Pre-Workshop Setup

Download these three files:

- https://bit.ly/2TySY5P - python-syntax.py
- https://bit.ly/2tbzIQt - presentation.pdf
- https://bit.ly/2GeRJG9 - smbus-doc.pdf

Power On Beaglebone Black Wireless

- Connect USB cable to board
- Connect other end of USB cable to PC

Connect to Beaglebone Black Wireless access point

- Wait ~4 minutes after your board has been powered on
- Connect to board's wifi access point
  - Wifi ssid is what you see on the board's label after ssid:
  - Wifi password is beaglebone_ and then you add the 3 characters after pw:

The above image means the ssid is bbw-108 and the password is beaglebone_jkr

Super Introduction to Embedded Linux Workshop

beaglebone

Franklin Cooper
University Marketing Manager

Quick Questionnaire

Who has used Python before?

Who has used a microcontroller before like Raspberry PI or Beaglebone?

What is Linux?

Remember Download these three files:

https://bit.ly/2TySY5P
https://bit.ly/2tbzIQt
https://bit.ly/2GeRJG9

Goals

Expose you to one of the most popular programming languages

Expose you to real world problems

Make you sweat

Have fun

Did you downloaded these three files?

https://bit.ly/2TySY5P https://bit.ly/2tbzIQt

https://bit.ly/2GeRJG9

Not Familiar with Python?

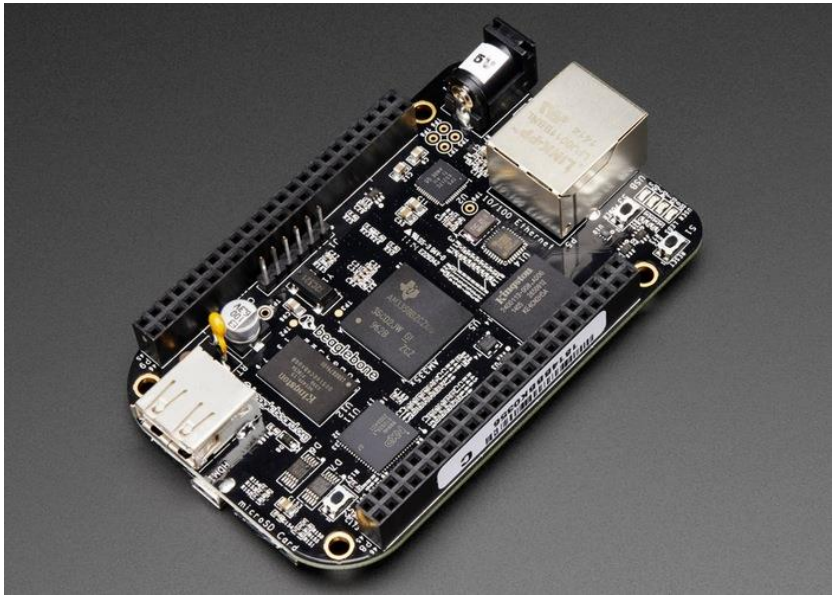
Look at python-syntax.py ← you downloaded this
Ask a neighbor
Google (on your phone)

I hope you downloaded these three files

https://bit.ly/2TySY5P https://bit.ly/2tbzIQt
https://bit.ly/2GeRJG9

Beaglebone Black Wireless Quick Summary

- Open Source Community Board
- Based on TI Technology
- Like the Raspberry Pi but better when interfacing with hardware
- Integrated Wifi
- Intergrated Bluetooth



Everything Is Running Off The Beaglebone

Router

Web Page

Python

IDE (Cloud 9)

Lets Get Started

Trouble Shooting

1. Make sure your board is plugged in to your computer via USB and is turned on
   Board powered on = Blinking blue leds on top right of board

2. Make sure your connected to the board via wifi
   Sometimes the connection drops and will reconnect to another wifi access point

3. Make sure the URL your using starts with http://192.168.7.2
   Always include http:// and don't use https://

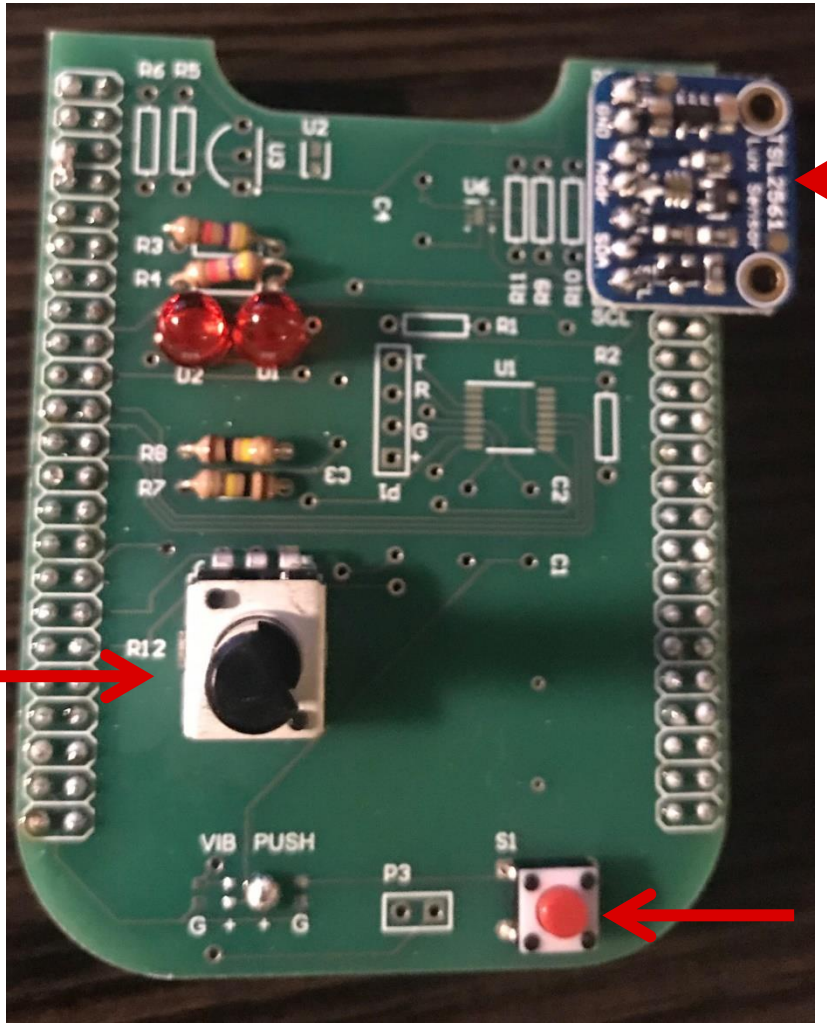
4. When using Cloud9 manually hit save File->Save or CTRL+S before running
   If you updated your program and the output hasn't changed. Then this is your problem

Play Beaglebone Based Bop It

Goal :

Play Bop It using Push Button, Potentiometer and Light Sensor

Beaglebone Workshop Cape



BLOCK IT!

