Taking the BeagleBone Cookbook recipes beyond BeagleBone Black

January 28, 2016

Mark Yoder and Jason Kridner

Authors of BeagleBone Cookbook and
BeagleBoard.org Foundation board members

Description

- BeagleBoards and BeagleBones are inexpensive web servers, Linux desktops, and electronics hubs that include all the tools you need to create your own projects—whether it's robotics, gaming, drones, or software-defined radio. This webcast will go over some of the recipes in the BeagleBone Cookbook that go beyond BeagleBone Black for connecting and talking to the physical world with this credit-card-sized computer.

- In this webcast you will learn:
  - What is BeagleBone Black? What can you do with BeagleBone Black?
  - What basic skills will “BeagleBone Cookbook” help me develop?
  - What are some other BeagleBoards coming out, including SeeedStudio BeagleBone Green, SanCloud BeagleBone Enhanced, BeagleBoard.org BeagleBone Blue and BeagleBoard.org BeagleBoard-X15
  - What recipes will work with these other boards and how do I apply them?

BeagleBone Black

Ready to explore and use in minutes

Truly flexible open hardware and software development platform

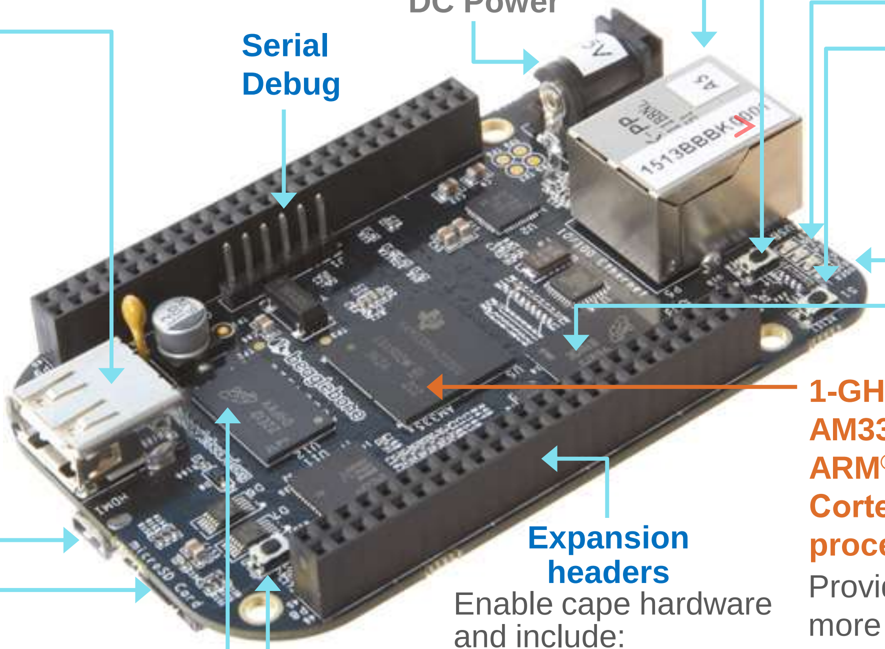
All you need is in the box

Proven ecosystem from prototype to product

- Ready to use
  - USB client network
  - Built-in tutorials
  - Browser based IDE
  - Flashed w/Debian
- Fast and flexible
  - 1-GHz Sitara ARM
  - 2x200-MHz PRUs
  - 512-MB DDR3
  - On-board HDMI
  - 65 digital I/O
  - 7 analog inputs
- Support for numerous Cape plug-in boards
  http://beaglebonecapes.com

BeagleBone Black – **the most flexible solution in open-source computing**

BeagleBone Black board features



10/100 Ethernet

USB Host
Easily connects to almost any everyday device such as mouse or keyboard

microHDMI
Connect directly to monitors and TVs

microSD
Expansion slot for additional storage

512MB DDR3
Faster, lower power RAM for enhanced user-friendly experience

DC Power

Serial Debug

Boot Button

Expansion headers
Enable cape hardware and include:
- 65 digital I/O
- 7 analog
- 4 serial
- 2 SPI
- 2 I2C
- 8 PWMs
- 4 timers
- And much much more!

1-GHz Sitara AM335x ARM® Cortex™-A8 processor
Provides a more advanced user interface and up to 150% better performance than ARM11

Power Button

LEDS

Reset Button

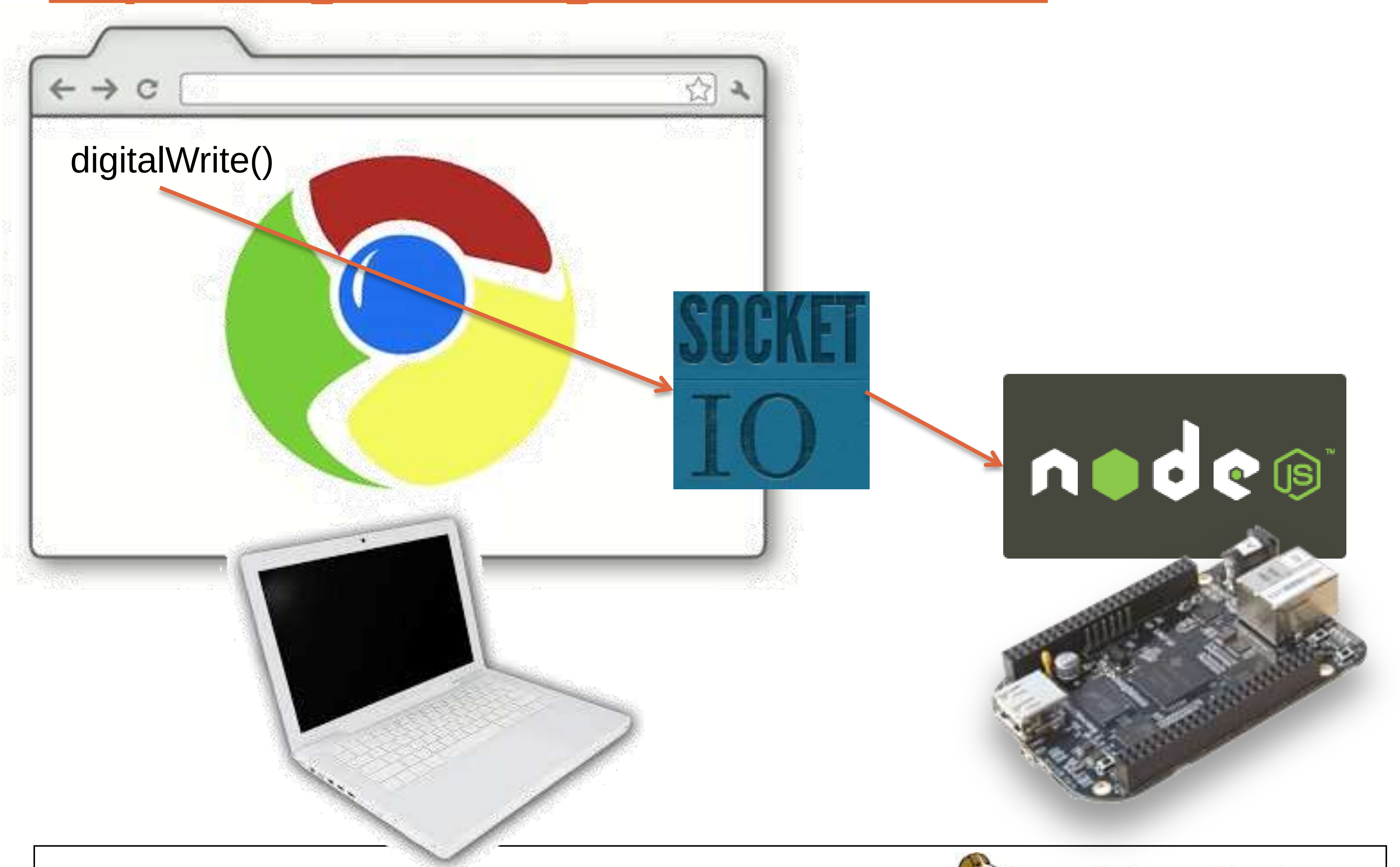
USB Client
Development interface and directly powers board from PC

4-GB on-board storage using eMMC
- Pre-loaded with Debian Linux Distribution
- 8-bit bus accelerates performance
- Frees the microSD slot to be used for additional storage for a less expensive solution than SD cards

Money saving extras:
- **Power over USB**
- **Included USB cable**
- **4-GB on-board storage**
- **Built-in PRU microcontrollers**

Simple browser-based interactions

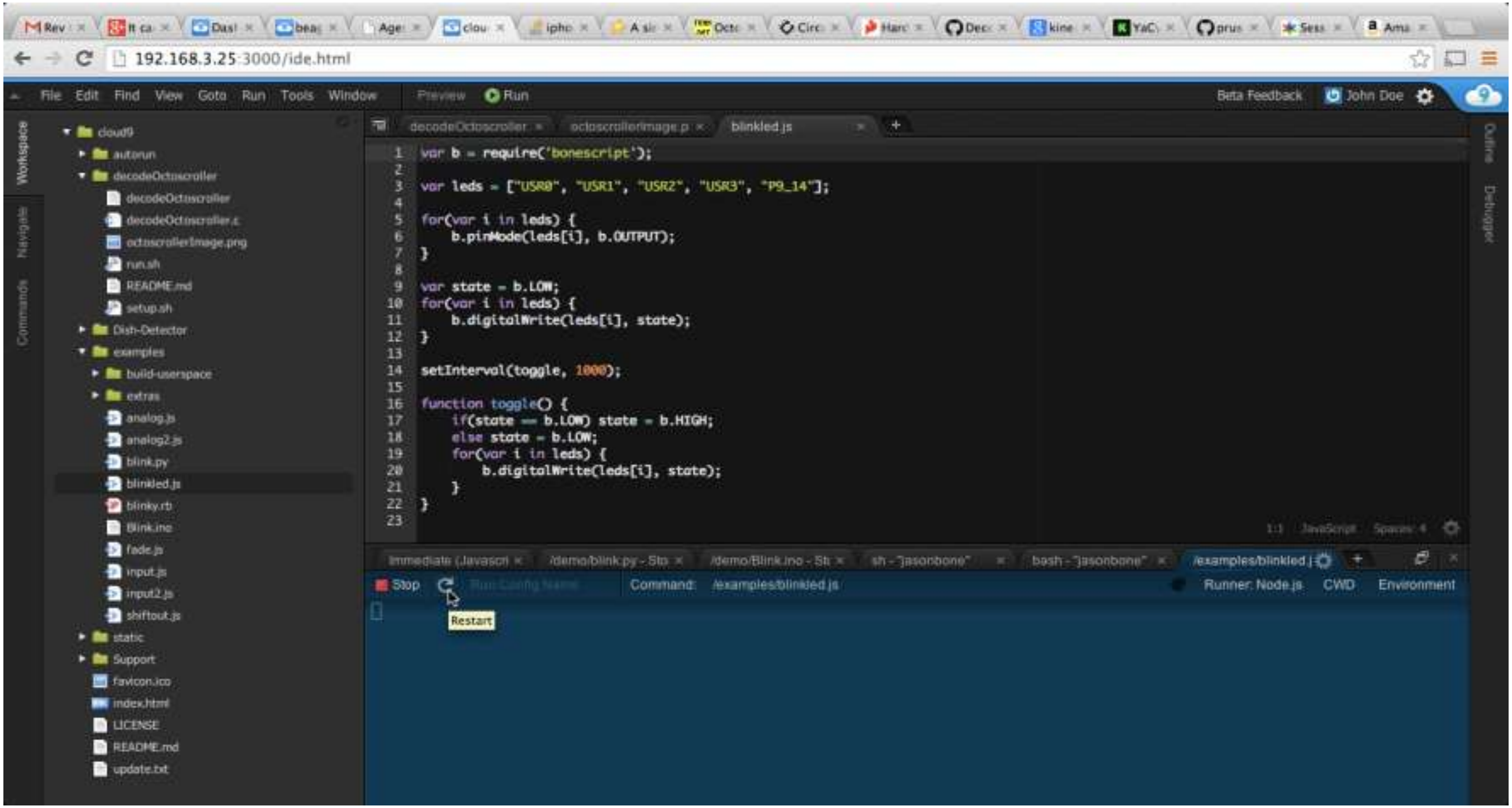
http://beagleboard.github.io/bone101



digitalWrite()

Cloud9 IDE hosted locally

Zero install and exposes command-line



```
var b = require('bonescript');

var leds = ["USR0", "USR1", "USR2", "USR3", "P9_14"];

for(var i in leds) {
    b.pinMode(leds[i], b.OUTPUT);
}

var state = b.LOW;
for(var i in leds) {
    b.digitalWrite(leds[i], state);
}

setInterval(toggle, 1000);

function toggle() {
    if(state == b.LOW) state = b.HIGH;
    else state = b.LOW;
    for(var i in leds) {
        b.digitalWrite(leds[i], state);
    }
}
```

