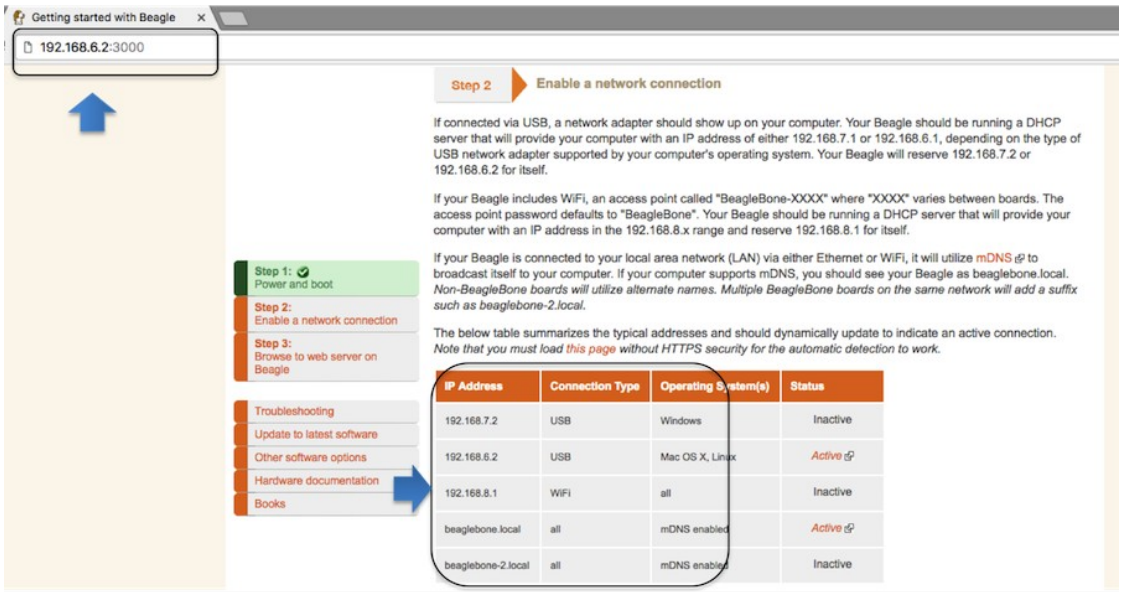
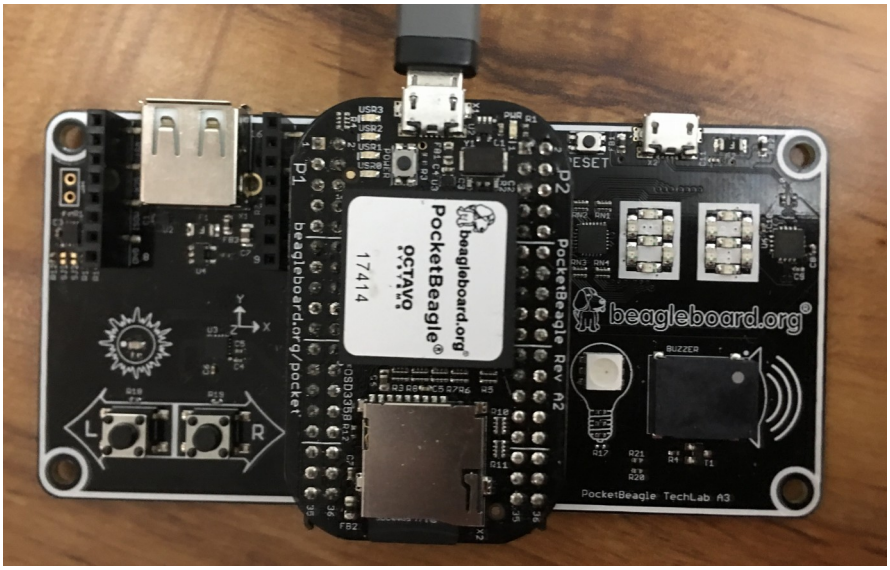
PocketBeagle® TechLab Cape Hands-On Coding Workshop

