

Embedded System Design Based on Beaglebone Black with Embedded Linux

Prof. Omer Farook, Purdue University Calumet (College of Technology)

Omer Farook is a member of the faculty of Electrical and Computer Engineering Technology at Purdue University, Calumet. Farook received the diploma of licentiate in mechanical engineering and B.S.M.E. in 1970 and 1972, respectively. He further received B.S.E.E. and M.S.E.E. in 1978 and 1983, respectively, from Illinois Institute of Technology. Farook's current interests are in the areas of embedded system design, hardware-software interfacing, digital communication, networking, image processing, and biometrics, C++, Python, PHP and Java languages. He has a keen interest in pedagogy and instruction delivery methods related to distance learning.

Dr. Jai P Agrawal, Purdue University, Calumet (College of Technology)

Jai P. Agrawal is a professor in electrical and computer engineering technology at Purdue University, Calumet. He received his Ph.D. in electrical engineering from University of Illinois, Chicago, in 1991, dissertation in power electronics. He also received M.S. and B.S. degrees in electrical engineering from Indian Institute of Technology, Kanpur, India, in 1970 and 1968, respectively. His expertise includes analog and digital electronics design, power electronics, nanophotonics, and optical/wireless networking systems. He has designed several models of high frequency oscilloscopes and other electronic test and measuring instruments as an entrepreneur. He has delivered invited short courses in Penang, Malaysia and Singapore. He is also the author of a textbook in power electronics, published by Prentice-Hall, Inc. His professional career is equally divided in academia and industry. He has authored several research papers in IEEE journals and conferences. His current research is focused on renewable energy technology, smart energy grid.

Prof. Ashfaq Ahmed, Purdue University Calumet (College of Technology)

Ashfaq Ahmed is a professor in the Electrical and Computer Engineering Technology Department at Purdue University Calumet. Ahmed received his Bachelor's of Science degree in Electrical Engineering from the University of Karachi in 1973 and Master's of Applied Science degree in 1978 from University of Waterloo. He is the author of a textbook in Power Electronics, published by Prentice-Hall. He is a registered Professional Engineer in the State of Indiana. He is senior member of IEEE. Ahmed's current interests include Embedded System Design, IoT, VHDL design AND Smart Grid.

Dr. Athula Kulatunga, Purdue University, West Lafayette

N. Athula Kulatunga, Ph.D., CEM is the Department Head of Engineering Technology at Purdue University Northwest in Hammond, Indiana, USA. Before becoming the head, he was the George W McNelly Professor in Electrical and Computer Engineering Technology at Purdue University, West Lafayette, Indiana, USA. He received a Ph.d. from Purdue University in 1995. He is the founder and director of two industry sponsored applied research labs: Power Electronics Development and Applications Lab (PEDAL) and Smart Meter Integration Lab (SMIL). He is the Principal Investigator of one of 10 Global Innovation projects funded by the US department of State, Rapid, Smart Grid Impact RSGI, partnering with DeMontfort University in Leicester, UK, and UNESP in Sao Paulo, Brazil. He has been a Certified Energy Manager (CEM) since 1998.

Mr. Naveen Kumar Koyi, Purdue University, West Lafayette

Naveen Kumar was born in INDIA in the state of Andhra Pradesh in the year 1993. He had completed Bachelors of Technology(B.Tech) at JNTU Kakinada in Andhra Pradesh in field of Electrical and Electronics Engineering in 2010. Then he was graduated from PURDUE UNIVERSITY in the department of Electrical and Computer Engineering in 2016. He had done projects on Data Encryption using Reversible Data Hiding (RDH) using Matlab simulation software. Being a co-author of this paper, he had familiarity with Cadence SPB 16.6 CIS software which is extremely useful in designing Schematics which are shown in this paper. He had familiarity with Homer Software, R-Programing etc.,. That is something outstanding about Naveen Kumar. Koyi, a co-author of this publication.



Mr. Hassan Abdullah Alibrahim, Purdue University Calumet

I have an associates degree with first honor in Electrical Engineering Technology from Jubail Industrial College, Saudi Arabia. Now, I am studying at Purdue University Calumet to accomplish my bachelor degree in Electrical and Computer Engineering Technology (Senior classification).

Mr. Mohammad Almenaies, Purdue University, Calumet

- I graduated from collage of technology in 2003 in major Transmission and Distribution Tower electricity (Diploma - 2.5 yr.) - Employed in Ministry of Electricity and Eater in Kuwait since 2004. - Got scholarship from ministry of electricity and water in 2013 to completion of studies (Bachelor.) - Current student on Purdue University Calumet in USA. - Expected graduation in spring 2016 in Electrical and Computer Engineering Technology major(ECET.)

EMBEDDED SYSTEM DESIGN BASED ON BEAGLEBONE BLACK WITH EMBEDDED LINUX

Abstract

The paper expounds the content of the course and further explores the context with which the course is delivered that finally turns over the ownership of the subject material to the learner in the form of final projects. This is a 400-level course that has retained its technological currency by climbing the evolutionary ladder of myriad of technological advances in hardware, software and OSs. In its current form the course has been totally redesigned based on BeagleBone Black. The BeagleBone Black is a compact, affordable, open-source Linux computing platform ideal for class room learning and designing. The operating system for the board is Linux and course is delivered with C++ and Python.

The paper elaborates the architecture of the board, its multiple processors and accompanying hardware components that are utilized in the course. It will further discuss the software methodologies utilized in the course with C++ and Python languages. Debian Linux is utilized in the course with command line environment. Paper discusses all the different tools employed like Putty and WinSCP and Eclipse IDE that will be needed in the execution of software Design on PC. Laboratory exercises covered the interfacing, controlling, and communicating with the physical environment.

Through this course the students in Electrical and Computer Engineering Technology program develop the design template that they utilize in a Capstone Senior Design two course sequence and become proficient system designers for tackling challenges of the industry. The pedagogy of the course delivery is based on “Interactive Learning model”, utilizing the methodology of Outcome Based Education. Outcome Based Education’s end result is the students’ design projects performed at the end of the course. The course is conducted in a lab or studio like settings, that integrates both lecture and laboratory work in the same settings. The paper elaborates the benefits derived through the pedagogical approaches of keeping the learner actively engaged in all aspects of discovery and design. The course interactively involves the learner in directing and defining the System Design under discourse.