Command: /examples/blinkled.js

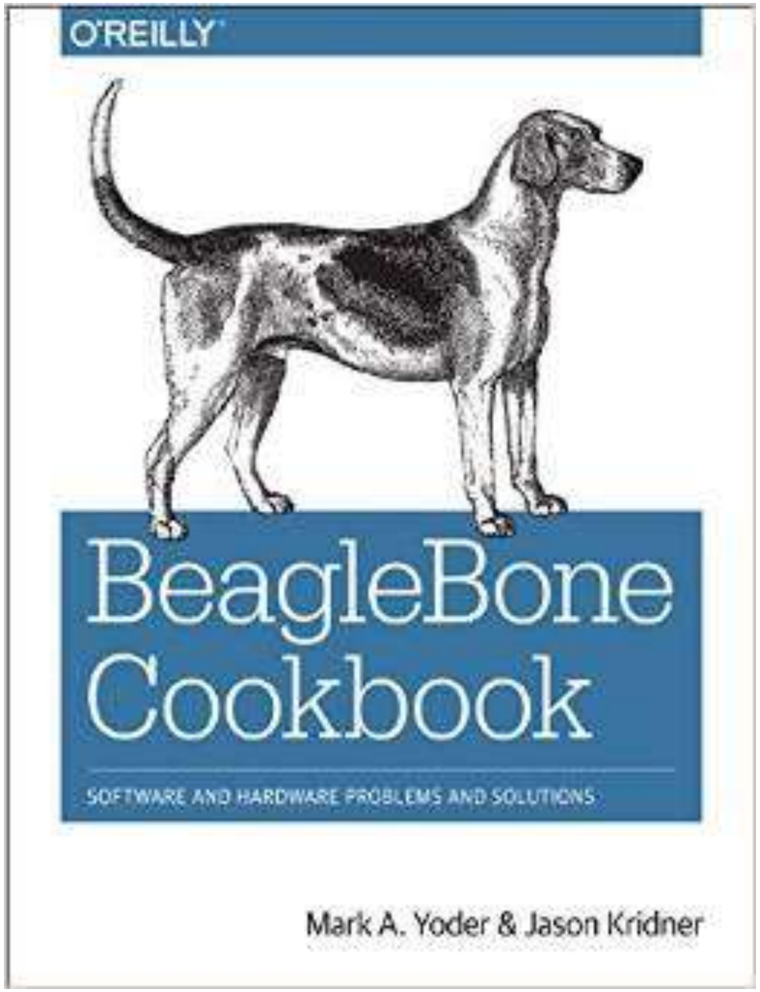
10,000s of developers building connected devices today

Medical
IoT
Gaming
BeagleBone Applications
Software Defined Radio
Robotics
Drones
Automotive
Citizen Scientist

- Medical analysis, assistance and information management
- Home information, automation and security systems
- Home and mobile entertainment and educational systems
- New types of communications systems
- Personal robotic devices for cleaning, upkeep and manufacturing
- Remote presence and monitoring
- Automotive information management and control systems
- Personal environmental exploration and monitoring

BeagleBone Cookbook

http://beagleboard.org/cookbook


O'REILLY
BeagleBone Cookbook
SOFTWARE AND HARDWARE PROBLEMS AND SOLUTIONS
Mark A. Yoder & Jason Kridner

- 99 recipes covering
  - Basics
  - Sensors
  - Displays and outputs
  - Motors
  - Internet of things
  - Kernel
  - Real-time I/O
  - Capes

Key take-aways from BeagleBone Cookbook

- Gain familiarity with electronic components you can integrate
  - Sensors, displays/lights, motors, networking and more
  - Quick success with known-good recipes
  - Go all the way to making your own PCB
- Build confidence working with a Linux system
  - Get the guided tour
  - Work with high-level languages like JavaScript and Python
  - Utilize Linux networking capabilities
  - Get introduced to working with real-time and kernel patching
  - Gain exposure to related industry tools

BeagleBoard.org to now

Fanless open computer
BeagleBoard

In 2008, BeagleBoard.org introduced the world to personally affordable open computing with the original BeagleBoard, spawning countless want-to-be designs inspired by open community collaboration

In 2010, BeagleBoard-xM provided extra MHz and memory, without extra cost

Now, BeagleBoard-X15, updates the full-featured BeagleBoard line for those wanting everything

Mint tin sized
BeagleBone

In 2011, BeagleBoard.org got down to the bare bones and a single cable development experience with the original BeagleBone at under $90

In 2013, BeagleBone Black again brought developers extra MHz and memory, restored the HDMI and all at a price below $50!

BeagleBoard.org Logo program

http://beagleboard.org/logo

- Third party product that licenses use of logo
- Verified to run BeagleBoard.org software image
- Open hardware design materials
- Targeting new applications

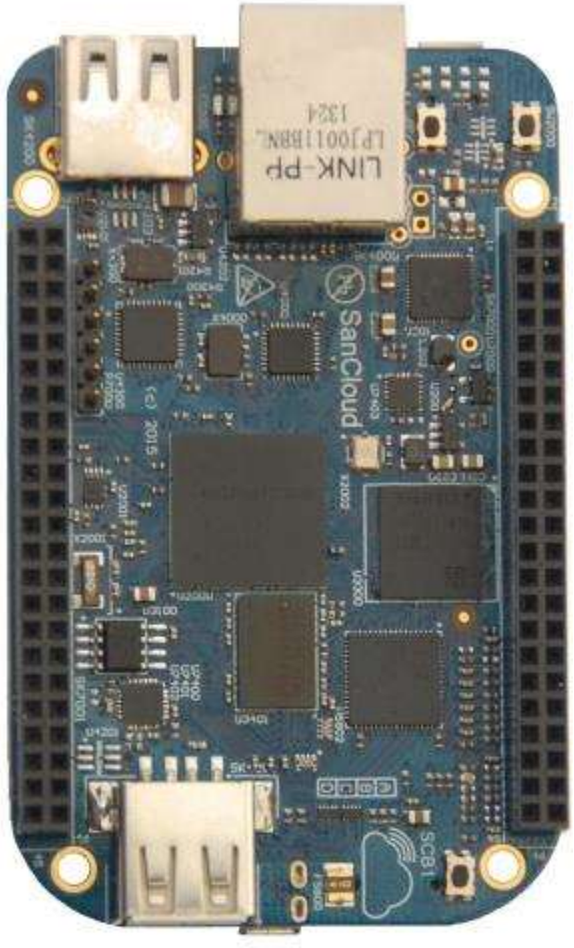
SeeedStudio BeagleBone Green

http://beagleboard.org/green

- Available now
- Compared to Black
  - Removes HDMI
  - Adds Grove connectors
- Affordable and great for quick-connect to I2C and UART sensors
- SCL = P9_19
  SDA = P9_20
- TXD = P9_21
  RXD = P9_22

SanCloud BeagleBone Enhanced

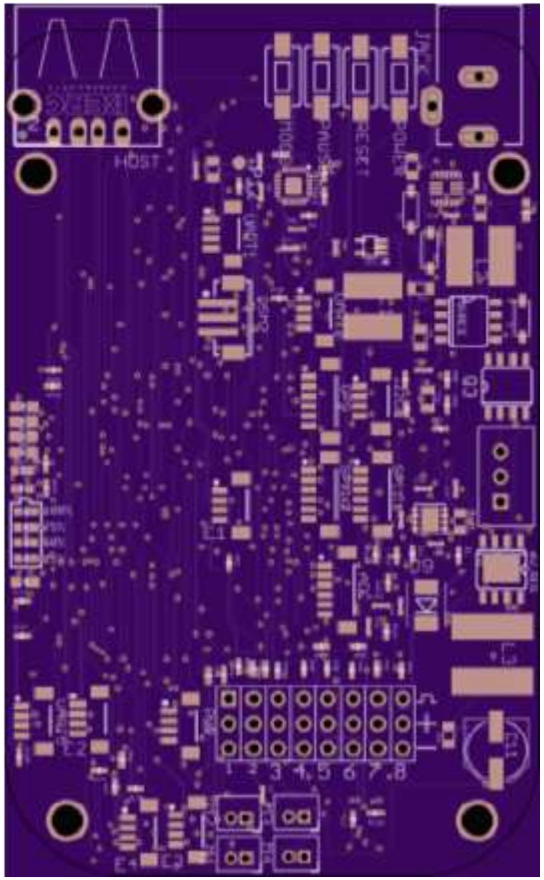
http://beagleboard.org/enhanced



- To be released soon
- Compared to Black
  - Adds RAM to 1GB
  - Ethernet to 1Gbit/s
  - Adds IMU, barometer, temperature sensors
  - Adds WiFi/Bluetooth via daughterboard
  - Adds 3 USB ports
- For those that want all the bells and whistles, but still BeagleBone compatibility

BeagleBoard.org BeagleBone Blue

http://beagleboard.org/blue



- To be released May 2016
- Compared to Black
  - Removes cape headers, HDMI and Ethernet
  - Adds wireless connectivity
  - Adds battery support
  - Adds DC and servo motor control
  - Adds IMU and barometer sensors
  - Adds CAN and several quick expansion connections
- Open robotics education solution

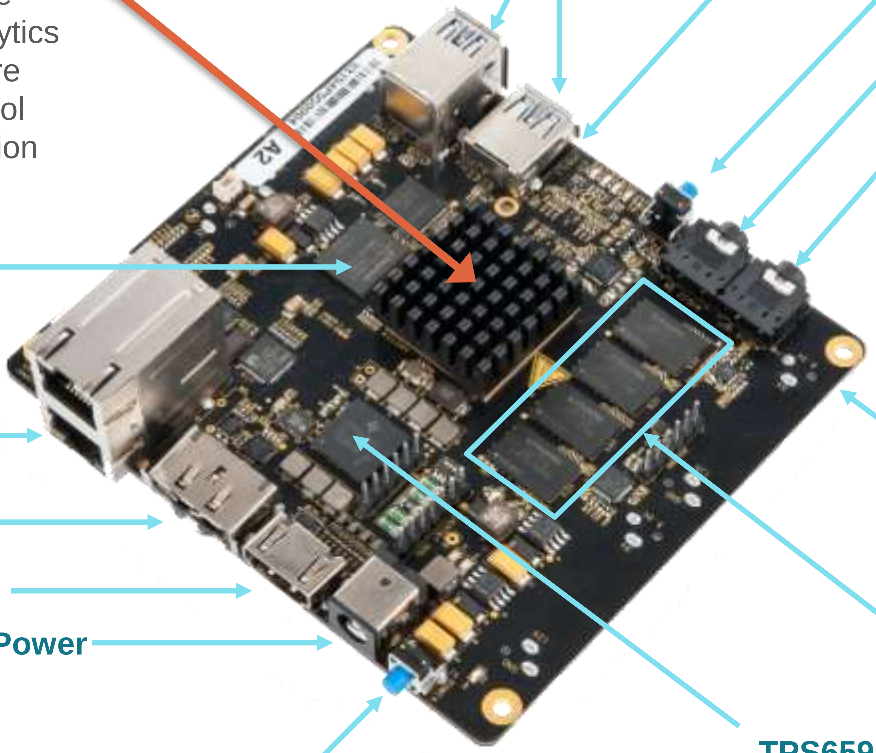
BeagleBoard.org BeagleBoard-X15

- To be released Feb 2016
- Compared to Black
  - Similar Debian Linux distribution
  - No cape interface
  - PRUs
  - Many more cores
  - Many more I/Os
  - Lots more connectivity
- The “what if” machine

BeagleBoard-X15

High performance Sitara™ AM5728 processor

- 1.5GHz ARM® Cortex®-A15 for processing and user interface
- 750-MHz C66x DSP for analytics
- Quad core PRU and dual core Cortex-M4 for real time control
- Video and Graphic acceleration



USB3.0 Host
3x HUB

Bottom side
µSD card slot
Micro USB 2.0 slave

Reset button

Audio in

Audio out

eMMC
4GB

2x Ethernet Ports
Gigabit

eSATA

HDMI

12V DC Power

Power button

Bottom side
Expansion connectors
20-pin ARM JTAG

DDR3L
2GB

TPS659037
Power
Management IC

Quick Compatibility Chart vs. Black

	Capes	HDMI	Flash	Special
BeagleBoard.org BeagleBone	Y	N	N	JTAG
BeagleBoard.org BeagleBone Black	Y	Y	Y	-
Arrow BeagleBone Black Industrial	Y	Y	Y	Industrial
Element14 BeagleBone Black Industrial	Y	Y	Y	Industrial
SeeedStudio BeagleBone Green	Y	N	Y	Grove
SanCloud BeagleBone Enhanced	Y	Y	Y	1GB, 1Gbit, wireless
BeagleBoard.org BeagleBone Blue	N	N	Y	Robotics
BeagleBoard.org BeagleBoard-X15	N	Y	N	Big jump in CPUs and I/O

Audio recipes

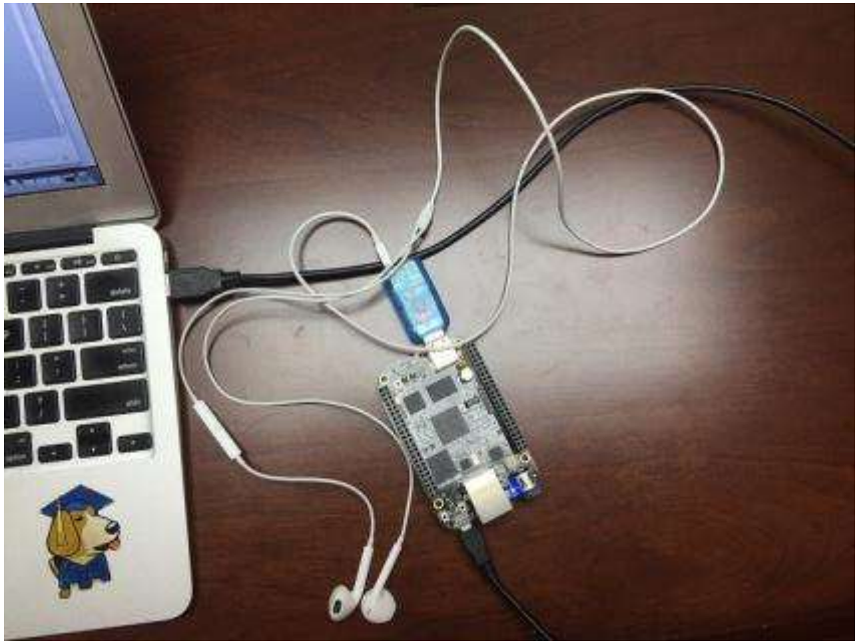
Possible audio solutions

- Built-in HDMI audio
  - connect to TV or HDMI-audio adapter
- Audio cape
  - SPI, I^2S and I^2C available
- USB Bluetooth dongles
  - BlueZ → https://wiki.debian.org/Bluetooth/Alsa
- USB audio adapter ← this will be our approach
  - Easy to find adapters on Amazon, etc.
    - http://www.amazon.com/s/ref=nb_sb_noss_2?url=search-alias%3Daps&field-keywords=linux+usb+audio