- MicroSD software image and other materials available from **bbb.io/techlab**

See **bbb.io/start** for instructions on using Etcher.io to write a microSD card

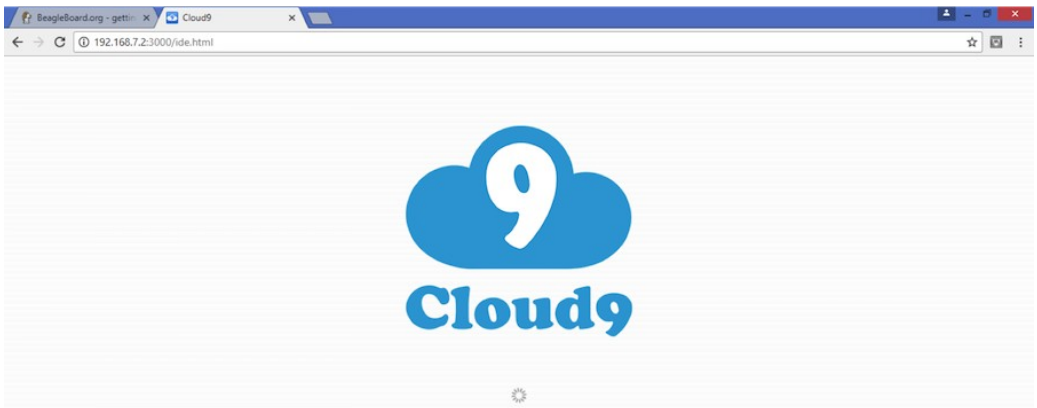




Plug into the microUSB on PocketBeagle to provide power and a network connection. Look for the "heartbeat" pulse on the USR0 LED to know the board has Linux up-and-running.

Get to the Cloud9 IDE

- Served on port 3000
- Windows: http://192.168.7.2:3000
- Linux/Mac: http://192.168.6.2:3000



The **BeagleBoard.org Foundation** is a 501(c)(3) non-profit corporation existing to provide education in and collaboration around the design and use of open-source software and hardware in embedded computing.

Blink PocketBeagle on-board USRx LED

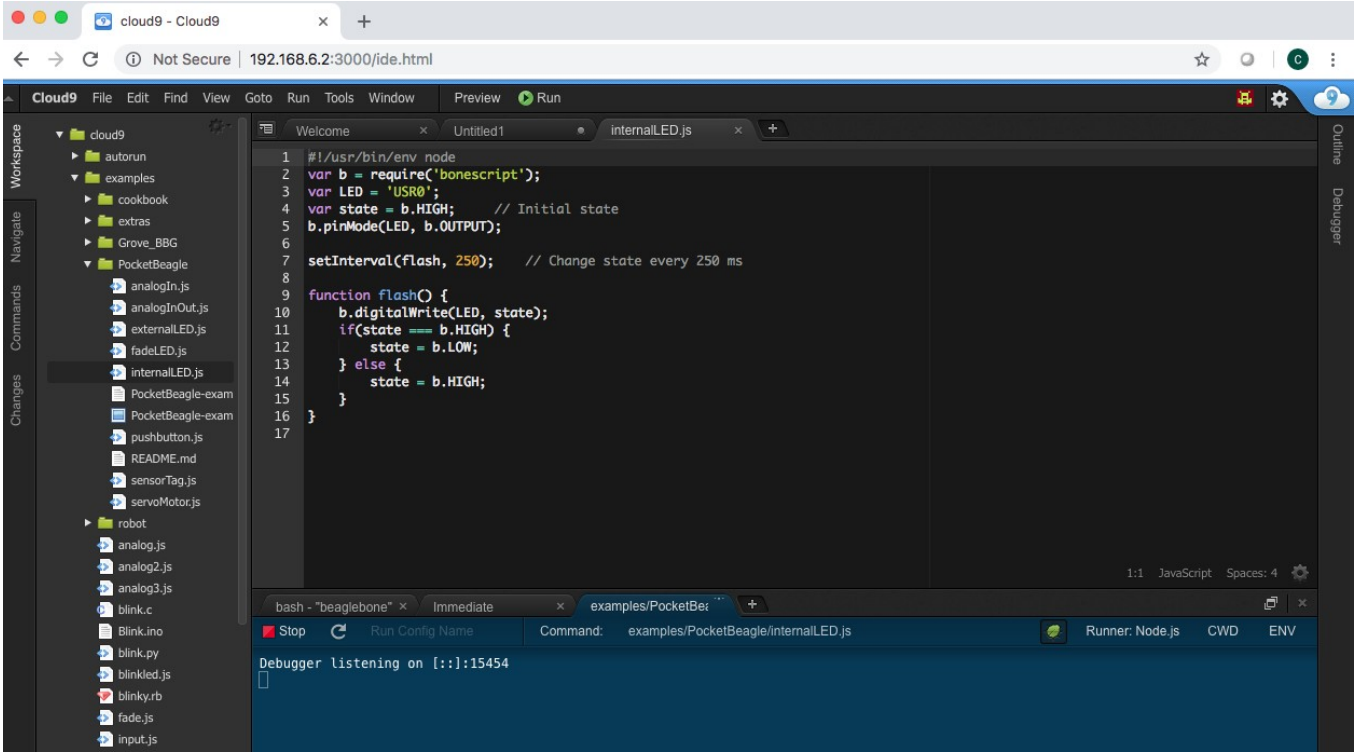
Goal: Blink USR3 LED on PocketBeagle.