I. Introduction

In the bygone days, the System Design courses were mostly centered on, a single processor. Before the era of Microcontrollers courses were designed with single Microprocessors(80*86), EPROM chip, and 2048-bit static ram with i/o ports and timer(8155) in a minimum mode. This era was followed by the availability of Microcontrollers and courses were designed around a single Microcontroller with all the needed functionality (A/D, Timer, RAM, ROM, and Direct availability of multiple ports) available on a single chip. This was followed with the courses that were centered on Personal Computers and Microcontrollers providing the System’s functionality together. These systems tap into the dedicated processor of Microcontrollers and utilized the power of the PC’s Operating System and other resources and code that was available. This was the era Multiprocessor System Design. Omnipresence of internet added the possibility of

Distributed System Design with Multiple Processors. Moving into the present times the System Design courses are centered around the powerful new system-on-a-chip (SOC) devices (BeagleBone Black¹ and Raspberry PI²) that were essentially capable of performing all the duties of a computer on a single chip. The need to go beyond the basics of providing an introductory course in the microprocessor or microcontroller in Engineering and Engineering Technology type curriculums has long been overdue. The subject matter covered in System Design has matured to the extent that it has been the subject of curriculum content in the form of two or more courses in most of the universities. The subject course which is the subject of this paper is a 400 level course in the Electrical and Computer Engineering Technology Department. This is preceded by two courses: 1) a C or C++, programming course, that covers the C or C++ language constructs with emphases on bit manipulation, 2) an introductory microcontroller based Embedded System design course, that covers the architecture of the microcontroller along with its software design implemented with C language. The Embedded System Design course is based on Atmel's Atmega 328 which is on Arduino UNO platform.

II. Choice between BeagleBone Black and Raspberry PI

The choice between the BeagleBone Black (BBB) and Raspberry PI (RPI) is easy to make if we carry on aside by side comparison of the features that are of relevance for a class room usage.

Comparing Raspberry Pi and BeagleBone Black³

	BeagleBone Black	Raspberry Pi
Base Price	45	35
Processor	1GHz TI Sitara AM3359 ARM Cortex A8	700 MHz ARM1176JZFS
RAM	512 MB DDR3L @ 400 MHz	512 MB SDRAM @ 400 MHz
Storage	2 GB on-board eMMC, MicroSD	SD
Video Connections	1 Micro-HDMI	1 HDMI, 1 Composite
Supported Resolutions	1280×1024 (5:4), 1024×768 (4:3), 1280×720 (16:9), 1440×900 (16:10) all at 16 bit	Extensive from 640×350 up to 1920×1200, this includes 1080p
Audio	Stereo over HDMI	Stereo over HDMI, Stereo from 3.5 mm jack
Operating Systems	Debian (Default), Ubuntu, Android, ArchLinux, Gentoo, Minix, RISC OS, others...	Raspbian (Recommended), Ubuntu, Android, ArchLinux, FreeBSD, Fedora, RISC OS, others...
Power Draw	210-460 mA @ 5V under varying conditions	150-350 mA @ 5V under varying conditions

GPIO Capability	65 Pins	8 Pins
Peripherals	1 USB Host, 1 Mini-USB Client, 1 10/100 Mbps Ethernet	2 USB Hosts, 1 Micro-USB Power, 1 10/100 Mbps Ethernet, RPi camera connector

The overwhelming advantage of BBB over RPI is the availability General Purpose I/Os (65) vs. 8. This is significant for System Design around it for interfacing transducers and control elements.

The rich interconnectivity offered by BBB vs. RPI along with storage and speed make us select BBB for our System Design class and is the subject of this paper.

BBB Interconnectivity and buses:

- 3 I2C buses
- CAN bus
- SPI bus
- 4 timers
- 5 serial ports
- 65 GPIO pins
- 8 PWM outputs
- 7 analog inputs (1.8V max 12 bit A/D converters)

RPI Interconnectivity and buses:

- 8 GPIO pins
- 1 UART interface
- 1 SPI bus
- 1 I2C bus

III. Definition of Multiprocessing Computer System Design

The term Multiprocessing, with regard to this paper is defined as the utilization of multiple processors or microcontrollers which provides or defines the system's hardware functionality. The system's software functionality is made up of a host of discrete software pieces that are implemented or targeted on host of microcontrollers or microprocessors. In case of distributed systems, the subject hardware and software are implemented at distributed geographical locales. Thus, Multiprocessing as described in this paper, is the use of two or more central processing units (CPUs) within a system and it further refers to the execution of multiple concurrent software processes in a system as opposed to a single process at any instant of time⁴.

IV. BeagleBone Black and host computer Platform

There are three utilities that students use in the lab, 1) PuTTY 2) WinScp 3) Nano Editor.

PuTTY⁵ is a free software application for Windows 95, 98, XP, Vista, and 7 which can be used to make an SSH connection to your server (BBB for instance). All the software design is performed with putty remotely on BBB. You can download the application PuTTY from <http://www.putty.org/>

WinSCP⁶ is an open source free SFTP client, FTP client, WebDAV client and SCP client for Windows. Its main function is file transfer between a local and a remote computer. Beyond this, WinSCP offers scripting and basic file manager functionality.

Nano Editor⁷ is a small and friendly text editor. Besides basic text editing, Nano offers many extra features like an interactive search and replace, go to line and column number, auto-indentation, feature toggles, internationalization support, and filename tab completion. This editor is part of the Linux distribution.

In the lab host PCs use Windows64 and they were tethered to BBB with USB mini cable. BBB out of the box is accompanied with “Quick Start” instruction set. This enables the user to install the drivers on the PC. Following the instructions you are starting a html session with BBB on Pc and you could continue in START.html.

V. Operating System - Linux

Added bonus of this approach is student learn and master Linux and get liberated from the confines of commercial operating systems and learn to design with the complexity of Linux which paves the way for efficiency of System Design . They could choose to write applications with a host of different computer languages.

The latest version of BBB are shipped with Debian⁸ distribution already installed. We opted to continue with this choice of distribution due to the fact that Debian distribution currently is the most supported distribution on this platform. About ¼ of the time is dedicated to learning and mastering different concepts and practices with the Linux. A short list of the topics follows.

- 1) Examining the Prompt and working with Putty
- 2) Concept of Remote access
- 3) Exploring the Linux File System
- 4) Understanding the directory tree
- 5) Listing files and directories
- 6) Checking file types
- 7) Creating, editing, and viewing files with Nano editors

- 8) Installing of Software (including updating to the latest version of your distribution itself)
- 9) Kernel Programming Concepts⁹

VI. Programming Languages

We have opted for Python and C++ as the languages of choice for the course. Students come to this particular course with the background of two courses on course in C++ and another course in Embedded System Design with Arduino. However for this course most Python was utilized due to the fact of ease of use with the Adafruit's BeagleBone Python library¹⁰, which can handle (GPIO, PWM, ADC, I2C, SPI and UART). This in a nut shell serves all of our class experimental needs and is part of the installation.

