High-Level Web Interface to Low-Level Linux I/O on the BeagleBoard

Embedded Linux Conference 2011, April 11, San Francisco, CA

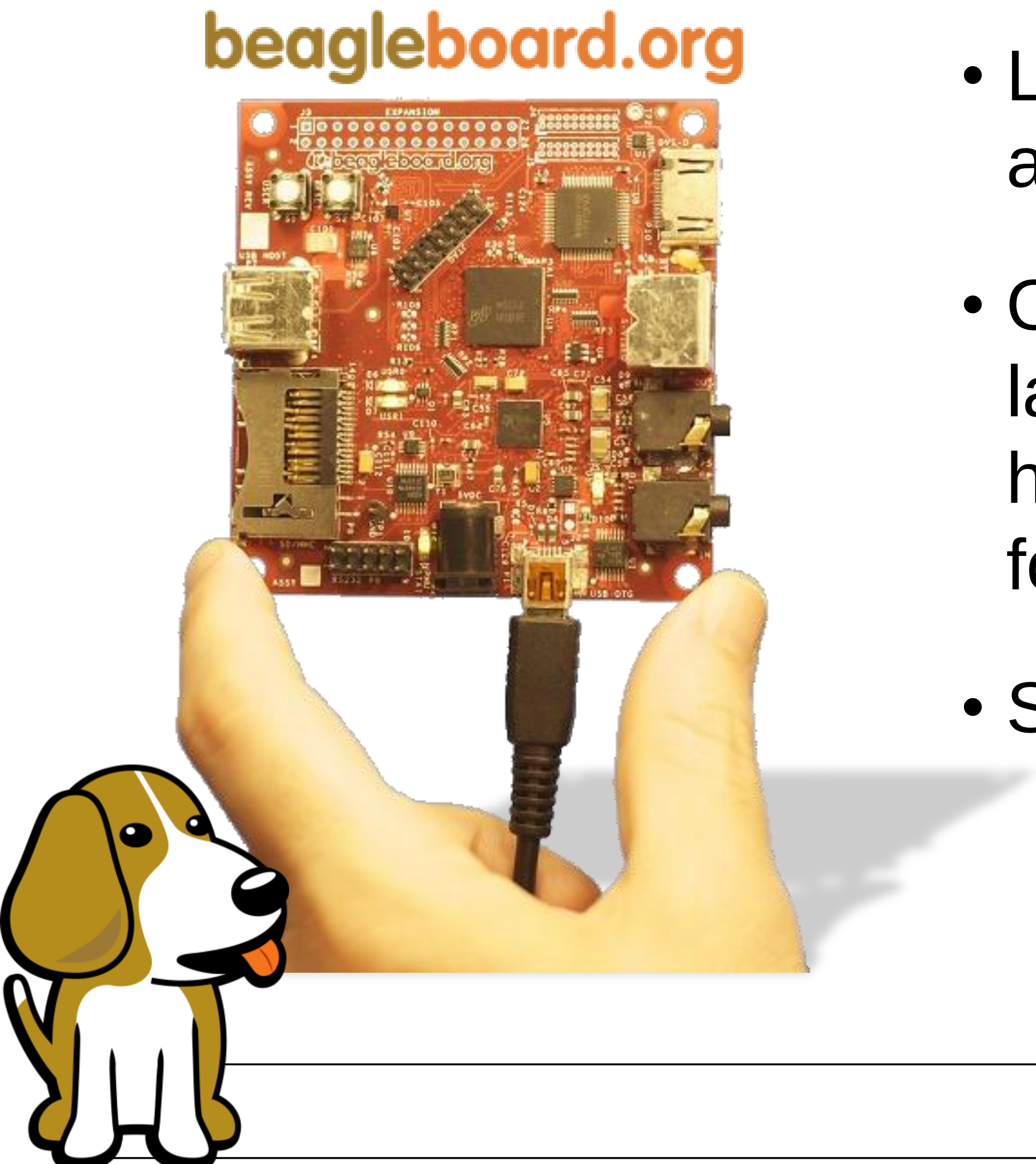
Jason Kridner, jdk@ti.com

Community Development Manager, Sitara™ ARM Microprocessors

Abstract

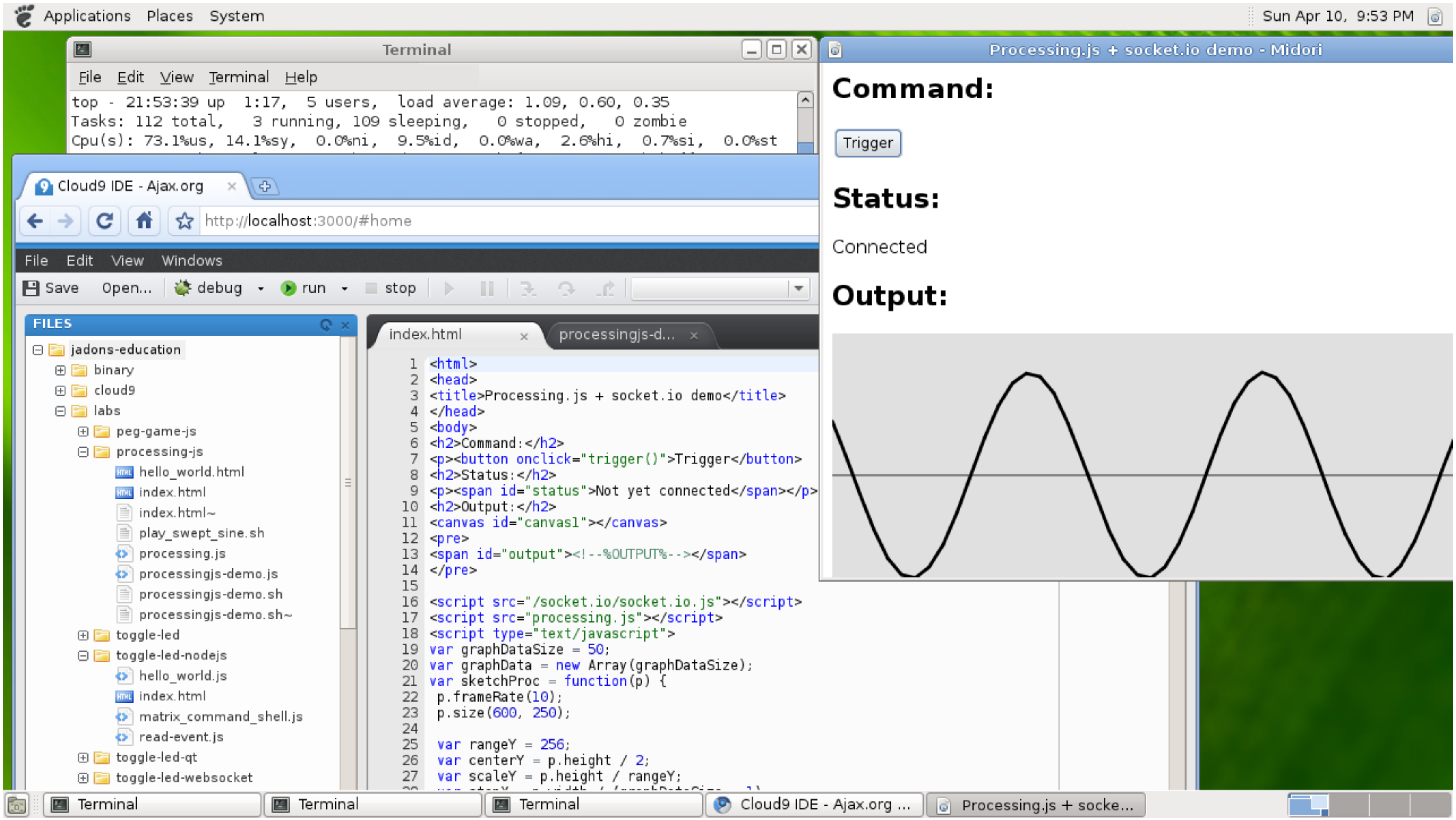
- We will walk through an open source project utilizing the BeagleBoard, the Trainer I/O expansion board and the Node.JS-based Cloud9 IDE to a provide synchronized and responsive JavaScript library and programming environment for performing low-level I/O. By hosting the IDE directly on the BeagleBoard and serving it up locally using QtWebkit, hobbyists can begin twiddling GPIOs and talking to I2C or SPI based sensor devices without installing any software. Only simple web programming