

# Teach, Learn and Connect with BeagleBone Black

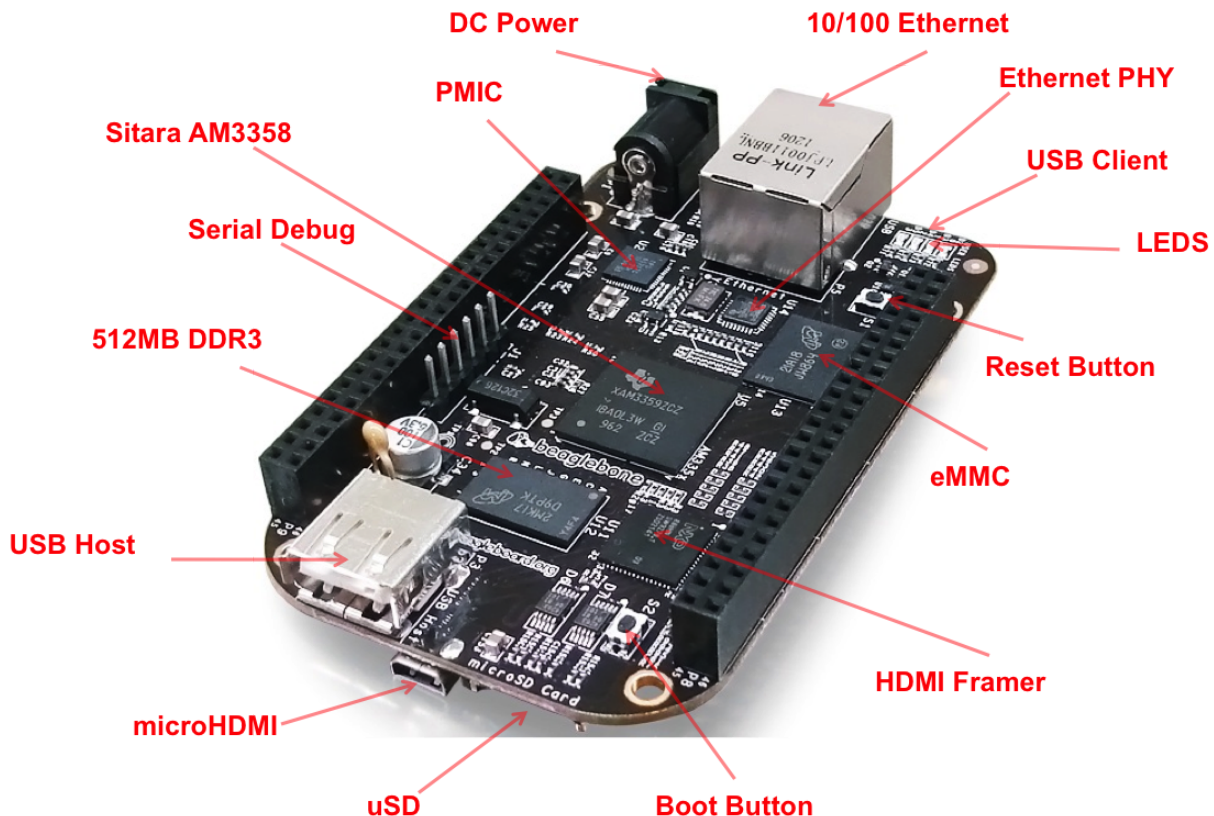
*A hands-on workshop from LEDs to the Internet of Things*

by

Mark A. Yoder

*Rose-Hulman Institute of Technology*

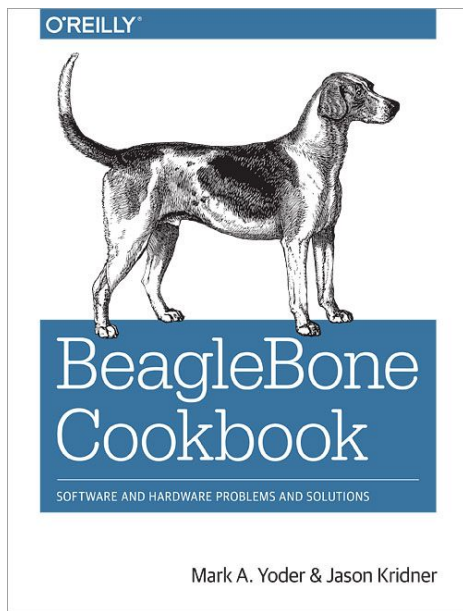
[Mark.A.Yoder@Rose-Hulman.edu](mailto:Mark.A.Yoder@Rose-Hulman.edu)