<https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/>

About ¼ of the time is dedicated to learning and mastering Python language. Student pick up Python language fairly quickly since they already have a background in C++. All the coding is done using strictly Structured Programming Methodology.

VII. Laboratory Experiments

About ¼ of the time is dedicated to performing of the lab experiments. The Laboratory experiments span over Linux operating system, Hardware Interfacing of Electronics, data gathering, analysis and control of local elements and remote elements over the NET.

VIII. Partial List of Experiments Performed¹¹

The following is a short list of experiments performed in the lab.

- 1) Introduction to BeagleBone Black
- 2) Cloud(Web IDE)Accessing the Linux Terminal
- 3) Directories, User Navigation and Blinking LEDsBasic LED Python Program
- 4) Reading Pushbutton States
- 5) Analog to Digital Conversion
- 6) USART Serial Communication
- 7) Data Manipulation Using Lists(Python) or Arrays(C++)
- 8) I2C bus interface
- 9) Java Runtime Environment on the BBB
- 10) Controlling of Stepper Motors
- 11) DC Motor Speed AND Direction Control
- 12) Interfacing with Text Files (Reading and Writing)
- 13) Cross-Compilation and the Eclipse IDE (Home setup, not performed in Lab)^x
- 14) Remote server Interface

IX. Sample Experiments Performed

We will provide in this section a few sample experiment performed by the students in the Laboratory.

A Sample Lab 1:

Design a System around BBB and 5 LEDs that would provide the apperance of chasing each other by turning them on and off.

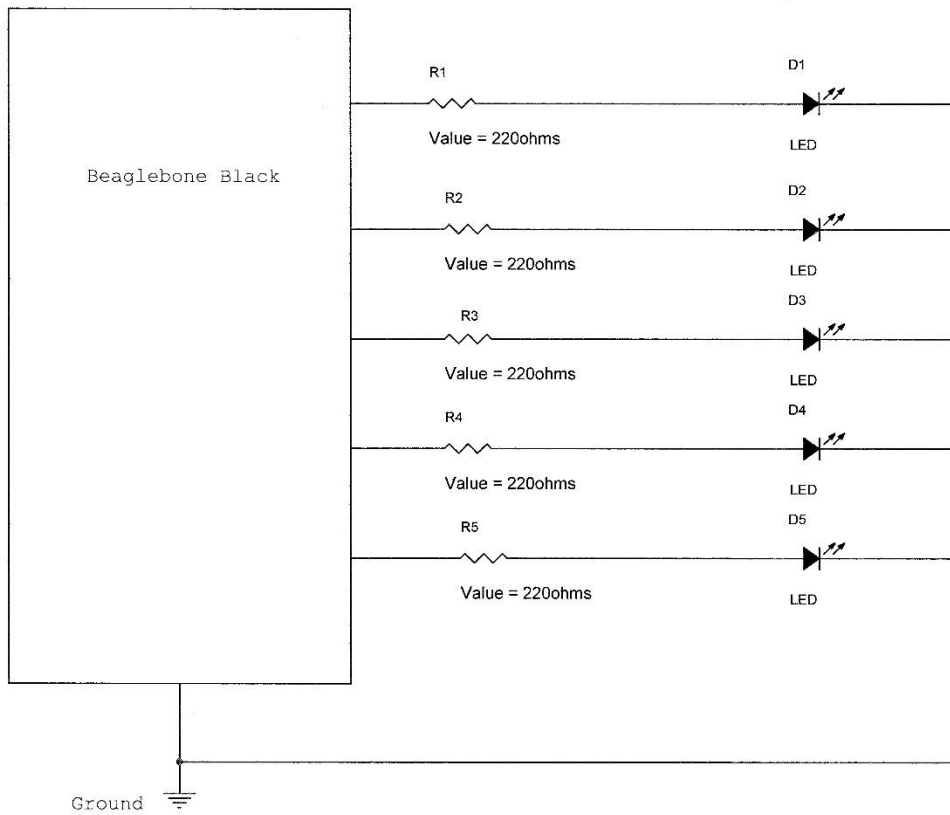


Figure 1. Schematic of BBB and 5 LEDs that are Chasing eachother

were chasing each other.

Code:

```
import Adafruit_BBIO.GPIO as GPIO
import time

GPIO.setup("P9_12",GPIO.OUT)
GPIO.setup("P9_14",GPIO.OUT)
GPIO.setup("P9_16",GPIO.OUT)
GPIO.setup("P9_11",GPIO.OUT)
GPIO.setup("P9_22",GPIO.OUT)

def blink():
    while True:
        GPIO.output("P9_12",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_12",GPIO.LOW)
        GPIO.output("P9_14",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_14",GPIO.LOW)
        GPIO.output("P9_16",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_16",GPIO.LOW)

        GPIO.output("P9_11",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_11",GPIO.LOW)

        GPIO.output("P9_22",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_22",GPIO.LOW)

        GPIO.output("P9_22",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_22",GPIO.LOW)

        GPIO.output("P9_11",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_11",GPIO.LOW)

        GPIO.output("P9_16",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_16",GPIO.LOW)

        GPIO.output("P9_14",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_14",GPIO.LOW)

        GPIO.output("P9_12",GPIO.HIGH)
        time.sleep(.25)
        GPIO.output("P9_12",GPIO.LOW)

def main():
    blink()

main()
```

Figure 2. Python Code of BBB and 5 LEDs that are Chasing eachother

A Sample Lab 2:

Design a System around BBB and LM34 temperature transducer so as to monitor ambient temperature in both Fahrenheit as well as Centigrade scales.

CODE:

```
import Adafruit_BBIO.ADC as ADC
import time

ADC.setup()

def sensor_value():
    analog_value = ADC.read("P9_40")
    print("The analog voltage is:")
    print (analog_value)
    analog_voltage = 1.8 * analog_value
    sensor_output_voltage = analog_voltage * 2
    F = sensor_output_voltage * 100
    return F

def measured_temps():
    while(1):
        F = sensor_value()
        C = (9/5)*(F - 32)
        print ("The temperature in the room is: ")
        print "%3.1fF, %3.1fC" % (F, C)
        time.sleep(1)

def main():
    measured_temps()
    sensor_value()

main()
```