Step #0 – Prerequisites

- Connect to the board per recipe 1.2
  - http://beagleboard.org/getting-started
- Verify the software image per recipe 1.3 and potentially updating per recipe 1.9
  - http://beagleboard.org/latest-images

Step #1 – Boot with USB audio adapter



- Power up with USB audio adapter inserted
  - Some kernels don't like USB hotplugging
  - USB power typically sufficient, but add a power adapter if you see issues
- Verify driver loaded
  - lsusb
  - dmesg

Step #2 – Test playback

- Discover devices
  - man aplay
  - aplay -l
  - aplay -L
- Playback samples
  - aplay -D "default:CARD=Device" /usr/share/sounds/alsa/Front_Center.wav

Step #3 – Test record

- Use the mixer to set the input gain
  - alsamixer
- Record a sample
  - man arecord
  - arecord -f dat -D "default:CARD=Device" test.wav

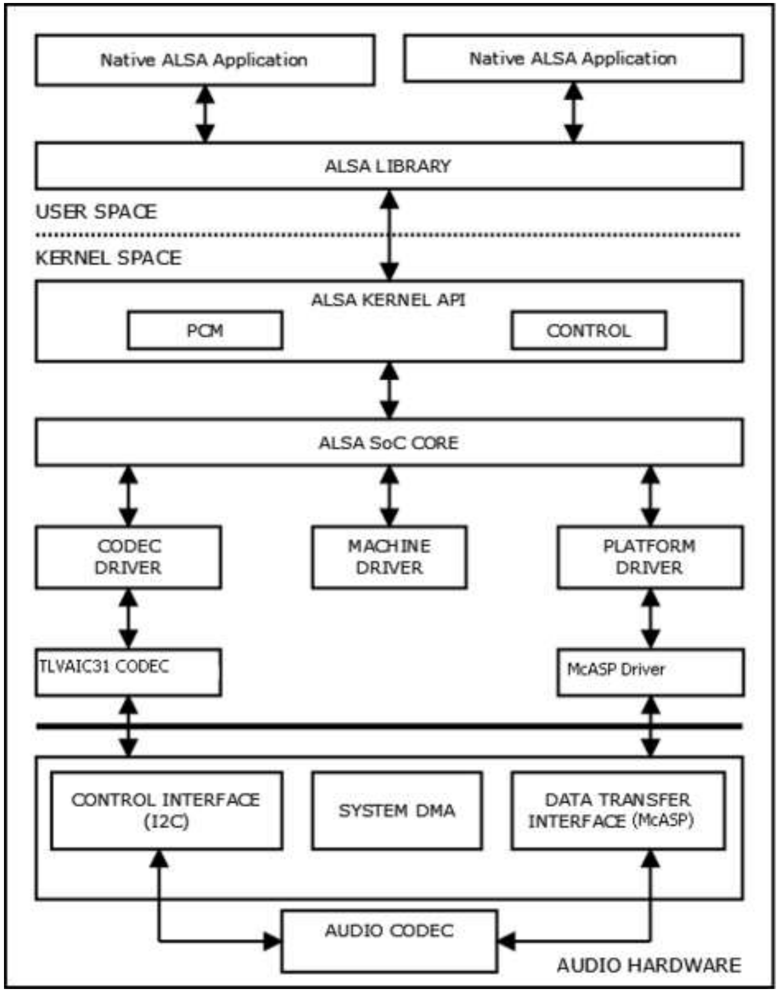
Step #4 – Set default audio

- Write to ~/.asoundrc
- Enables you to use applications without specifying the card each time
- Example
  requires ‘apt-get install flite’
  - flite –t "Hello!"

```
pcm.!default {
    type plug
    slave {
        pcm "hw:1,0"
    }
}
ctl.!default {
    type hw
    card 1
}
```

More about ALSA

Advanced Linux Sound Architecture - http://alsa-project.org



- Includes user space library for application programming
- Supports many devices
- ALSA SoC supports adding codecs to embedded boards

More

- Nice set of tutorials from 13-year old Alek Mabry
  - http://einsteiniumstudios.com/speak.html
- Shortcuts to updates and examples from the book
  - http://beagleboard.org/cookbook

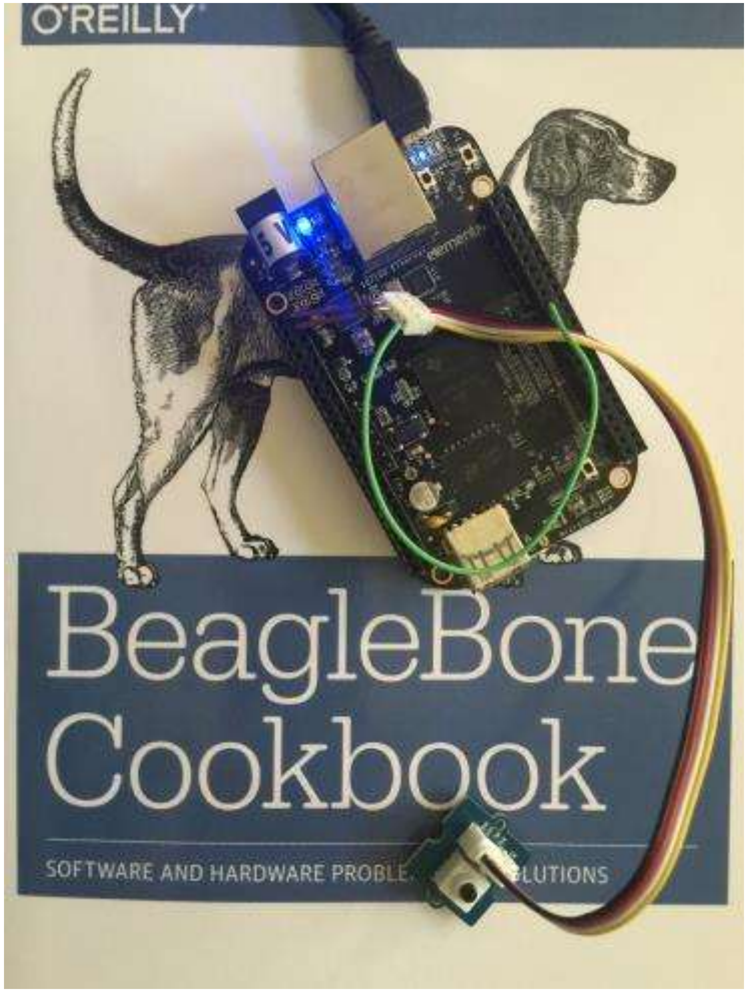
Web interaction recipes

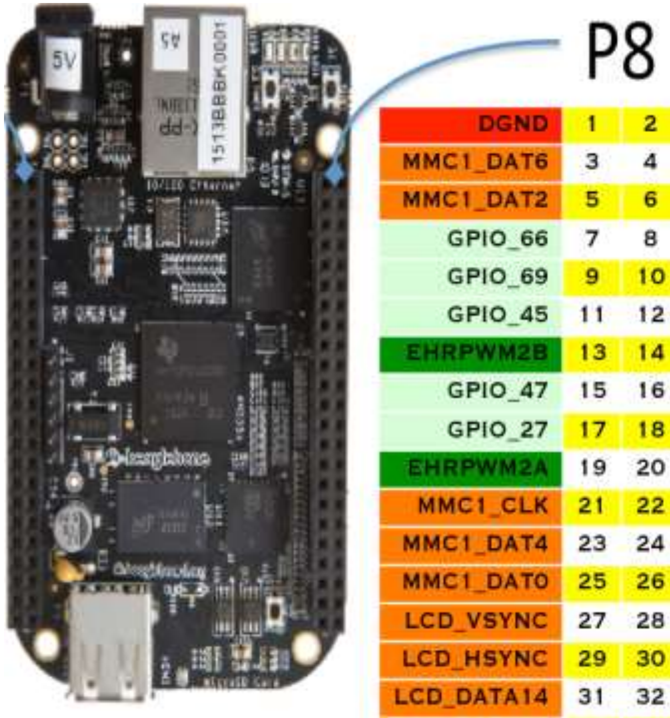
Prerequisites

- Connect to the board per recipe 1.2
  - http://beagleboard.org/getting-started
- Verify the software image per recipe 1.3 and potentially updating per recipe 1.9
  - http://beagleboard.org/latest-images

Connect a button to GPIO P8_19

http://beagleboard.org/Support/bone101/#headers





DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DAT0	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

LEGEND
- Power/Ground/Reset
- Available Digital
- Available PWM
- Shared I2C Bus
- Reconfigurable Digital
- Analog Inputs (1.8V)

Recipe 6.6: Continuously Displaying the GPIO Value

https://github.com/BeagleBoneCookbook/firstEdition/blob/master/06iot/jQueryDemo.html

```
<html>
<head>
  <title>BoneScript jQuery Demo</title>
  <script src="/static/jquery.js"></script>
  <script src="/static/bonescript.js"></script>
  <script src="jQueryDemo.js"></script>
</head>

<body>
<h1>BoneScript jQuery Demo</h1>
<p>buttonStatus = <span id="buttonStatus">-</span>
</p>
</body>
</html>
```

https://github.com/BeagleBoneCookbook/firstEdition/blob/master/06iot/jQueryDemo.js

```
setTargetAddress('192.168.7.2',
    {initialized: run}
);
function run() {
    var b = require('bonescript');
    b.pinMode('P8_19', b.INPUT);
    getButtonStatus();
    function getButtonStatus() {
        b.digitalRead('P8_19', onButtonRead);
    }
    function onButtonRead(x) {
        $('#buttonStatus').html(x.value);
        setTimeout(getButtonStatus, 20);
    }
}
```

Stepping back to recipe 6.3

Interacting with the Bone via a Web Browser

https://github.com/BeagleBoneCookbook/firstEdition/blob/master/06iot/server.js

```
var port=9090, h=require('http'),
    u=require('url'), f=require('fs');
var s=h.createServer(servePage);
s.listen(port);

function servePage(req, res) {
 var p = u.parse(req.url).pathname;
 f.readFile(__dirname+p,
  function (err, data) {
   if (err) return;
   res.write(data, 'utf8');
   res.end();
  }
 );
}
```

- BeagleBone Black ships with Debian and Node.JS
- Using Node.JS is easy to serve up a simple web page
- Run with:
  node server.js
- Browse to port 9090 and a local file

Recipe 6.4 adds hardware interaction

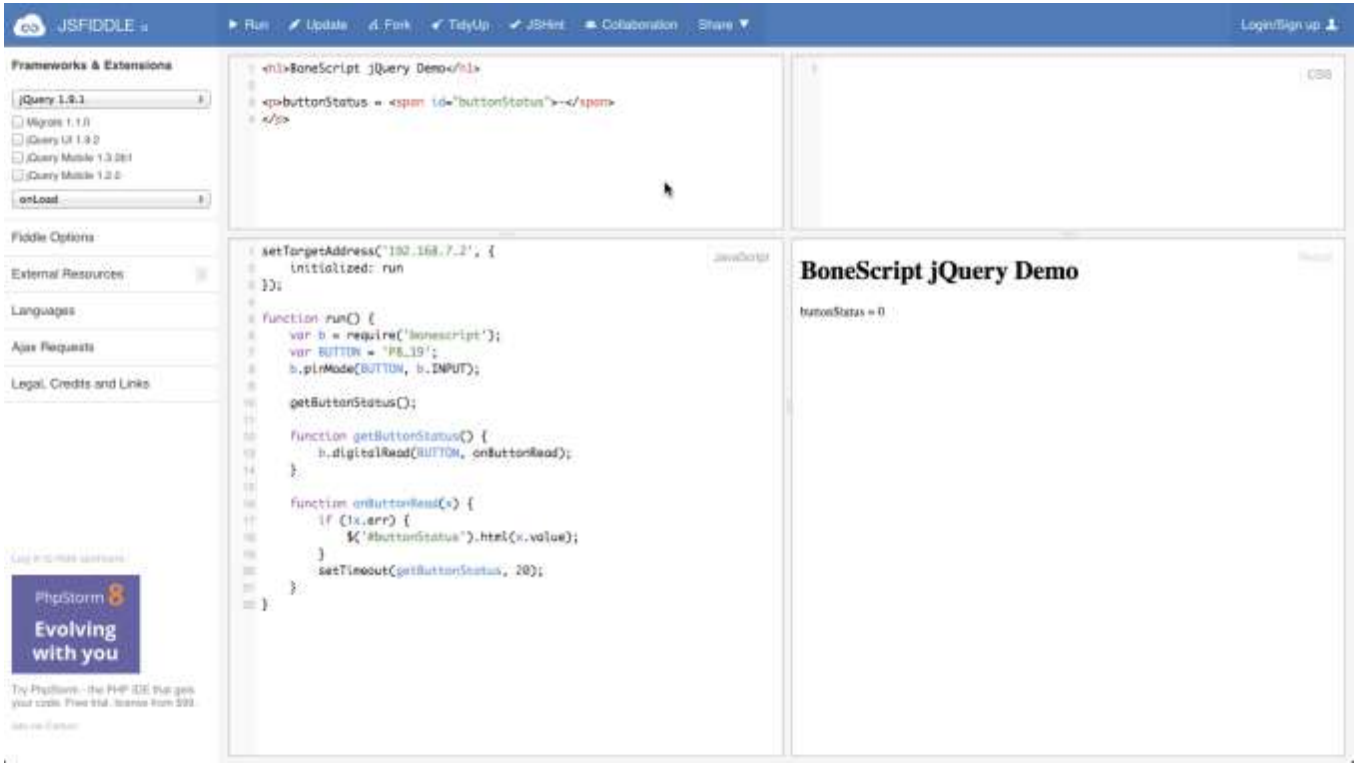
https://github.com/BeagleBoneCookbook/firstEdition/blob/master/06iot/GPIOserver.js

```
var h=require('http'),f=require('fs'),
    b=require('bonescript'),
    g='P8_19', p=9090;

var htmlStart = "<!DOCTYPE html>\
<html><body><h1>" + g + "</h1>data = ";
var htmlEnd = "</body></html>";
var s = h.createServer(servePage);

b.pinMode(g, b.INPUT);
s.listen(p);

function servePage(req, res) {
    var data = b.digitalRead(g);
    res.write(htmlStart + data + htmlEnd, 'utf8');
    res.end();
}
```

