Project Assignment #3

CSCI502/702 – HARDWARE/SOFTWARE CO-DESIGN

Spring Semester 2025

GETTING STARTED WITH BEAGLEBONE BLACK AND IMU SENSOR INTERFACING

DUE TIME AND DATE

- **Sunday, 2 March, 23:59**.

LEVEL OF COLLABORATION ALLOWED

You will be working in groups of 3 students.

DELIVERABLES REQUIRED

Your group is required to prepare and submit a PDF report with Linux terminal screenshots as well as BBB+IMU photos in-operation, and video-demonstration of working tasks (total 3-5 minutes) to Moodle. Only one group member must submit. The names of all team members must be written on the cover page of your report. No name – no grade.

REFERENCES

- **Derek Molloy, "Exploring BeagleBone. Tools and Techniques for Building with Embedded Linux", 2nd edition, Wiley, 2019 (available on Moodle)**
- **Derek Molloy, "Exploring BeagleBone. Tools and Techniques for Building with Embedded Linux", 1st edition, Wiley, 2015 (available in library and on Moodle)**
- Note: some Linux commands and packages mentioned in the textbook(s) can be already deprecated. So, searching for alternatives on the web is required sometimes.

ASSIGNMENT DETAILS

All instructions in this manual are based on Ubuntu as the operating system use case. You can complete these tasks using Windows OS (or macOS) as well; however, a little (or no) guidance will be provided in this case as we did not test them on other operating systems.

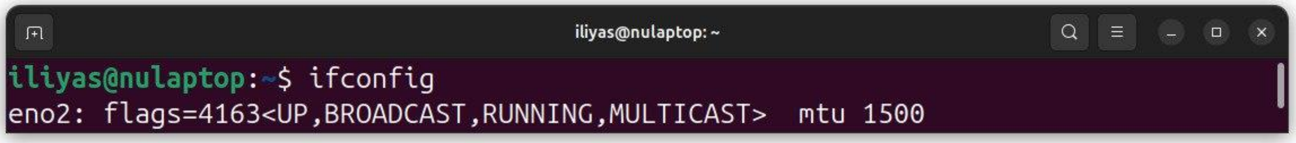
TASK 1. CONNECTING TO AND COMMUNICATING WITH BBB (15 POINTS)

This task aims to ensure that you can connect to your BeagleBone Black ***(hereafter, BBB for short)***. Related reference textbook material (available on Moodle):

- *Chapter 1. The BeagleBone Hardware*
- *Chapter 2. The BeagleBone Black Software*

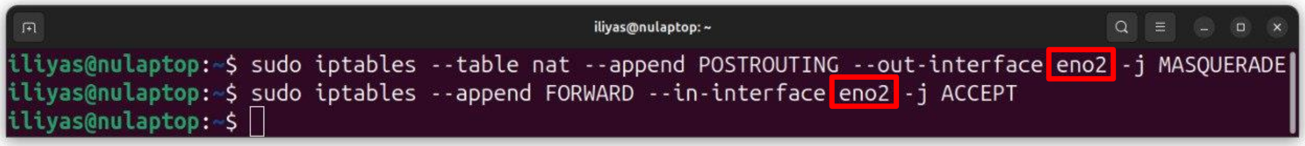
There are three main ways to connect to and communicate with the board over a wired network, each with its own advantages and disadvantages. The **first** way is to use **Internet-over-USB (the primary methods used in the class)**, which creates a "private" virtual LAN using a single USB cable. The second way is to use *regular Ethernet*, and the third is to use an *Ethernet crossover cable*. Connecting to the board over a network can be a stumbling block for beginners. It is usually straightforward if you are working at home with control of your own network; however, complex networks, such as those in universities, can have multiple subnets for wired and wireless communication. In such complex networks, routing restrictions may make it difficult, if not impossible, to connect to the board over regular Ethernet.