## Getting Started

1. Plug the BeagleBone into your computer via USB.
2. Open the new drive that appears.
3. Open **START.htm** with Chrome or Firefox.
4. If you are using your own laptop, install driver for your OS. (You'll have to click **Install** several times.). The drive is already installed on the workshop laptops.
5. Return to browser window with **START.htm** and scroll down to **Step 3** to find <http://192.168.7.2> and click on it.
6. Explore.
7. Click on the title **Cloud9 IDE** (<http://192.168.7.2:3000>).
8. Continue with lab handouts.

A hands-on workshop from LEDs to the Cloud



## The BeagleBone Cookbook

The examples used in this workshop are from the **BeagleBone Cookbook**

(<https://ssearch.oreilly.com/?q=beaglebone+cookbook>).

The Cookbook is full of recipes which include simple wiring diagrams and example code to get you started.

# Cape Expansion Headers

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	MMC1_DAT6	3	4	MMC1_DAT7
VDD_5V	5	6	VDD_5V	MMC1_DAT2	5	6	MMC1_DAT3
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BTN	9	10	SYS_RESETN	GPIO_69	9	10	GPIO_68
UART4_RXD	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
UART4_TXD	13	14	EHRPWM1A	EHRPWM2B	13	14	GPIO_26
GPIO_48	15	16	EHRPWM1B	GPIO_47	15	16	GPIO_46
SPI0_CS0	17	18	SPI0_D1	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	EHRPWM2A	19	20	MMC1_CMD
SPI0_DO	21	22	SPI0_SCLK	MMC1_CLK	21	22	MMC1_DAT5
GPIO_49	23	24	UART1_TXD	MMC1_DAT4	23	24	MMC1_DAT1
GPIO_117	25	26	UART1_RXD	MMC1_DAT0	25	26	GPIO_61
GPIO_115	27	28	SPI1_CS0	LCD_VSYNC	27	28	LCD_PCLK
SPI1_DO	29	30	GPIO_122	LCD_HSYNC	29	30	LCD_AC_BIAS
SPI1_SCLK	31	32	VDD_ADC	LCD_DATA14	31	32	LCD_DATA15
AIN4	33	34	GND_ADC	LCD_DATA13	33	34	LCD_DATA11
AIN6	35	36	AIN5	LCD_DATA12	35	36	LCD_DATA10
AIN2	37	38	AIN3	LCD_DATA8	37	38	LCD_DATA9
AIN0	39	40	AIN1	LCD_DATA6	39	40	LCD_DATA7
GPIO_20	41	42	ECAPPWM0	LCD_DATA4	41	42	LCD_DATA5
DGND	43	44	DGND	LCD_DATA2	43	44	LCD_DATA3
DGND	45	46	DGND	LCD_DATA0	45	46	LCD_DATA1

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

A hands-on workshop from LEDs to the Cloud

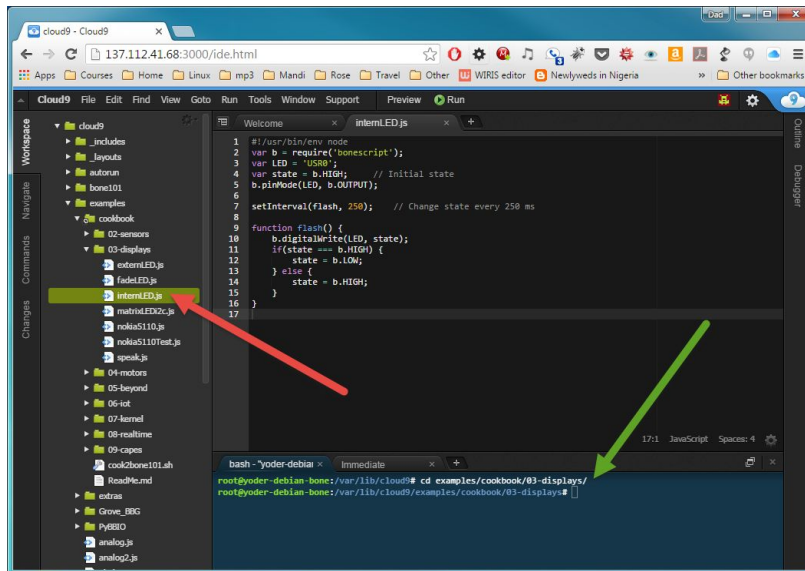
# Blink an LED

**Goal:** Blink some LEDs.

**Overview:** Bonescript is an Arduino-like language, built on JavaScript, that makes interacting with the physical world easy. Here we will blink the Bone's right-most built in LED using a script from page 90 of the BeagleBone Cookbook.