↑

Light Sensor

Connected to:
P9_19
P9_20

TWIST IT!

↑

Potentiometer

Connected to:
P9_36

PUSH IT!

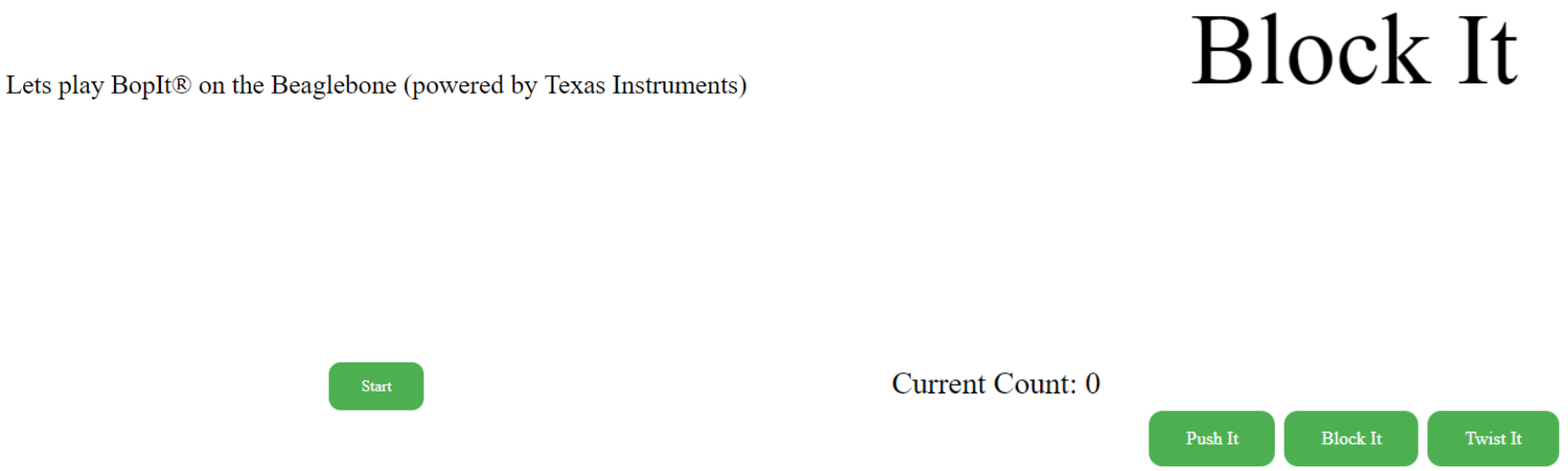
↑

Push Button

Connected to:
P9_29

Task 1 – Play TI Beaglebone BopIt

- Goto: http://192.168.7.2:8080/workshop/bopit/
- Turn down your computer's speaker (**IMPORTANT**)
- Hit Start
- You will see and/or hear a command
- Hit the correct button below that corresponds with the command



Cloud 9 – Quick Summary

Browser Base IDE

Supported Languages:

- Python
- C
- C++
- and much more

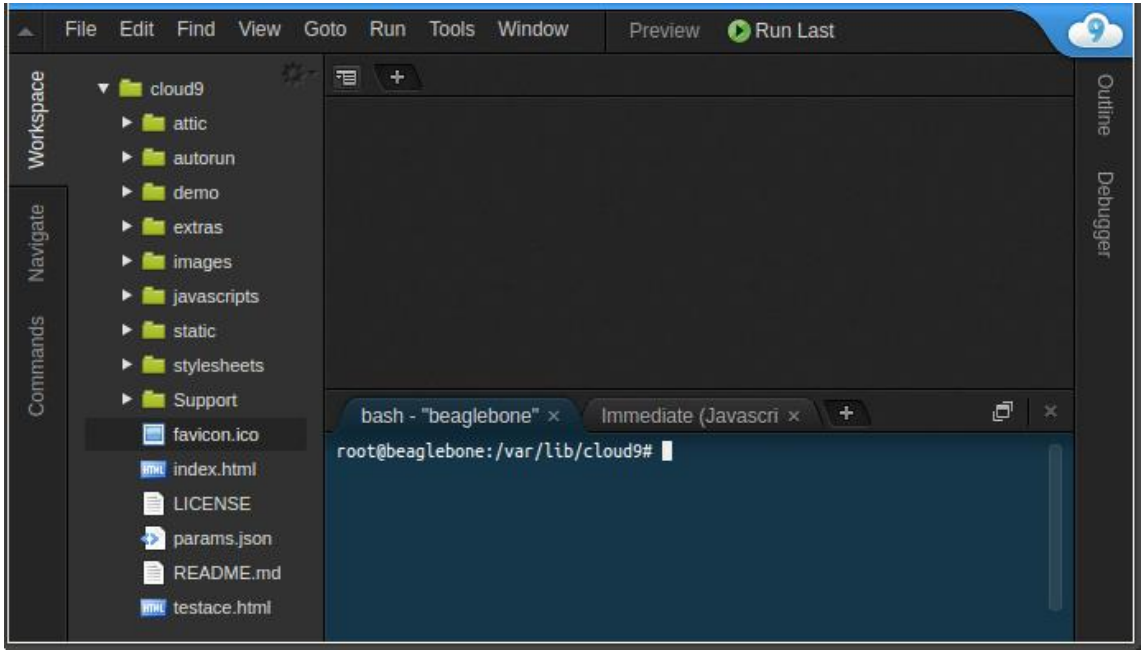
- Supports Debugging

Task 2 – Connect to Cloud 9 IDE

Open up a web browser (don't use IE)

- Go to http://192.168.7.2:3000/ (May take 2-3 minutes for page to load)
- You should see something like the below