Overview: BoneScript is a Node.js library customized for the Beagle family and featuring familiar Arduino function calls. Here we will use it to blink an LED built into your PocketBeagle.

Do this:

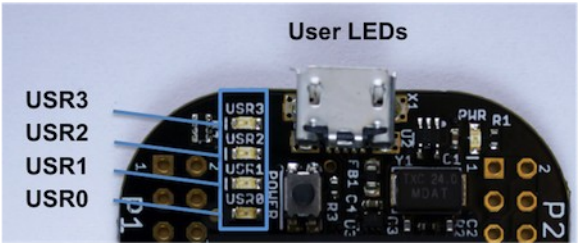
1. Navigate to **examples/TechLab/internalLED.js** and double-click on it.
2. Click the Run button in the toolbar to execute the script in the active file window
3. You will see the run configuration window open with a Stop button. Click the Stop button to halt the program.
4. Try changing the LED or blink time, save the program and run again.

TIP: Click the green bug to disable the debugger and begin execution quicker.



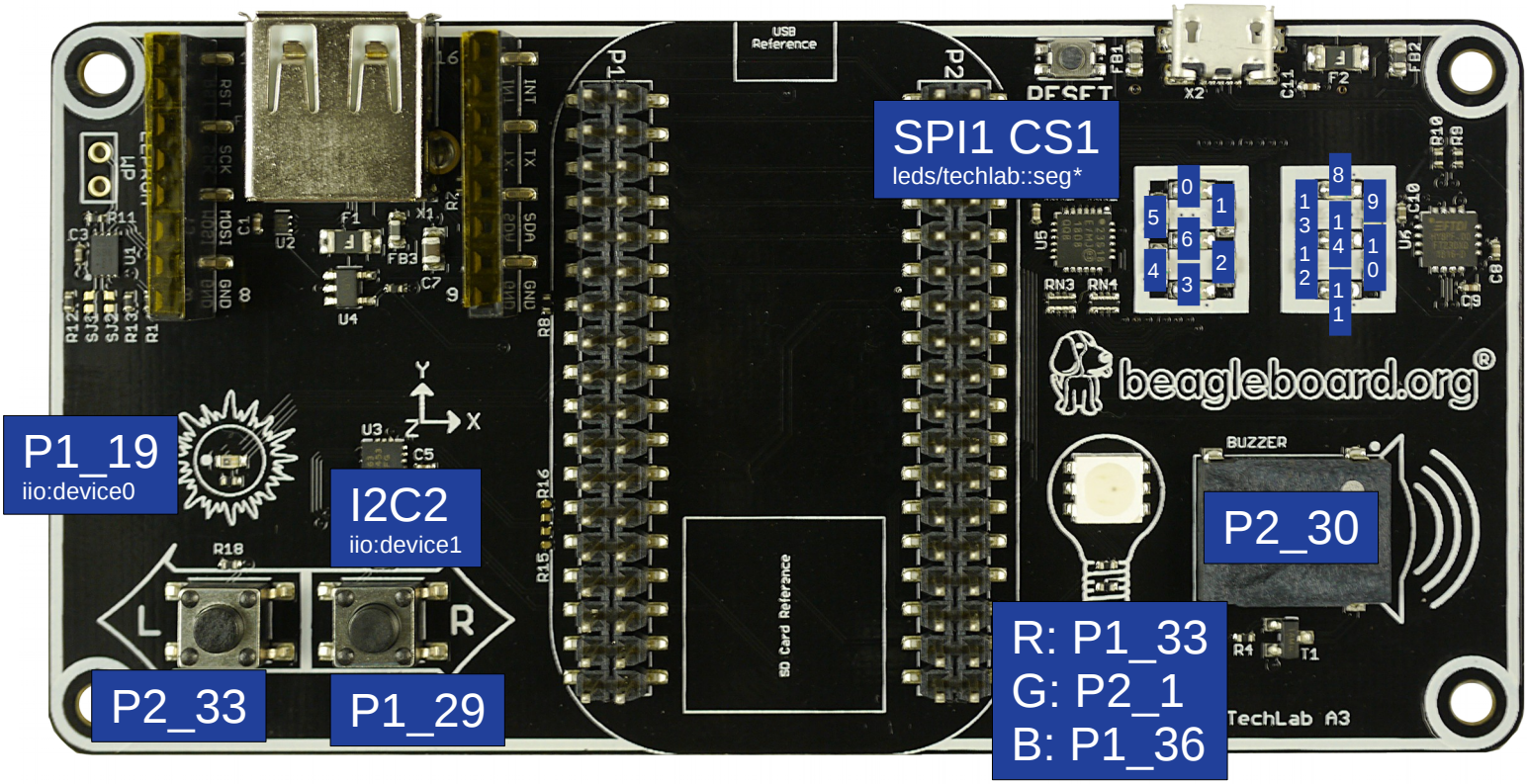
internalLED.js

```
#!/usr/bin/env node
var b = require('bonescript');
var LED = 'USR3';
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;
    }
}
```