## Do this

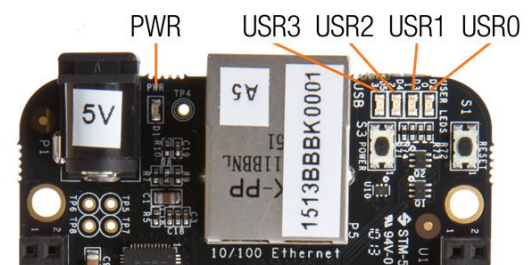
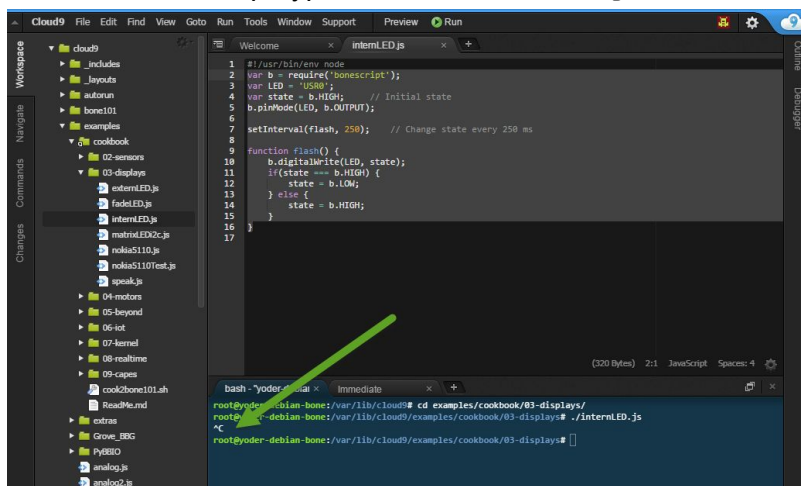
1. Open *Cloud9* by pointing your browser to <http://192.168.7.2:3000/>
2. Navigate to **examples/cookbook/03-displays/internalLED.js** and double-click on it.



3. In the window pointed to by the **Green Arrow** above, type:

**cd examples/cookbook/03-displays**

4. To run the script type: **./internalLED.js**

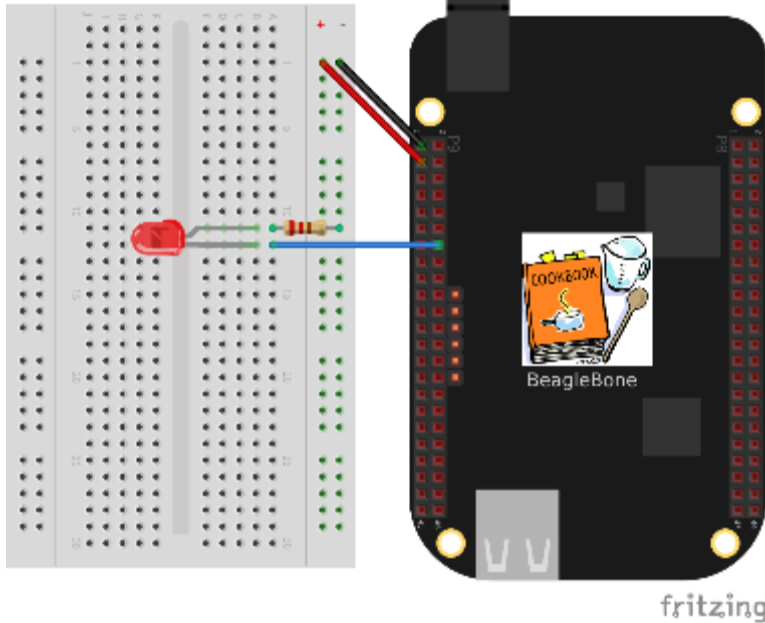


5. After a few seconds watch the right-most blue internal LED blink.
6. To stop, enter **^c** (Ctrl-C).

A hands-on workshop from LEDs to the Cloud

## Extras

- Change the blink time.
- Create a blink pattern.
- Wire an external LED. (Note the short lead of the LED goes to the resistor.) Try running **cookbook/03-displays/externalLED.js** (From page 92 of the Cookbook.)