* Type the above link as is. So you must include http:// and do not use https://

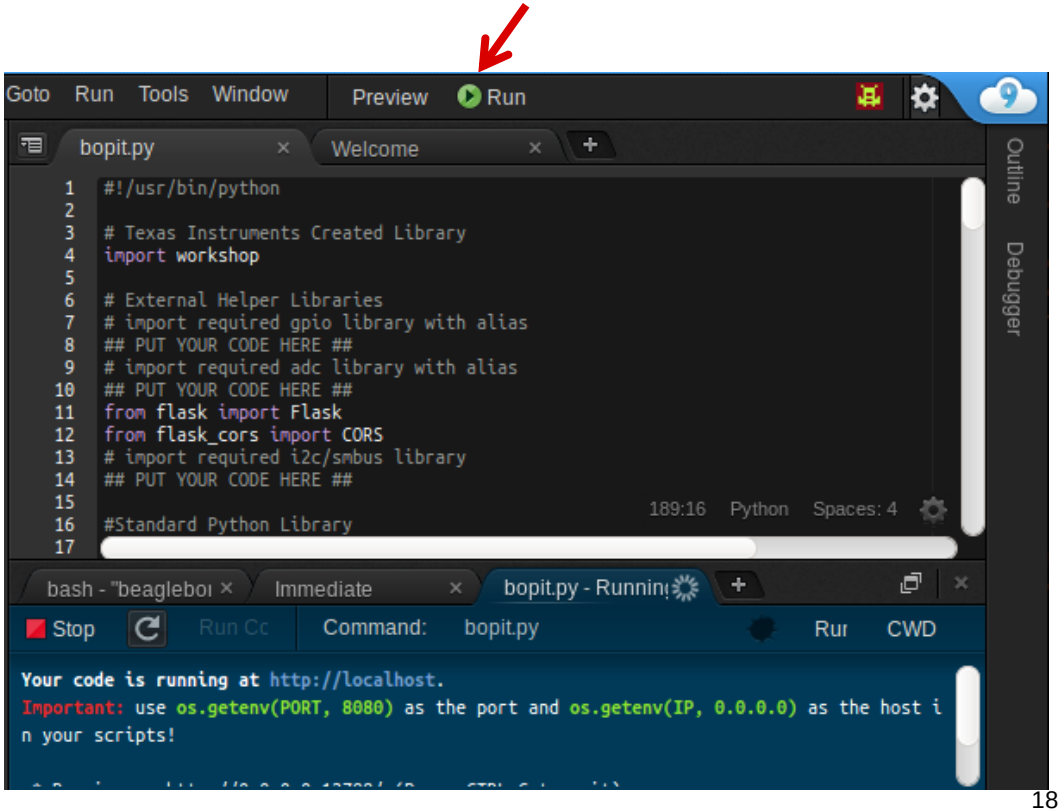


Cloud 9 – Running Program

Make sure the Python file you want to run is currently displayed.
Hit Run Button to “run your program”

* **Important Tip***

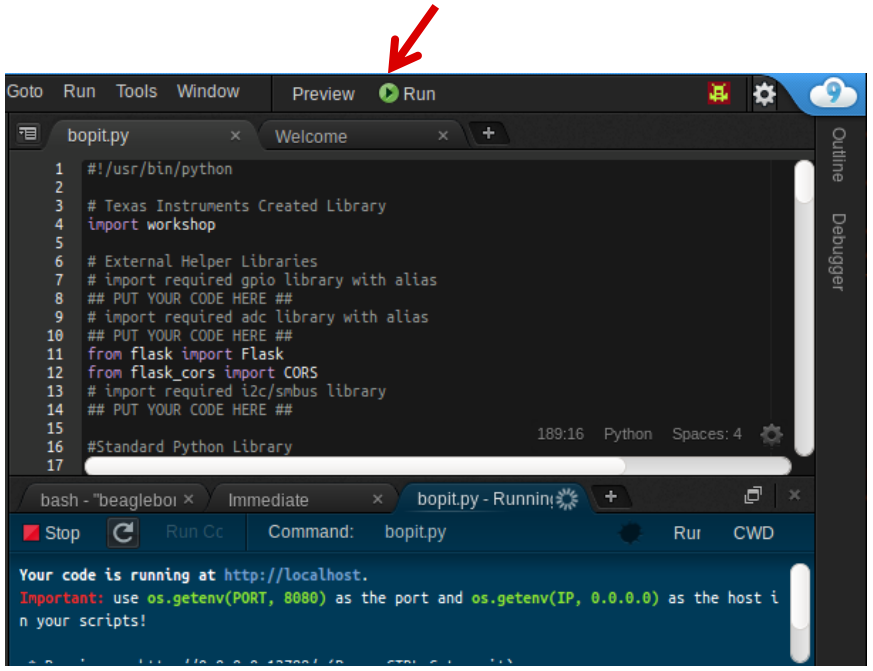
- Before running save your file first.



Task 3 – Run Bopit_board_test.py Program

Run the bopit_board_test.py program.

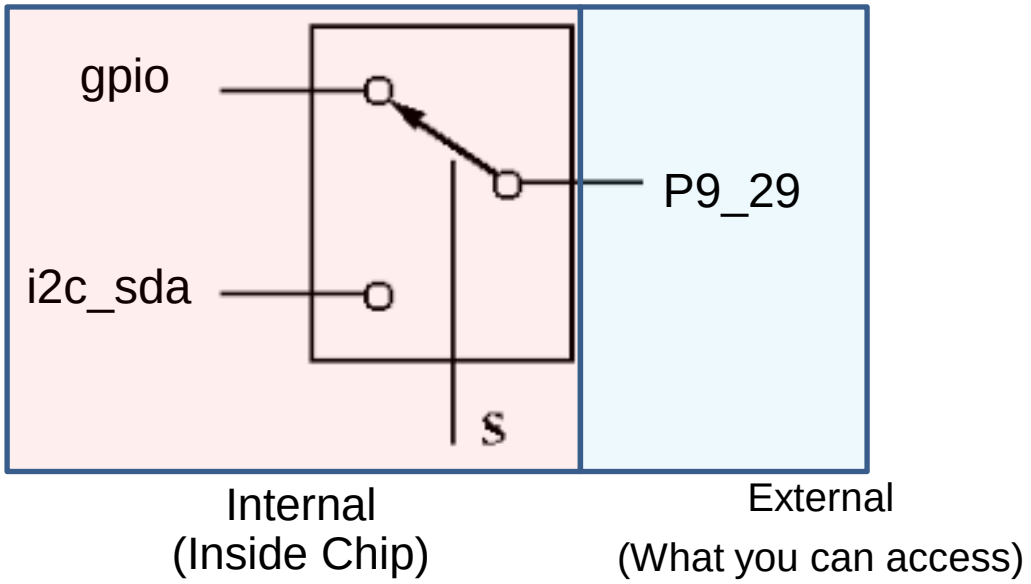
- This will help verify your board is in working order.
- Look at the console and follow the commands.
- If you get any failures then alert your instructor.



