O'REILLY®
oscon
open source convention

JULY 16–20, 2012 PORTLAND, OR

#oscon

Bonescript

Simplified Physical Computing with Node.JS

Thursday, July 19, 2012
5:00PM, Open Hardware, D137

Jason Kridner
Software Architecture and Community Development Manager
Sitara ARM Processors
Texas Instruments
jkridner@beagleboard.org or jdk@ti.com

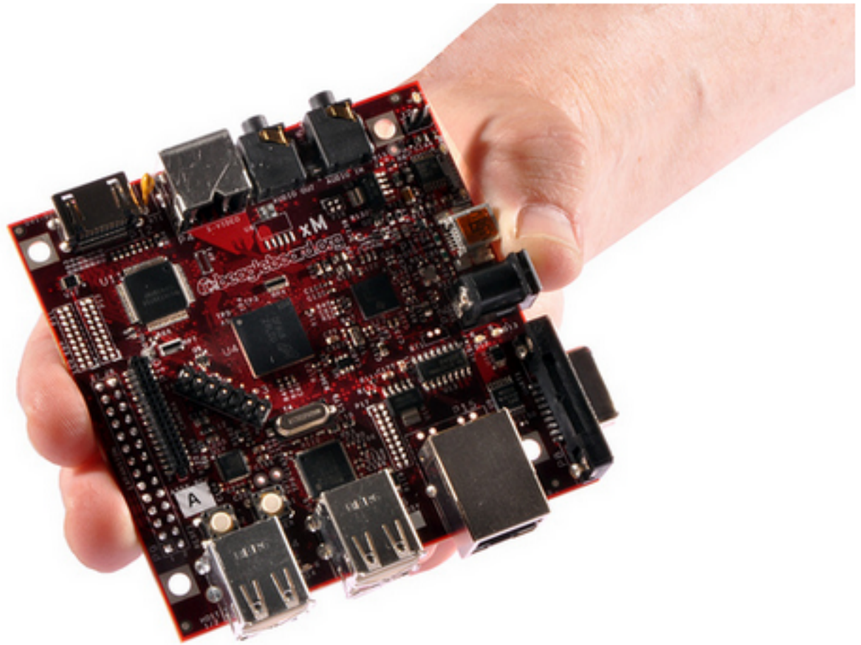
Code at
https://github.com/jadonk/bonescript/tree/oscon-2012

BeagleBoard.org is open access to ARM processors

http://beagleboard.org/static/flyer_latest.pdf

BeagleBoard.org is an open hardware project leveling access to low-power mobile processing

BeagleBoard-xM: Open software desktop experience with extra MHz and extra memory



BeagleBone: Hardware I/O focus with single cable development experience

Arduino reignites electronics interest for non-experts

Not the original BeagleBoard inspiration, but should it have been?

- Removes cognitive barriers to demonstrating hardware interactions
  - *Avoids C++ boilerplate such as "int main(int argc, char* argv) ..."*
  - *Eliminates manual linking to libraries*
- Derived from Processing and then Wiring
- Minimal time to Hello-World-for-hardware: toggling of LED
- Lots of examples on interfacing
- Hundreds of "shields", breakout-boards and tools like Fritzing enable step-by-step examples

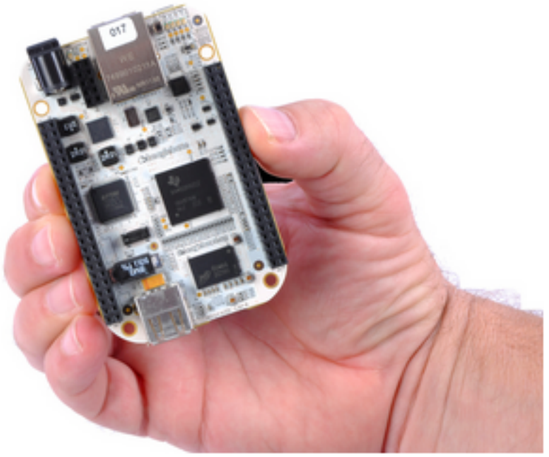
BeagleBone: open hardware expandable computer