- Try running **cookbook/03-displays/fade.js**. This uses the Bone's PWM hardware to fade an LED. (From page 95 of the Cookbook.)

## The Code

```
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;
  }
}
```



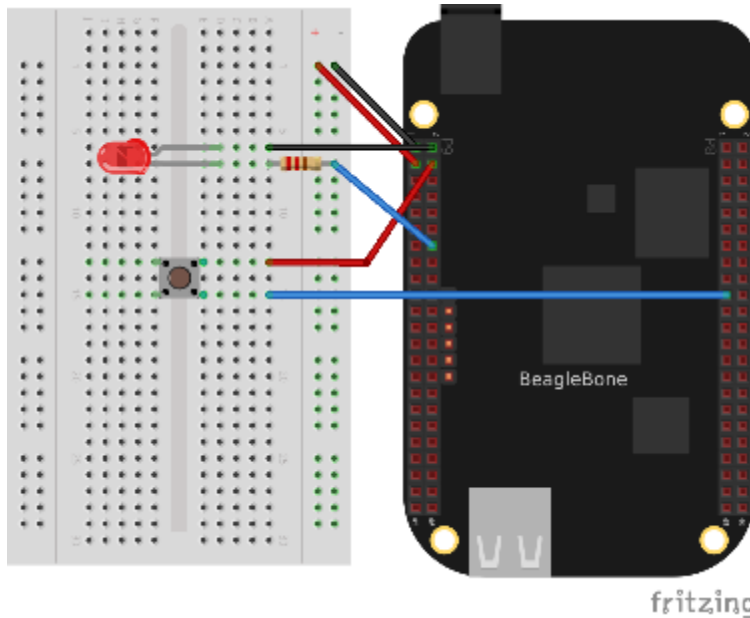
# Read a switch

**Goal:** Sense the external world by reading a switch.

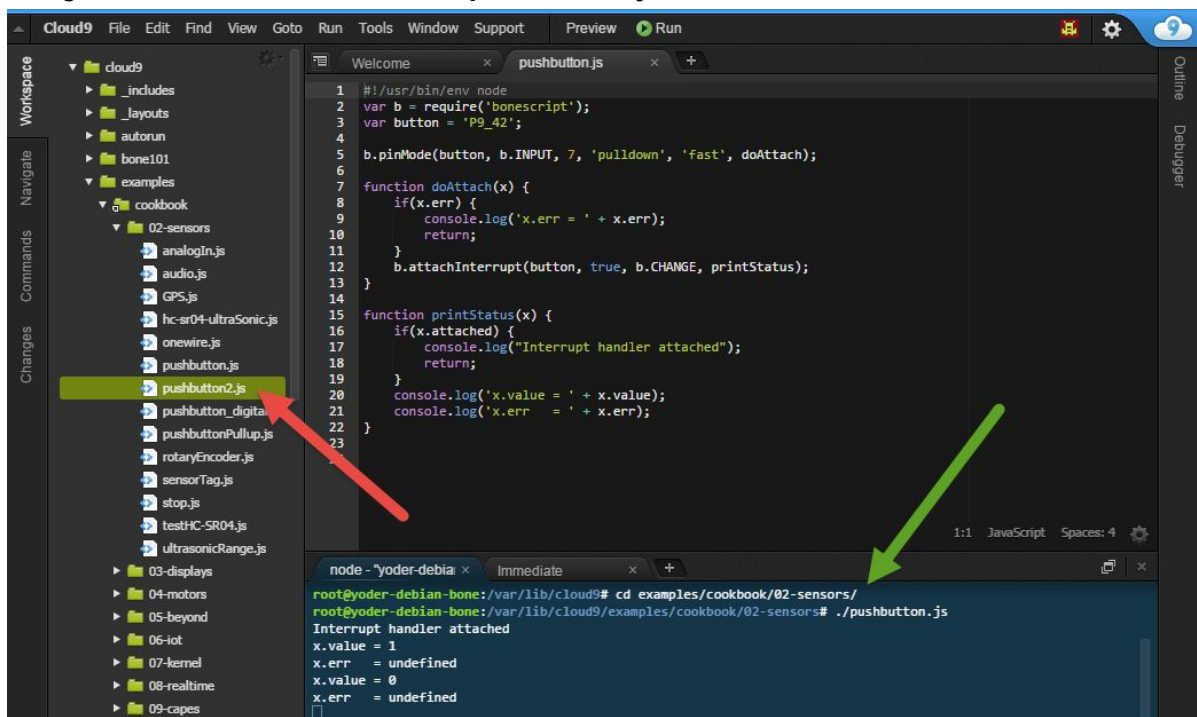
**Overview:** Reading a switch attached to a GPIO port is as easy as configuring the port as an input and attaching an interrupt handler to it. This is from page 50 of the BeagleBone Cookbook