- Builds on simple Node.JS web server
- BoneScript library utilized on server
- Content served using variables, not files
- Full example uses URL path
  - distinguish content
- Refresh manually

Recipe 6.5 introduces jQuery

http://jsfiddle.net/n5j3p32o/1/



jQuery 1.9.1

onLoad

```
<h1>BoneScript jQuery Demo</h1>

<p>buttonStatus = <span id="buttonStatus">-</span>
</p>
```

```
setTargetAddress('192.168.7.2', {
    initialized: run
});

function run() {
    var b = require('bonescript');
    var BUTTON = 'P8_19';
    b.pinMode(BUTTON, b.INPUT);

    getButtonStatus();

    function getButtonStatus() {
        b.digitalRead(BUTTON, onButtonRead);
    }

    function onButtonRead(x) {
        if (!x.err) {
            $('#buttonStatus').html(x.value);
        }
        setTimeout(getButtonStatus, 20);
    }
}
```

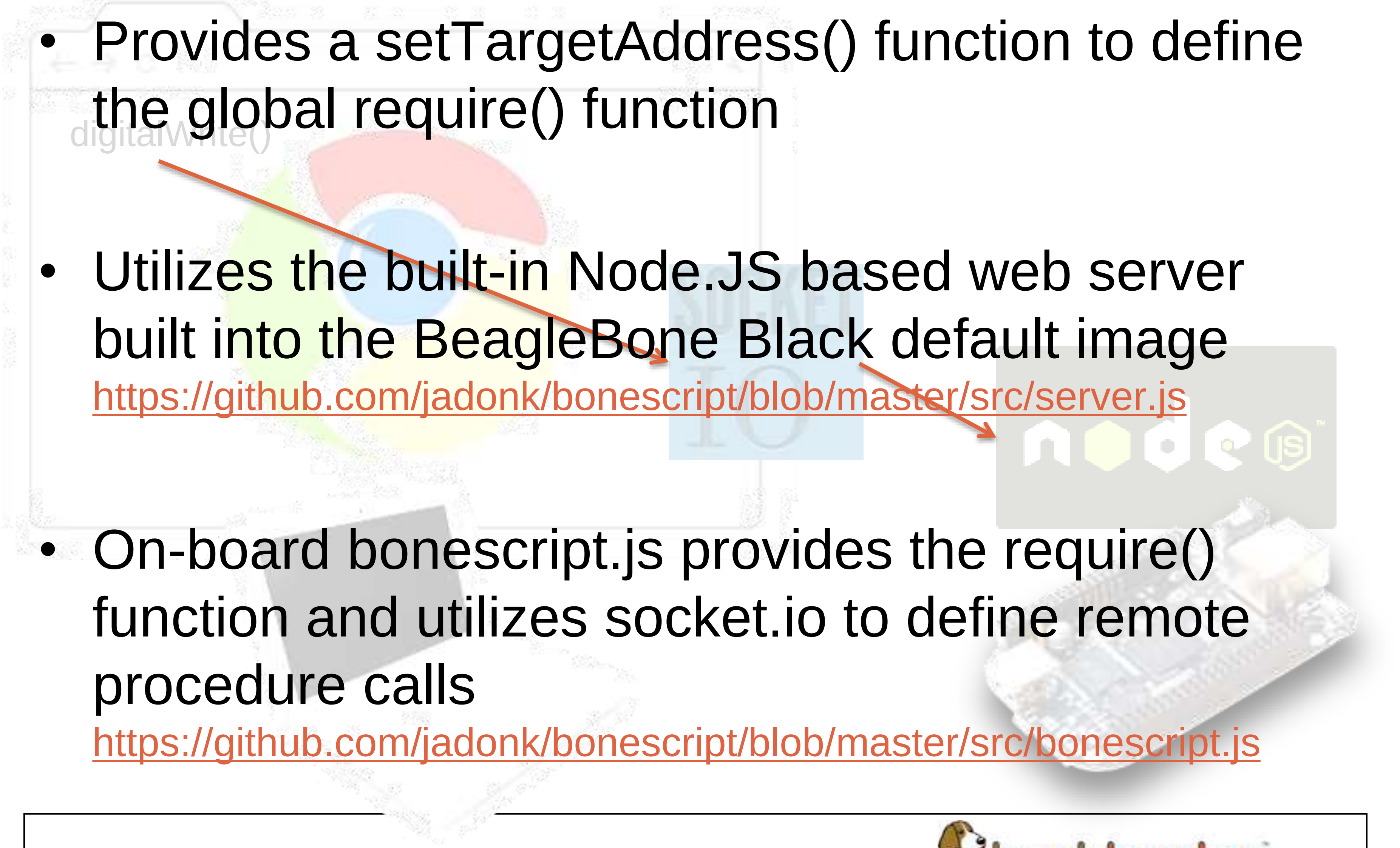
BoneScript jQuery Demo

buttonStatus = 0

- Great tool to make content dynamic
- jsfiddle.net provides a playground for learning
- Learn more about the API at jquery.com

How BoneScript works in the browser

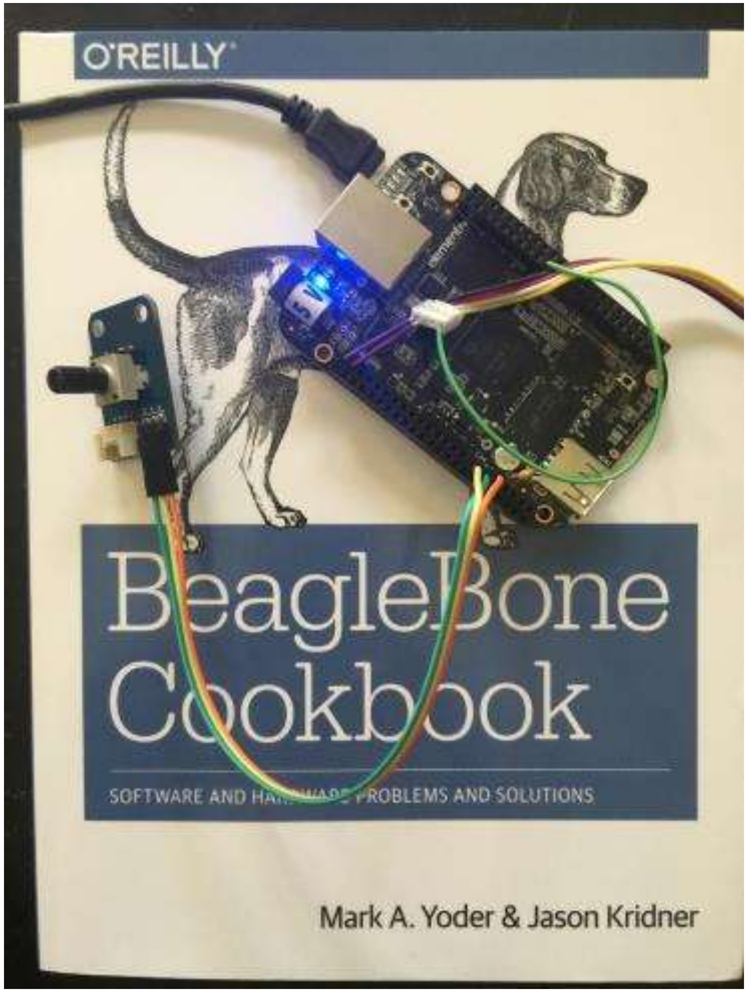
http://beagleboard.org/static/bonescript.js

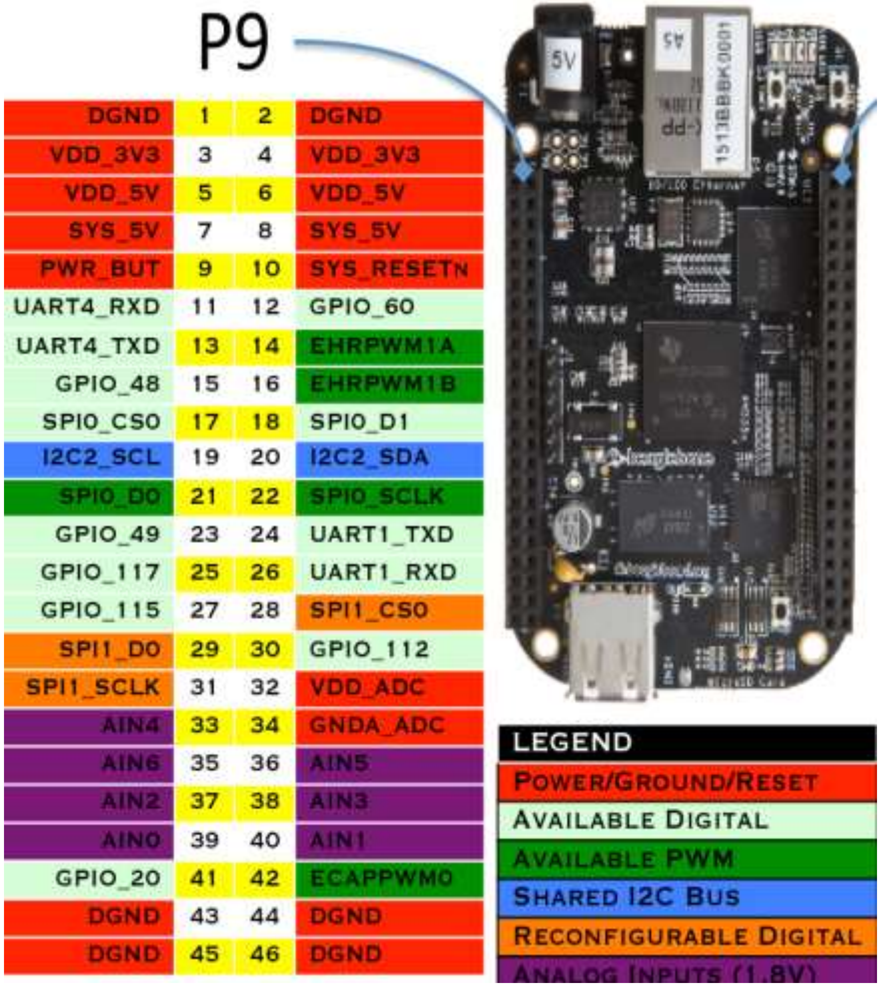


- Provides a setTargetAddress() function to define the global require() function
- Utilizes the built-in Node.JS based web server built into the BeagleBone Black default image
  https://github.com/jadonk/bonescript/blob/master/src/server.js
- On-board bonescript.js provides the require() function and utilizes socket.io to define remote procedure calls
  https://github.com/jadonk/bonescript/blob/master/src/bonescript.js

Connect a potetiometer to ADC P9_36

http://beagleboard.org/Support/bone101/#headers


O'REILLY
BeagleBone Cookbook
SOFTWARE AND HA PROBLEMS AND SOLUTIONS
Mark A. Yoder & Jason Kridner



DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPI0_CS0	17	18	SPI0_D1
I2C2_SCL	19	20	I2C2_SDA
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SPI1_CS0
SPI1_D0	29	30	GPIO_112
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPPWM0
DGND	43	44	DGND
DGND	45	46	DGND

LEGEND
- POWER/GROUND/RESET
- AVAILABLE DIGITAL
- AVAILABLE PWM
- SHARED I2C BUS
- RECONFIGURABLE DIGITAL
- ANALOG INPUTS (1.8V)

Recipe 6.7: Plotting Data

- See demo code at
  - https://github.com/BeagleBoneCookbook/firstEdition/blob/master/06iot/flotDemo.js
  - https://github.com/BeagleBoneCookbook/firstEdition/blob/master/06iot/flotDemo.html
- This is just the beginning
  - Lots of different types of hardware interactions
  - Lots of different visualizations possible in the browser

More

- JavaScript tricks
  - http://beagleboard.org/project/javascript-tricks/
- Shortcuts to updates and examples from the book
  - http://beagleboard.org/cookbook

