Embedded Audio
synthesis and processing from cheap to steep

Dan Overholt
Aalborg University Copenhagen
dano@create.aau.dk
Interactive Audio in Devices

- Annoyed with the annoying beeps some devices make?
  - Let’s bring higher quality audio into sketching!
    - Tools to enable this need to be
      - accessible to designers
      - convenient for users
    - -> Embedded rather than laptop-based
Music Interaction Design: ‘The Fingers’

Example device sketched “the traditional way” (USB interface with laptop generating sound)…

Inspired by Michel Waisvisz ‘The Hands’

Transmits data from MicroNav360 sensors to MaxMSP at 1000 Hz (7ms latency, low jitter)

Simple subtractive synth patch demo that runs in Max/MSP on a laptop

But let’s go back and start cheap!

Blimps at the Click Festival in Helsingør, Denmark

- Built with ATTinys
  (~ $5 each in parts)
- Fast to build, PWM out
  (20 mins each incl cones)
- Sequencer onboard
  (blimps controlled via IR)
Synthlib written by DZL (similar contributed to Arduino)
http://clickfestival.dk/program-2013%20/elevatedaudioworkshop
Wavetable synth in C

Markus Gritsch’s open source polyphonic (multi-voice) musical synth on PIC32

’dead bug’ style, code inspired by http://elm-chan.org/

Similar open source project in Arduino language for PIC32: http://youtu.be/8LdxwfSsjZM

MIDIbox SD Card Polyphonic Sample Player

- An open source polyphonic (multi-voice) musical sampler on ARM-Cortex M3 (devboards available for around $20 from Embedded Artists)

http://www.midibox.org/dokuwiki/doku.php?id=midibox_sd_card_sample_player
Interactive sampler

- Philip Burgess’ open source polyphonic sample-based synth, with delay effect
- Arduino-language with a ChipKIT PIC32 board

Pure Data running on a RPi

- $35, Edgar Berdhal’s “Satellite CCRMA” SD-card image has Pd already installed (+ ChucK, SuperCollider, Audacity, etc…)

- Raspbian GUI only via X11 to optimize audio performance
• Spencer Salazar’s “Spectrum Overdrive” is a basic overdrive distortion guitar effects pedal with one twist — it displays a real-time spectrum of the over-driven signal via a small pico-projector
Android / iOS options

- Steep ($$?)
  - Well, some Android ‘TV-sticks’ are not so expensive, but generally phone/tablet options are much more costly
    - Download free apps like MobMuPlat (Mobile Music Platform) that enables Pd patches to run on iOS…
    - Compile libPD for Android or iOS, SuperCollider, etc.
  - Powerful compared to many embedded options (but still smaller than a laptop)
    - Note: Android currently (still!!) has latency problems…
OK, that was cheap-> steep

- Now a bit about the CUI32Stem board + wireless…
  - Teaching approach: start with BASIC, move on when needed
CUI32Stem (YADB)

- $30, is this somewhat ‘steep’ now?
- Continuation of past research...
- Make it as simple as possible to sketch prototypes
  - Integration with GROVE system of sensors/actuators
  - Focus on making it easy for beginners + versatile for advanced users
CUI32Stem - versatility

Programmable in **BASIC**, **Arduino** or **C-languages** for versatility (enabled by a multi-platform bootloader)...

- **BASIC language**: StickOS operating system is included, with an on-chip BASIC compiler, line editor, debugger, profiler, and in-line help system to create new firmware programs, save them and run them (IDE = any terminal emulator).

- **Arduino language**: Code can be compiled and run on the CUI32Stem using MPIDE - Multi-Platform Integrated Development Environment, a spin-off of the official Arduino IDE (may become part of the official distribution eventually)...

- **C-language**: MPLAB-X (IDE runs on Windows, Linux, and MacOS), many example projects included in the “Microchip Application Libraries”, such as USB-soundcard, USB-MIDI, USB-HID, etc…
CUI32Stem Grove Dash Kit

(~$100)
Individual Grove Elements

Arduinos connect via shield (as you probably knew already):

StickOS overview…

- While C-language or Arduino-language are better suited for high-performance audio applications, StickOS is better suited for some prototyping:

  - StickOS BASIC Features - StickOS was created by Rich Testardi
    - access all on-chip peripheral modules: ADC, PWM, TIMERS, UARTS, I2C, SPI, etc...
    - trace or single-step program execution
    - use profiling to see where the program is spending its time
    - use breakpoints, assertions, and watchpoints
    - use live variable (and pin) manipulation and examination while the program is stopped