## Do this

1. Wire as switch as shown. (Wiring the LED is optional.)



2. Open *Cloud9* by pointing your browser to <http://192.168.7.2:3000/>.
3. Navigate to **cookbook/02-sensors/pushbutton.js**



A hands-on workshop from LEDs to the Cloud

4. In the window pointed to by the **Green Arrow** above, type:  
`cd /var/lib/cloud9/examples/cookbook/02-sensors`
5. Then type: `./pushbutton.js`
6. Once the message **Interrupt handler attached** appears, push the button. (You may have to wait it bit the first time.)
7. Can you figure what the code does?
8. To stop, enter `^c` (Ctrl-C).

## Extras

- Make an LED (internal or external) respond.
- Use a different input port (Hint: <http://192.168.7.2/bone101/Support/bone101/#headers> or <http://beagleboard.org/Support/bone101/#headers>).

## The Code

```
var b = require('bonescript');
var button = 'P9_42';

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

function doAttach(x) {
  if(x.err) {
    console.log('x.err = ' + x.err);
    return;
  }
  b.attachInterrupt(button, true, b.CHANGE, printStatus);
}

function printStatus(x) {
  if(x.attached) {
    console.log("Interrupt handler attached");
    return;
  }
  console.log('x.value = ' + x.value);
  console.log('x.err = ' + x.err);
}
```