skills are required to create a GUI, transparently remote it anywhere on the web and utilize Linux I/O. The audience includes open source developers looking to teach embedded I/O using the Linux kernel and hobbyists looking to rapidly add electronic controls to their applications. Information will be given on how to setup the environment, toggle GPIOs, synchronously update a web page upon getting a Linux input event efficiently using a JavaScript closure, measure the latency and contribute to the project.

Hi Embedded, meet full-distro Linux



- Low-power and affordable (and small)
- Open hardware with large community having many hardware forks
- Support in many distros

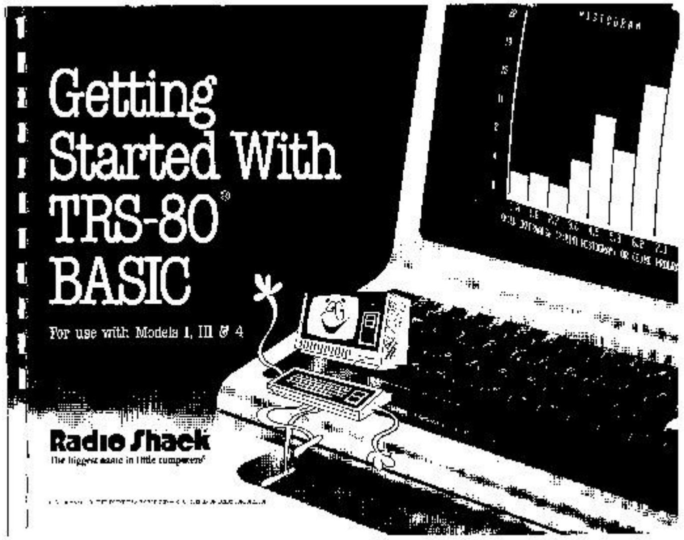
Zero-install visualization tools and editor anywhere on the web in under 2 Watts