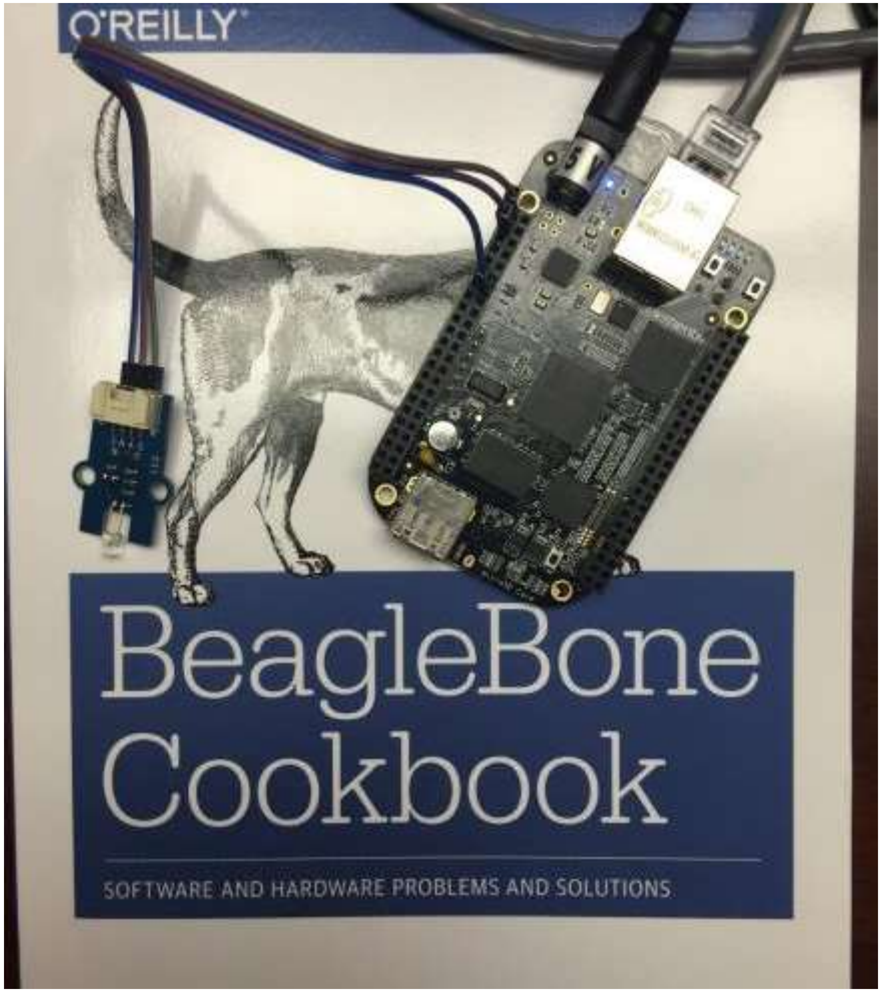
Node-RED

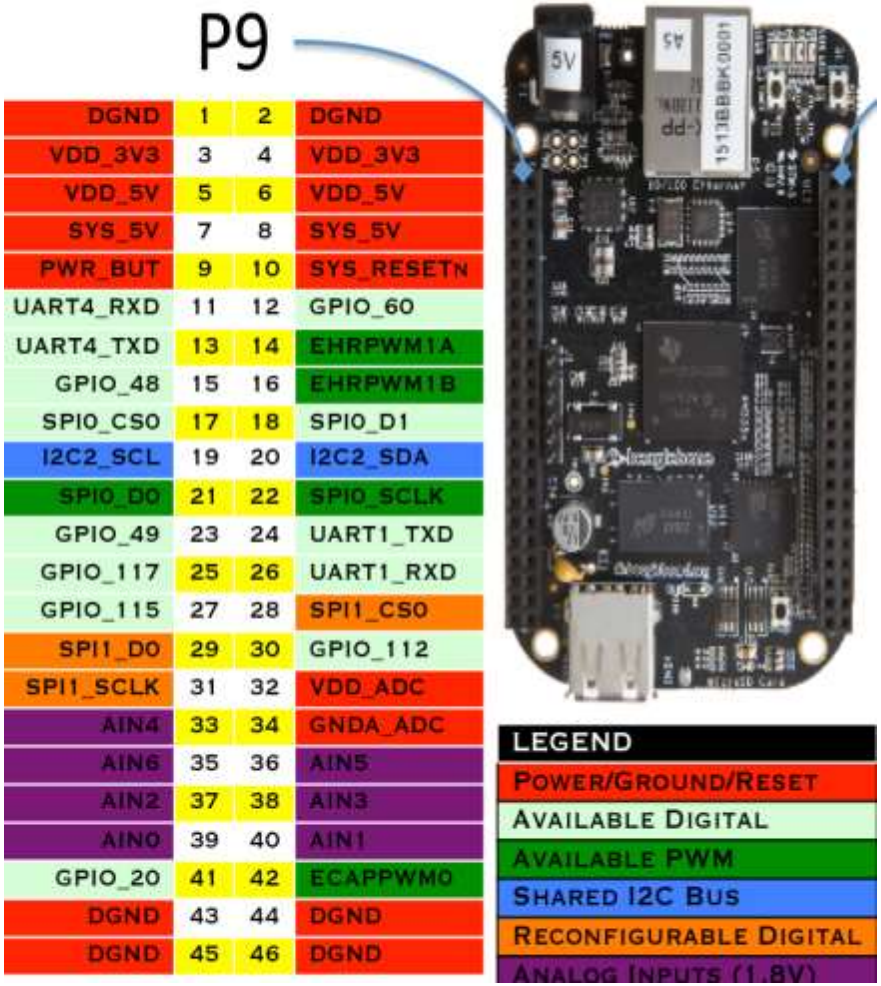
Prerequisites

- Connect to the board per recipe 1.2
  - http://beagleboard.org/getting-started
- Verify the software image per recipe 1.3 and potentially updating per recipe 1.9
  - http://beagleboard.org/latest-images
- Establish an Ethernet-based Internet connection per recipe 5.11 or a WiFi-based Internet connection per recipe 5.12
  - WiFi adapters: http://bit.ly/1EbEwUo

Connect an LED to GPIO P9_14

http://beagleboard.org/Support/bone101/#headers





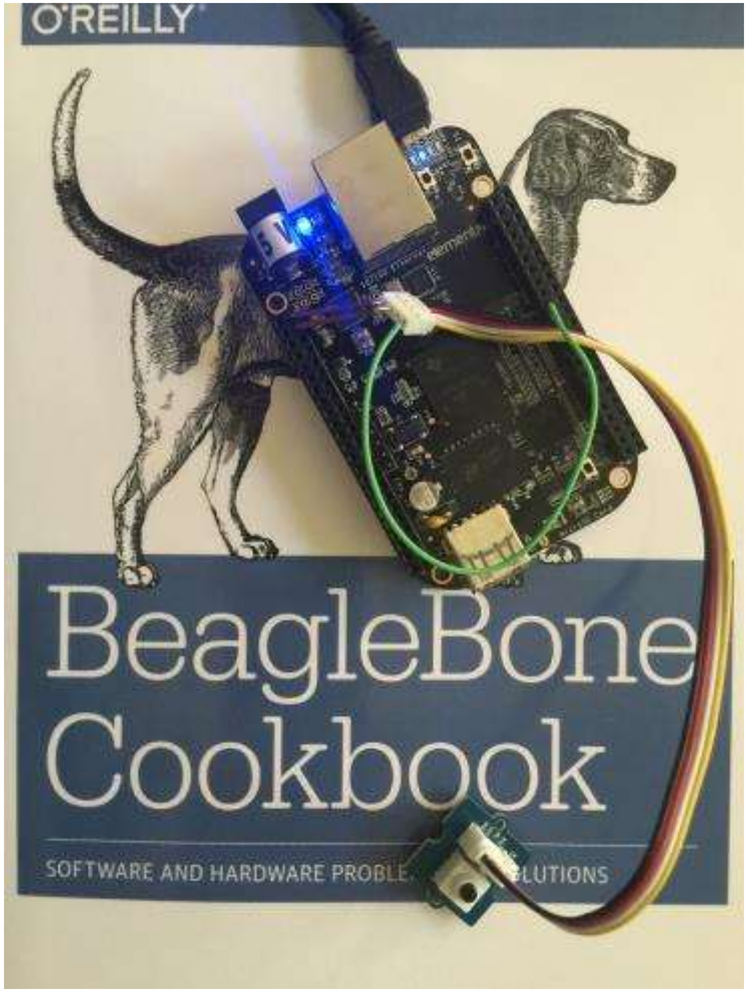
DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPI0_CS0	17	18	SPI0_D1
I2C2_SCL	19	20	I2C2_SDA
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SPI1_CS0
SPI1_D0	29	30	GPIO_112
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPPWM0
DGND	43	44	DGND
DGND	45	46	DGND

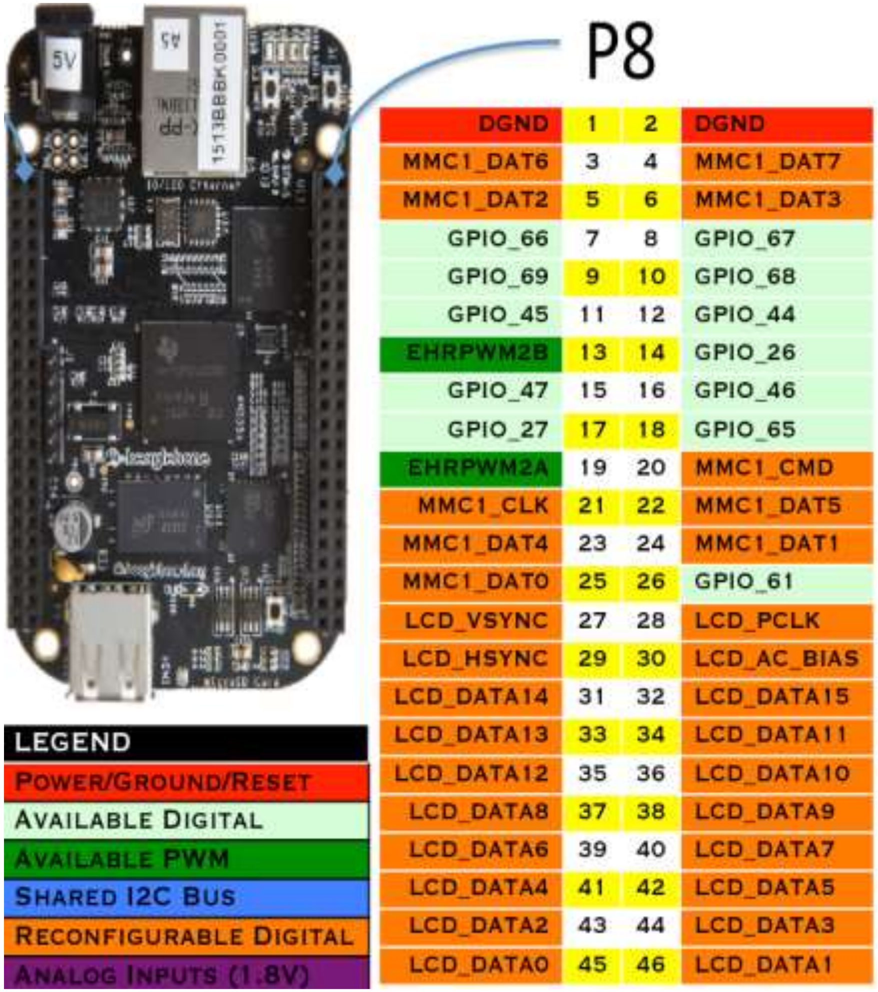
LEGEND

- Power/Ground/Reset
- Available Digital
- Available PWM
- Shared I2C Bus
- Reconfigurable Digital
- Analog Inputs (1.8V)