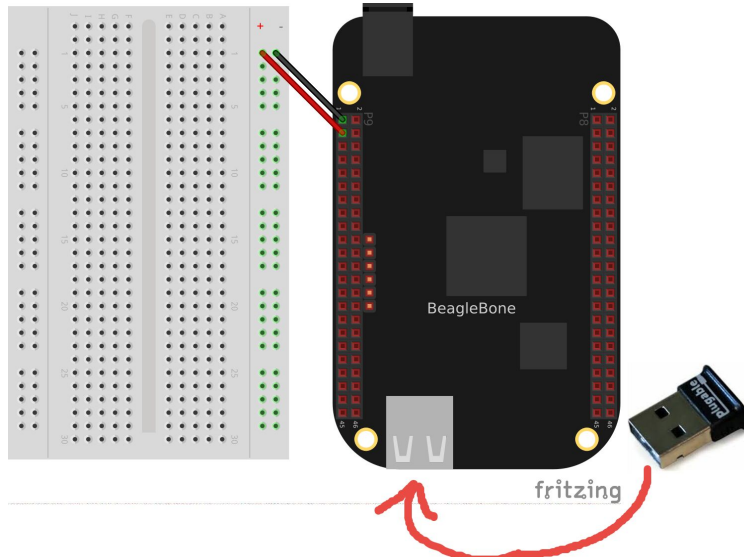
# SensorTag

**Goal:** Sense the world with a SensorTag.

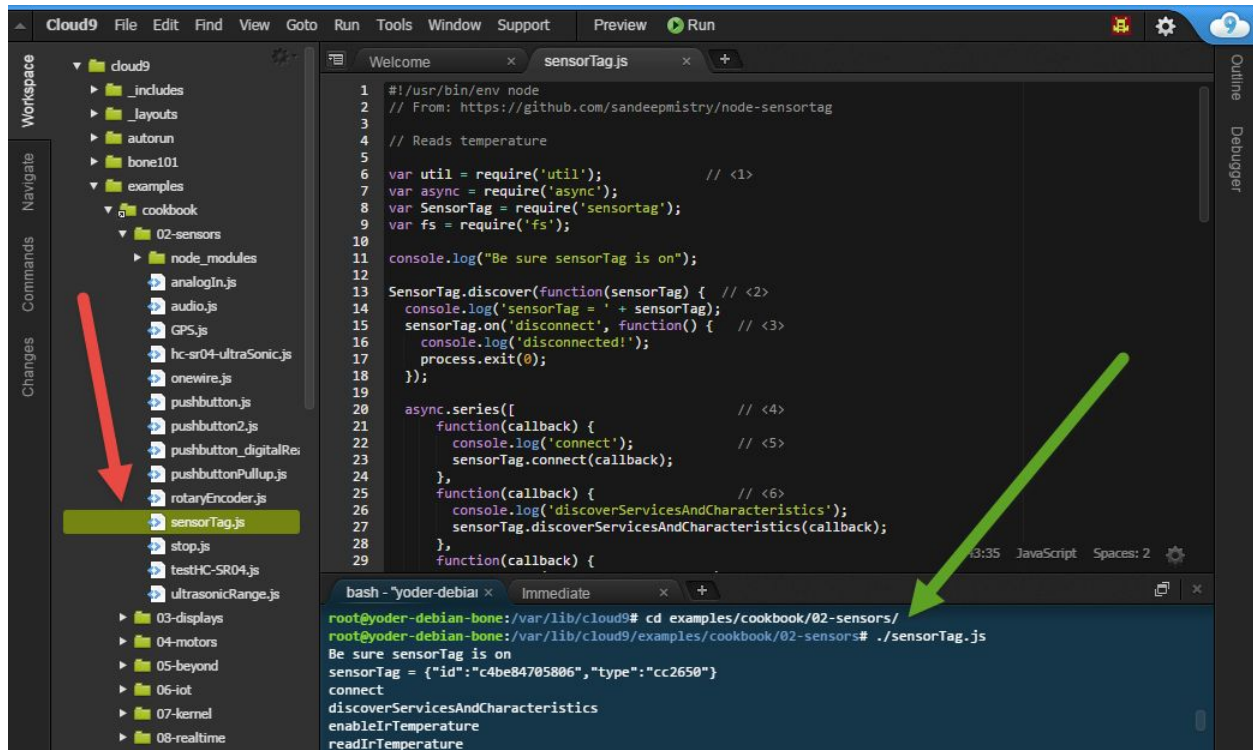
**Overview:** The SensorTag has numerous sensors that can be accessed via Bluetooth. This is from page 81 of the BeagleBone Cookbook.

## Do this

1. Plug the Bluetooth USB dongle into the BeagleBone.



2. Navigate to `cookbook/02-sensors/sensorTag.js`.



A hands-on workshop from LEDs to the Cloud

1. In the window pointed to by the **Green Arrow** above, type:  

```
cd /var/lib/cloud9/examples/cookbook/02-sensors
```
2. Then type: `./sensorTag.js`
3. Once the message **Be sure sensorTag is on** appears, press the “side” button on the SensorTag.



1. Wait a moment while it connects to the SensorTag.
2. The SensorTag has many sensors. Here we are only reading the IR and ambient temperature sensors. Removing it from the red case may give better readings.
3. Try clicking the button you turned it on with. Try clicking the button on the other side. Try clicking both at the same time.



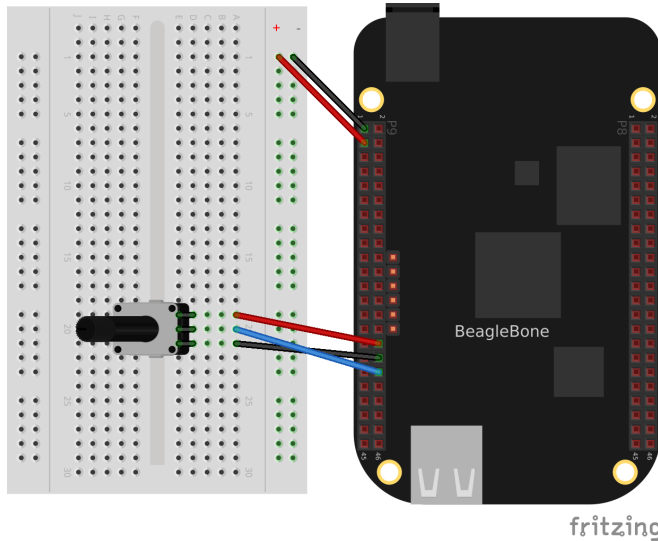
# Read a Variable Resistor

**Goal:** Sense the analog world by reading a variable resistor.

**Overview:** The Bone has seven analog inputs that are read with **analogRead()**. This is from page 55 of the BeagleBone Cookbook.