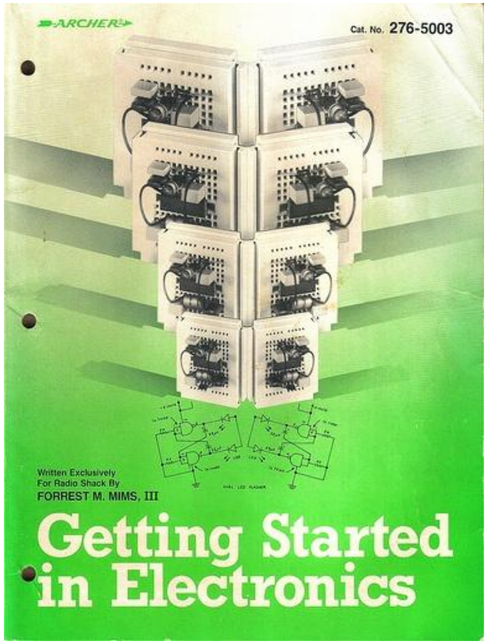


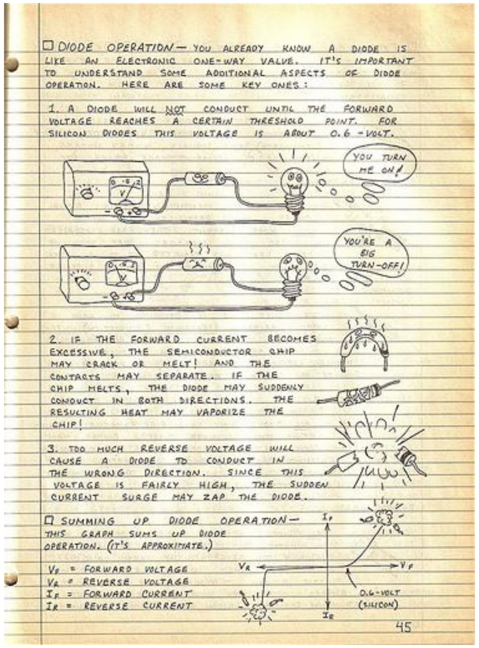
https://gitorious.org/~Jadon/jadons-education

This is where I started learning electronics

- A computer was something you programmed, not just a collection of applications
- Experimentation was encouraged, without fear of destroying the family photo album







☐ DIODE OPERATION — YOU ALREADY KNOW A DIODE IS LIKE AN ELECTRONIC ONE-WAY VALVE. IT'S IMPORTANT TO UNDERSTAND SOME ADDITIONAL ASPECTS OF DIODE OPERATION. HERE ARE SOME KEY ONES:

1. A DIODE WILL NOT CONDUCT UNTIL THE FORWARD VOLTAGE REACHES A CERTAIN THRESHOLD POINT. FOR SILICON DIODES THIS VOLTAGE IS ABOUT 0.6-VOLT.

2. IF THE FORWARD CURRENT BECOMES EXCESSIVE, THE SEMICONDUCTOR CHIP MAY CRACK OR MELT! AND THE CONTACTS MAY SEPARATE. IF THE CHIP MELTS, THE DIODE MAY SUDDENLY CONDUCT IN BOTH DIRECTIONS. THE RESULTING HEAT MAY VAPORIZE THE CHIP!

3. TOO MUCH REVERSE VOLTAGE WILL CAUSE A DIODE TO CONDUCT IN THE WRONG DIRECTION. SINCE THIS VOLTAGE IS FAIRLY HIGH, THE SUDDEN CURRENT SURGE MAY ZAP THE DIODE.

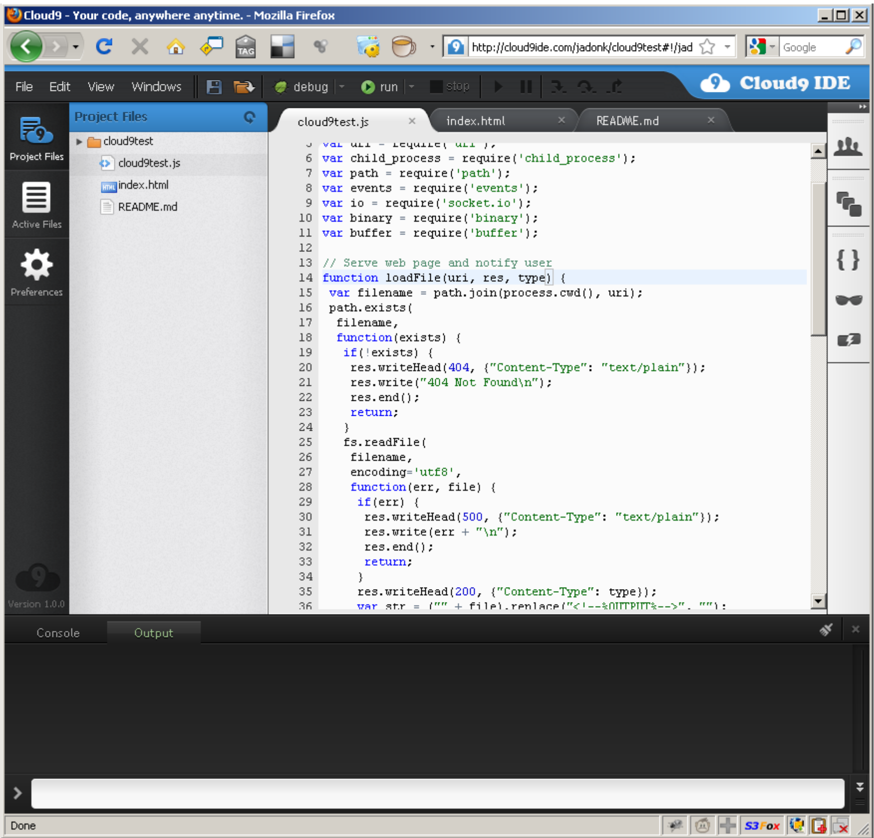
☐ SUMMING UP DIODE OPERATION — THIS GRAPH SUMS UP DIODE OPERATION. (IT'S APPROXIMATE.)