PocketBeagle® TechLab Cape Hands-On Coding Workshop

TechLab Wiring Summary



PocketBeagle Expansion Header Pin-out

P1

			SYS	VIN	1	2	87		6	AIN 3.3V	9	PRU1	
		USB1	V_EN	GPIO	109	3	4	89			11		
				VBUS	5	6	5	GPIO	CS	SPI0	TX	PRU	
				VIN	7	8	2		CLK		RX	UART2	
				DN	9	10	3		MISO		TX		
		USB1		DP	11	12	4		MOSI		RX	PRU	
				ID	13	14	3.3V	SYS					
				GND	15	16	GND						
				REF-	17	18	REF+	AIN 1.8V					
				0	19	20	20	GPIO			16(in)	PRU0	
		AIN 1.8V		1	21	22	GND	SYS					
				2	23	24	VOUT						
				3	25	26	12	GPIO	SDA	I2C2	TX	CAN0	
				4	27	28	13		SCL		RX		
PRU0	7	QEP0	STRB	GPIO	117	29	30	43	TX	UART0	15	PRU1	
PRU0	4		A		114	31	32	42	RX		14		
PRU0	1	PWM0	B		111	33	34	26					
PRU1	10				88	35	36	110	A	PWM0	0	PRU0	

P2

			PWM1	A	GPIO	50	1	2	59	GPIO			
			PWM2	B		23	3	4	58				
			UART4	RX		30	5	6	57				
				TX		31	7	8	60				
CAN1	RX	I2C1		SCL		15	9	10	52				
CAN1	TX			SDA		14	11	12	PWR BTN	SYS			
					SYS	VOUT	13	14	VIN	BAT			
						GND	15	16	TEMP				
					GPIO	65	17	18	47	GPIO	STRB	QEP2	15i	PRU0
						27	19	20	64					
					SYS	GND	21	22	46		IDX	QEP2	14(in)	PRU0
						3.3V	23	24	44		A		14(out)	
CAN1	RX	SPI1	MOSI		GPIO	41	25	26	NRST	SYS				
CAN1	TX		MISO			40	27	28	116	GPIO	IDX	QEP0	6	PRU0
PRU	eCAP		CLK			7	29	30	113				3	
PRU1	16(in)		CS			19	31	32	112				2	
PRU0	15(out)	QEP2	B			45	33	34	115		B	QEP0	5	
PRU1	8	AIN 3.3V	5			86	35	36	7	AIN 1.8V				

Great getting started information is at beagleboard.org/pocket

Read a button

Goal: Sense the external world by reading a digital input.

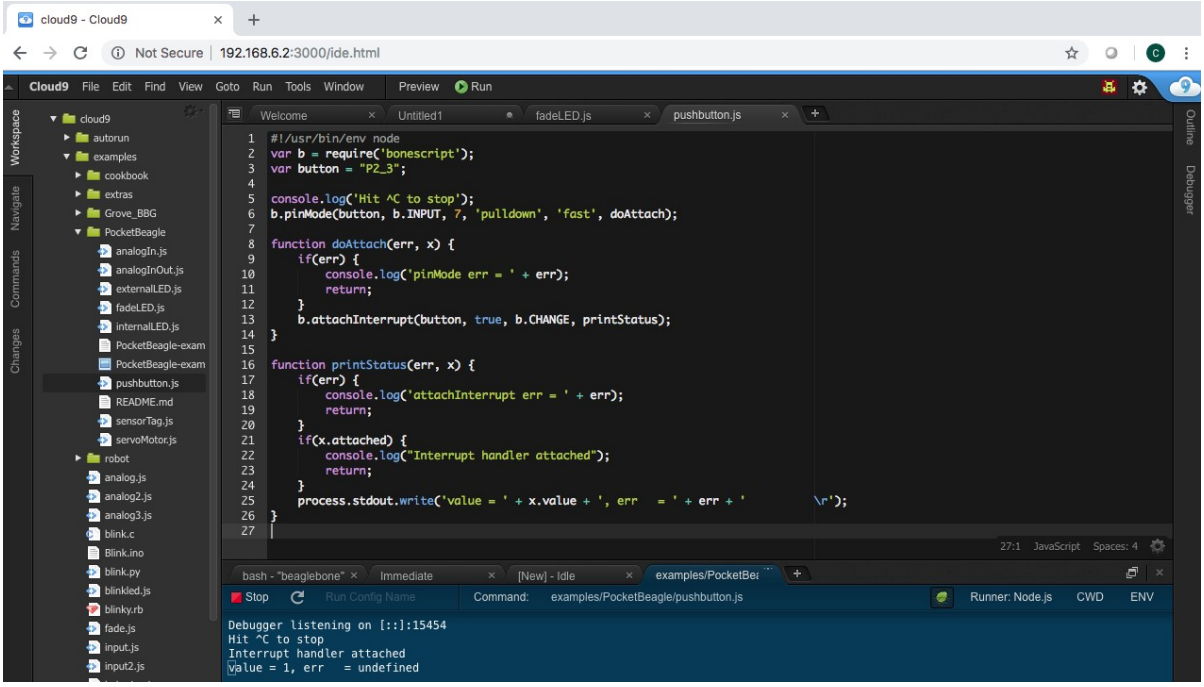
Overview: Reading a switch attached to a GPIO (general purpose input/output) port is as easy as configuring the port as an input and attaching an interrupt handler to it. Note the buttons are “active low”.