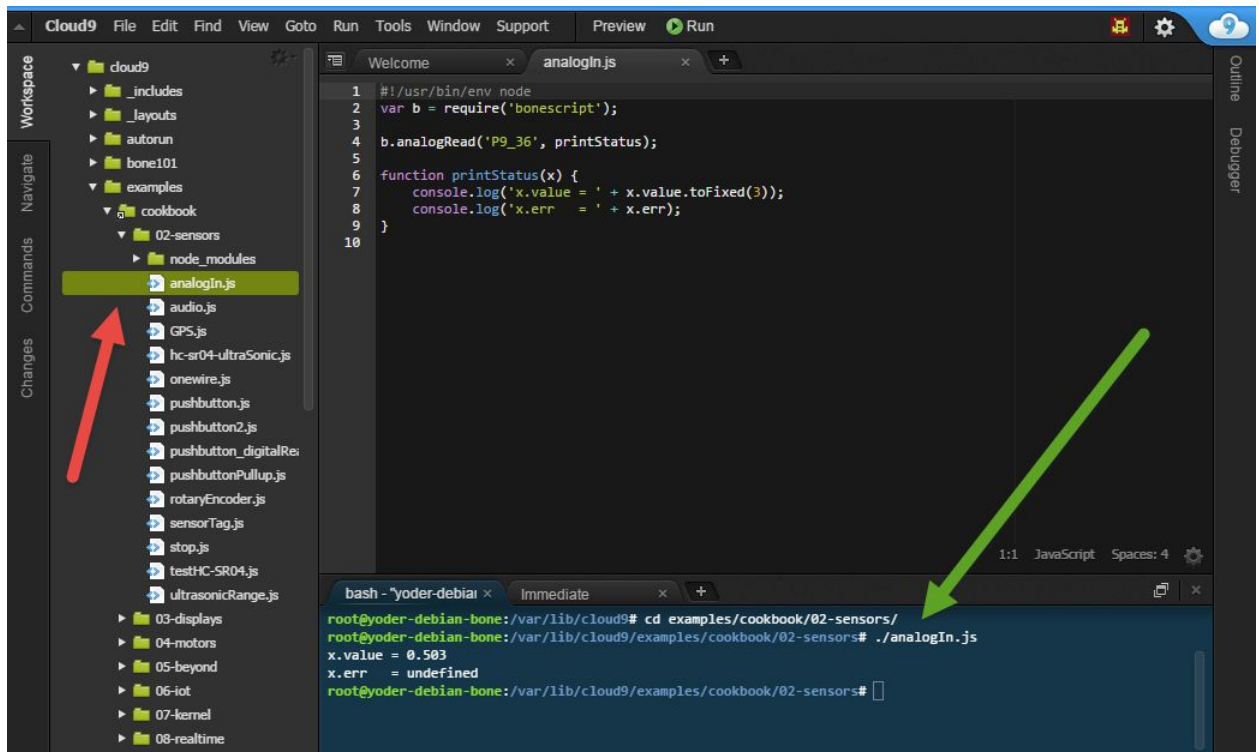
## Do this

1. Wire the variable resistor as shown. (If you wired the LED, leave it in place.)



fritzing

2. Open *Cloud9* by pointing your browser to <http://192.168.7.2:3000/>.
3. Navigate to `/var/lib/cloud9/examples/cookbook/02-sensors/analogIn.js`



4. In the window pointed to by the **Green Arrow** above, type:

A hands-on workshop from LEDs to the Cloud

```
cd /var/lib/cloud9/examples/cookbook/02-sensors
```

5. Then type: `./analogIn.js`
6. Can you figure what the code does?
7. Turn the pot and run again.
8. Can you make it take readings at a set interval?

## Extras

- If you wired up the LED, open and run **demo/analog.js**. It uses the variable resistor to control the brightness of the LED.
- Use a different analog input. (Hint: <http://192.168.7.2/bone101/Support/bone101/#headers> or <http://beagleboard.org/Support/bone101/#headers>)