V_F = FORWARD VOLTAGE
V_R = REVERSE VOLTAGE
I_F = FORWARD CURRENT
I_R = REVERSE CURRENT

45

Making the web programmable for all

Cloud9 IDE utilizing the
Ajax.org Code Editor



JavaScript-based in-browser editor merged with Mozilla Bespin/Skywriter project

- Kids and grandparents alike “get” the browser
- A URL is universal
- Eats its own JavaScript dog food
- HTML provides a quick and easy UI
- Possible to encourage good version control

JavaScript provides native closures

```
root@beagleboard:~# node
>   by = function(x) {
...       var multiplier = x;
...       return function(y) {
...        return(y*multiplier);
...      }};
[Function]
>   by4 = by(4);
[Function]
>   by5 = by(5);
[Function]
>   by4(3);
12
>   by5(3);
15
```

- A function reference isn’t just a pointer
- The reference keeps a copy of the local state
- Garbage collection is used to clean up the references

HelloWorld in node.js

```
var http = require('http');
http.createServer(
 function(req, res) {
  res.writeHead(200, {
   'Content-Type':
   'text/plain'
  });
  res.end('Hello World\n');
 }
).listen(8124, "127.0.0.1");
console.log('Server running at
  http://127.0.0.1:8124/');
```

```
setTimeout(function() {
 console.log("world");
}, 2000);
console.log("hello");
```

- Node utilizes the V8 JavaScript interpreter written by Google
- The node.js is CommonJS and its standard libraries make use of closures
- The libraries focus on enabling networking
- Many, many other libraries available via ‘npm’

Event-based programming with node.js

- The node.js event loop runs as long as there are callbacks remaining
- I/O requests are given callbacks instead of leaving it to the operating system to try to find something else to do while waiting on the hardware
- There is no need to create threads to do something when blocked
- events.EventEmitter is a class that you can use to make new objects that allow registering callbacks for events (event listeners)
- JavaScript closures make all of these callbacks really easy to create, unlike interrupt handlers that you'd have to carefully guard how they communicate to the rest of the system
- The approach keeps node.js responding quickly even with many callbacks, open sockets, and event listeners as shown in benchmarks against apache

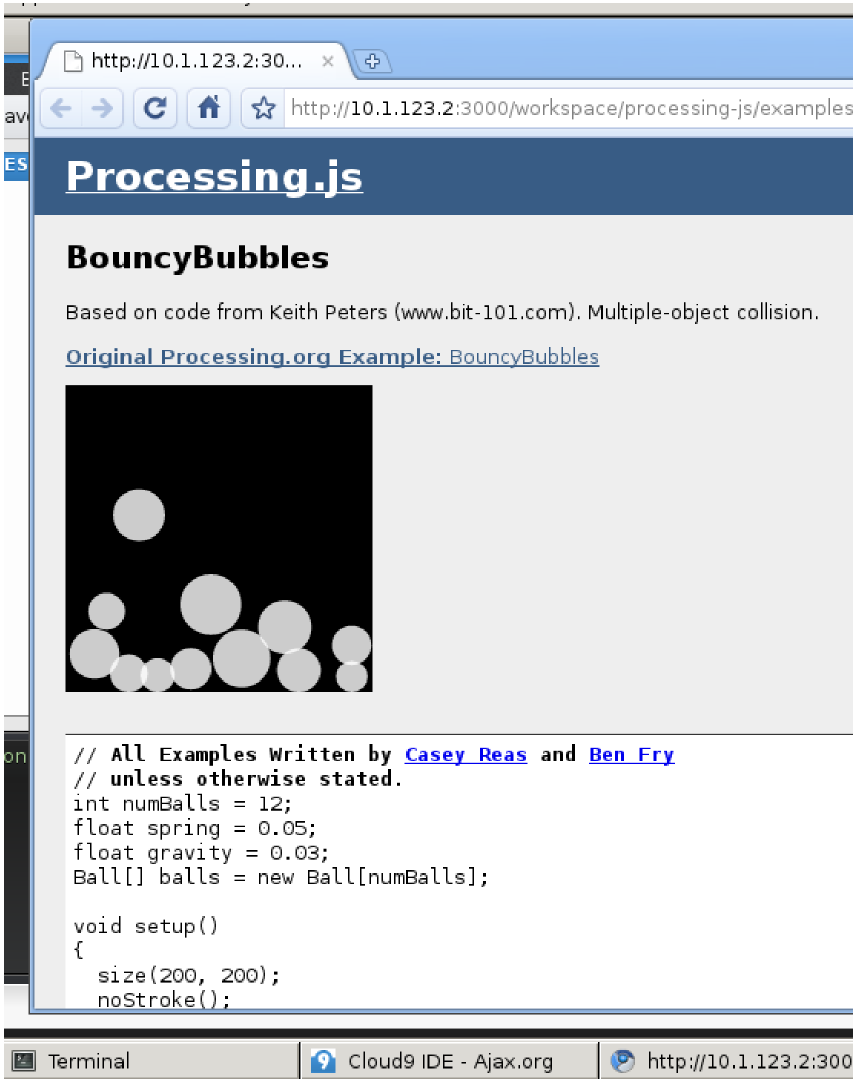
Reading events from Linux device nodes

```
var socket = io.listen(server)
socket.on('connection', function(client) {
 // new client is here!
 sys.puts("New client connected");

 // Function for parsing and forwarding events
 var myListener = function (data) {
  var myEvent = binary.parse(data)
   .word32lu('time1')
   .word32lu('time2')
   .word16lu('type')
   .word16lu('code')
   .word32lu('value')
   .vars;
  myEvent.time = myEvent.time1 + (myEvent.time2 / 1000000);
  var myEventJSON = JSON.stringify(myEvent);
  client.send(myEventJSON + "\n");
 };

 // initiate read
 var myStream = fs.createReadStream(
  '/dev/input/event2',
  {
   'bufferSize': 16
  }
 );
 myStream.addListener('data', myListener);
 myStream.addListener('error', function(error) {
  sys.puts("Read error: " + error);
 });
```