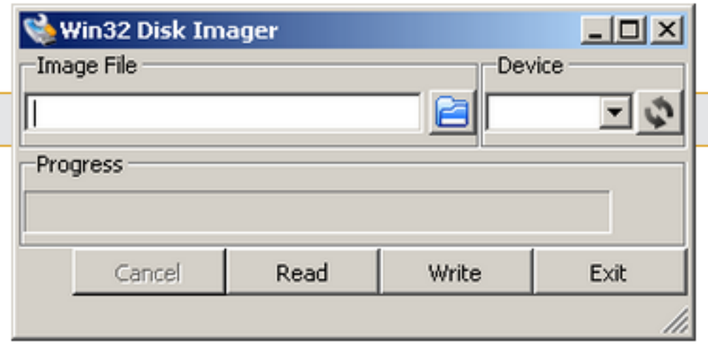
Latest ARM open source focused on easy hardware experimentation

- Ships ready to use
  - *Angstrom Distribution with C++, Perl, Python, ...*
  - *Built-in USB-to-serial adapter with JTAG and device access*
  - *Linux drivers support countless USB peripherals*
- Open source means options
  - *Texas Instruments releases: Android 4.0, Linux, StarterWare (no OS)*
  - *Linux: Angstrom, Ubuntu, Debian, ArchLinux, Gentoo, Sabayon, Buildroot, Erlang, Fedora, TimeSys*
  - *Other: QNX Neutrino, FreeBSD*
- SD card images like get-out-of-jail-free card

```
xzcat XXX.img.xz | sudo dd of=/dev/sdX
```

  - *Can be used just as easily for backups*
  - *Booted from device ROM, so you can't "brick" it*





Bonescript enables physical computing exploration

Linux handles the interfaces, JavaScript provides the glue

- Builds on two key familiar programming environments: web and Arduino
  - *JavaScript is language of web and first logical choice for new programmers*
  - *HTML5 and libraries provides rapid prototyping of GUIs and ability to access remotely*
- Benefits from being run on complete Linux computer
  - *Networking and USB stacks*
  - *Extensive libraries and processing performance, such as OpenCV*
  - *Web-based zero-install experience*
  - *Avoids need to get low-level permissions on shared machine*
- Event-based model of NodeJS simplifies asynchronous interactions
  - *I/O interactions are relatively slow compared to the CPU*
  - *Lends itself naturally to many event handlers*
- Enables single-language environment, including IDE
- Goal is a self-documenting interactive teaching web experience
  - *Currently using W3C Slidy for presentation view*

First experience of Bonescript

Familiar Arduino function calls, exported to the browser

The links below are live and will impact the USR3 LED on your BeagleBone. The exact code used in the browser is below and will send messages to your board using Socket.IO.

Set pin high: run

```
pinMode(bone.USR3, 'out');
digitalWrite(bone.USR3, 1);
```

Set pin low: run

```
digitalWrite(bone.USR3, 0);
```

All GPIOs accessible via digitalWrite and digtalRead.

Native sketch-like applications

Bonescript application code uses the typical setup/loop global functions

Some boilerplate required, but fairly minimal. Below is blinking LED app with the familiar blocking style:

```
require('bonescript');
ledPin = bone.USR3;
setup = function() {
    pinMode(ledPin, OUTPUT);
};
loop = function() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
};
```

Installation and execution of above 'blinkled.js' with node 0.6.x on a supported platform would look something like:

```
# npm install bonescript
# node blinkled.js
```

Interactively add new loops

Bonescript adds addLoop function and multiple loops

- Arduino sketches are designed to block
- Web apps don't want to run all loops at beginning of time
- Each loop is a fork using NodeJS 'cluster' feature
  - *Not in the spirit of NodeJS, but goal is simplicity and growth*
  - *Also added blocking 'delay' call for familiarity*
  - *Works well for independent processes*
- Vision is to enable moving loops to dedicated processors or RT threads
  - *Loops are local strings with one-time global reference*
  - *Callbacks forwarded when loop returns a non-false value*
  - *Performance is reasonable today for many interactive apps*

addLoop/getLoops/removeLoop

Run something repeatedly

Blink USR3 LED: run

```
pinMode(bone.USR3, 'out');
addLoop(function() {
  digitalWrite(bone.USR3, 1);
  delay(100);
  digitalWrite(bone.USR3, 0);
  delay(100);
});
```

Halt all loops: run

```
getLoops(function(x) {
  for(var loop in x.loops) {
    removeLoop(loop);
  }
});
```

setInterval/clearInterval/setTimeout

Run something repeatedly without invoking a process

- Starts faster than addLoop because it isn't invoking a process
- Consumes less CPU on target
- Puts load on browser machine and communications in this example
- Here we demonstrate on the client browser, but NodeJS provides same functions

Blink USR3 LED for 30 seconds: run

```
value = 0;
pinMode(bone.USR3, 'out');

interval = setInterval(function() {
  value = value ? 0 : 1;
  digitalWrite(bone.USR3, value);
}, 100);

setTimeout(function() {
 clearInterval(interval);
}, 30000);
```

doEval

Run something remotely just once

What if we want to invoke the previous example on the target, rather than the browser?