Do this:

1. Navigate to **examples/TechLab/pushbutton.js** and double-click on it.
2. Click the Run button in the toolbar to execute the script in the active file window
3. Press the “L” button on TechLab and check the output in the configuration window. Click the Stop button on the IDE to halt the program.

Challenge #1: Can you modify the program to read from the “R” button?

Challenge #2: Can you modify the program to toggle the USR3 LED?



pushbutton.js

```
#!/usr/bin/env node
var b = require('bonescript');
var button = "P2_33";

console.log('Hit ^C to stop');
b.pinMode(button, b.INPUT, 7, null, null, doAttach);

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

function printStatus(err, x) {
  if(err) {
    console.log('attachInterrupt err = ' + err);
    return;
  }
  if(x.attached) {
    console.log("Interrupt handler attached");
    return;
  }
  process.stdout.write('value = ' + x.value + '          \r');
}
```

Read an analog sensor

Goal: Sense the external world by reading a variable analog input

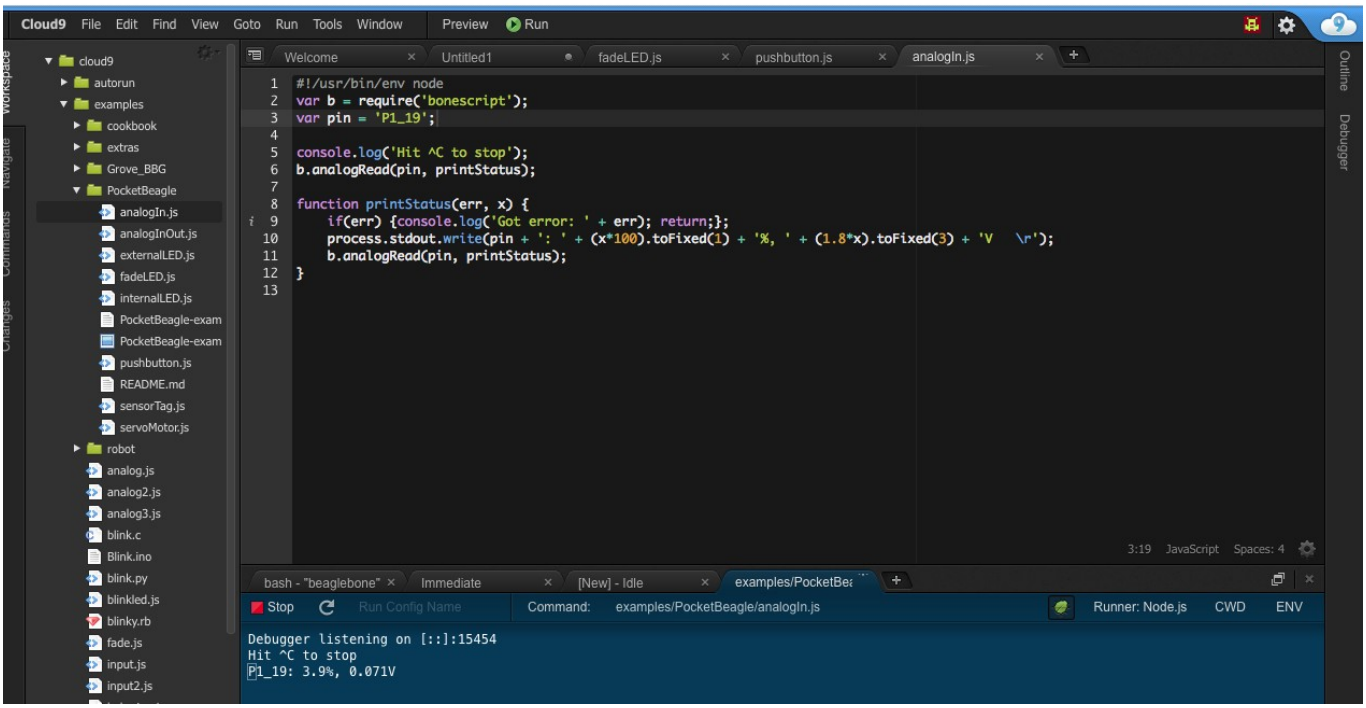
Overview: Reading a light sensor attached to an analog input pin.