- A new connection will trigger creation of myListener function to send data to the browser
- The standard node.js binary module enables simple parsing
- When the device node provides more data, myListener is called

Texas Instruments

Processing.JS visualization language



- Processing is a Java-based language used as starting point for the Arduino language
- Processing.JS utilizes HTML5 Canvas and JavaScript to do the same thing
- Designed for artists and animators, the language boiler plates many programming tasks

HelloWorld in processing.js

```
f = loadFont("Arial");
size(250, 250);
frameRate(15);

// Main draw loop
void draw(){
 textFont(f, 32);
 r = 100*sin(frameCount / 4);
 background(220);
 fill(0);
 text("hello world", 30, 130+r);
 line(30, 135+r, 180, 135+r);
}
```

- More graphical than textual, so you need to load a font
- draw() loop repeated every frame
- Lots of useful graphical functions and transforms
- Integrates easily with JavaScript syntax

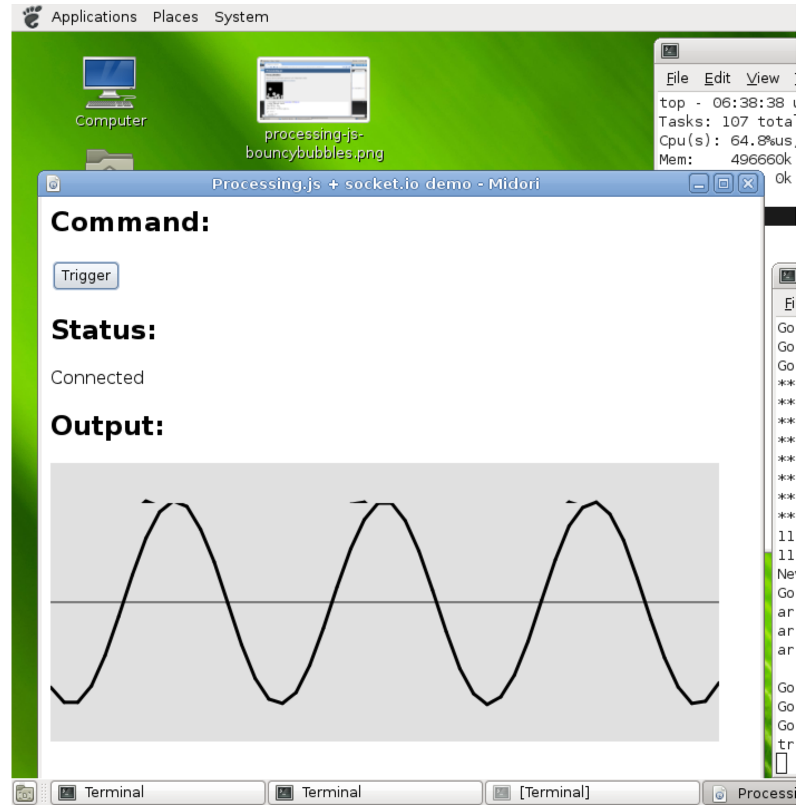
Streaming audio data to processing.js

```
// socket.io
var socket = io.listen(server)
socket.on('connection', function(client) {
 // new client is here!
 sys.puts("New client connected");

 // initiate read
 try {
  var child = child_process.spawn(
   "/usr/bin/arecord",
   [
    "-cl", "-r8000", "-fS8", "-traw",
    "--buffer-size=200", "--period-size=200", "-N"
   ]
  );
  child.stdout.setEncoding('base64');
  child.stdout.on('data', function(data) {
   client.send(data);
  });
  child.stderr.on('data', function(data) {
   sys.puts("arecord: " + data);
  });
  child.on('exit', function(code) {
   sys.puts("arecord exited with value " + code);
  });
 } catch(err) {
  sys.puts("arecord error: " + err);
 }

 // on message
 client.on('message', function(data) {
  sys.puts("Got message from client:", data);
  if(data.match(/trigger/)) {
   child_process.exec(
    "play -bl -cl -r8000 -n synth 10 sine create 200-800 0 0
    function (err, stdout, stderr) {}
   );
  }
 });

 // on disconnect
 client.on('disconnect', function() {
  child.kill('SIGHUP');
  sys.puts("Client disconnected.");
 });
});
```

```

var canvas = document.getElementById("canvas1");
var processing = new Processing(canvas, sketchProc);

var socket = new io.Socket(null, {port: 3001, rememberTransport: false});
socket.connect();
socket.on('connect', function() {
 document.getElementById("status").innerHTML="Connected";
});
socket.on('message', function(data) {
 var myData = window.atob(data);
 for(var i=0; i<graphDataSize; i++) {
  var b = myData.charCodeAt(i);
  if(b>=128) {
   b = b-256;
  }
  window.graphData[i] = b;
 }
});
socket.on('disconnect', function() {
 document.getElementById("status").innerHTML="Disconnected";
 p.exit();
});

trigger = function() {
 socket.send("trigger");
};
```