Blink USR3 LED for 30 seconds: run

```
doEval(function(callback) {
 var value = 0;
 pinMode(bone.USR3, 'out');

 var interval = setInterval(function() {
  value = value ? 0 : 1;
  digitalWrite(bone.USR3, value);
 }, 100);

 setTimeout(function() {
  clearInterval(interval);
  callback({'notice': "Removing doEval demo interval function"});
 }, 30000);

 return({'notice': "Launched doEval demo interval function"});
}, doAlert);
```

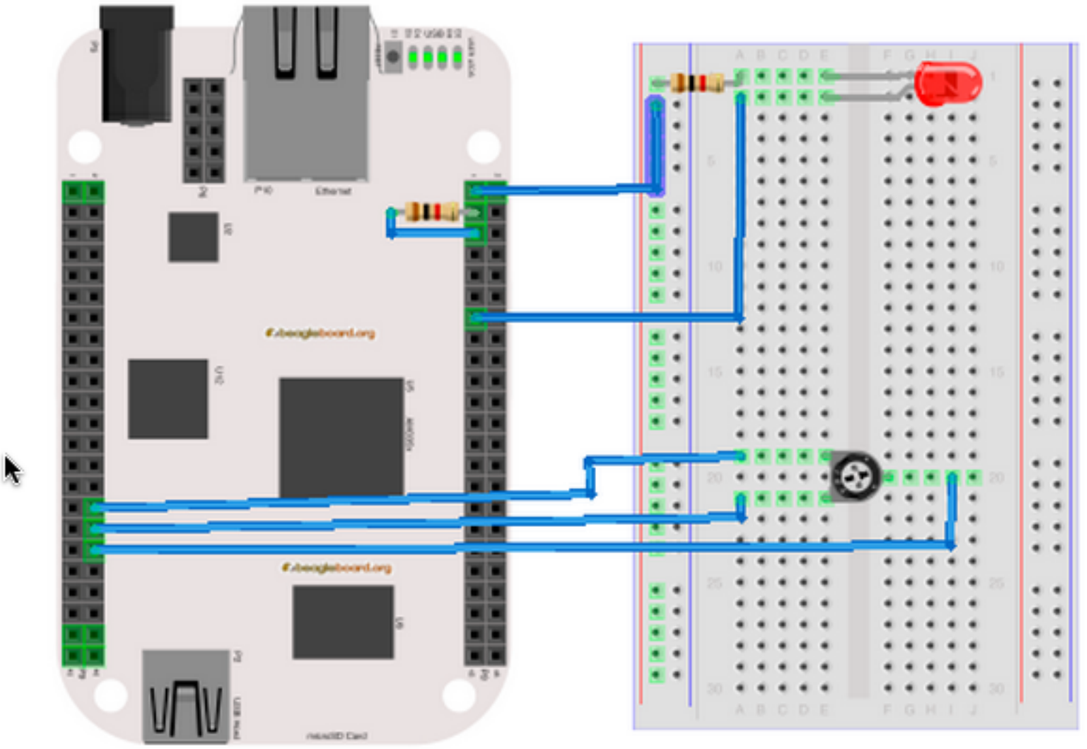
- Function converted to string when passed from client to server
- On client, 'callback' is a proxy for the 'doAlert' callback provided to doEval

Connecting up some inputs

Wire up this Fritzing drawing to give us some stimulus

- 2x 1kohm resistor
- 1x LED
- 1x potentiometer

You can download the BeagleBone Fritzing part



Made with Fritzing.org

attachInterrupt/detachInterrupt

Detect input changes

Please connect P8-3 to P8-5 with a 1khom resistor

Attach interrupt and toggle P8-3 for 30 seconds: run

Last value read was: **0**

```
pinMode(bone.P8_5, 'in');
pinMode(bone.P8_3, 'out');
value = 0;
digitalWrite(bone.P8_3, value);
function interruptHandler(x) { return({'value': x.value}); }
function interruptCallback(x) {
 if(x.output) $('#attachInterruptValue').text(x.output.value);
 else doAlert(x);
}
function toggleP8_3() {
  value = value ? 0 : 1;
  digitalWrite(bone.P8_3, value);
}
attachInterrupt(bone.P8_5, interruptHandler, 'both', interruptCallback);
var interval = setInterval(toggleP8_3, 200);
setTimeout(function() {
 clearInterval(interval);
 detachInterrupt(bone.P8_5, doAlert);
}, 30000);
```