Do this:

1. Navigate to examples/pocketbeagle/analogIn.js and double-click on it.
2. Click the Run button in the toolbar to execute the script in the active file window
3. Cover the light sensor and check the output in the configuration window. Click the Stop button to halt the program.

Challenge #1: Can you activate the USR3 LED based upon a voltage threshold from the light sensor?

Challenge #2: Try using the I2C accelerometer input from /sys/bus/iio/devices/iio:device1/in_accel_x_raw.



analogIn.js

```
#!/usr/bin/env node
var b = require('bonescript');
var pin = 'P1_19';

console.log('Hit ^C to stop');
doAnalogRead();

function printStatus(err, x) {
  if(err) {console.log('Got error: ' + err); return;};
  process.stdout.write(pin + ': ' + (x*100).toFixed(1) +
    '%, ' + (1.8*x).toFixed(3) + 'V   \r');
  setTimeout(doAnalogRead, 100);
}

function doAnalogRead() {
  b.analogRead(pin, printStatus);
}
```

Fade an LED

Goal: Utilize a hardware pulse-width-modulator (PWM) to light an LED with variable brightness

Overview: Linux provides LED drivers that understand how to utilize PWM drivers, making use of PocketBeagle's built-in PWM hardware. They are controlled with simple text files where you can set the brightness.

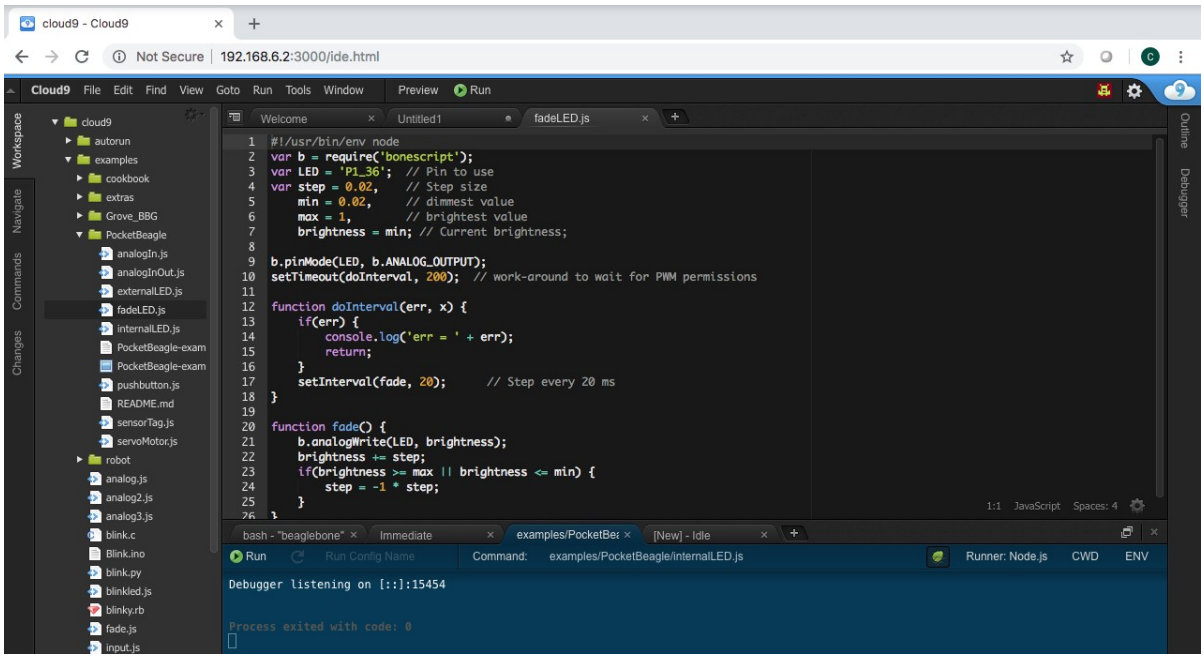
Do this:

1. Navigate to **examples/TechLab/fadeLED.js** and double-click on it.
2. Click the Run button in the toolbar to execute the script in the active file window
3. You will see the run configuration window open with a Stop button. Click the Stop button to halt the program.

Challenge #1: Try changing the fade interval, save the program and run again.

Challenge #2: Try using the light sensor input to set the LED brightness.