Performing the display of the data

```
<script src="/socket.io/socket.io.js"></script>
<script src="processing.js"></script>
<script type="text/javascript">
var graphDataSize = 50;
var graphData = new Array(graphDataSize);
var sketchProc = function(p) {
 p.frameRate(10);
 p.size(600, 250);

 var rangeY = 256;
 var centerY = p.height / 2;
 var scaleY = p.height / rangeY;
 var stepX = p.width / (graphDataSize - 1);

 // The draw function will be called frameRate times per second
 p.draw = function() {
  // erase background
  p.background(224);

  // draw axis
  p.stroke(25);
  p.strokeWeight(1);
  p.line(0, centerY, p.width, centerY);

  // draw graph
  p.stroke(0);
  p.strokeWeight(3);
  //p.line(0, centerY+1, p.width, centerY+1);
  var lastX = 0, nextX = 0, lastY, nextY;
  for(var point in graphData) {
   nextY = centerY - (graphData[point] * scaleY);
   if(point != 0) {
    p.line(lastX, lastY, nextX, nextY);
    lastX += stepX;
   }
   nextX += stepX;
   lastY = nextY;
  }
 };
}

var canvas = document.getElementById("canvas1");
var processing = new Processing(canvas, sketchProc);
```



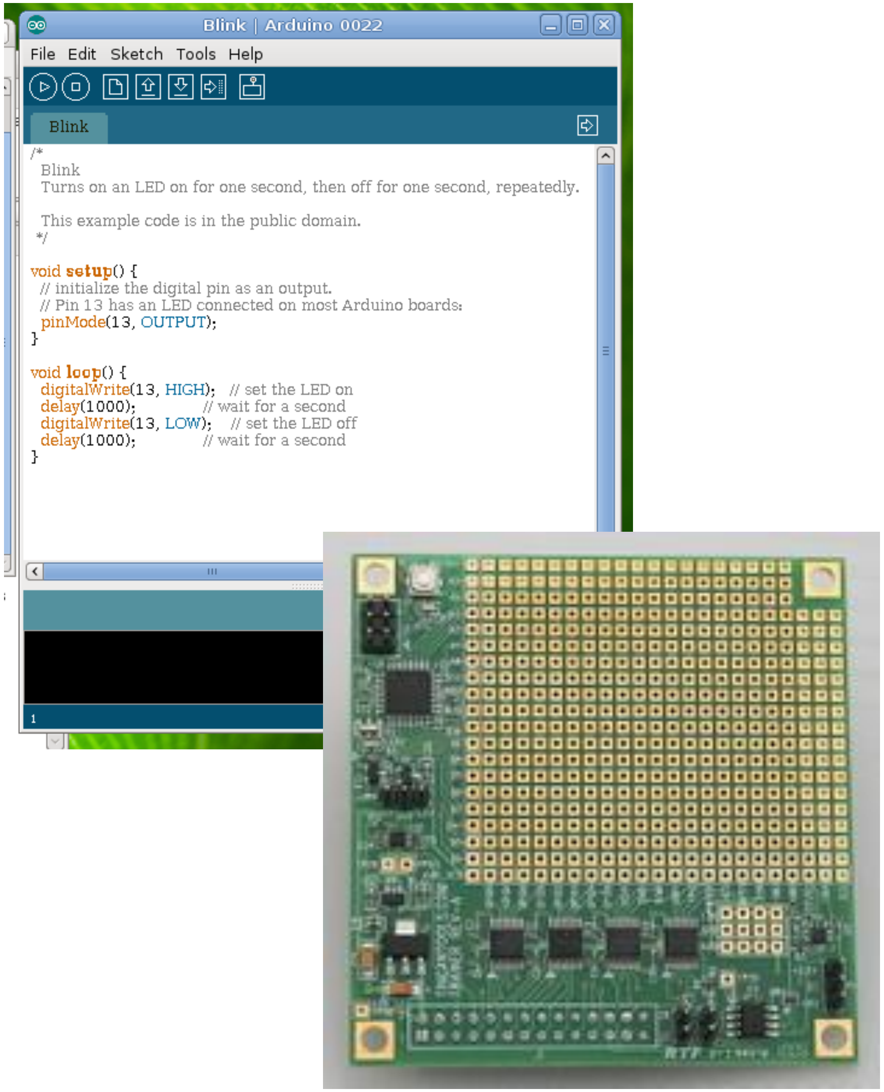
Round-trip performance with DOM update

- A quick check of the performance showed that typical responses were under 20ms for round-trip message→DOM update→message loops across the IP over USB connection to Windows PC running Firefox or Chrome
- Running Midori locally showed a typical response of under 35ms

{"time1":1302500648,"time2":504489,"type":1,"code":108,"value":1,"time":1
302500648.504489,"start":1302500648534}
{"time1":1302500648,"time2":504489,"type":0,"code":0,"value":0,"time":130
2500648.504489,"start":1302500648540}
Got ack from client after 19 milliseconds
{"time1":1302500648,"time2":753144,"type":1,"code":108,"value":2,"time":1
302500648.753144,"start":1302500648755}
{"time1":1302500648,"time2":753205,"type":0,"code":0,"value":1,"time":130
2500648.753205,"start":1302500648759}
Got ack from client after 17 milliseconds
{"time1":1302500648,"time2":792238,"type":1,"code":108,"value":2,"time":1
302500648.792238,"start":1302500648800}
{"time1":1302500648,"time2":792269,"type":0,"code":0,"value":1,"time":130
2500648.792269,"start":1302500648803}
Got ack from client after 13 milliseconds
{"time1":1302500648,"time2":831392,"type":1,"code":108,"value":2,"time":1
302500648.831392,"start":1302500648833}
{"time1":1302500648,"time2":831422,"type":0,"code":0,"value":1,"time":130
