Beaglebone Black & Python Scripts:

1. Update the beaglebone black Angstrom/Ubuntu Snappy system:
Make sure the ETH is connected to the BBB system. Angstrom comes pre-installed with the dropbear ssh package by default, instead of the more common openssh package. The easiest way to generate ssh keys is to simply install the openssh-keygen client.

Angstrom:
$ opkg update
$ opkg install openssh-keygen
OR
Ubuntu/Debian:
$ apt-get update
$ apt-get install openssh-keygen

2. Update the date and time:
$ /usr/bin/ntpdate -b -s -u pool.ntp.org

3. SSH to BeagleBone Black over USB
A. Install BBB USB Driver:
For Windows:
On Windows 64 bit, install this:
http://beagleboard.org/static/Drivers/Windows/BONE_D64.exe
Windows 32 bit,
http://beagleboard.org/static/Drivers/Windows/BONE_DRV.exe
For linux:
http://beagleboard.org/static/Drivers/Linux/FTDI/mkudevrule.sh
For MAC:
http://beagleboard.org/static/Drivers/MacOSX/RNDIS/HoRNDIS-rel4.pkg
and the serial driver which is here:
http://beagleboard.org/static/Drivers/MacOSX/FTDI/FTDI_Ser.dmg

B. Check if BBB is alive and get the IP address by trying to connect to the BBB with a browser. In the browser type http://192.168.Y.X, where Y is your subnet and X is your device.
C. Having downloaded and installed Putty (TIVACWARE labs!), run the program. Enter the IP address 192.168.Y.X and click 'Connect'. Make sure your port is 22. Acknowledge the warning. Log in with a username of 'root' and no password (just hit return). Now that you have command-

line access to your BBB, you can browse the file system, install software, monitor processes etc, just as you would with the command line on any Linux computer.

4. Setting up IO Python Library on BeagleBone Black in Windows:
Connecting to your BeagleBone Black (SSH) as above. You can also use http://beaglebone.local instead of http://192.168.Y.X.
Commands to setup and install Adafruit_BBIO
$ opkg update && opkg install python-pip python-setuptools python-smbus
$ pip install Adafruit_BBIO

5. Test your Installation
$ python -c "import Adafruit_BBIO.GPIO as GPIO; print GPIO"
you should see this or similar:
<module 'Adafruit_BBIO.GPIO' from '/usr/local/lib/python2.7/dist-packages/Adafruit_BBIO/GPIO.so'>

6 Manual Install: (skip if 4 and 5 work)
You can also install Adafruit_BBIO by cloning the git repository. The following commands should get it installed as well:
$ git clone git://github.com/adafruit/adafruit-beaglebone-io-python.git
#set the date and time
$ /usr/bin/ntpdate -b -s -u pool.ntp.org #install dependency
$ opkg update && opkg install python-distutils python-smbus cd adafruit-beaglebone-io-python
$ python setup.py install

7. Setting up IO Python Library on BeagleBone Black in Linux:
Install IO Python Lib:
$ sudo apt-get update
$ sudo apt-get install build-essential python-dev python-setuptools python-pip python-smbus -y
$ sudo pip install Adafruit_BBIO
Test:
$ sudo python -c "import Adafruit_BBIO.GPIO as GPIO; print GPIO"
you should see this or similar:
<module 'Adafruit_BBIO.GPIO' from '/usr/local/lib/python2.7/dist-packages/Adafruit_BBIO/GPIO.so'>

8. GPIO Example with python script:
1. LED Blink/GPIO:
Connection:
The top two connections on the BBB expansion header we are using (P8) are both GND. The other lead is connected to pin 10, which is the right-hand connector on the fifth row down. The pins are numbered left to right, 1, 2 then on the next row down 3,4 etc.
Launch the Python Console in BBB by typing:
$ python
————————Output————————————
Python2.7.3(default,Apr 32013,21:37:23)
[GCC 4.7.3 20130205 (prerelease)] on linux2
Type "help", "copyright", "credits" or "license" for more information.