Connect a button to GPIO P8_19

http://beagleboard.org/Support/bone101/#headers



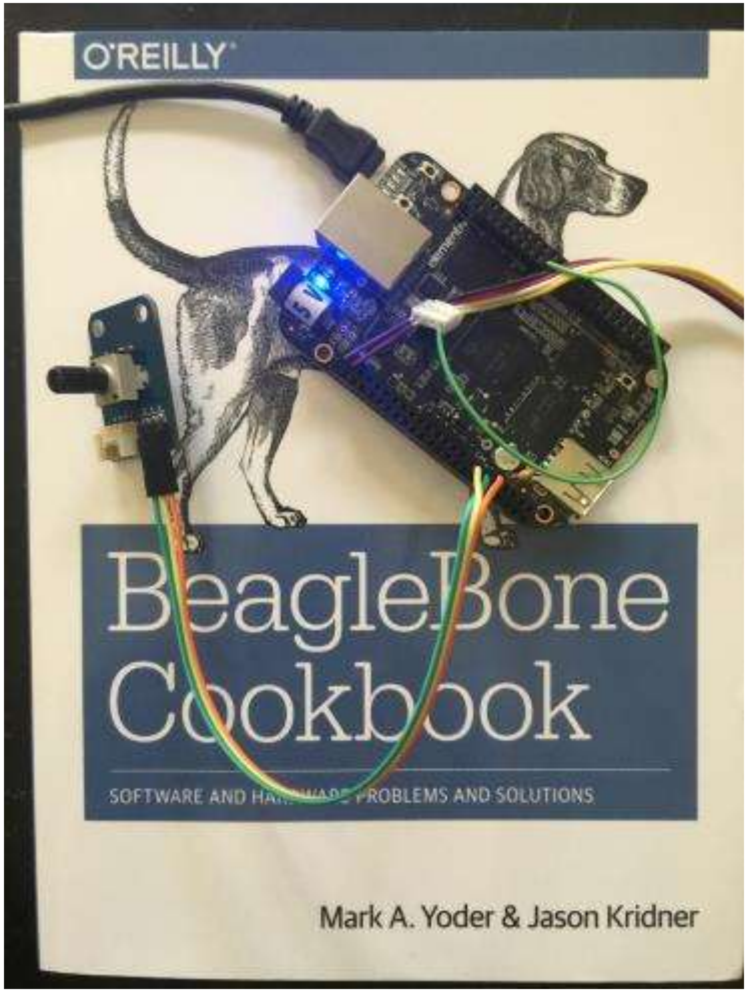


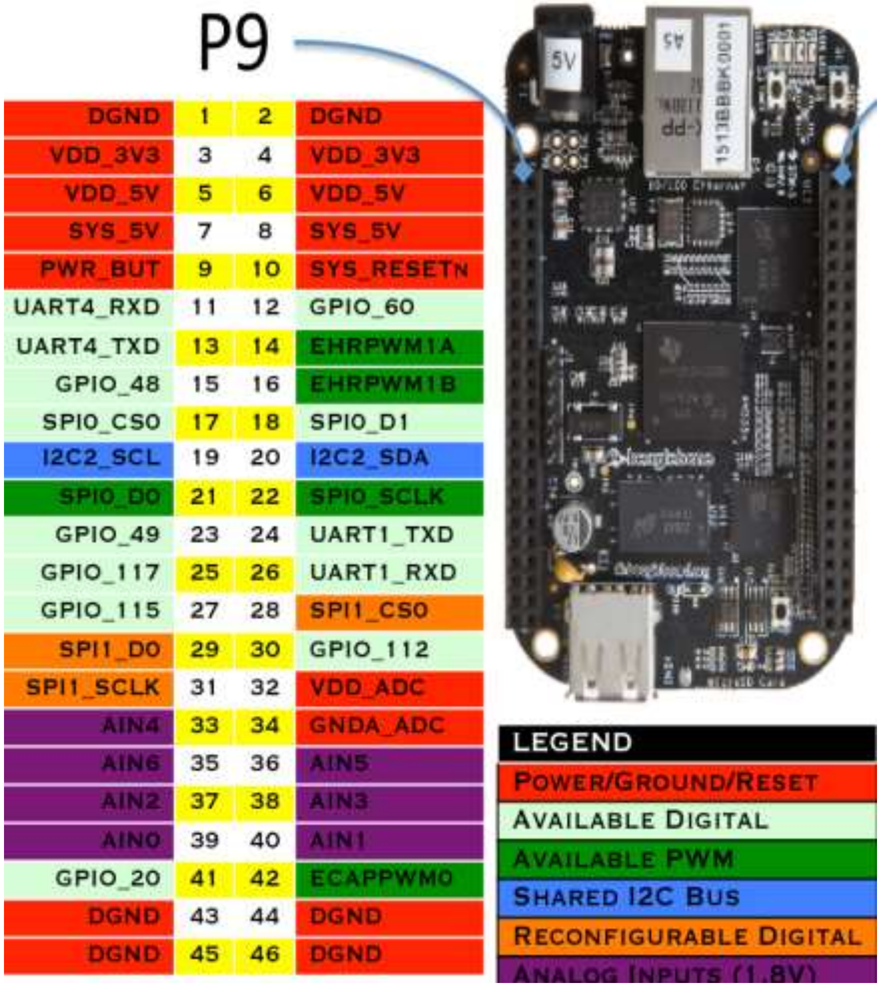
DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DAT0	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

LEGEND
Power/Ground/Reset
Available Digital
Available PWM
Shared I2C Bus
Reconfigurable Digital
Analog Inputs (1.8V)

Connect a potentiometer to ADC P9_36

http://beagleboard.org/Support/bone101/#headers


O'REILLY
BeagleBone Cookbook
SOFTWARE AND HARDWARE PROBLEMS AND SOLUTIONS
Mark A. Yoder & Jason Kridner



P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPI0_CS0	17	18	SPI0_D1
I2C2_SCL	19	20	I2C2_SDA
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SPI1_CS0
SPI1_D0	29	30	GPIO_112
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPPWM0
DGND	43	44	DGND
DGND	45	46	DGND

LEGEND

- Power/Ground/Reset
- Available Digital
- Available PWM
- Shared I2C Bus
- Reconfigurable Digital
- Analog Inputs (1.8V)

Install and start Node-RED

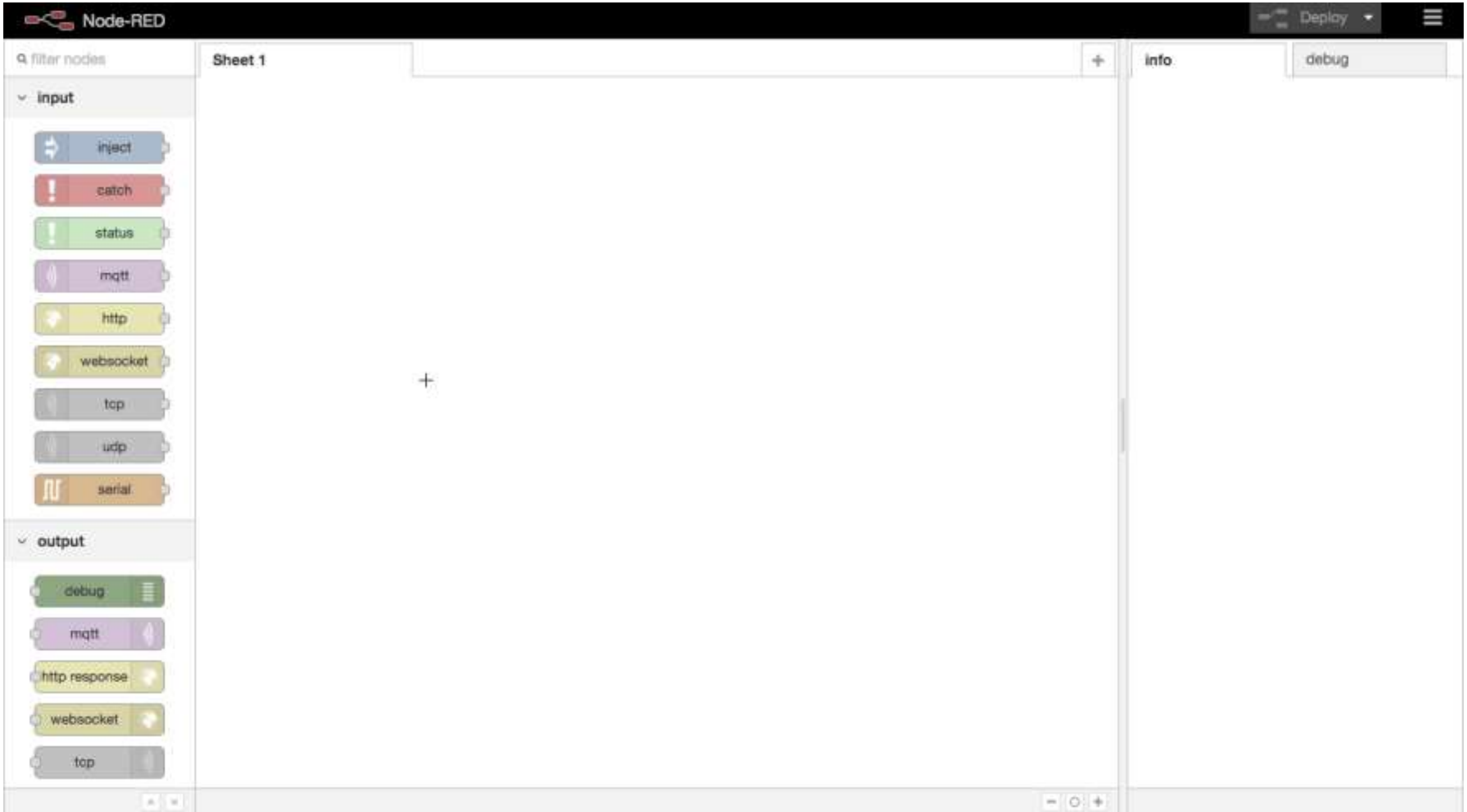
- Installation is simple, but requires a network connection
- Installing the developer version has changed slightly with a build step, but it is easier just to install using ‘npm’
- Requires a live Internet connection
- Steps to install and run from root prompt

```
bone# npm install --unsafe-perm -g node-red@0.12.1
bone# node-red
```

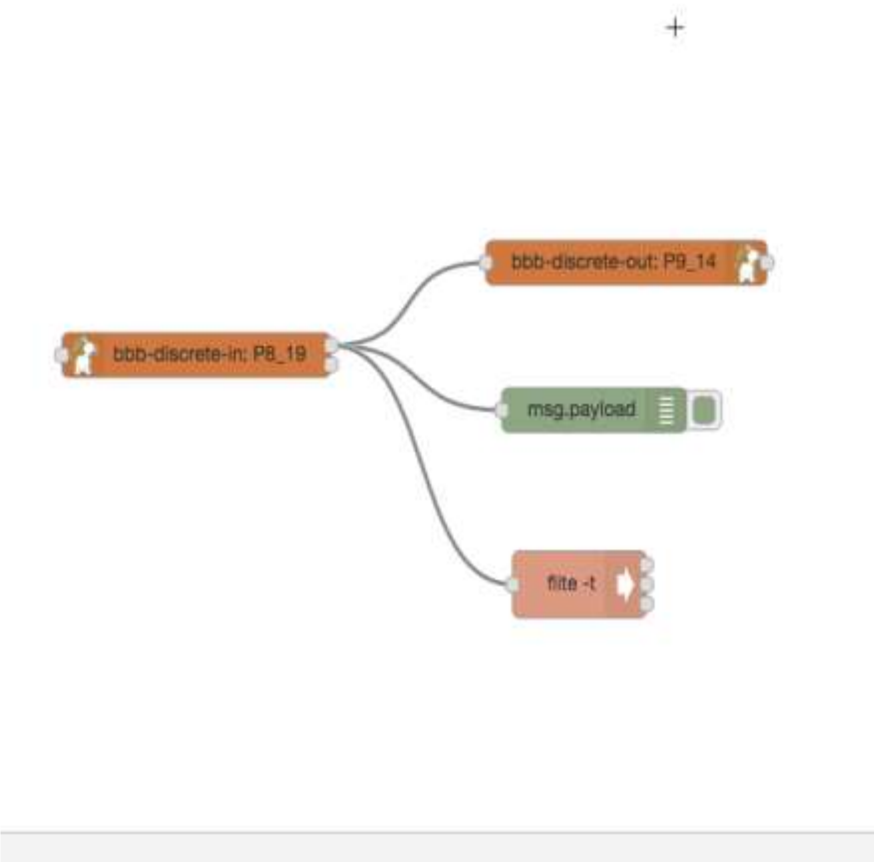
- Add BeagleBone specific nodes

```
bone# cd ~/.node-red
bone# npm install node-red-node-beaglebone
```

Node-RED on port 1880

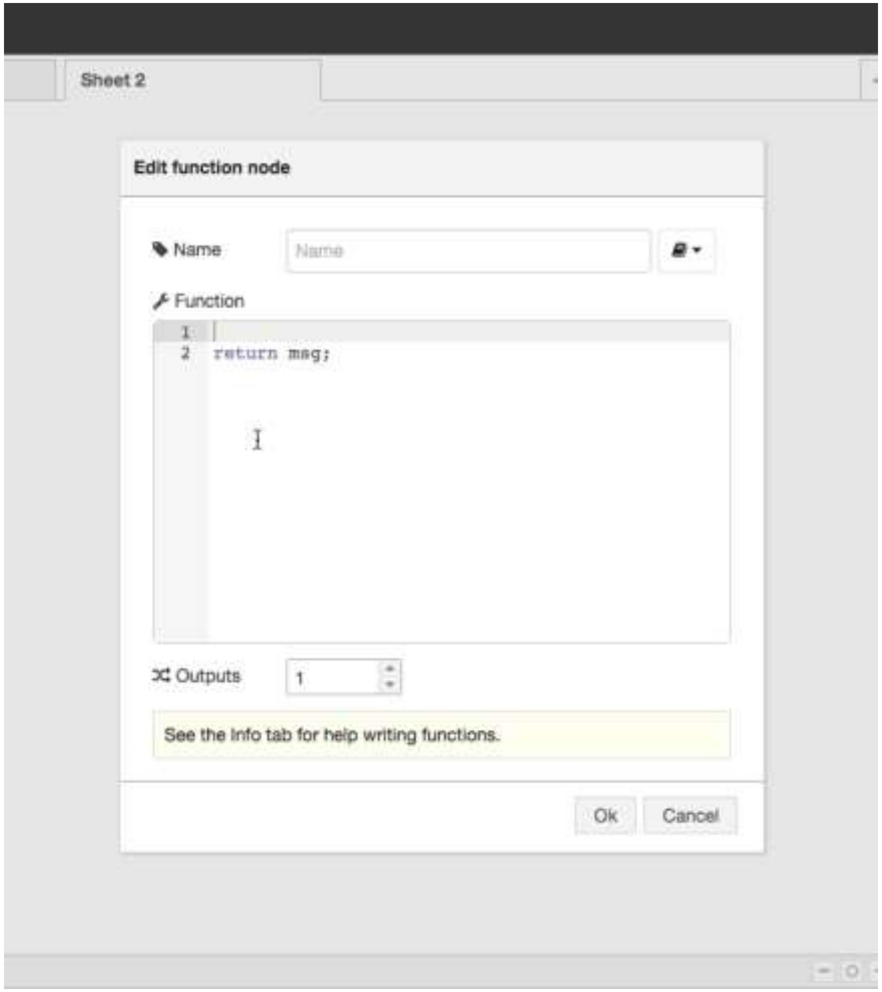


Creating flows



- Drag nodes from the left side into the sheet to add them
- Configure the nodes
- Use debug nodes to test the outputs
- Be sure to click ‘Deploy’ to start the app

Functions add fun



Name: Name

Function:

```
return msg;
```

Outputs: 1

See the Info tab for help writing functions.