*When twisting make sure you turn it by a significant amount

Hardware – Pinmux

Pinmuxing is when a signal can be connected via a mux (switch) to various other signals



Pinmuxing is used on almost every signal pin on the Beaglebone

Software – Pinmux

To simplify configure a pin's pinmuxing you can use a program/script called config-pin

debian@beaglebone:~$ config-pin P9_29 gpio ← Set P9_29 to gpio

Python version (What you will be using)

workshop.setPinMux("P9_29","gpio")

Set P9_29 to gpio mode.

Modes you will use in this workshop:
gpio and i2c

Peripheral – GPIO: Basic Electrical Information

- Input -
  - High (1) : voltage >= 2 V
  - Low (0) : voltage <= .8 V
- Output
  - High (1) : 3.3 V
  - Low (0) : 0 V

65 possible digital I/Os

P9

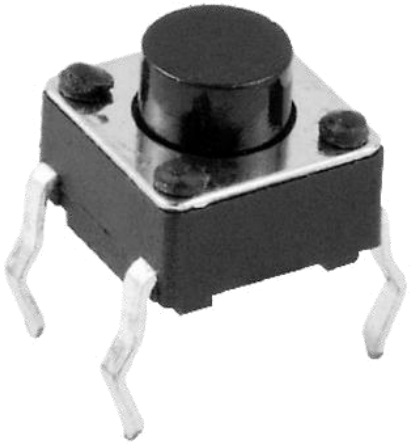
DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETN
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	GPIO_113
GPIO_111	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

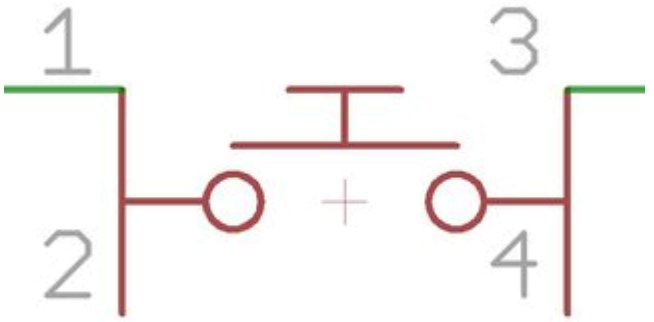
P8

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

Hardware - Push Button: Introduction

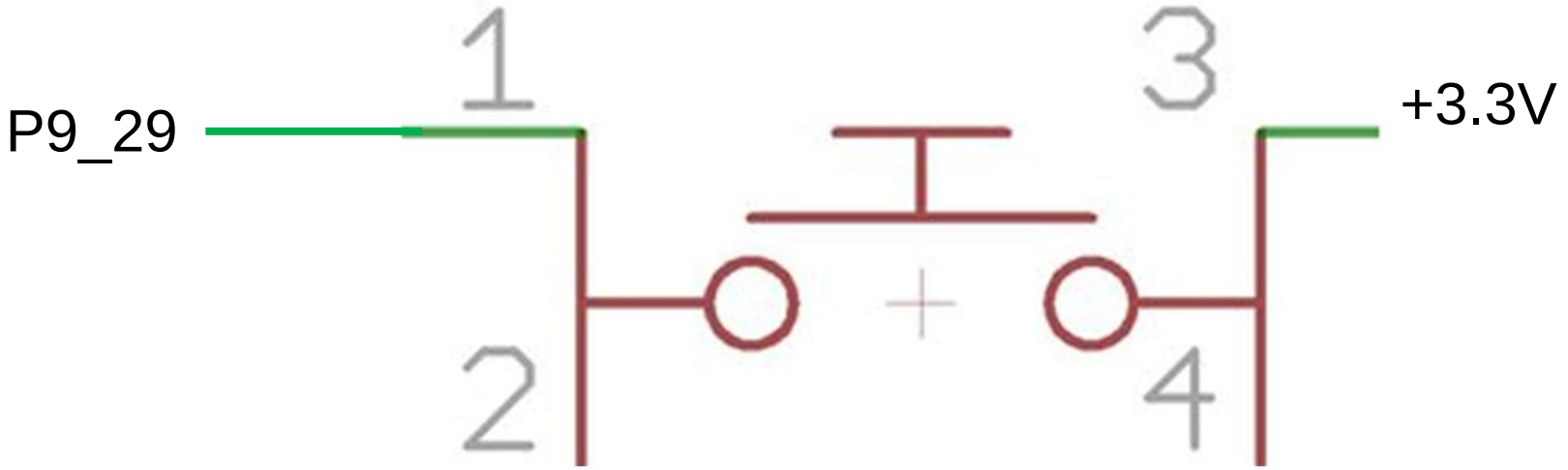
Basic Switch – by default pins 1 and 2 are not connected to pins 3 and 4.





Hardware – Push Button: Wiring

BBB Header Pin P9_29 is connected to one side of tactile switch. VCC is connected to other side of tactile switch.