## The Code

```
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);
}
```

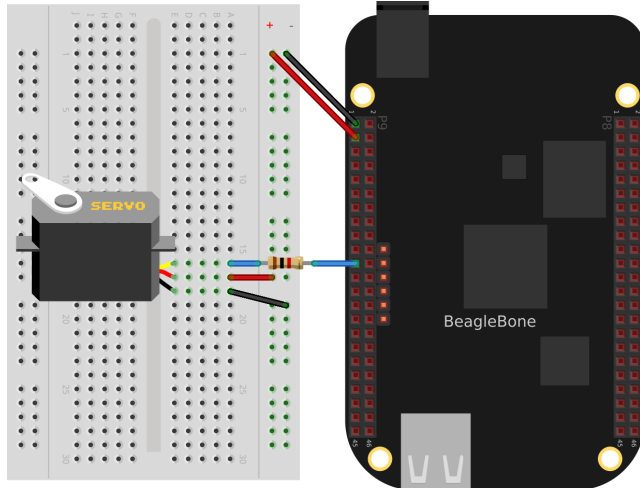
# Turn a Servo Motor

**Goal:** Make a small servo motor turn..

**Overview:** The **analogWrite()** command on the Bone uses PWM for it's output. This means it can also drive a small servo motor. This is from page 110 of the BeagleBone Cookbook.

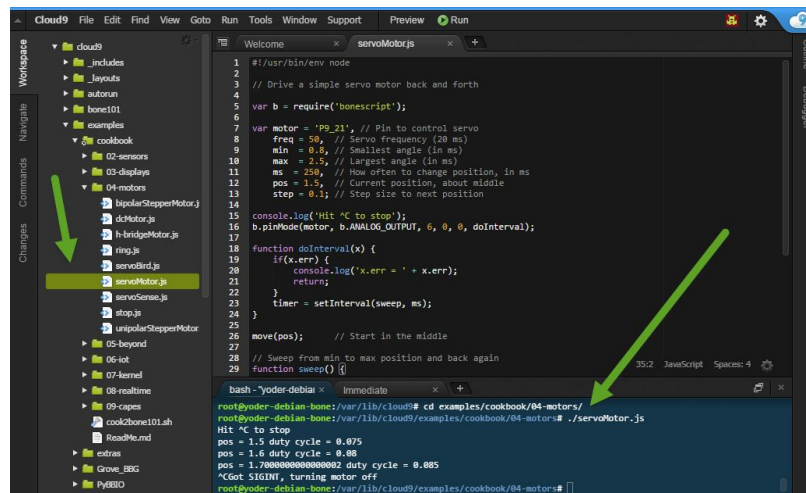
## Do this

1. Wire the servo as show. The yellow wire connects to the resistor, the center wire connects to the 3.3V supply and the brown wire is ground.



fritzing

2. Open *Cloud9* by pointing your browser to <http://192.168.7.2:3000/>.
1. Navigate to `/var/lib/cloud9/examples/cookbook/04-motor/servoMotor.js`



3. In the window pointed to by the **Green Arrow** above, type:  
`cd /var/lib/cloud9/examples/cookbook/04-motors`
4. Then type: `./servoMotor.js`
5. The servo should start turning.
6. To stop, enter `^C` (Ctrl-C).

A hands-on workshop from LEDs to the Cloud

## Extras

- Make the variable resistor set the position of the servo.
- Use a different PWM port. (Hint: <http://192.168.7.2/bone101/Support/bone101/#headers> or <http://beagleboard.org/Support/bone101/#headers>)

## The Code

```
var b = require('bonescript');
var motor = 'P9_21', // Pin to control servo
    freq = 50, // Servo frequency (20 ms)
    min = 0.8, // Smallest angle (in ms)
    max = 2.5, // Largest angle (in ms)
    ms = 250, // How often to change position, in ms
    pos = 1.5, // Current position, about middle
    step = 0.1; // Step size to next position
console.log('Hit ^C to stop');
b.pinMode(motor, b.ANALOG_OUTPUT, 6, 0, 0, doInterval);
function doInterval(x) {
    if(x.err) {
        console.log('x.err = ' + x.err);
        return;
    }
    timer = setInterval(sweep, ms);
}
move(pos); // Start in the middle
// Sweep from min to max position and back again
function sweep() {
    pos += step; // Take a step
    if(pos > max || pos < min) {
        step *= -1;
    }
    move(pos);
}
function move(pos) {
    var dutyCycle = pos/1000*freq;
    b.analogWrite(motor, dutyCycle, freq);
    console.log('pos = ' + pos + ' duty cycle = ' + dutyCycle);
}
process.on('SIGINT', function() {
    console.log('Got SIGINT, turning motor off');
    clearInterval(timer); // Stop the timer
    b.analogWrite(motor, 0, freq); // Turn motor off
});
```

A hands-on workshop from LEDs to the Cloud