analogRead/analogWrite

Use ADCs and PWMs to read/write analog values

Monitor analog in: run

```
pinMode(bone.P8_13, 'out', 4);
addLoop(function() {
 var value = analogRead(bone.P9_36);
 analogWrite(bone.P8_13, value);
}, 25, doAlert);
```

Fade in and out: run

```
 pinMode(bone.P8_13, 'out', 4);
 addLoop(function() {
  if(!this.awDirection) { this.awValue = 0.01; this.awDirection = 1; }
  analogWrite(bone.P8_13, this.awValue);
  this.awValue = this.awValue + (this.awDirection*0.01);
  if(this.awValue > 1.0) { this.awValue = 1.0; this.awDirection = -1; }
  else if(this.awValue < 0.01) { this.awValue = 0.01; this.awDirection = 1; }
 }, 10, doAlert);
```

Halt all loops: run

```
 getLoops(function(loops){for(var loop in loops.loops){removeLoop(loop, doAlert);}});
```

Using a slider with analogWrite

Please connect P8-13 to LED through 1khom resistor

Attach to slider: run

```
pinMode(bone.P8_13, 'out', 4);
$("#slider1").bind("slidechange", function(event, ui) {
 analogWrite(bone.P8_13, ui.value/100.0);
})
```

Using a slider with analogRead

Please connect P9-36 to potentiometer and adust from 0V to 1.8V

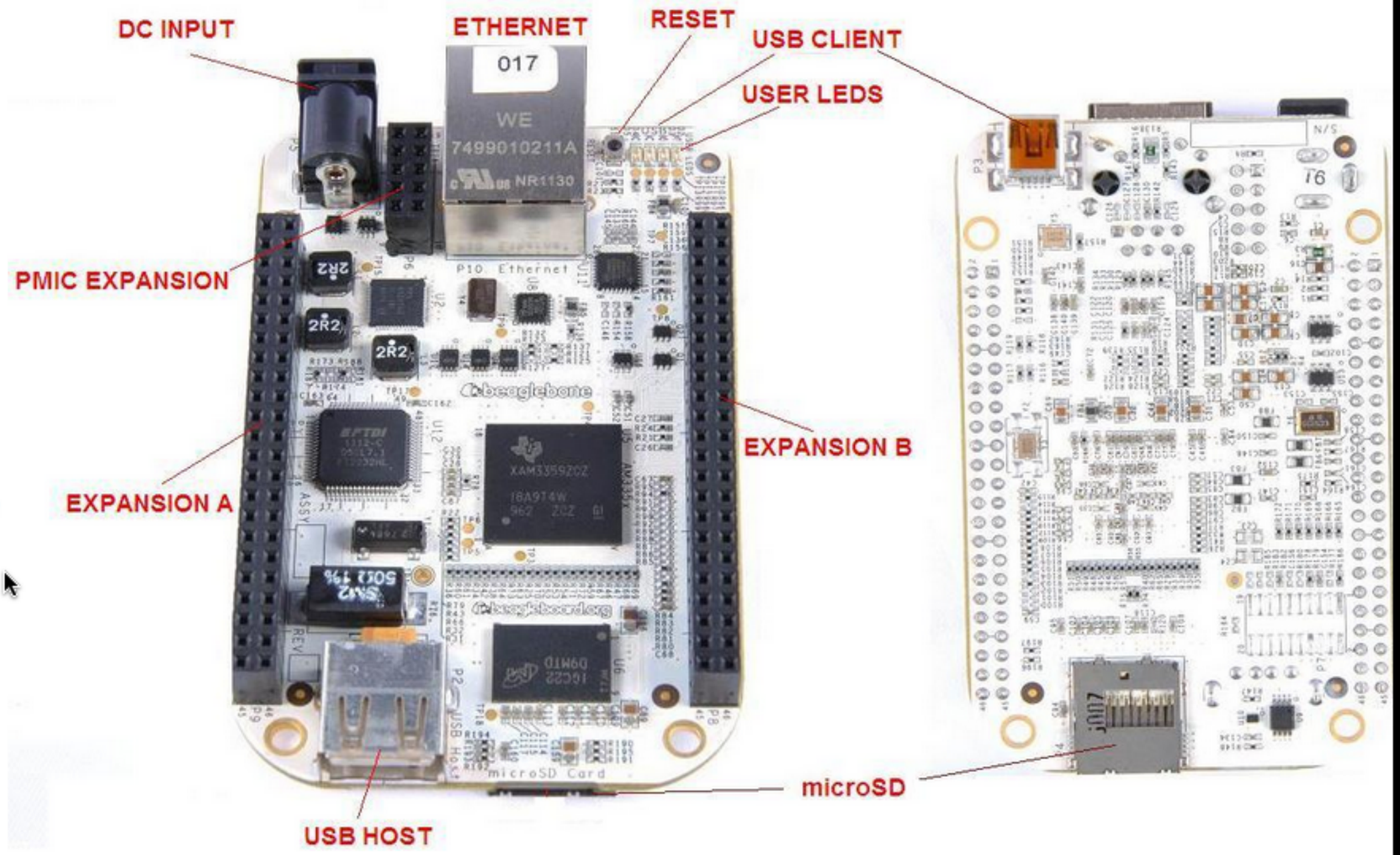
Attach to slider for 30 seconds: run

```
arUpdateSlider = function(x) {
 if(x.value) $("#slider2").slider("option", "value", x.value*100);
 if(x.halt) alert("Halting analogRead slider update");
};

doEval(function(callback) {
 var interval = setInterval(function() {
  analogRead(bone.P9_36, callback);
 }, 100);

 setTimeout(function() {
  clearInterval(interval);
  callback({'halt':true});
 }, 30000);
}, arUpdateSlider);
```