$exit()

Open the nano editor in BBB:
$ nano blink.py

Type the following code in the editor:
```
import Adafruit_BBIO.GPIO as GPIO import time
GPIO.setup("P8_10", GPIO.OUT)
while True:
GPIO.output("P8_10", GPIO.HIGH)
time.sleep(0.5)
GPIO.output("P8_10", GPIO.LOW)
time.sleep(0.5)
```

Save and exit the editor using CTRL-x and the Y to confirm. To start the program, led if connected to P8 will blink when you enter the command:
$ python blink.py
When you want to stop the blinking, use CTRL-c to exit the program.

2. Push Button Input & LED Blink/GPIO:
Connection:
The blue lead is connected from this GND (0V) connection to one end of the resistor. The red lead is connected to pin 3 of the other connector (3.3V) and the orange lead to pin 12 (P8.12), which is the right-hand connector on the sixth row down. The resistor 'pulls down the input pin (P8.12) so that it is at 0V (GND) until the push button is pressed, at which

point it will be at 3.3V.

Enter the following command to create a new files called switch.py
$ nano switch.py

```
import Adafruit_BBIO.GPIO as GPIO import time
GPIO.setup("P8_12", GPIO.IN)
old_switch_state = 0
while True:
new_switch_state = GPIO.input("P8_12")
if new_switch_state == 1 and old_switch_state == 0 :
print('Do not press this button again!')
time.sleep(0.1)
old_switch_state = new_switch_state
```

To start the program, enter the command:
$ python switch.py

Each time you press the button, you should see a message.
$ python switch.py
Do not press this button again!
Do not press this button again!

When you want to stop the program, use CTRL-c.

Change the code to turnON and OFF the LED every time the button is pressed.

3. PWM and Servo Motor Control:
Connect an 5V 100mA power supply to the servo motor. Or use a open USB to power the servo motor. Only the PWM signal is feed directly to the servo motor.
We will use pin P8_13 as the PWM output to control the servo. The only other connection that we need from the BBB is GND.

Enter the following command to create a new files called servo.py
$ nano servo.py

Now paste the code below into the editor window.

```
import Adafruit_BBIO.PWM as PWM
```

```
servo_pin = "P8_13"
duty_min = 3
duty_max = 14.5
duty_span = duty_max - duty_min
PWM.start(servo_pin, (100-duty_min), 60.0)
while True:
        angle = raw_input("Angle (0 to 180 x to exit):")
        if angle == 'x':
                PWM.stop(servo_pin)
                PWM.c leanup() break
        angle_f = float(angle)
        duty = 100 - ((angle_f / 180) * duty_span + duty_min)
        PWM.set_duty_cycle(servo_pin, duty)
```

To start the program, enter the command:

```
$ python servo.py
Angle (0 to 180 x to exit):90
Angle (0 to 180 x to exit):180
Angle (0 to 180 x to exit):0
Angle (0 to 180 x to exit):x
$
```

Entering a value between 0 and 180 will set the servo's angle accordingly. When you want to stop the program, enter 'x'.

4. ADC using python script:

The red lead is connected from pin 3 of the other connector (3.3V) to the positive supply pin of the TMP36 and the orange lead to pin P9.40.

Enter the following command to create a new files called adc.py

```
$ nano adc.py
```

Now paste the code below into the editor window.

```
import Adafruit_BBIO.ADC as ADC import time
sensor_pin = 'P9_40'
ADC.setup()
while True:
        print('mv=%d C=%d F=%d' % (millivolts))
        time.sleep(1)
```

Save and exit the editor using CTRL-x and the Y to confirm.

To start the program, enter the command:
$ python tmp36.py

You will then see a series of readings when the pot is adjusted.
0.389999985694885 25
0.388333320617675 78
0.397222220897674 56
0.425000011920928 96
Warning: The analog inputs of the BBB operate at 1.8V. There is a potential for the BBB to be damaged if the voltage in millivolts exceeds 1.8V. Always choose the value of the resistor and pot equal.