01-3 – Blink an LED the Easy Way

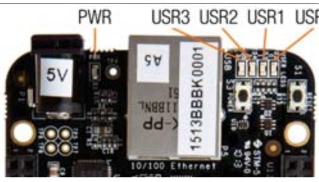
Much of this is from

BeagleBone Cookbook

BoneScript

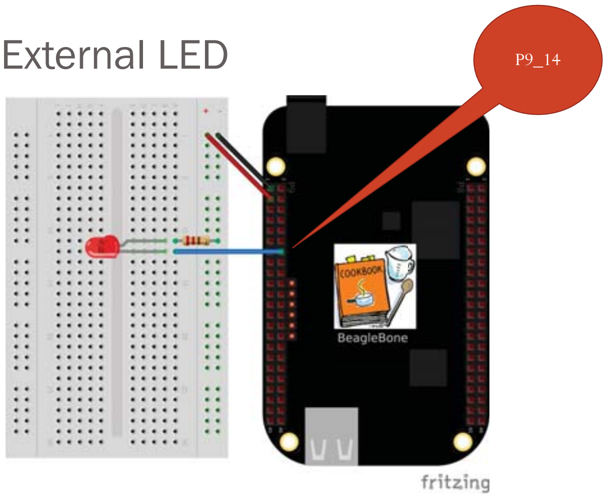
- http://beagleboard.org/Support/BoneScript/
- BoneScript is a Node.js library specifically optimized for the Beagle family and featuring familiar *Arduino* function calls.
- http://nodejs.org/
- Node.js® is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.
- Node.js uses an event-driven, non-blocking I/O model.
- https://code.google.com/p/v8/
- V8 is Google's open source JavaScript engine.
- V8 is written in C++.

Blink an LED



```
#!/usr/bin/env node
var b = require('bonescript');
var LED = 'USR0';
var state = b.HIGH;     // Initial state
b.pinMode(LED, b.OUTPUT);

setInterval(flash, 250);    // Change state every 250 ms

function flash() {
    b.digitalWrite(LED, state);
    if(state === b.HIGH) {
        state = b.LOW;
    } else {
        state = b.HIGH;
    }
}
```

External LED



Fritzing

- http://fritzing.org/home/
- Fritzing is an *open-source hardware initiative* that makes electronics accessible as a creative material for anyone.

Blink an LED

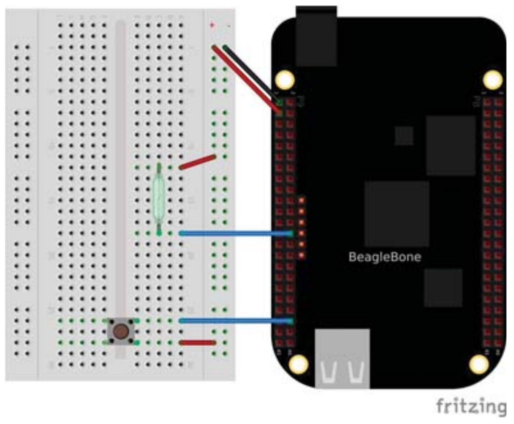
External

```
#!/usr/bin/env node
var b = require('bonescript');
var LED = 'P9_14';
var state = b.HIGH
b.pinMode(LED, b.OUTPUT);

setInterval(flash, 250);

function flash() {
    b.digitalWrite(LED, state);
    if(state === b.HIGH) {
        state = b.LOW;
    } else {
        state = b.HIGH;
    }
}
```

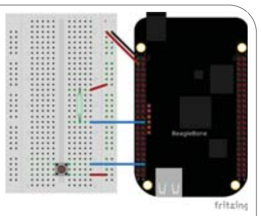
Internal

```
#!/usr/bin/env node
var b = require('bonescript');
var LED = 'USR0';
var state = b.HIGH;
b.pinMode(LED, b.OUTPUT);

setInterval(flash, 250);

function flash() {
    b.digitalWrite(LED, state);
    if(state === b.HIGH) {
        state = b.LOW;
    } else {
        state = b.HIGH;
    }
}
```

Read a button

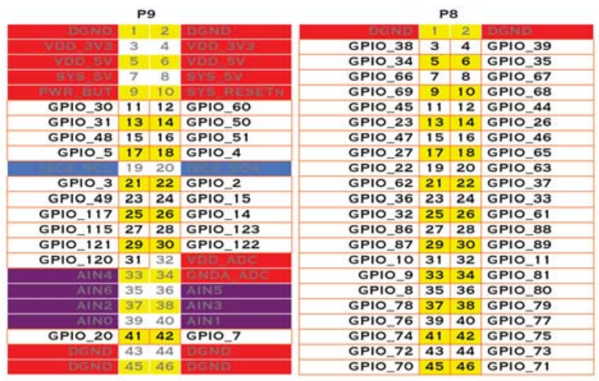


Button via Interrupts



```
#!/usr/bin/env node
var b = require('bonescript');
var button = 'P9_42';

b.pinMode(button, b.INPUT, 7, 'pulldown');

b.attachInterrupt(button, true,
      b.CHANGE, printStatus);

function printStatus(x) {
    console.log('x.value = ' + x.value);
    console.log('x.err   = ' + x.err);
}
```

65 possible digital I/Os



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_123
GPIO_121	29	30	GPIO_122
GPIO_120	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