- ‘msg’ is a JavaScript object
- ‘msg’ contains the element ‘payload’, which is what you most likely want to manipulate

More

- Learn more about Node-RED
  - http://nodered.org
- Shortcuts to updates and examples from the book
  - http://beagleboard.org/cookbook

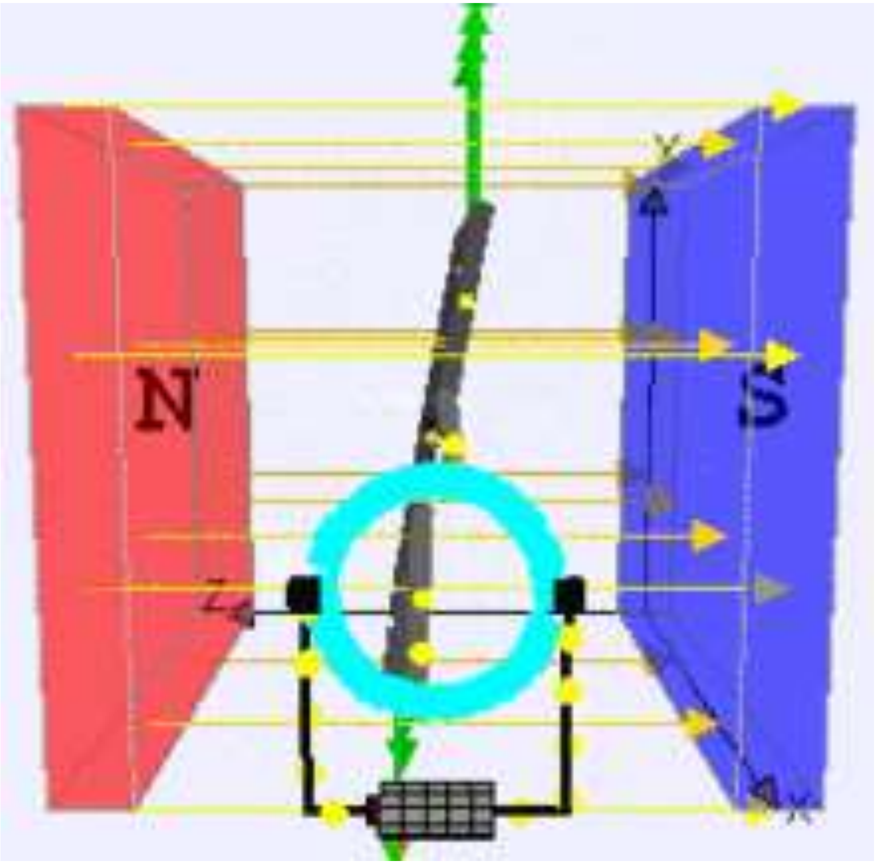
DC motor control recipes

Prerequisites

- Connect to the board per recipe 1.2
  - http://beagleboard.org/getting-started
- Verify the software image per recipe 1.3 and potentially updating per recipe 1.9
  - http://beagleboard.org/latest-images
- Components
  - BeagleBone Black
  - L293D H-Bridge IC
  - 5V DC motor
    - For other voltages, verify H-bridge compatibility
  - Breadboard and jumper wire
    - Alternatively, I've had a PCB fabricated

Direct Current (DC) Motor

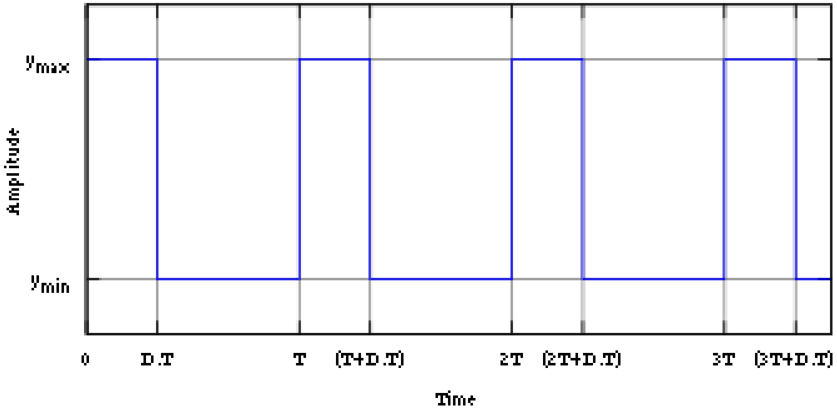
https://en.wikipedia.org/wiki/DC_motor



- DC voltage causes motor to turn
- Brush contact resets drive after partial revolution
- Drive strength is proportional to input voltage
- There’s a maximum input voltage
- Reversing voltage reverses direction
- BeagleBone Black doesn’t supply enough current on its I/O pins

Pulse-Width Modulation (PWM)

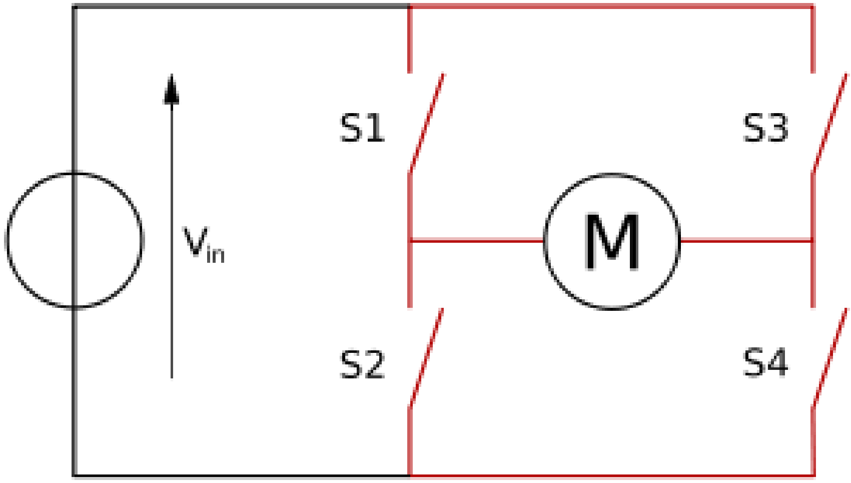
https://en.wikipedia.org/wiki/Pulse-width_modulation



- Enables approximating a voltage by turning on and off quickly
- BeagleBone Black has 8 hardware PWMs
- PRU can produce another 25 more with appropriate firmware

H-Bridge

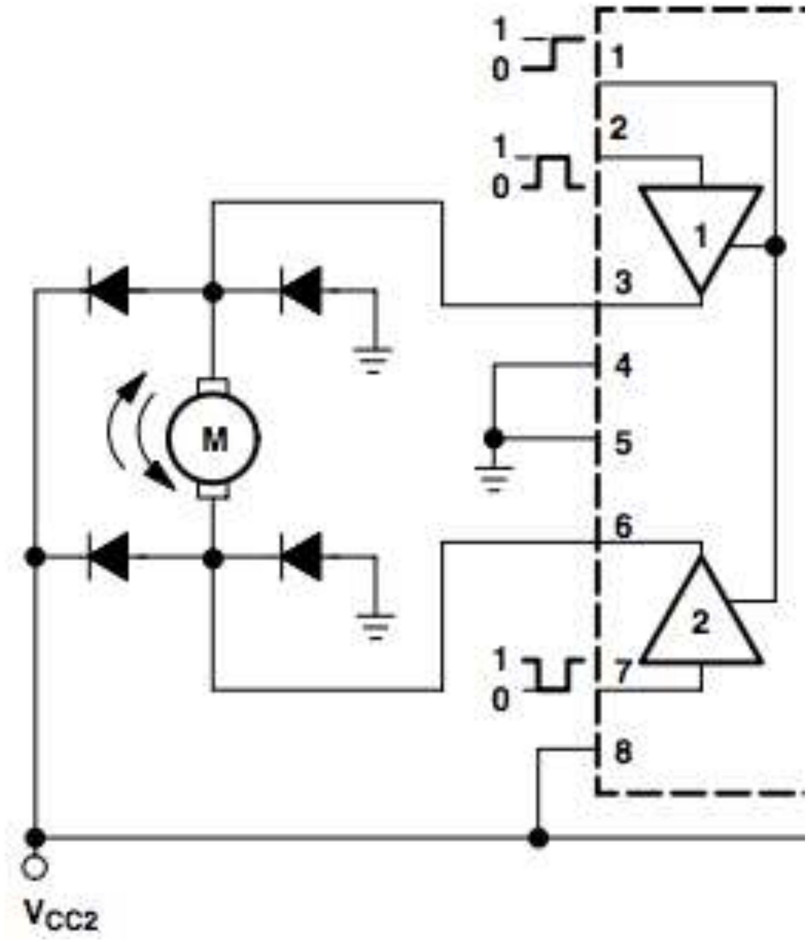
https://en.wikipedia.org/wiki/H_bridge



- Enables reversing direction of the motor
- Integrates driver as well

L293D Block Diagram

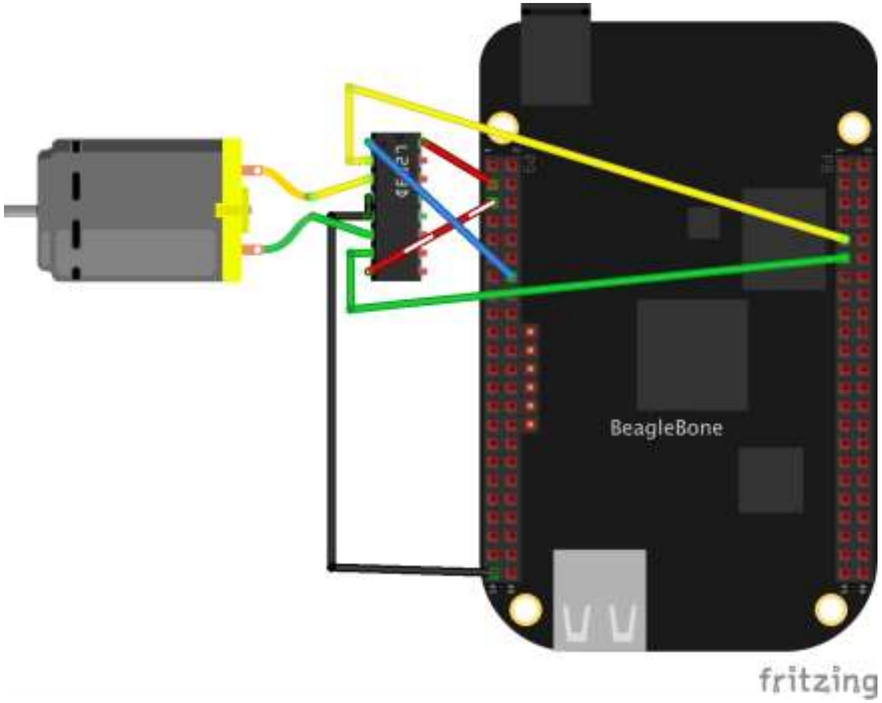
http://www.ti.com/lit/ds/symlink/l293d.pdf



- Pin 1 is the speed control
- Pin 2 is the forward drive
- Pin 7 is the backward drive

Connect your L293D H-bridge

http://beagleboard.org/Support/bone101/#headers



- Pin 1 to P9_14 “EN”
- Pin 2 to P8_9 “FWD”
- Pin 3 to “Motor +”
- Pin 4 and 5 to DGND
- Pin 6 to “Motor -”
- Pin 7 to P8_11 “BWD”
- Pin 8 to VDD_5V
- Pin 9 to VDD_3V3

Recipe 4.3: Controlling the motor

https://github.com/BeagleBoneCookbook/firstEdition/blob/master/04motors/h-bridgeMotor.js

```
var b = require('bonescript');
var motor = { SPEED: 'P9_14', FORWARD:
'P8_9', BACKWARD: 'P8_11' };
var FREQ = 50;
var STEP = 0.1;
var count = 0;
var stop = false;

b.pinMode(motor.FORWARD, b.OUTPUT);
b.pinMode(motor.BACKWARD, b.OUTPUT);
b.analogWrite(motor.SPEED, 0, FREQ, 0, 0);

var timer = setInterval(updateMotors, 100);

function updateMotors() {
    var speed = Math.sin(count*STEP);
    count++;
    Mset(motor, speed);
}
```

- Define the pins
- Keep track of state
- Setup pins initially
- Use a 100ms timer to update the motors
- Use a sine wave to increment/decrement the speed for test
- Call ‘Mset’ to update the PWM and direction

Recipe 4.3: Controlling the motor

https://github.com/BeagleBoneCookbook/firstEdition/blob/master/04motors/h-bridgeMotor.js

```
function Mset(motor, speed) {
    speed = (speed > 1) ? 1 : speed;
    speed = (speed < -1) ? -1 : speed;
    //console.log("Setting speed = " + speed);
    b.digitalWrite(motor.FORWARD, b.LOW);
    b.digitalWrite(motor.BACKWARD, b.LOW);
    if(speed > 0) {
     b.digitalWrite(motor.FORWARD, b.HIGH);
    } else if(speed < 0) {
     b.digitalWrite(motor.BACKWARD, b.HIGH);
    }
    b.analogWrite(motor.SPEED,
                             Math.abs(speed), FREQ);
}
```

- Put a cap on the maximum and minimum at 1 and -1
- Set the drive signals for direction
- Adjust PWM based upon the speed

Recipe 4.3: Controlling the motor

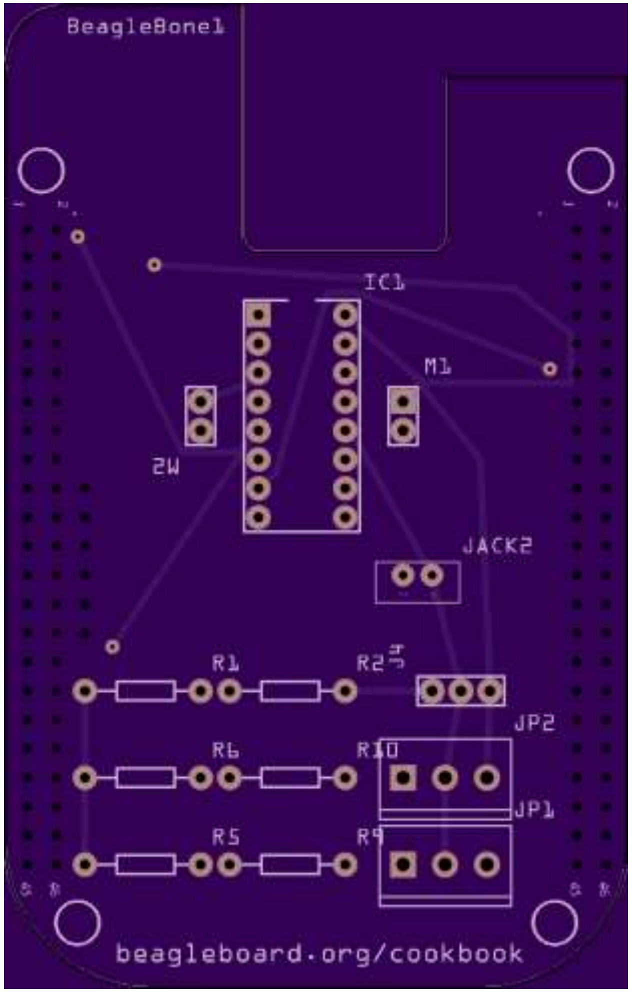
https://github.com/BeagleBoneCookbook/firstEdition/blob/master/04motors/h-bridgeMotor.js