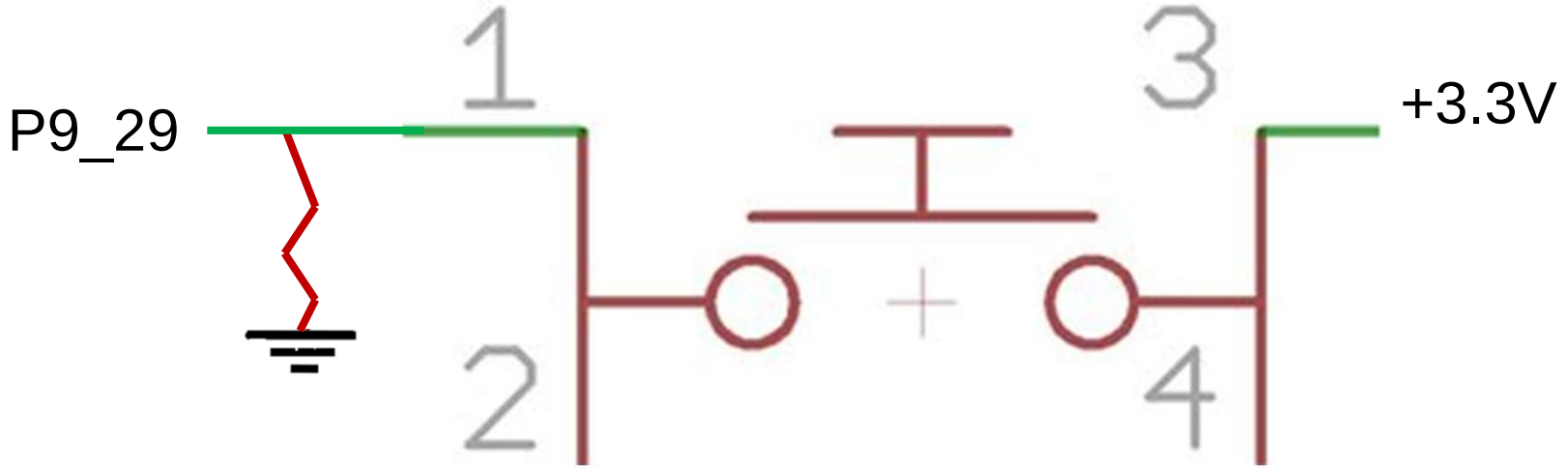


Quiz 1 - GPIO

What is the value of a input pin when nothing is driving it (outputting a voltage to it)?

Hardware – Pullup and Pull Down Resistors

- Typical Usage:
  - Provide a steady/default value
- BBB Pin Configuration:
  - Supports internal pullup and internal pulldown
  - Adafruit library allows you to configure internal pull up and pull down



Adafruit Library – Quick Introduction

Adafruit created a peripheral library for BBB:

- Python Based
- Simple to Use
- Lose some flexibility

Adafruit Library – GPIO (Input)

Including/Import

import Adafruit_BBIO.GPIO as GPIO

What does this mean?

Adafruit Library – GPIO (Input)

Including/Import

```
import Adafruit_BBIO.GPIO as GPIO
```

Library alias (reduce the amount of typing you do)

Adafruit Library – GPIO (Input)

Including/Import

import Adafruit_BBIO.GPIO as GPIO

Configuration:

GPIO.setup("<PIN_NAME>", GPIO.IN)

Configures GPIO as an input (pull up resistor by default)

GPIO.setup("<PIN_NAME>", GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

Configures GPIO as an input (pull down by default)

Usage:

GPIO.input("<PIN_NAME>")

Returns 1/True if voltage is high and returns 0/False if voltage is low

Adafruit Library – GPIO (Output)

Including/Import

import Adafruit_BBIO.GPIO as GPIO

Configuration:

GPIO.setup("<PIN_NAME>", GPIO.OUT)

Configure pin as an output

Usage:

GPIO.output("<PIN_NAME>", GPIO.HIGH)

Set pin to high

GPIO.output("<PIN_NAME>", GPIO.LOW)

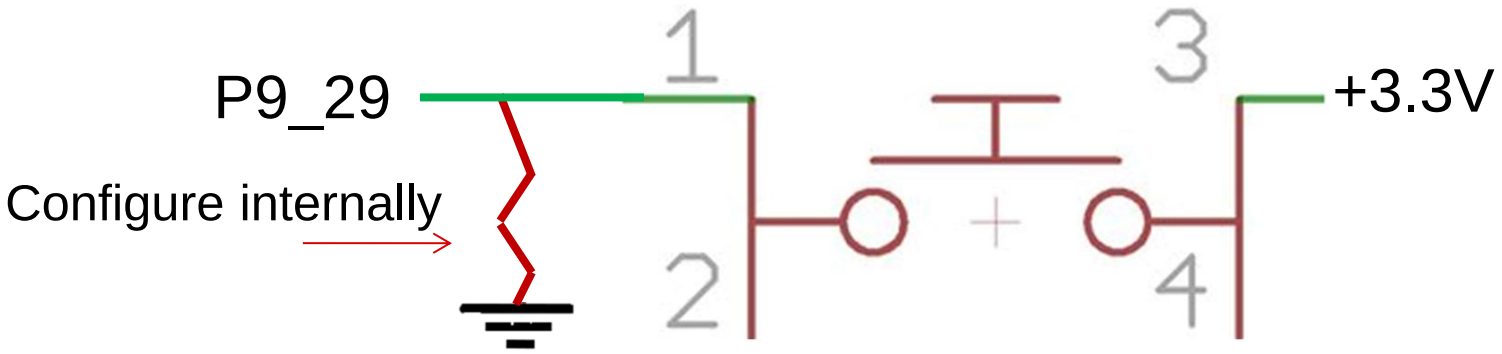
Set pin to low

Task 4 – Basic GPIO Program

Goal: print a message every time you push the button

1.Open gpio_input.py file that you previously loaded into Cloud 9.
2.Replace all the lines that says ## PUT YOUR CODE HERE ## with your code
3.Read the comment to understand what your being asked to do for the lines your writing code for
4.Use the information I provided in the previous slides to do this
5.Run the program and tap the button quickly to see if you achieve your goal

Remember:



Try:
Tap push button once as fast as possible and see what your program outputs

Task 5 – Basic GPIO Program (Solution)