BeagleBone hardware details

http://beagleboard.org/bone

- Processor
  - *720MHz super-scalar ARM Cortex-A8*
  - *armv7 and NEON SIMD instructions*
  - *3D graphics accelerator*
- Connectivity
  - *USB client: power, debug and device*
  - *USB host, Ethernet*
  - *2x46 pin headers*
    - 7xADC, 66xGPIO
    - 2xI2C, 5xUART, SPI, CAN, 8xPWM
    - LCD, parallel, MMC/SDIO
- Software
  - *4GB microSD card with Angstrom Distribution*



Expansion header state

Pin mux can be read and altered with pinMode and getPinMode

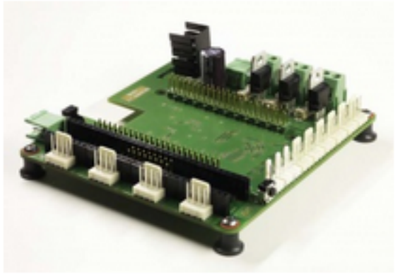
P9

	1	2	
DGND			DGND
VDD_3V3			VDD_3V3
VDD_5V			VDD_5V
SYS_5V			SYS_5V
PWR_BUT			SYS_RESETn
0: gpmc_wait0			7: gpio1_28
7: gpio0_31			7: gpio1_18
0: mii1_rxd3			7: gpio1_19
2: i2c1_scl			2: i2c1_sda
3: i2c2_scl			3: i2c2_sda
7: gpio0_3			7: gpio0_2
7: gpio1_17			7: gpio0_15
7: gpio3_21			7: gpio0_14
7: gpio3_19			3: spi1_cs0
3: spi1_d0			3: spi1_d1
3: spi1_sclk			VDD_ADC
AIN4			GNDA_ADC
AIN6			AIN5
AIN2			AIN3
AIN0			AIN1
3: clkout2			7: gpio0_7
DGND			DGND
DGND			DGND
	45	46	