Figure 3. Python Code for LM34 Temperature Monitoring System

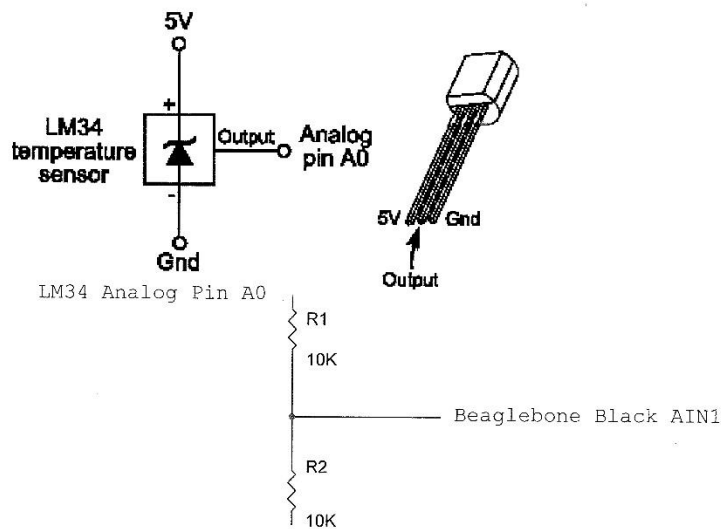


Figure 4. Schematic Diagram for LM34 Temperature Monitoring System

OUTPUT:

```

192.168.7.2 - PuTTY
0.204444438219
The temperature in the room is:
73.6F, 41.6C
The analog voltage is:
0.204999998212
The temperature in the room is:
73.8F, 41.8C
The analog voltage is:
0.204999998212
The temperature in the room is:
73.8F, 41.8C
The analog voltage is:
0.204444438219
The temperature in the room is:
73.6F, 41.6C
The analog voltage is:
0.204444438219
The temperature in the room is:
73.6F, 41.6C
The analog voltage is:
0.204444438219
The temperature in the room is:
73.6F, 41.6C

```

Figure 5. Typical Output for LM34 Temperature Monitoring System

A Sample Lab 3:

Design a System around BBB and LM34 temperature transducer and a DC motor, that will monitor the temperature and report it on the screen. The system is to run the motor above 78 degrees Fahrenheit, and will stop the motor below 78 degrees Fahrenheit.

```
Code:
import Adafruit_BBIO.ADC as ADC
import Adafruit_BBIO.GPIO as GPIO
import time

GPIO.setup("P9_5",GPIO.OUT)
GPIO.setup("P9_33",GPIO.IN)
GPIO.setup("P9_12",GPIO.OUT)

sensor_pin="P9_33"

ADC.setup()

def temperature():
    while True:

        reading = ADC.read(sensor_pin)
        millivolts = reading * 1.8
        voltage=millivolts*2
        temp_f= voltage*100
        temp_c = (temp_f-32)*5/9
        print ("%3.1fV %3.1f %3.1fC" % (voltage,temp_f, temp_c))
        if temp_f >=78:
            GPIO.output("P9_12",GPIO.HIGH)
            print("Motor ON")
        else:
            GPIO.output("P9_12",GPIO.LOW)
            print("Motor OFF")
            time.sleep(1)

def main():
    temperature()

main()
```