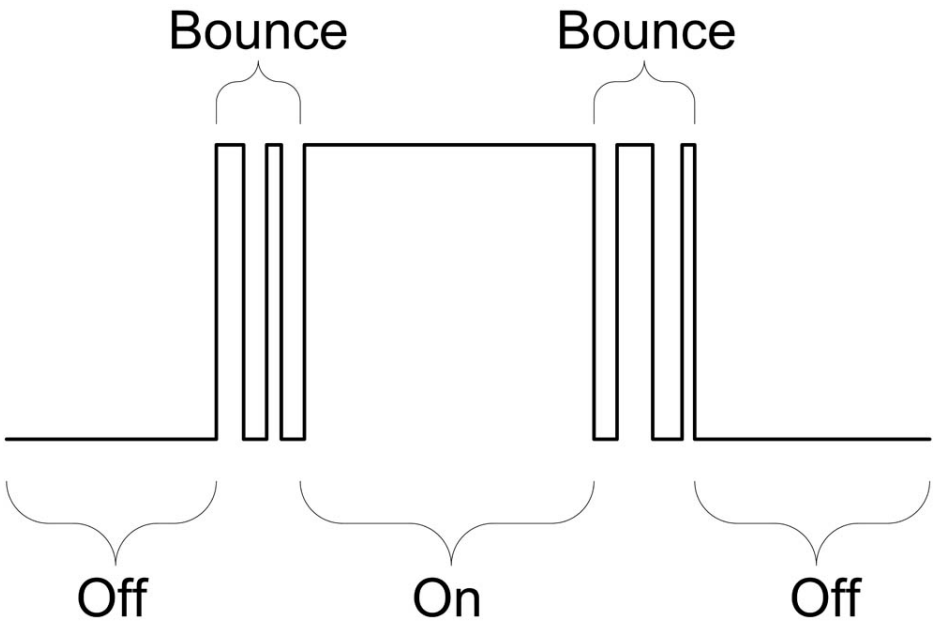
```
#!/usr/bin/python

# Texas Instruments Created Library
import workshop

# External Helper Libraries
#import Adafruit GPIO library with alias GPIO
import Adafruit_BBIO.GPIO as GPIO ## Solution

#Standard Python Library
import time

# Use workshop.setPinMux to set "P9_29" to "gpio" mode.
workshop.setPinMux("P9_29","gpio") ## Solution

# Call GPIO.setup and configure "P9_29" as an input with a pull down resistor
GPIO.setup("P9_29", GPIO.IN,pull_up_down=GPIO.PUD_DOWN) ## Solution


while True:
    # Print "Button Pressed" when the button has been pressed
    if GPIO.input("P9_29"): ## Solution
        print "Button Pressed" ## Solution
```

Hardware - Issue

Switch Bouncing:

- Occurs when pushing or release buttons/switch

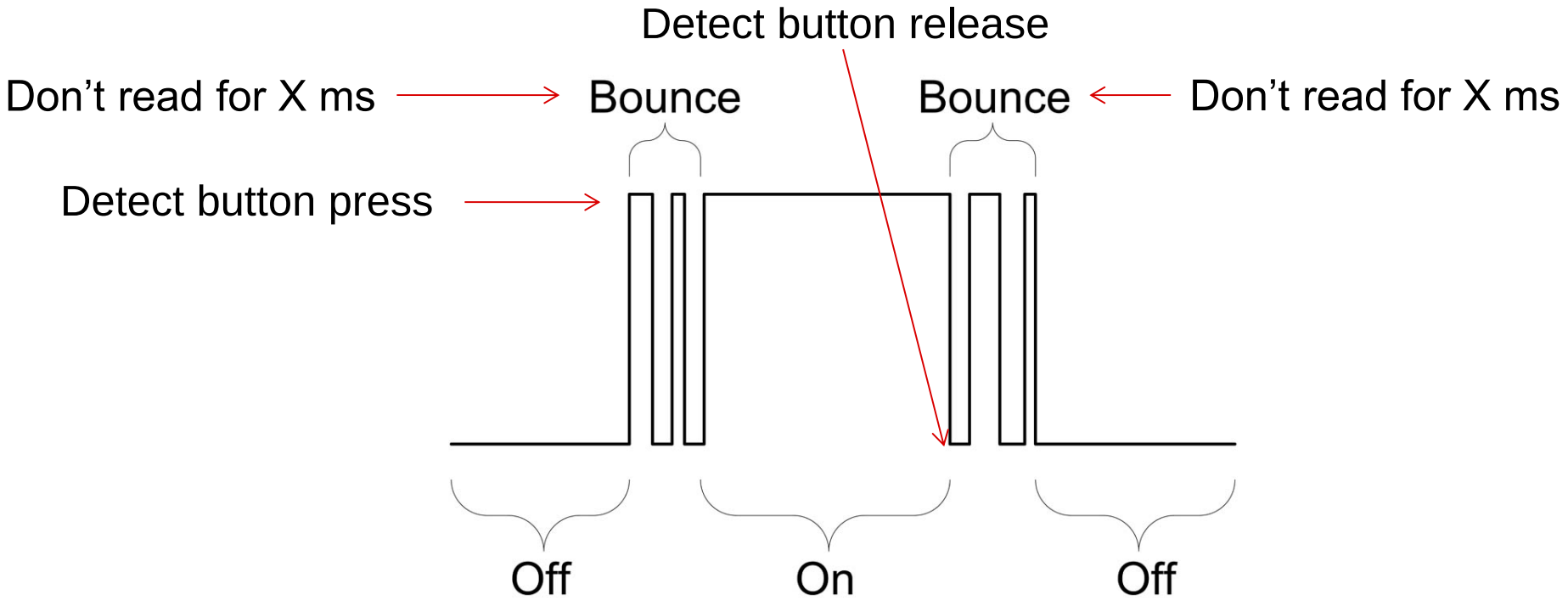
How do you solve it?



Hardware – Debounce Solution

Simple software solution:

- When state change ignore any future state changes for X ms

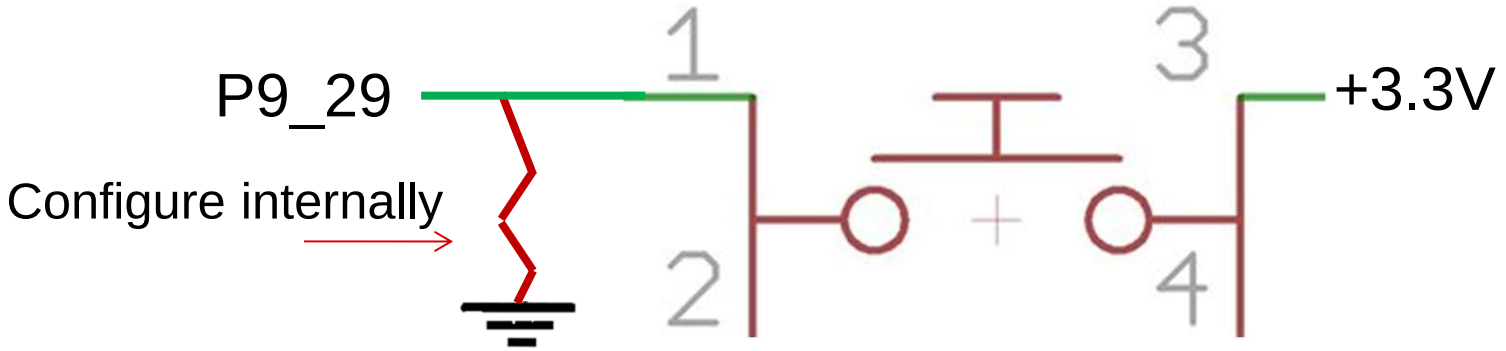


Task 6 – Debounce GPIO Program

Goal: print a message ONCE (ie debounce) every time you push the button

1.Open gpio_input_debounce.py file that you previously loaded into Cloud 9.
2.Replace all the lines that says ## PUT YOUR CODE HERE ## with your code
3.Read the comment to understand what your being asked to do for the lines your writing code for
4.Use the information I provided in the previous slides to do this
5.Run the program and tap the button quickly to see if you achieve your goal