Challenge #3: Try using the I2C accelerometer input from /sys/bus/iio/devices/iio:device1/in_accel_x_raw.



fadeLED.js

```
#!/usr/bin/env node
var b = require('bonescript');
var LED = '/sys/class/leds/techlab::blue/brightness';
var step = 10,        // Step size
    min = 0,          // dimmest value
    max = 255,        // brightest value
    brightness = min; // Current brightness;

doInterval();

function doInterval(err, x) {
    if(err) {
        console.log('err = ' + err);
        return;
    }
    setInterval(fade, 20);       // Step every 20 ms
}

function fade() {
    b.writeTextFile(LED, brightness);
    brightness += step;
    if(brightness >= max || brightness <= min) {
        step = -1 * step;
    }
}
```

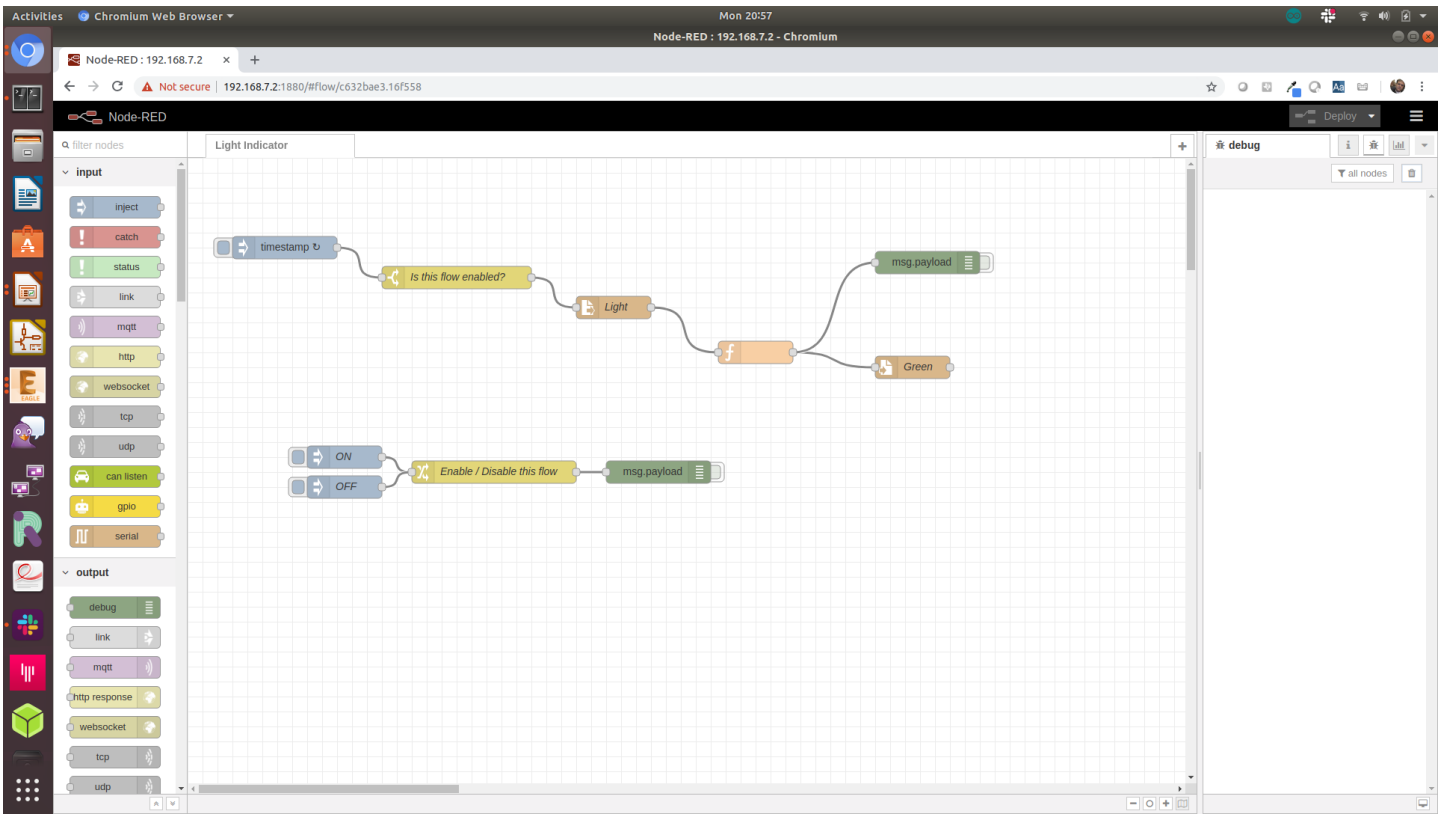
Using Node-RED to read and write files

Goal: Read light sensor data and output to green LED brightness

Overview: Node-RED is a flow-based development tool developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a browser-based flow editor, which can be used to create JavaScript functions. Linux turns devices into virtual files, making Node-RED well suited to interacting with the physical world.