Figure 6. Python Code for LM34 Temperature Monitoring Motor Control System

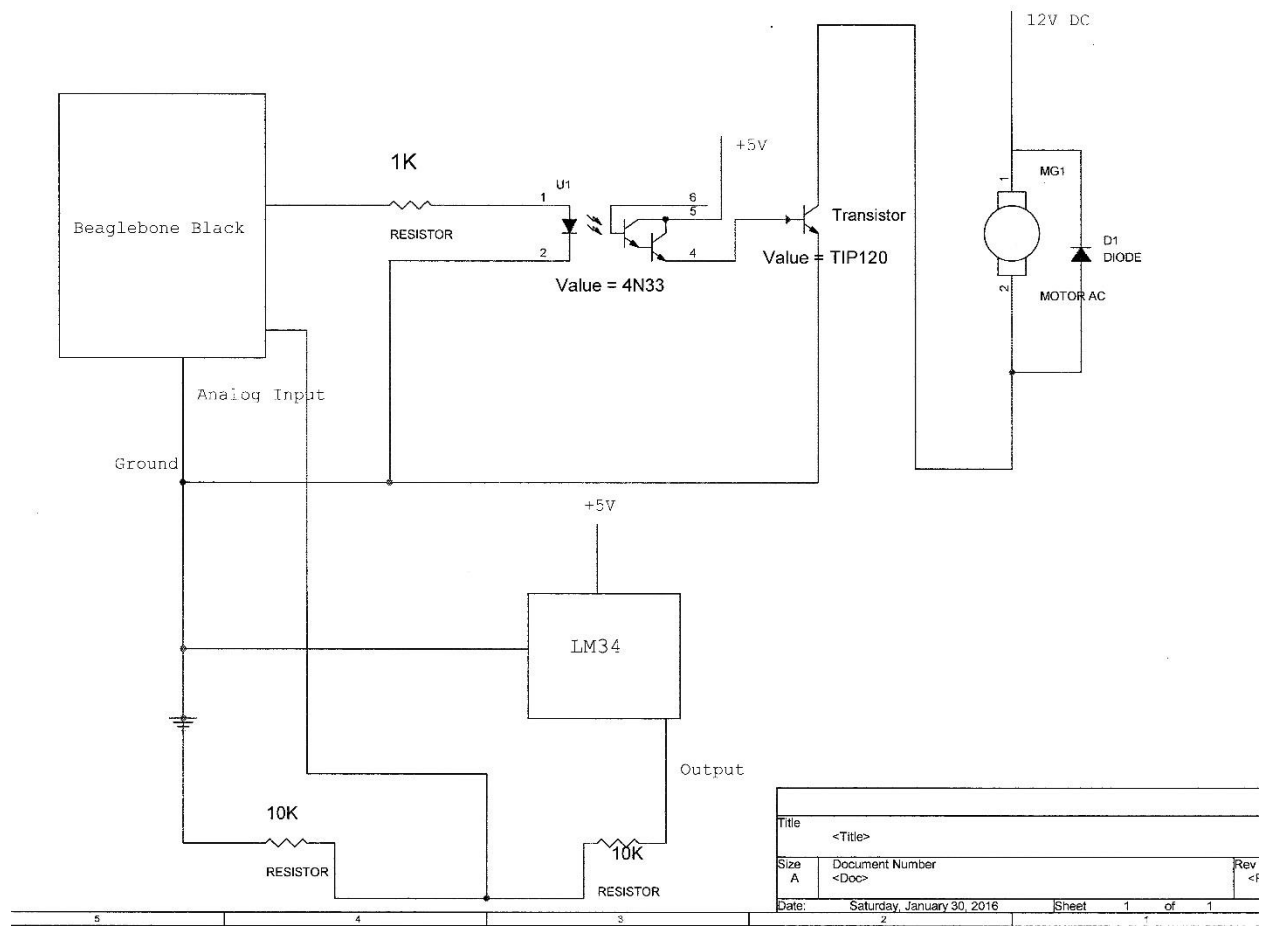


Figure 7. Schematics for LM34 Temperature Monitoring Motor Control System

```
0.8V 77.8F 25.4C
Motor OFF
0.8V 78.8F 26.0C
Motor ON
0.8V 79.4F 26.3C
Motor ON
0.8V 80.0F 26.7C
Motor ON
0.8V 80.6F 27.0C
Motor ON
0.8V 81.0F 27.2C
Motor ON
0.8V 81.4F 27.4C
Motor ON
0.8V 82.0F 27.8C
Motor ON
0.8V 82.0F 27.8C
Motor ON
0.8V 82.4F 28.0C
Motor ON
0.8V 82.2F 27.9C
Motor ON
0.8V 81.6F 27.6C
Motor ON
0.8V 81.0F 27.2C
Motor ON
0.8V 80.4F 26.9C
Motor ON
0.8V 79.8F 26.6C
Motor ON
0.8V 79.6F 26.4C
Motor ON
0.8V 79.0F 26.1C
Motor ON
0.8V 78.6F 25.9C
Motor ON
0.8V 78.2F 25.7C
Motor ON
0.8V 78.2F 25.7C
Motor ON
0.8V 77.8F 25.4C
Motor OFF
0.8V 77.4F 25.2C
Motor OFF
```

Figure 8. Typical Output in Putty for LM34 Temperature Monitoring Motor Control System

XII. Pedagogy of the Course

The pedagogy of the course is based on Outcome Based Education⁶, and utilizes the interactive model of learning. All the students maintain an online portfolio of their work. The system designed in the laboratory to perform a specific task is the core measurement as the learning outcome of the course. The laboratory performance of the course is performed in teams of three students. This mode provides a platform for horizontal learning through active and engaged discourse and discussion. Students are empowered to charter their learning and feed their curiosity. The course culminates in a Final Project which is assessed based upon its comprehensiveness and originality. Students are required to master the soft skills of comprehensive report writing on a weekly basis and of Technical Project Report writing and project oral presentation based upon the Team's Final Project. These classroom practices and laboratory environment provides a challenging and invigorating environment that prepares them for a lifelong learning process and career path.

Each of the student after covering $\frac{3}{4}$ of the time in class spends the last $\frac{1}{4}$ of the time in performing a Final Project for the class based upon integration of all elements of the course content. A student work sample of the System Design with BBB performed as the Final Project for the class is presented in the Appendix.

XIII. Conclusion

The paper has provided the reader the philosophical framework and turnkey pathway for the subject of Multiprocessing System Design. The subject matter is to be pursued in parallel at dual tiers of hardware and software system design details. The subject matter Multiprocessing System Design provides the student in class room many of the same realistic challenges faced by a System Design practitioner. The authors sincerely hope that many academicians by offering such a course in their respective curriculums will provide students a frame work to integrate hardware and software and learn the challenges faced while working with multiple processors and embedded Operating System are to be integrated into a whole system.

XIV. Appendix – A Sample Final Project Report Demonstrating Outcome of the Course

Introduction:

This Final Project is integration of BNO055 (the BNO055 is a *micro-electro-mechanical* system (MEMS) of accelerometer, magnetometer and gyroscope and putting them on a single die with a high speed ARM Cortex-M0 based processor) with BeagleBone Black (BBB) to collect 3D orientation rotational data. BNO055 communicates with BBB via I2C bus. The BBB based subject designed system further communicates with a remote server Python module (hosted by Adafruit) in controlling a 3D model on a webpage.

The BNO055 can output the following sensor data¹:

- **Absolute Orientation** (Euler Vector, 100Hz)
Three axis orientation data based on a 360° sphere
- **Absolute Orientation** (Quaternion, 100Hz)
Four point quaternion output for more accurate data manipulation
- **Angular Velocity Vector** (100Hz)
Three axis of 'rotation speed' in rad/s
- **Acceleration Vector** (100Hz)
Three axis of acceleration (gravity + linear motion) in m/s²
- **Magnetic Field Strength Vector** (20Hz)
Three axis of magnetic field sensing in micro Tesla (uT)
- **Linear Acceleration Vector** (100Hz)
Three axis of linear acceleration data (acceleration minus gravity) in m/s²
- **Gravity Vector** (100Hz)
Three axis of gravitational acceleration (minus any movement) in m/s²
- **Temperature** (1Hz)
Ambient temperature in degrees Celsius

Using the python module, one is able to talk to the BNO055 sensor from one's BeagleBone Black and bring on board the magic of orientation sensing to the project. This report shows how to connect a BNO055 absolute orientation sensor to a BeagleBone Black. The system is detecting the orientation of the sensor and accordingly rotating a 3D model.

Hardware:

1. Adafruit BNO055 absolute orientation sensor breakout.
2. BeagleBone Black.
3. Breadboard & hookup wires.
4. Soldering tools.

Parts:

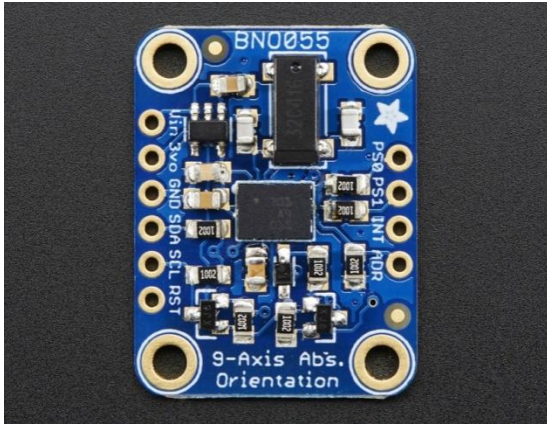


Figure 1: BNO055 Sensor

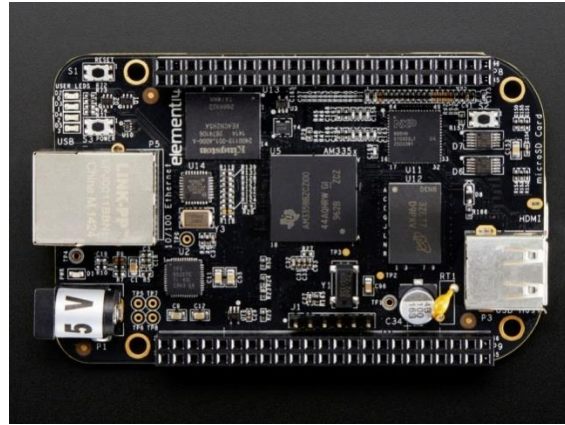


Figure 2: BeagleBone Black

Bno055 Pinouts:

Power Pins:

- VIN: 3.3-5.0V power supply input.
- 3VO: 3.3V output from the on-board linear voltage regulator, you can grab up to about 50mA as necessary.
- GND: The common/GND pin for power and logic.