Remember:



Try:
Feel free to change the debounce timeout value to see how it impacts the program

Task 6 – Debounce GPIO Program Solution

```
#!/usr/bin/python

import workshop
import Adafruit_BBIO.GPIO as GPIO ## Solution

import time

def getTimeMS():
    return (int(round(time.time() * 1000)))

# Set pin P9_29 pinmux to gpio
workshop.setPinMux("P9_29","gpio") ## Solution

# Configure P9_29 as a gpio input with an internal pull down
GPIO.setup("P9_29", GPIO.IN,pull_up_down=GPIO.PUD_DOWN) ## Solution


state = 0
# How many ms to wait before reading gpio again
gpio_debounce = 100

# Time last read the gpio
last_read_gpio = 0

while True:
    current_time = getTimeMS()
    time_delta = current_time - last_read_gpio

    # Read pin P9_29 and set variable "btnVal" to the pins value
    btnVal = GPIO.input("P9_29") ## Solution

    if state == 0: ## Solution
        if btnVal == 1: ## Solution
            print "Button pressed"  ## Solution
            state = 1  ## Solution
            last_read_gpio = current_time  ## Solution

    if state == 1: ## Solution
        if btnVal == 0: ## Solution
            state = 2 ## Solution

    if state == 2:
        if time_delta > gpio_debounce: ## Solution
            state = 0 ## Solution
```

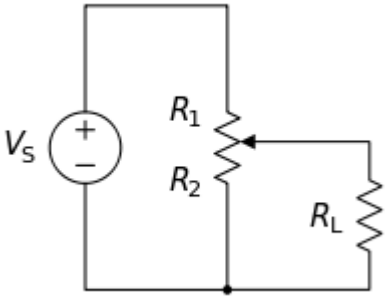
Task 7 – Update Bopit to use Push Button

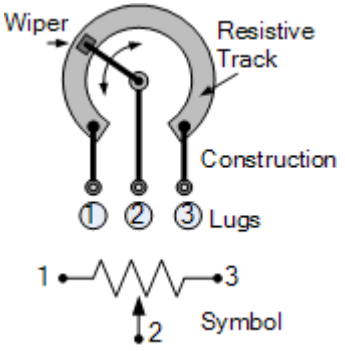
1.If bopit.py is running make sure you stop it.
2.Replace all the lines that says “## PUT YOUR CODE HERE ##” within the functions “gpioSetup()” and getPushButtonVal()”.
3.Run bopit.py program
4.Go to http://192.168.7.2:8080/workshop/bopit/

What to expect:
Push It – Push the push button on the green board
Twist It – Click the button with the mouse on web page
Block It – Click the button with the mouse on web page

Hardware: Potentiometer

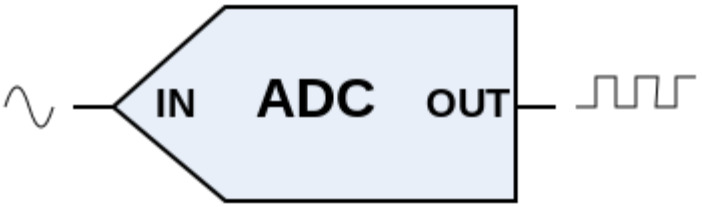
- What is it:
  - Variable Resistor
  - Increase/decrease resistance via knob
  - Can be used as a voltage divider
    - Middle pin is output of divider





Peripheral – Analog to Digital Converter (ADC)

- Purpose
  - Converts analog voltage to digital value
  - Useful for sensors that output voltage only
- Beaglebone Black's ADC is a 12 bit ADC:
  - Digital Value Range
    - 0 – 4095
  - Max support input voltage:
    - 1.8 V



Adafruit Library – ADC

Including/Import

import Adafruit_BBIO.ADC as ADC

Configuration:

ADC.setup()

Usage:

ADC.read(“<PIN_NAME>”)

Returns a decimal value from 0.0 – 1.0

ADC.read_raw(“<PIN_NAME>”)

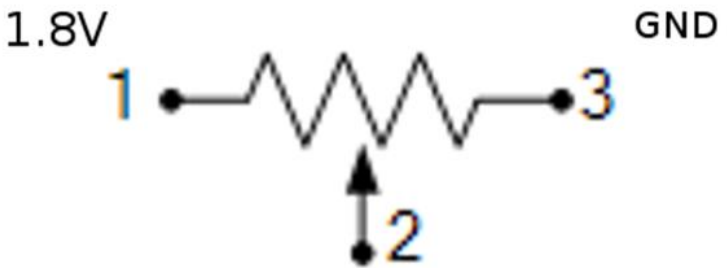
Returns a digital value from 0 - 4095

Task 8 – Read Potentiometer

Goal: Print the voltage being read by the potentiometer

1.Open adc.py file that you previously loaded into Cloud 9.
2.Replace all the lines that says ## PUT YOUR CODE HERE ## with your code
3.Read the comment to understand what your being asked to do for the lines your writing code for
4.Use the information I provided in the previous slides to do this
5.Run the program and tap the button quickly to see if you achieve your goal

Remember:



P9_36

Task 8 – Read Potentiometer Solution

```
#!/usr/bin/python

# Texas Instruments Created Library
import workshop

# External Helper Libraries
#import Adafruit ADC library with alias
import Adafruit_BBIO.ADC as ADC

#Standard Python Library
import time


# Call ADC Setup (Must be done only once)
ADC.setup() ## Solution

while True:
    # Read ADC value from pin (P9_36) and set retrieved value to variable called "val"
    val = ADC.read("P9_36") ## Solution

    # Print value of "val" variable
    print "ADC: ",val ## Solution

    # Convert digital value stored in "val" variable to analog (voltage) value.
    # Store the converted value back into "val" variable.
    # Important note: Max analog value possible is 1.8V
    val = val * 1.8 ## Solution

    # Print value of "val" variable (should contain analog value from (0 - 1.8V)
    print "Voltage: ",val ## Solution

    time.sleep(1)
```