Do this:

1. Open Node-RED by pointing your browser to **http://192.168.7.2:1880**



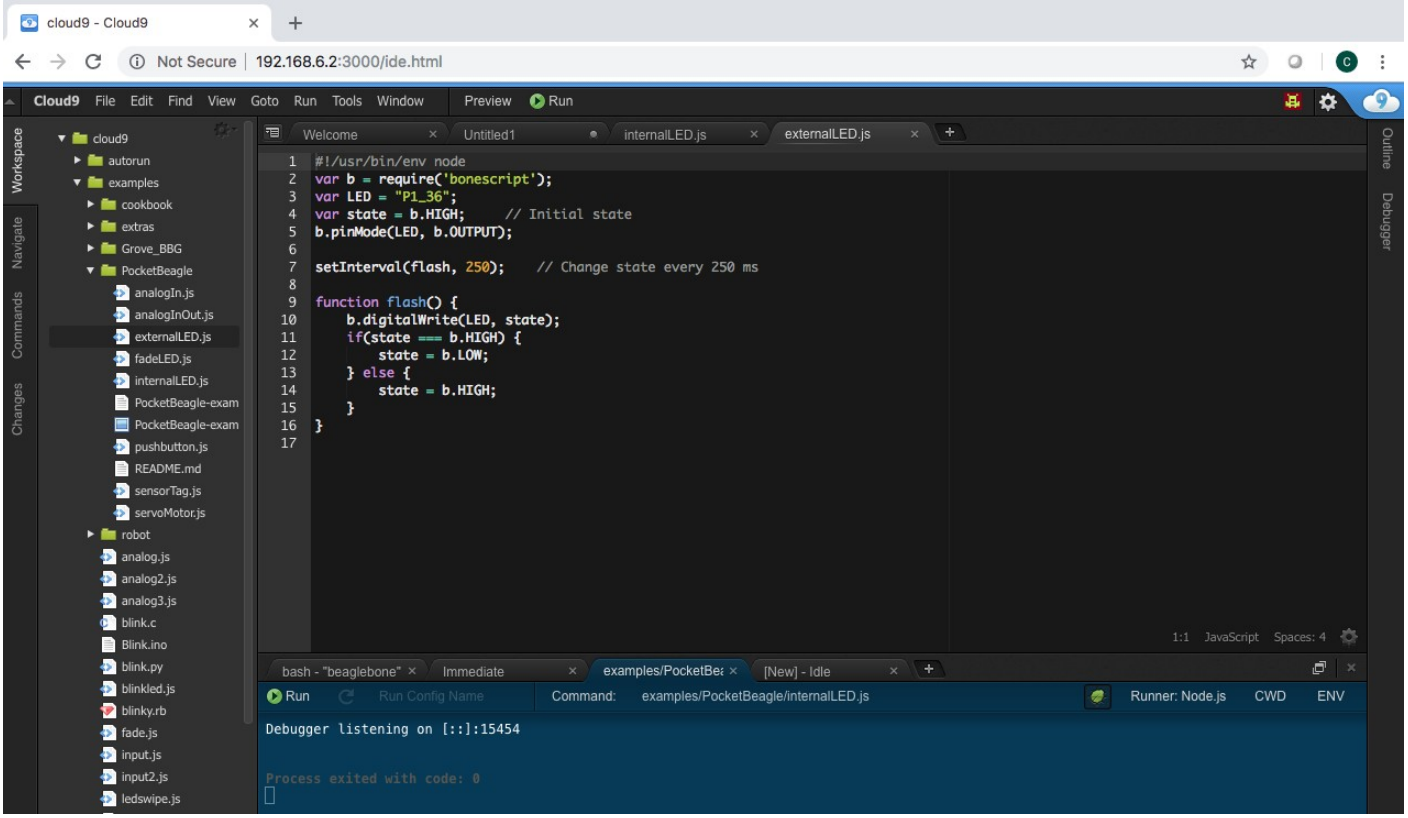
Explore the Linux command line

Goal: Blink an LED connected via breadboard to PocketBeagle

Overview: Similar to the internalLED.js here we will use Bonescript to blink an LED connected to a GPIO pin on PocketBeagle. The LED used in this exercise has the resistor built in simliar to the Adafruit “sequins”

Do this:

1. Open Cloud9 by pointing your browser to http://192.168.8.1:3000
2. Navigate to examples/pocketbeagle/externalLED.js and double-click on it.
3. Click the Run button in the toolbar to execute the script in the active file window
4. You will see the run configuration window open with a Stop button. Click the Stop button to halt the program.
5. Try changing the LED or blink time, save the program and run again



externalLED.js

```
#!/usr/bin/env node
var b = require('bonescript');
var LED = "P1_36";
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;
    }
}
```