I2C Pins:

- SCL - I2C clock pin, connect to your microcontrollers I2C clock line. This pin can be used with 3V or 5V logic, and there's a 10K pullup on this pin.
- SDA - I2C data pin, connect to your microcontrollers I2C data line. This pin can be used with 3V or 5V logic, and there's a 10K pullup on this pin.

Other Pins Utilized:

- RST: Hardware reset pin. Set this pin low then high to cause a reset on the sensor. This pin is 5V safe.
- INT: The HW interrupt output pin, which can be configured to generate an interrupt signal when certain events occur like movement detected by the accelerometer, etc. (not currently

supported in the Adafruit library, but the chip and HW is capable of generating this signal). The voltage level out is 3V

- ADR: Set this pin low to change the default I2C address for the BNO055 if you need to connect two ICs on the same I2C bus. The default address is 0x28. If this pin is connected to 3V, the address will be 0x29
- PS0 and PS1: These pins can be used to change the mode of the device (it can also do HID-I2C and UART) and also are provided in case Bosch provides a firmware update at some point for the ARM Cortex M0 MCU inside the sensor. They should normally be left unconnected.

Assembly:

Assembly of the system's components are as follows:

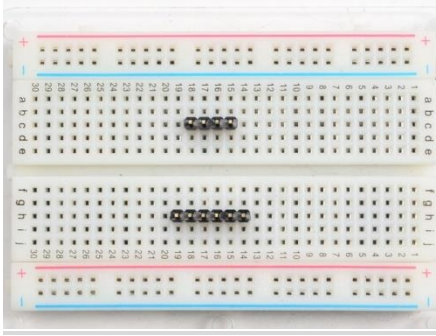


Figure 3: Sensor Head

Step 1: Preparing the header strip.

By cutting the strip to length if necessary. It will be easier to solder if we insert it into a breadboard - **long pins down**.

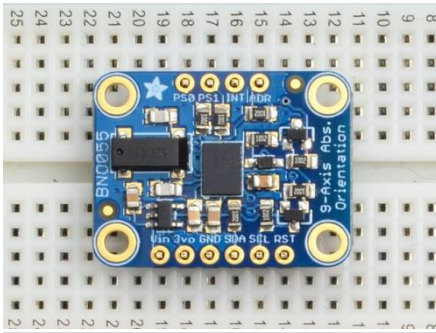


Figure 4: Sensor Place

Step 2: Adding the breakout board.

By placing the breakout board over the pins so that the short pins poke through the breakout pads and the long pins through the breadboard.

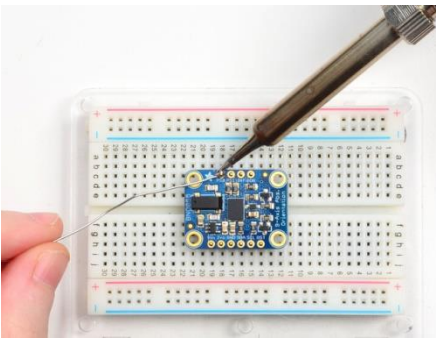


Figure 5: Sensor Sold

Step 3: Start soldering.

By soldering all pins for reliable electrical contact. Solder the longer power strip first.

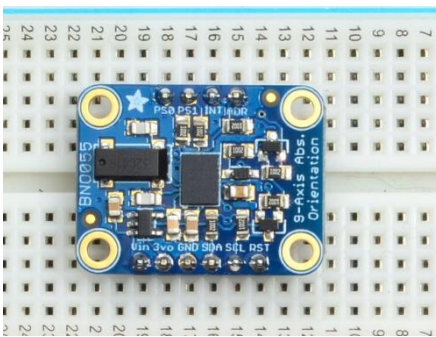


Figure 6: Sensor Done

Step 4: We're done!

Checking the solder joints visually.

Wiring:

On a BeagleBone Black, the BNO055's I2C communication mode can be used as the hardware fully supports I2C clock stretching. The BeagleBone Black is connected to the BNO055 as follows:

- BNO055 Vin to BeagleBone Black 3.3V power pin P9_3.
- BNO055 GND to BeagleBone Black ground pin P9_1.
- BNO055 SDA to BeagleBone Black I2C2_SDA pin P9_20.
- BNO055 SCL to BeagleBone Black I2C2_SCL pin P9_19.

Wiring diagram:

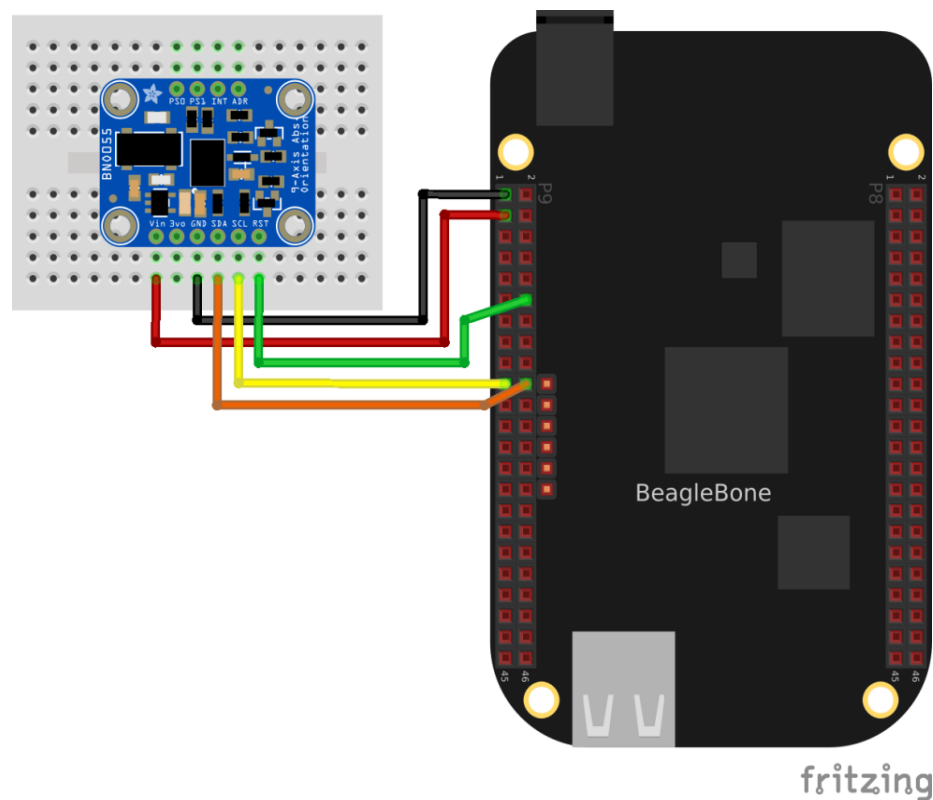


Figure 7: Wiring diagram for Beaglebone black & BNO055 sensor

Software:

Installation:

Before writing the code for this project, few things must be checked. First, the BeagleBone Black should be running the latest operating system. Second, the board is connected to the internet through a wireless or wired network connection. Using the command line terminal, run the following commands to install the necessary dependencies:

```
sudo apt-get update
sudo apt-get install -y build-essential python-dev python-smbus python-pip
git
```

Next run the following commands to download and install the latest version of the [BNO055 Python module code from GitHub](#):

```
cd ~

git clone https://github.com/adafruit/Adafruit_Python_BNO055.git

cd Adafruit_Python_BNO055

sudo python setup.py install
```

- Note: if you get error while installing from website, writ the following:

```
wget --no-check-certificate https://github.com/adafruit/Adafruit_Python_BNO055.git
```

Usage:

To learn how to use the BNO055 Python module I'll walk through running the included simpletest.py example below. Before you get started make sure you've wired up the sensor and installed the library code following the steps above. Also note if you're using the sensor on a Raspberry Pi or other device in UART mode be sure you've followed the steps to disable the kernel from accessing the serial port at the same time as the BNO055 sensor! Connect to the board in a command terminal and navigate to the library examples folder by running the following command:

```
cd ~/Adafruit_Python_BNO055/examples
```

Open the file using the nano text editor by running:

```
nano simpletest.py
```

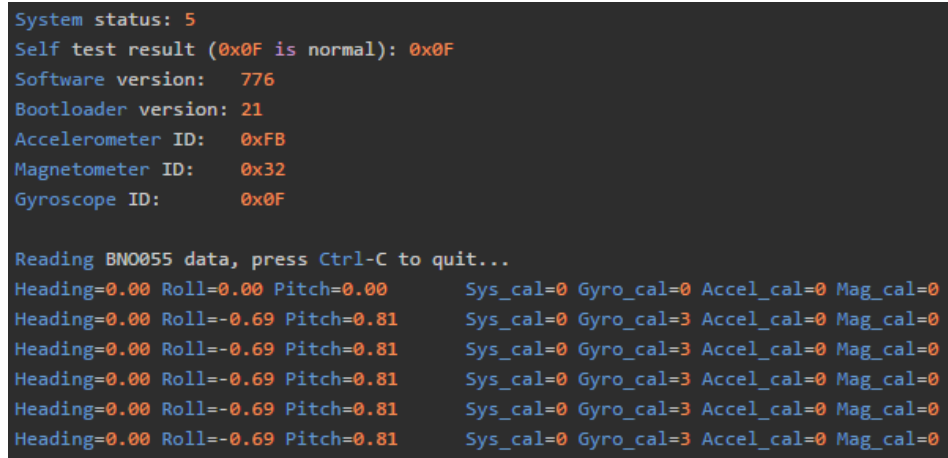
Then navigate down to this section of the example code then leave only this line uncommented:

```
bno = BNO055.BNO055(rst='P9_12')
```

Now run the code as a root user by executing: `/Adafruit_Python_BNO055/examples#`

```
python simpletest.py
```

If everything is working correctly you should see the following:



```
System status: 5
Self test result (0x0F is normal): 0x0F
Software version: 776
Bootloader version: 21
Accelerometer ID: 0xFB
Magnetometer ID: 0x32
Gyroscope ID: 0x0F

Reading BNO055 data, press Ctrl-C to quit...
Heading=0.00 Roll=0.00 Pitch=0.00 Sys_cal=0 Gyro_cal=0 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81 Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81 Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81 Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81 Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
Heading=0.00 Roll=-0.69 Pitch=0.81 Sys_cal=0 Gyro_cal=3 Accel_cal=0 Mag_cal=0
```

Figure 8. Data Captured

Once the example code is running you can see it start out by printing diagnostic details about the BNO055 sensor. You can learn more about the sensor status and other values in its datasheet, but a status of 5 means the fusion algorithm is running and a self-test result of 0x0F means all of the sensors are working as expected.

Every second the orientation data for the sensor is printed as Euler angles that represent the heading, roll, and pitch of the sensor in degrees.

In addition to orientation the calibration level of each sensor is printed each second. Calibrating the BNO055 sensor is very important to ensure you get good orientation readings. You can see the system (the fusion algorithm) and each sensor has a separate calibration level. A level of 0 is uncalibrated and 3 are fully calibrated (with 1 and 2 being levels of partial calibration).

To learn how to use the library we will walk through the code for it in a little more detail. Stop running the `simpletest.py` example (by pressing Ctrl-C) and then open the file in a text editor (like nano). The file starts by importing some required dependencies and the `Adafruit_BNO055` module:

```
import logging

import sys

import time

from Adafruit_BNO055 import BNO055
```

Next the code creates and configures a BNO055 sensor instance. We have already seen this part of the code described above when preparing to run the example. After creating and configuring the BNO055 sensor the code optionally turns on Python's logging module output (if a -v parameter is passed when running the script--this is useful to see the raw commands sent and received with the BNO055 sensor), and then initializes the BNO055 sensor.

```
if len(sys.argv) == 2 and sys.argv[1].lower() == '-v':  
  
logging.basicConfig(level=logging.DEBUG)  
  
if not bno.begin():  
  
raise RuntimeError('Failed to initialize BNO055! Is the sensor connected?')
```

It's very important to initialize the BNO055 sensor by calling the **begin()** function. This function will return true if it succeeds in initializing the sensor and false if it fails for some reason. The function might also throw an exception that provides more details about why the sensor failed to initialize. Next we can see how status and diagnostic information is retrieved using the **get_system_status()** and **get_revision()** functions.

```
status, self_test, error = bno.get_system_status()  
  
print('System status: {}'.format(status))  
  
print('Self test result (0x0F is normal): 0x{0:02X}'.format(self_test))  
  
if status == 0x01:  
  
print('System error: {}'.format(error))  
  
print('See datasheet section 4.3.59 for the meaning.')  
  
sw, bl, accel, mag, gyro = bno.get_revision()  
  
print('Software version:  {}'.format(sw))  
  
print('Bootloader version: {}'.format(bl))  
  
print('Accelerometer ID:   0x{0:02X}'.format(accel))  
  
print('Magnetometer ID:     0x{0:02X}'.format(mag))  
  
print('Gyroscope ID:        0x{0:02X}\n'.format(gyro))
```

Now the main loop of the program runs. WE can see the **read_euler()** function returns a tuple of heading, roll, and pitch data. In addition the **get_calibration_status()** function is called to retrieve the calibration level for each part of the sensors.

```
print('Reading BNO055 data, press Ctrl-C to quit...')

while True:

    heading, roll, pitch = bno.read_euler()

    sys, gyro, accel, mag = bno.get_calibration_status()

print('Heading={0:0.2F} Roll={1:0.2F} Pitch={2:0.2F}\tSys_cal={3}

    Gyro_cal={4} Accel_cal={5} Mag_cal={6}'.format(

heading, roll, pitch, sys, gyro, accel, mag))
```

That's really all we need to use, initialize, and get orientation from the BNO055 sensor!