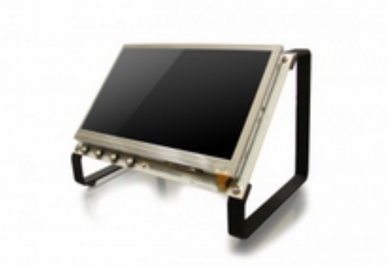
P8

	1	2	
DGND			DGND
7: gpio1_6			0: gpmc_ad7
7: gpio1_2			7: gpio1_3
0: gpmc_advn_			0: gpmc_oen_re
0: gpmc_ben0_			0: gpmc_wen
7: gpio1_13			7: gpio1_12
4: ehrpwm2B			7: gpio0_26
7: gpio1_15			7: gpio1_14
7: gpio0_27			7: gpio2_1
7: gpio0_22			7: gpio1_31
7: gpio1_30			0: gpmc_ad5
0: gpmc_ad4			0: gpmc_ad1
0: gpmc_ad0			0: gpmc_csn0
7: gpio2_22			7: gpio2_24
7: gpio2_23			7: gpio2_25
7: gpio0_10			7: gpio0_11
7: gpio0_9			7: gpio2_17
7: gpio0_8			7: gpio2_16
7: gpio2_14			7: gpio2_15
7: gpio2_12			7: gpio2_13
7: gpio2_10			7: gpio2_11
7: gpio2_8			7: gpio2_9
7: gpio2_6			7: gpio2_7
	45	46	