2500648.831422,"start":1302500648841}
Got ack from client after 7 milliseconds
{"time1":1302500648,"time2":870485,"type":1,"code":108,"value":2,"time":1
302500648.870485,"start":1302500648872}
{"time1":1302500648,"time2":870515,"type":0,"code":0,"value":1,"time":130
2500648.870515,"start":1302500648880}
Got ack from client after 6 milliseconds
{"time1":1302500648,"time2":909364,"type":1,"code":108,"value":2,"time":1
302500648.909364,"start":1302500648911}
{"time1":1302500648,"time2":909395,"type":0,"code":0,"value":1,"time":130
2500648.909395,"start":1302500648919}
Got ack from client after 15 milliseconds
{"time1":1302500648,"time2":948549,"type":1,"code":108,"value":2,"time":1
302500648.948549,"start":1302500648950}
{"time1":1302500648,"time2":948579,"type":0,"code":0,"value":1,"time":130
2500648.948579,"start":1302500648958}
Got ack from client after 7 milliseconds
{"time1":1302500648,"time2":987550,"type":1,"code":108,"value":2,"time":1
302500648.98755,"start":1302500648989}
{"time1":1302500648,"time2":987581,"type":0,"code":0,"value":1,"time":130
2500648.987581,"start":1302500648998}
Got ack from client after 6 milliseconds
{"time1":1302500649,"time2":26704,"type":1,"code":108,"value":2,"time":13
02500649.026704,"start":1302500649028}
{"time1":1302500649,"time2":26765,"type":0,"code":0,"value":1,"time":1302
500649.026765,"start":1302500649036}
Got ack from client after 12 milliseconds

https://gitorious.org/~Jadon/beagleboard-validation/jadons-education/blobs/master/labs/toggle-led-nodejs/read-event-timed.js

TinCanTools Trainer-xM



- I2C interface (3.3V/5V)
- SPI interface (3.3V)
- GPIOs (3.3V)
- Arduino-compatible Atmega328 processor
  - Connected to the BeagleBoard's 2nd serial
  - Works with avrdude

Plans

- Finish getting the Arduino tools up on the Trainer-xM
- Create simple apps on the Arduino to exercise the BeagleBoard I/Os
  - I2C, SPI and serial data streams with interesting patterns
  - Add A/D conversion on the AVR with triggers to make a better scope demo
  - Invoke Arduino tools from the Cloud9 environment
- Create libraries for the I2C, SPI and serial I/O
- Create an explorer for SYSFS and the kernel device nodes
  - Ultimately want to create a full training on kernel features
- Hook up the interface into QtWebkit browser that loads at boot time
  - Provide SD card images
  - Hopefully update the shipping image to enable zero-install

