for CUI32. Proc of the 2012 New Interfaces for Musical Expression conference, Ann Arbor, Michigan USA 2012.