P8

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

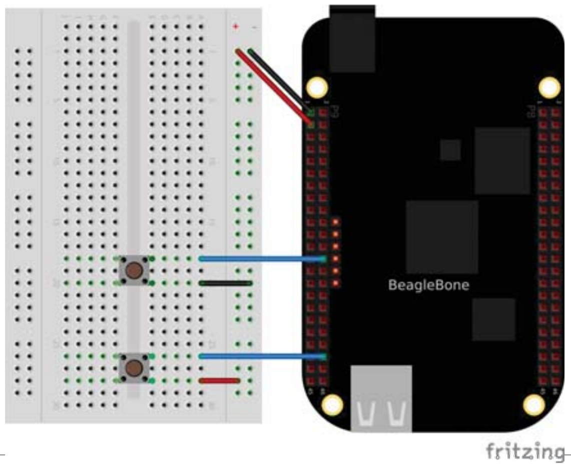
Button via Read

```
#!/usr/bin/env node
var b = require('bonescript');
var button = 'P9_42';
var state;      // State of pushbutton

b.pinMode(button, b.INPUT, 7, 'pulldown');

state = b.digitalRead(button);
console.log('button state = ' + state);
```

Button via Callback

```
#!/usr/bin/env node
var b = require('bonescript');
var button = 'P9_42';

b.pinMode(button, b.INPUT, 7, 'pulldown');

b.digitalRead(button, printStatus);

function printStatus(x) {
    console.log('x.value = ' + x.value);
    console.log('x.err   = ' + x.err);
}
```

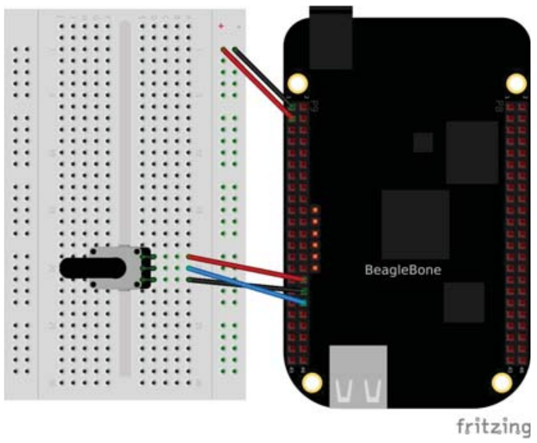
Pull up/down resistors



Pull up/down code

```
#!/usr/bin/env node
var b = require('bonescript');
var buttonTop = 'P9_26';
b.pinMode(buttonTop, b.INPUT, 7, 'pullup');
b.attachInterrupt(buttonTop, true, b.CHANGE, printStatus);

var buttonBot = 'P9_42';
b.pinMode(buttonBot, b.INPUT, 7, 'pulldown');
b.attachInterrupt(buttonBot, true, b.CHANGE, printStatus);

function printStatus(x) {
    console.log('x.value = ' + x.value);
    console.log('x.err   = ' + x.err);
}
```

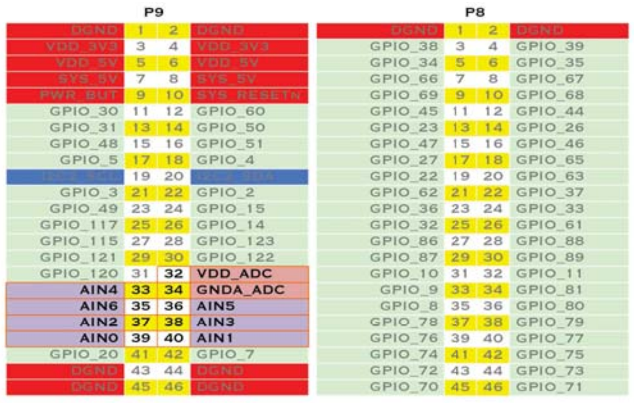
Analog in



Analog Code

```
#!/usr/bin/env node
var b = require('bonescript');

b.analogRead('P9_36', printStatus);

function printStatus(x) {
    console.log('x.value = ' +
          x.value.toFixed(3));
    console.log('x.err   = ' + x.err);
}
```

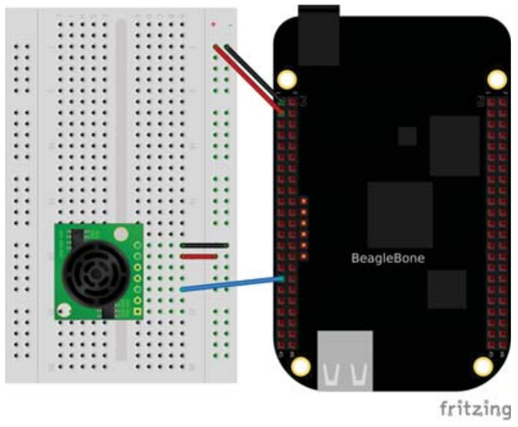
7 analog inputs (1.8V)



	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_123
GPIO_121	29	30	GPIO_122
GPIO_120	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

	P8		
DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

Range Finder



Range Finder Code

```
#!/usr/bin/env node
var b = require('bonescript');
var ms = 250;  // Time in milliseconds

setInterval(readRange, ms);

function readRange() {
    b.analogRead('P9_33', printStatus);
}
function printStatus(x) {
    console.log('x.value = ' + x.value);
}
```