BACKUP

BeagleBoard-xM details

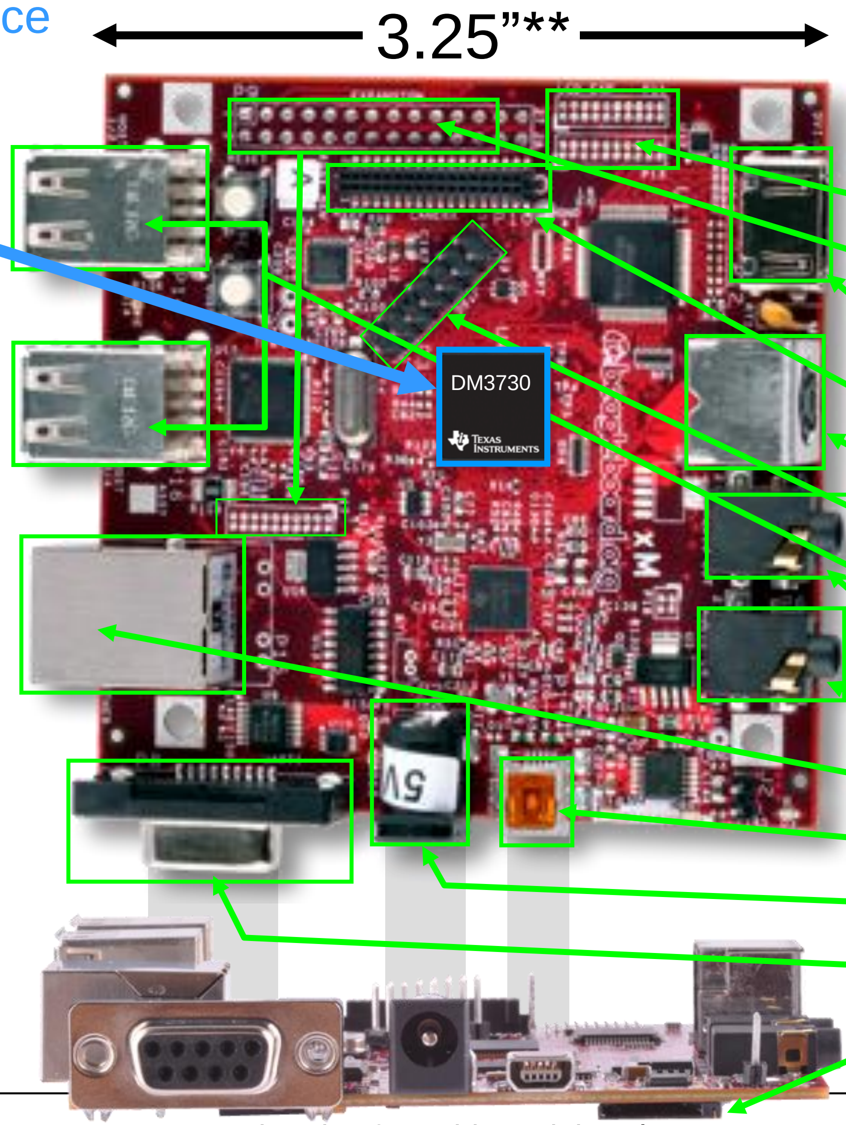
Laptop-like performance

DM3730 processor (AM37x-compatibile)

- 1GHz** superscaler ARM® Cortex ™-A8
- More than 2,000** Dhrystone MIPS
- Up to 20** Million polygons per sec graphics
- 512KB** L2$
- HD video capable C64x+™ DSP core

POP Memory

- 512MB** LPDDR RAM



Desktop-style USB peripherals and embedded style expansion

- LCD Expansion
- I²C, I²S, SPI, MMC/SD Expansion
- DVI-D
- Camera Header**
- S-Video
- JTAG
- 4-port USB 2.0 Hub**
- Stereo Out
- Stereo In
- 10/100 Ethernet**
- USB 2.0 HS OTG*
- Alternate Power
- RS-232 Serial*
- microSD Slot*

* Supports booting from this peripheral
** Change between Rev C4 and BeagleBoard-xM

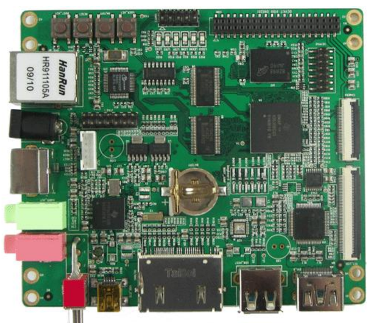
Development Boards Based on BeagleBoard

Compulab

TechNexion

Kwikbyte

EBV Beagle



Devkit8000

Gumstix

RealTime DSP

Variscite

Buglabs

Magniel

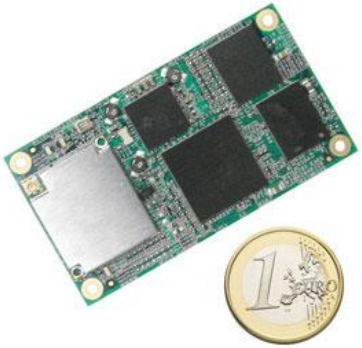
ISEE

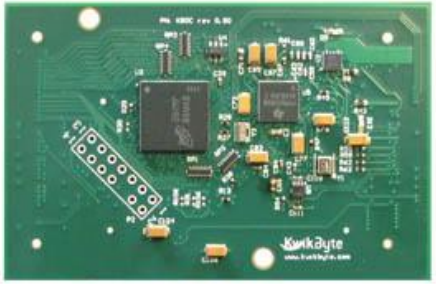
Modules Based on BeagleBoard

Compulab

Analog&Micro

Variscite





TechNexion

Gumstix

Kwikbyte

Add-On Boards and Accessories