WebGL Example:



Figure 9. Adafruit BNo55 Absolute Orientation Sensor Demo - A

In this example, we can run the absolute orientation sensor through browser by executing the code as a root user by executing to: `~/Adafruit_Python_BNO055/examples/webgl_demo#`

Run `python server.py`

Then open the browser and write the following include port number: (preferred to use Google Chrome browser)

<http://192.168.7.2:5000/>

Now you should see the following webpage:

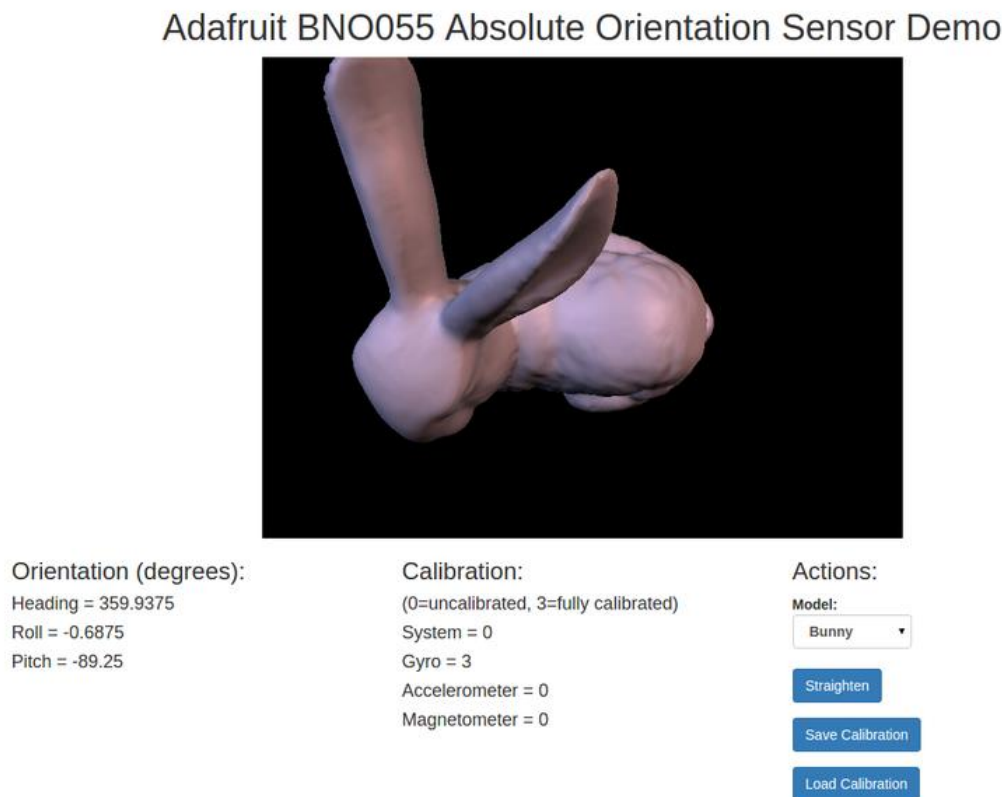


Figure 10. Adafruit BNo55 Absolute Orientation Sensor Demo - A

Review Steps:

- 1- `sudo apt-get update`
- 2- `sudo apt-get install -y build-essential python-dev python-smbus python-pip git`
- 3- `cd ~`
- 4- `git clone https://github.com/adafruit/Adafruit_Python_BNO055.git`
OR `wget --no-check-certificate https://github.com/adafruit/Adafruit_Python_BNO055.git`
- 5- `cd Adafruit_Python_BNO055`

- 6- `sudo python setup.py install`
- 7- `cd ~/Adafruit_Python_BNO055/examples` then write (`nano simpletest.py`)
- 8- Find then leave this line uncommented : `bno = BNO055.BNO055(rst='P9_12')`
- 9- Run file `/Adafruit_Python_BNO055/examples# python simpletest.py`
- 10- To run the WebGL, execute to : `~/Adafruit_Python_BNO055/examples/webgl_demo#`
- 11- Run the file `python server.py`
- 12- In the browser write <http://192.168.7.2:5000/>

References for the Report of Sample Final Project:

- [1] <https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/overview>
 - [2] <https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor?view=all>
-

Bibliography

- [1] <http://beagleboard.org/hardware/design>
- [2] <https://www.raspberrypi.org>
- [3] <http://makezine.com/2014/02/25/how-to-choose-the-right-platform-raspberry-pi-or-beaglebone-black/>
- [4] Farook, O., & Sekhar, C. R., & Agrawal, J. P., & Ahmed, A. (2012, June), *Multiprocessor Embedded System Design: A Course with Hardware-Software Integration* Paper presented at 2012 ASEE Annual Conference, San Antonio, Texas. <https://peer.asee.org/21718>
- [5] <https://mediateemple.net/community/products/dv/204404604/using-ssh-in-putty->
- [6] <https://winscp.net/eng/docs/introduction>
- [7] <http://www.nano-editor.org/dist/v2.2/nano.html>
- [8] http://elinux.org/Beagleboard:BeagleBoneBlack_Debian
- [9] “Exploring Beagle Bone Tools and Techniques”, by Derek Molloy. John Wiley & Sons Publication 2015.
- [10] <https://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/installation-on-ubuntu>
- [11] http://inspire.logicsupply.com/p/tutorials_3943.html
- [12] Omer Farook, Jai P. Agrawal, Chandra R. Sekhar, Essaid Bouktache, Ashfaq Ahmed and Mohammad Zahraee “Outcome Based Education And Assessment”, Proceedings of the 2006 American Society for Engineering Education Annual Conference & Exposition June 20 -23, 2006. Chicago, IL.