- StickOS is more approachable, transparent, and forgiving than many other environments
  - Nonetheless, even casual users will learn the same fundamental concepts that are used by career microcontroller experts (but without the career investment).
StickOS Quick Reference (v1.90)

http://www.cpustick.com

Expressions

The following operators are supported as in C, in order of decreasing precedence:

- \( \) - decimal constant
- \( \) - hexadecimal constant
- \( \) - character constant
- \( \) - simple variable
- \( \) - array variable element
- \( \) - length of array or string
- \( \) - grouping
- \( \) - logical not, bitwise not
- \( \) - times, divide, mod
- \( \) - add, subtract
- \( \) - shift right, left
- \( \) - inequalities
- \( \) - equal, not equal
- \( \) - bitwise or, xor, and
- \( \) - logical or, xor, and

Strings

V5 is a null-terminated view into a byte array v5.

string statements:

<table>
<thead>
<tr>
<th>dim, input, set, print, vprintf</th>
</tr>
</thead>
</table>

if expression relation expression then while expression relation expression do until expression relation expression

string expressions:

<table>
<thead>
<tr>
<th>&quot;literal&quot; - literal string</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables[v], variables[variables] - variable substring</td>
</tr>
<tr>
<td>+ - concatenates strings</td>
</tr>
</tbody>
</table>

string relations:

| = | <> |
|-----------------------------|
| equal, not equal |

contains, does not contain

Variables

All variables must be dimensioned. Variables dimensioned in a sub are local to that sub. Simple variables are passed to sub params by reference. Array variable indices start at 0. v is the same as v[0], except for input/print/stdio statements.

ram variables:

<table>
<thead>
<tr>
<th>dim var[i] = [0];</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim var[i] as (byte short);</td>
</tr>
</tbody>
</table>

flash parameter variables:

<table>
<thead>
<tr>
<th>dim varflash[i] as flash</th>
</tr>
</thead>
</table>

pin alias variables:

<table>
<thead>
<tr>
<th>dim varpin[i] as pin name;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(digital input output);</td>
</tr>
<tr>
<td>(digital output);</td>
</tr>
<tr>
<td>(digital output);</td>
</tr>
</tbody>
</table>

absolute variables:

<table>
<thead>
<tr>
<th>dim var[i] as address addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim var[i] as (byte short) at address addr</td>
</tr>
</tbody>
</table>

system variables (read-only):

<table>
<thead>
<tr>
<th>analog getter keypad;</th>
</tr>
</thead>
</table>

Pin Use the "help pin" command to see MCU-specific pin names and capabilities; use the "pin" command to see display pin assignments.

Commands

cbind: C

clear [line]
clear [line] changes the line to clear.
clear [line] changes the line to clear.

del: delete

delete [line] deletes the line.
delete [line] deletes the line.

dis: display

display [line] displays the line.
display [line] displays the line.

e: enter

e [line] enters the line.
e [line] enters the line.

Expressions

The following operators are supported as in C, in order of decreasing precedence:

- \( \) - decimal constant
- \( \) - hexadecimal constant
- \( \) - character constant
- \( \) - simple variable
- \( \) - array variable element
- \( \) - length of array or string
- \( \) - grouping
- \( \) - logical not, bitwise not
- \( \) - times, divide, mod
- \( \) - add, subtract
- \( \) - shift right, left
- \( \) - inequalities
- \( \) - equal, not equal
- \( \) - bitwise or, xor, and
- \( \) - logical or, xor, and

Strings

V5 is a null-terminated view into a byte array v5.

string statements:

<table>
<thead>
<tr>
<th>dim, input, set, print, vprintf</th>
</tr>
</thead>
</table>

if expression relation expression then while expression relation expression do until expression relation expression

string expressions:

<table>
<thead>
<tr>
<th>&quot;literal&quot; - literal string</th>
</tr>
</thead>
<tbody>
<tr>
<td>variables[v], variables[variables] - variable substring</td>
</tr>
<tr>
<td>+ - concatenates strings</td>
</tr>
</tbody>
</table>

string relations:

| = | <> |
|-----------------------------|
| equal, not equal |

contains, does not contain

Variables

All variables must be dimensioned. Variables dimensioned in a sub are local to that sub. Simple variables are passed to sub params by reference. Array variable indices start at 0. v is the same as v[0], except for input/print/stdio statements.

ram variables:

<table>
<thead>
<tr>
<th>dim var[i] = [0];</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim var[i] as (byte short);</td>
</tr>
</tbody>
</table>

flash parameter variables:

<table>
<thead>
<tr>
<th>dim varflash[i] as flash</th>
</tr>
</thead>
</table>

pin alias variables:

<table>
<thead>
<tr>
<th>dim varpin[i] as pin name;</th>
</tr>
</thead>
<tbody>
<tr>
<td>(digital input output);</td>
</tr>
<tr>
<td>(digital output);</td>
</tr>
<tr>
<td>(digital output);</td>
</tr>
</tbody>
</table>

absolute variables:

<table>
<thead>
<tr>
<th>dim var[i] as address addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>dim var[i] as (byte short) at address addr</td>
</tr>
</tbody>
</table>

system variables (read-only):

<table>
<thead>
<tr>
<th>analog gettau keypad;</th>
</tr>
</thead>
</table>

Pin Use the "help pin" command to see MCU-specific pin names and capabilities; use the "pin" command to see display pin assignments.
Using StickOS for simple I/O

- interfacing the CUI32 with desktop / laptop for sound generation...

- In C (MPLAB-X), HID / USB-audio, USB-MIDI, etc. are possible, but for simple ‘serial’ data StickOS is easiest.

```
150 dim o as pim on14 for analog input
160 dim p as pim on15 for analog input
170 configure timer 0 for 10 ms
180 on timer 0 do print "A",a,b,c,d,e,
190 while 1 do
200 endwhile
```
Sketching embedded interaction

Inspiration: “Knock Clock”, a screen-less clock by CIID students
Knock Clock – in StickOS BASIC

```
10 dim mins, secs, counter
20 dim timeTrigger as pin re7 for digital input debounced
30 rem --- re7 is the PROG switch on the CUI32
40 dim timeTeller as pin re0 for digital output
50 rem --- re0 is the STATUS LED on the CUI32

60 let mins = 0, secs = 0
70 rem --- setting the current time above

80 configure timer 0 for 1 s
90 on timer 0 do gosub timeKeeper

100 while 1 do
110    if timeTrigger=0 then
120        for counter = 0 to mins-1
130            let timeTeller = 0
140            sleep 100 ms
150            let timeTeller = 1
160            sleep 400 ms
170        next
180    sleep 800 ms
190    for counter = 0 to secs-1
200        let timeTeller = 0
210        sleep 100 ms
220        let timeTeller = 1
230        sleep 200 ms
240    next
250 endif
260 endwhile

270 sub timeKeeper
280    let secs = secs+1
290    if secs>60 then
300        let mins = mins+1
310        let secs = 0
320 endif
330 endsub
```
Wireless - motivation

- Urban Musical Game has been created by researchers in the Real-Time Musical Interactions team at IRCAM
Wireless: Bluetooth & Nordic

- Latency: 10s of milliseconds? nRF possibly faster.
- Max 7 slaves paired with 1 master with Bluetooth.
- Bluetooth re-pairing can be annoying…

  - BlueSMiRF Gold, $65
  - GROVE Serial Bluetooth, $20
  - Dealextreme/E-bay, $6
  - Bluegiga modules, $?

- 100 meter range
- 10 meter range
- 10 meter range
- 10 meter range

Note: Bluetooth 4.0 is only connectivity option for iOS devices (without joining Apple’s MFi program)
Wireless options: ZigFlea

- StickOS implements a subset of the ZigBee protocol (ZigFlea does not do node-hopping).  Example:

  On a CUI32Stick 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 CUI32Stick 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
Wireless options: Wi-Fi

- Wi-Fi allows the CUI32Stem to send raw UDP and/or TCP-based OSC packets, and communicate easily with any software that supports Open Sound Control.

- Can be used directly with any iOS or Android device, without having to use a laptop as a ‘bridge’.

- “WiFly” or Xbee Wi-Fi modules can broadcast ‘adhoc’ base-stations or join existing networks.

- can use telnet to connect to CUI32Stem and remotely program it.
Future Directions

- Considering high-quality 16/24-bit audio CODEC as an extension for the CUI32Stem, or with a DIP-package PIC32 (these include I2S support, instead of relying on the PWM ‘hack’ for audio output)…

- This is my personal focus - other platforms need high quality audio sketching capabilities, too! I’d love to hear from all of you about past approaches you’ve used, or any related upcoming plans…


<- MikroElektronika’s Audio Codec Board
Next year we’ll all be sound designers?

- No matter what the platform you’re using (Arduino Due, ???), let’s pursue adding higher quality sound to our sketching platforms…

- A lot of you probably already are there (e.g., CIID’s ‘motors &music’ board includes a 12-bit DAC), but…
Embedded Audio
synthesis and processing from cheap to steep

Thanks!

Dan Overholt
Aalborg University Copenhagen
dano@create.aau.dk