Capes extend BeagleBone capability

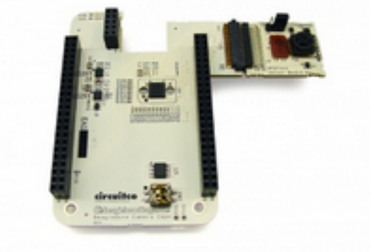


BeBoPr Cape
AES electronics



BeagleBone LCD7 Cape
BeagleBoardToys

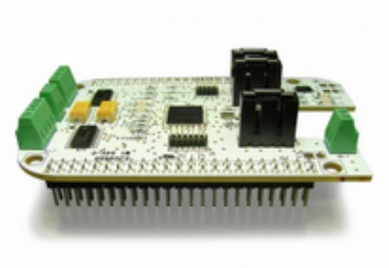
TT3201 CAN Cape
TowerTech



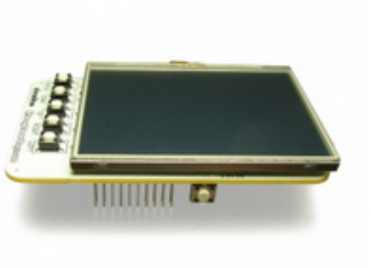
BeagleBone Camera Cape
BeagleBoardToys

Proto Cape Kit
Adafruit

BeagleBone VGA Cape
BeagleBoardToys



BeagleBone MSTP Cape
Plano CAD



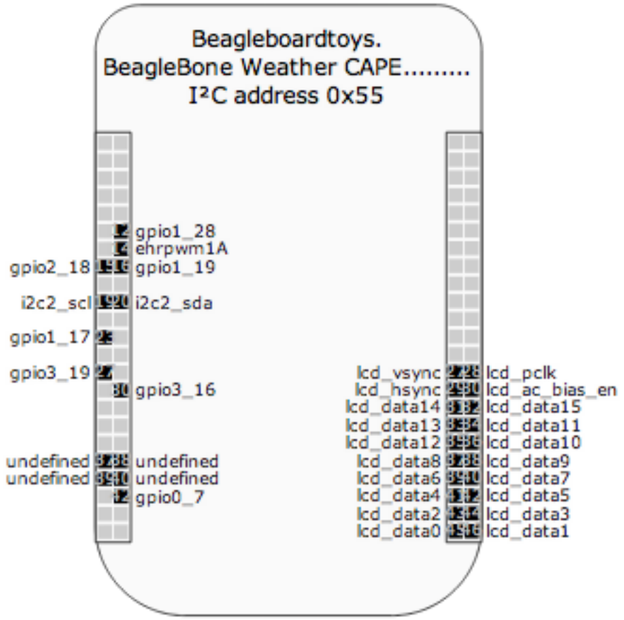
BeagleBone LCD3 Cape
BeagleBoardToys

Attached capes

Any attached capes with ID EEPROMs would be listed here

BeagleBone A3

Serial number: 4211BB000012



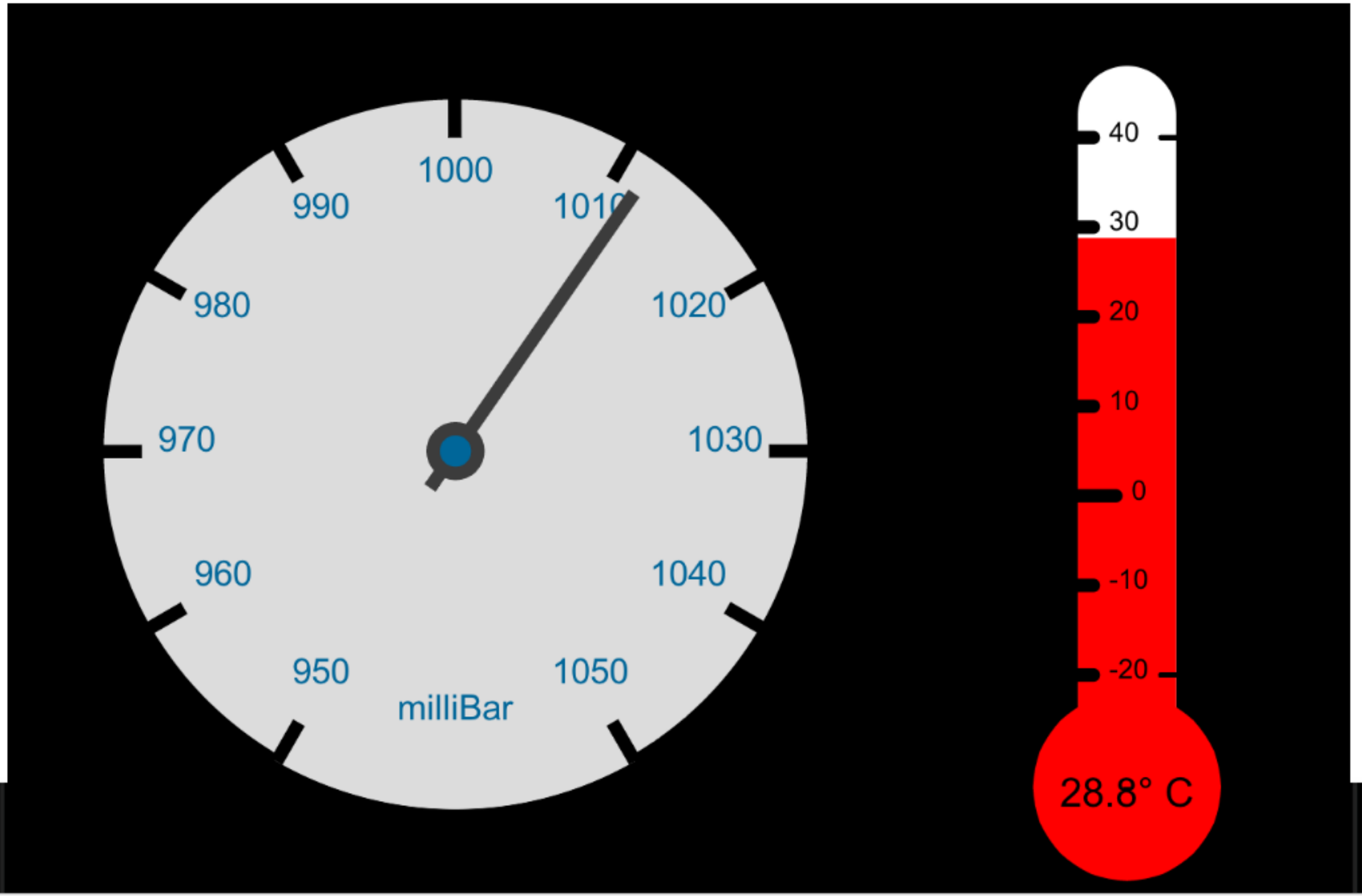
Weather station setup

Here we initialize the weather station demo: run

```
function updateWeather(x) {
 if(x.startup) { pressureRead(); tempRead(); }
 else if(x.temp) { temperatureUpdate(x.value); setTimeout(tempRead, 500); }
 else if(x.pressure) { pressureUpdate(x.value); setTimeout(pressureRead, 3000);}
};

function tempRead() {
 doEval(function(callback) {
  fs.readFile("/sys/bus/i2c/drivers/bmp085/3-0077/temp0_input", function(err, data) {
   if(!err) callback({'temp': true, 'value': '' + data});
  });
 }, updateWeather);
};

function pressureRead() {
 doEval(function(callback) {
  fs.readFile("/sys/bus/i2c/drivers/bmp085/3-0077/pressure0_input", function(err, data) {
   if(!err) callback({'pressure': true, 'value': '' + data});
  });
 }, updateWeather);
};

doEval(function(callback) {
 try {
  fs.writeFileSync('/sys/class/i2c-adapter/i2c-3/new_device', 'bmp085 0x77', encoding='ascii');
  callback({'startup':'bmp085 driver successfully loaded'});
 } catch(ex) {
  callback({'startup':'bmp085 driver load failed: ' + ex});
 }
}, updateWeather);
```