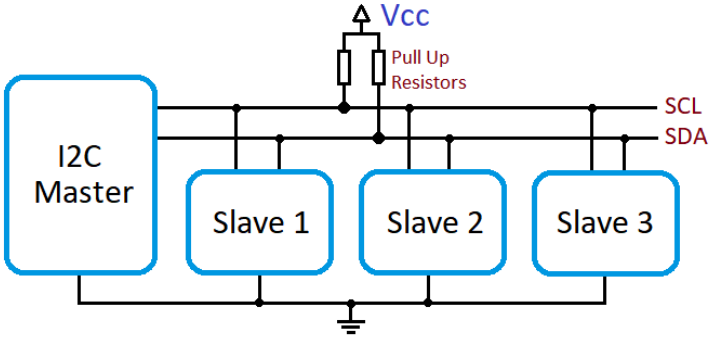
Task 9 – Update Bopit to use Potentiometer

1.If bopit.py is running make sure you stop it.
2.Replace all the lines that says “## PUT YOUR CODE HERE ##” within “adcSetup()” and “readPotentiometerVal()” function.
3.Go to http://192.168.7.2:8080/workshop/bopit/

What to expect:
Push It – Push the push button on the green board
Twist It – Turn (a lot) the potentiometer on the green board
Block It – Click the button with the mouse on web page

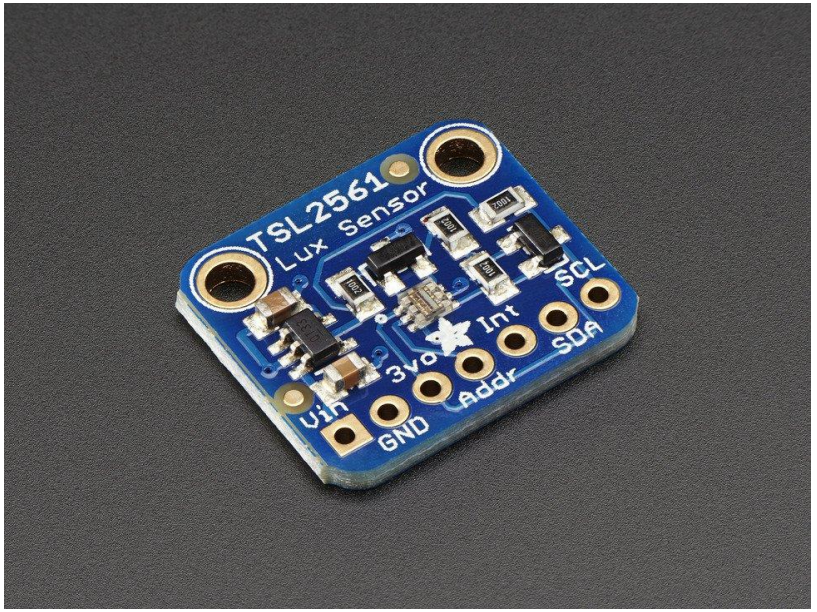
Peripheral – I2C Quick Summary

- Quick Summary
  - Uses 2 Pins
    - SCL – Clock
    - SDA – Data
  - Half Duplex
  - Speed “Typical” Ranges 1K to 400K
  - Supports Multiple Nodes
  - Communicate to specific node via unique address
  - Multi Master



Hardware – Light Intensity Sensor

- Summary:
  - Default address 0x29
  - Requires Configuration
  - Has defined sample rate
  - Returns ambient light
  - levels



Task 10 – I2C Light Intensity

Goal: print light intensity every second

1.Open i2c.py file that you previously loaded into Cloud 9.
2.Replace all the lines that says ## PUT YOUR CODE HERE ## with your code
3.Update the code based on the comments that explain what is expected.
4.Use the information I provided in the previous slides to do this as well as the smbus pdf included in the bopit.zip you extracted earlier.
5.Run the program and tap the button quickly to see if you achieve your goal

Remember/Hint:

- Device is on bus 2 (not 0 which the pdf uses)
- Device address is 0x29

Try:
Feel free to change the debounce timeout value to see how it impacts the program

Task 10 – I2C Light Intensity Solution

```
#!/usr/bin/python

# Texas Instruments Created Library
import workshop
import smbus  ## Solution

import time

def convertToU16(value,little_endian=True):
    value = value & 0xFFFF

    if not little_endian:
        value = ((value << 8) & 0xFF00) + (value >> 8)
    return value

workshop.setPinMux("P9_19","i2c") ## Solution
workshop.setPinMux("P9_20","i2c") ## Solution

bus = smbus.SMBus(2) ## Solution

TSL2561_I2C_ADDR          = (0x29)    # Default address (pin left floating)
TSL2561_COMMAND_BIT         = (0x80)    # Must be 1
TSL2561_WORD_BIT            = (0x20)    # 1 = read/write word (rather than byte)
TSL2561_CONTROL_POWERON     = (0x03)
TSL2561_CONTROL_POWEROFF    = (0x00)
TSL2561_REGISTER_CONTROL    = 0x00
TSL2561_DELAY_INTTIME_13MS  = (15) / 1000
TSL2561_REGISTER_CHAN0_LOW  = 0x0C

bus.write_byte_data(TSL2561_I2C_ADDR,TSL2561_COMMAND_BIT | TSL2561_REGISTER_CONTROL, TSL2561_CONTROL_POWEROFF) ## Solution

time.sleep(1)

bus.write_byte_data(TSL2561_I2C_ADDR, TSL2561_COMMAND_BIT | TSL2561_REGISTER_CONTROL, TSL2561_CONTROL_POWERON) ## Solution

val = 0
while True:

    val = bus.read_word_data(TSL2561_I2C_ADDR,TSL2561_COMMAND_BIT | TSL2561_WORD_BIT | TSL2561_REGISTER_CHAN0_LOW) ## Solution
    val = convertToU16(val)
    time.sleep(TSL2561_DELAY_INTTIME_13MS) ## Solution
    print val
    time.sleep(1)
```

Task 11 – Update Bopit to use Light Sensor

1.If bopit.py is running make sure you stop it.
2.Replace all the lines that says “## PUT YOUR CODE HERE ##” with your code within functions “i2cSetup()” and “getLightValue()”.
3.Run bopit.py program
4.Go to http://192.168.7.2:8080/workshop/bopit/

What to expect:
Push It – Push the push button on the green board
Twist It – Turn (a lot) the potentiometer on the green board
Block It – Cover and uncover light sensor on the green board