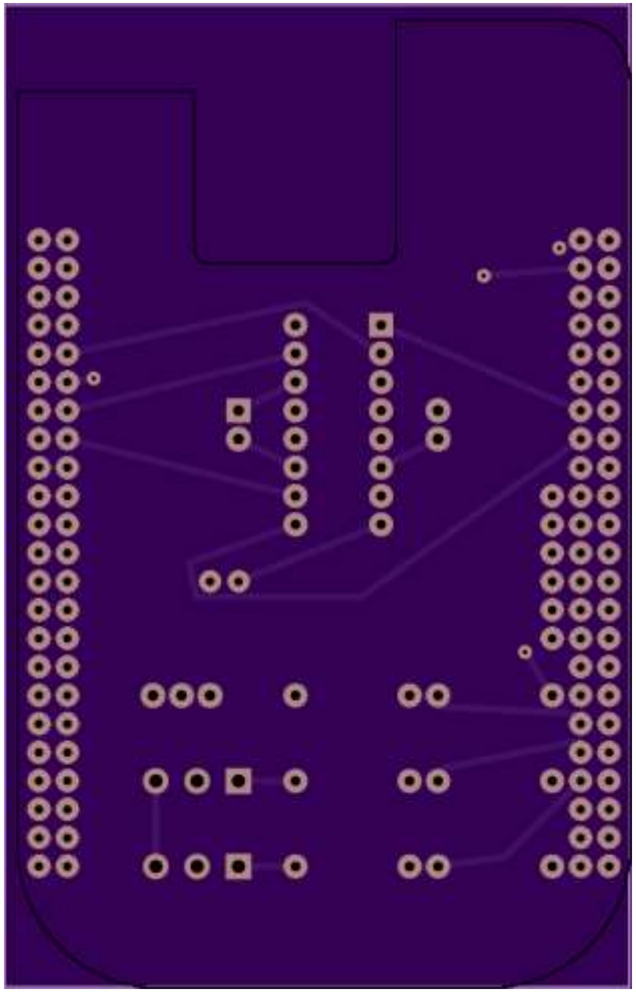
```
function doStop() {
    clearInterval(timer);
    Mset(motor, 0);
}

process.on('SIGINT', doStop);
```

- Detect when program is being stopped by a ^C
- Stop the timer and disable the motor

My quick-hack PCB

See recipe 9.7



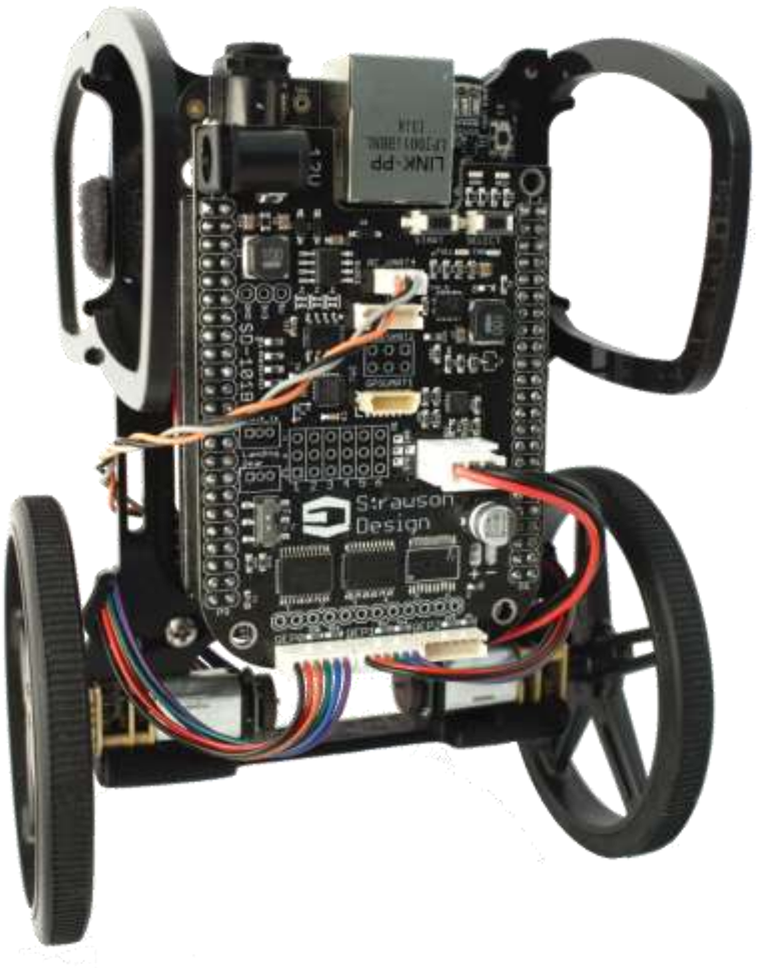


More

- Learn more about H-Bridges and motors
  - https://itp.nyu.edu/physcomp/lessons/dc-motors/dc-motors-the-basics/
- My simple PCB
  - https://oshpark.com/shared_projects/Mz40o0aN
- Shortcuts to updates and examples from the book
  - http://beagleboard.org/cookbook

I/O with mmap()

Understanding Real-Time



- Throughput vs. latency
- Hard, soft and firm
- Context switching
- Task scheduling
- Linux RT_PREEMPT
- Using ‘strace’ and ‘oprofile’

What are /dev/mem and mmap()?

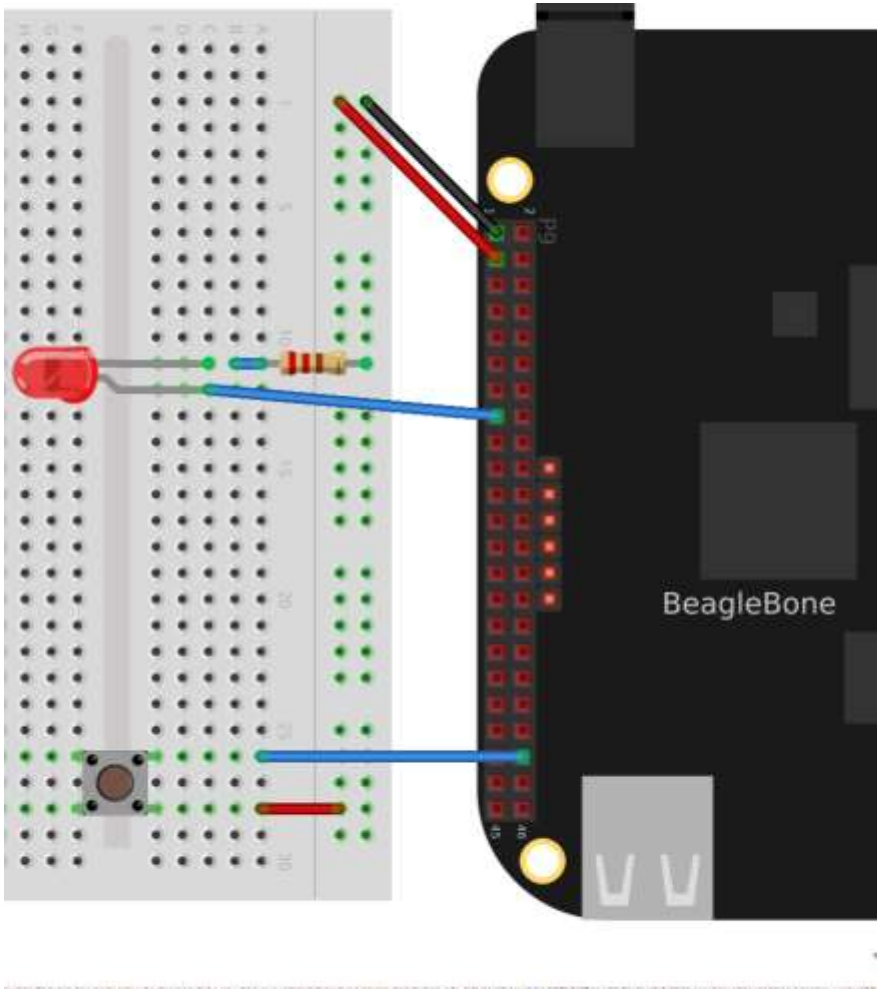
- /dev/mem is a character device that is an image of the main physical memory of the computer
- mmap() is a system function to map devices into (virtual) memory
- Together, they can be used to provide an application that has only a virtual memory space with access to specific physical addresses
- Directly accessing the registers bypasses system calls and avoids context switches
- This is really just a step towards writing your own device driver

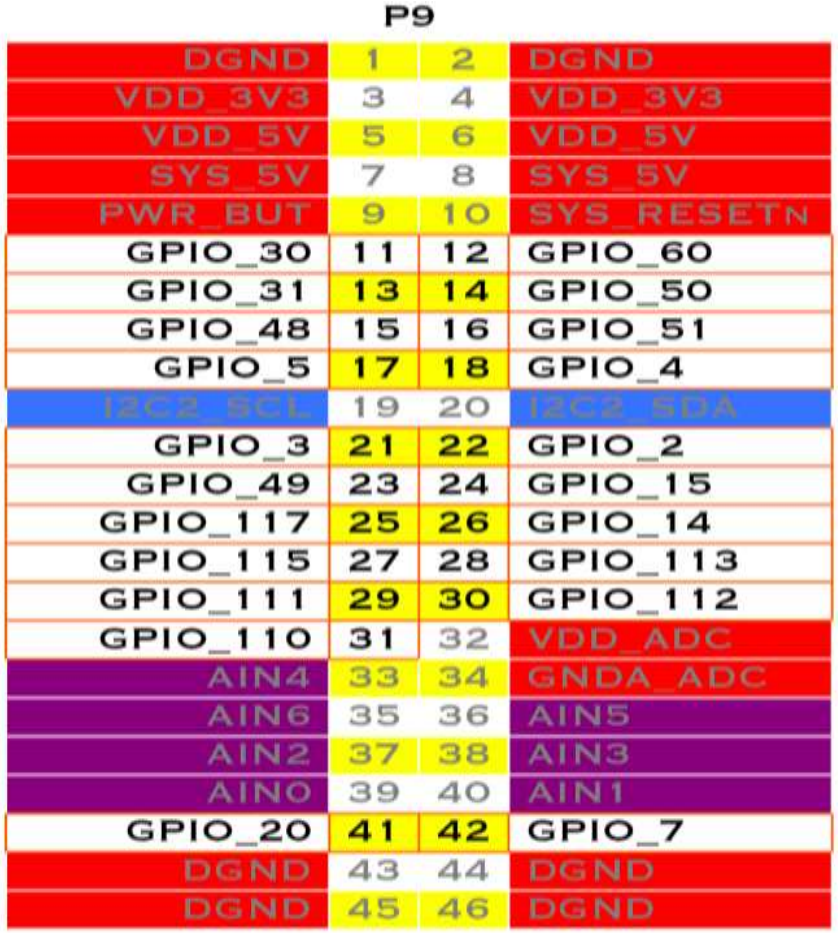
Prerequisites

- Connect to the board per recipe 1.2
  - http://beagleboard.org/getting-started
- Verify the software image per recipe 1.3 and potentially updating per recipe 1.9
  - http://beagleboard.org/latest-images
- Components
  - BeagleBone Black
  - Push button or 3.3V function generator
  - Jumper wire
  - LED with resistor or (preferred) oscilloscope

Connect a button and an LED

http://beagleboard.org/Support/bone101/#headers-gpio





P9

DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

Input on GPIO_7 and output on GPIO_31

Recipe 8.4: I/O with devmem2

```
bone# wget http://free-electrons.com/pub/mirror/devmem2.c
bone# gcc -o devmem2 devmem2.c && mv devmem2 /usr/local/bin/
bone# ln -s /sys/class/gpio
bone# echo 31 > gpio/export
bone# echo out > gpio/gpio31/direction
bone# echo 1 > gpio/gpio31/value
bone# echo 0 > gpio/gpio31/value
bone# devmem2 0x44E07138
bone# devmem2 0x44E07190 w 0x80000000
bone# devmem2 0x44E07194 w 0x80000000
bone# devmem2 0x44E07138
```

Recipe 8.4: I/O with C and mmap()

```
bone# wget https://raw.githubusercontent.com/BeagleBoneCookbook/firstEdition/master/08realtime/pushLEDmmap.c
bone# wget https://raw.githubusercontent.com/BeagleBoneCookbook/firstEdition/master/08realtime/pushLEDmmap.h
bone# gcc -O3 –o pushLEDmmap pushLEDmmap.c
bone# ./pushLEDmmap
^C
```

More

- AM335x Technical Reference Manual
  - http://bit.ly/1B4Cm45
- StarterWare for Sitara
  - http://www.ti.com/tool/starterware-sitara
- Enabling RT_PREEMPT
  - http://elinux.org/Beagleboard:BeagleBoneBlack_Debian#4.1.x-ti
- Learning to write a device driver in Recipe 7.2
- Program GPIO with PRU in Recipe 8.6
- Shortcuts to updates and examples from the book
  - http://beagleboard.org/cookbook

Thanks!

http://beagleboard.org/cookbook