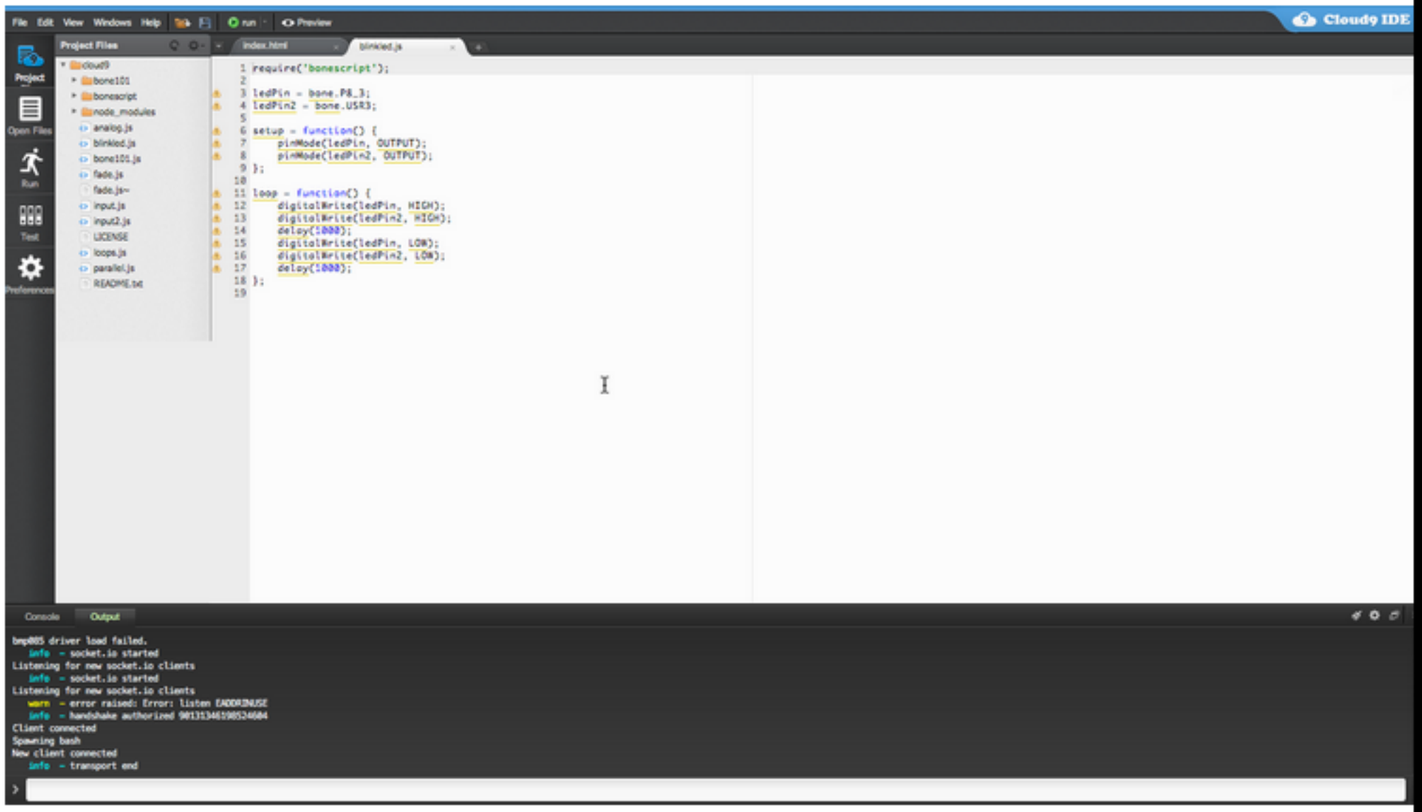
Weather station demo

Click here to start running the code on the previous page: run



Cloud9 experience

- Running on boot-up on the BeagleBone
- Starts-up with directory of Bonescript examples



Giving back!

http://github.com/jadonk/bonescript

- There's more to come. Please join me in improving this project for the benefit of both yourself and others.
- Pull down the latest using git

```
cd /var/lib/cloud9
git pull origin master
```

- File bug reports via the github interface
- License is MIT-like with contributions welcome