1. Get familiar with the **safety rules** when working with BBB on p. 21 of the reference textbook. Important note:
    a. Do not shut the BBB down by pulling out the power jack/USB power. You should shut down the board correctly using a software shutdown (e.g., by pressing the power button once) or by holding the power button for about eight seconds for a "hard" power down. Alternatively, you can use `sudo shutdown now` command in BBB terminal. Pulling out the power jack/USB power **can easily destroy the board.**
    b. Do not place a powered BBB on metal surfaces! Shorting the pins on the P8/P9 headers **can easily destroy the board.**
2. Read Section "Communication with the BBB" pp. 25-36 of the reference textbook.
3. Attach your BBB to a host Linux PC (your PC/laptop or a PC from 7.322 classroom) with a USB cable provided in the kit. It can take up to 1-2 minutes for BBB to fully load. By default, the host PC IP address is `192.168.7.1` and the BBB IP address is `192.168.7.2`. Verify if BBB is connected by typing `192.168.7.2` in an Internet Browser window (refer to p. 27 of the reference textbook).
4. For setting up Internet-over-USB connection to BBB, you need to activate Network Sharing on the host PC as described on p. 29 of the reference textbook (for Windows PC, follow p. 28).
    a. With BBB attached, type `ifconfig` in a terminal, which results in a display of the attached network interfaces. Try to understand what each one of the interfaces means.



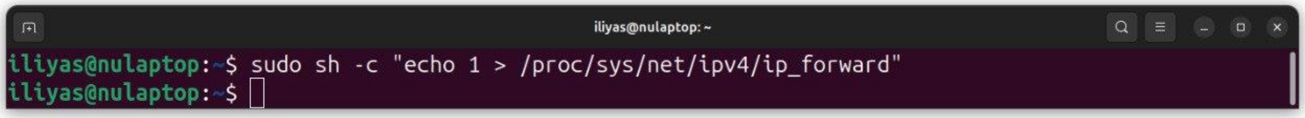
```
iliyas@nulaptop:~$ ifconfig
eno2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

    b. Find your main adapter (e.g., `eth0`) and Internet-over-USB adapter (e.g., `eth1`). In my case, the name of the network interface is `eno2`.
    c. Use the iptables program to configure the Linux kernel firewall rules. Note: I have used `eno2` in the command as it is the name of my wired internet network interface.



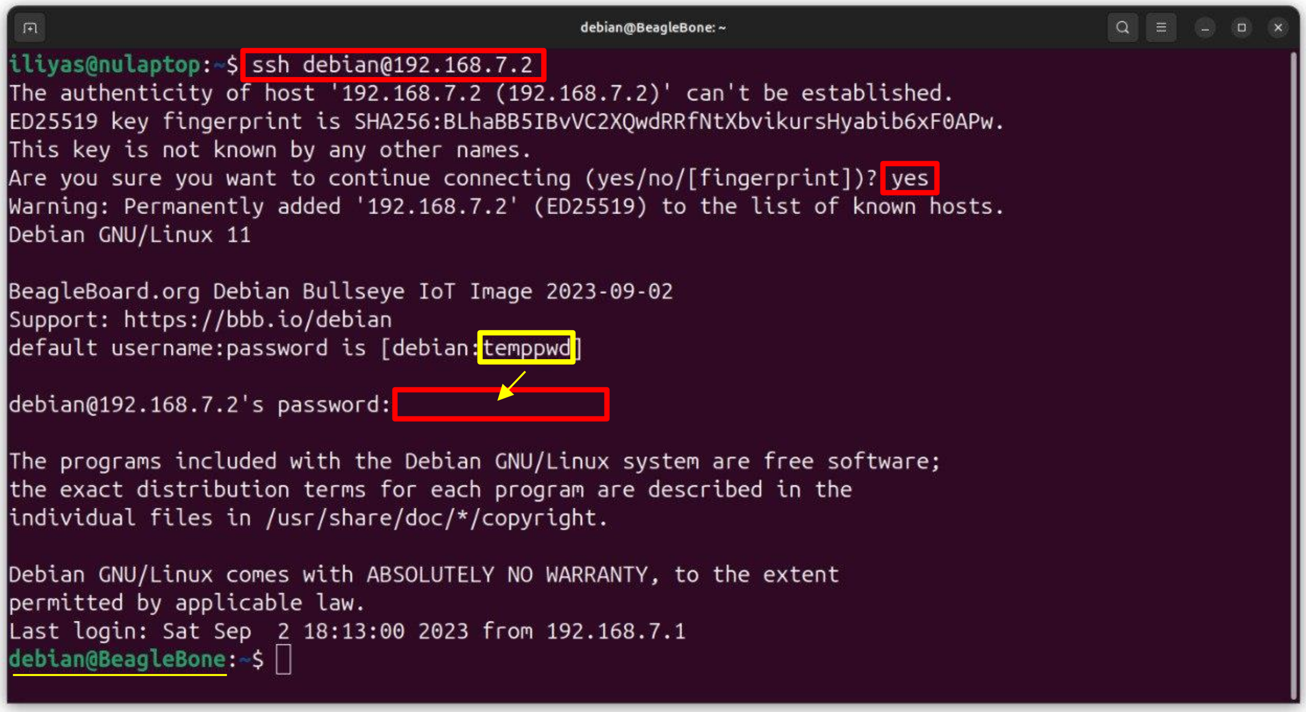
```
iliyas@nulaptop:~$ sudo iptables --table nat --append POSTROUTING --out-interface eno2 -j MASQUERADE
iliyas@nulaptop:~$ sudo iptables --append FORWARD --in-interface eno2 -j ACCEPT
iliyas@nulaptop:~$ 
```

d. Turn on IP forwarding:



```
iliyas@nulaptop:~$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
iliyas@nulaptop:~$ 
```

5. Connect to BBB through SSH using the following command prompts (pp. 33-36):

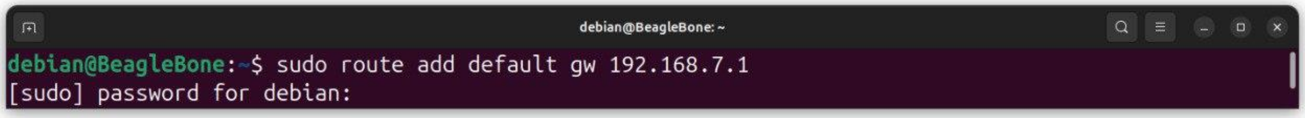


```
iliyas@nulaptop:~$ ssh debian@192.168.7.2
The authenticity of host '192.168.7.2 (192.168.7.2)' can't be established.
ED25519 key fingerprint is SHA256:BLhaBB5IBvVC2XQwdRRfNtXbvikursHyabib6xF0APw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.7.2' (ED25519) to the list of known hosts.
Debian GNU/Linux 11

BeagleBoard.org Debian Bullseye IoT Image 2023-09-02
Support: https://bbb.io/debian
default username:password is [debian:temppwd]

debian@192.168.7.2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep  2 18:13:00 2023 from 192.168.7.1
debian@BeagleBone:~$ 
```

Note: for password type - `temppwd`

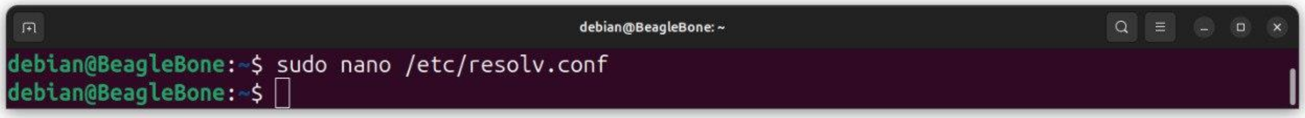
Note: for Windows, you can establish SSH connection using `cmd` (Control Prompt)

6. Set up and verify the Internet-over-USB connection on BBB side (p. 43) using the following prompt:

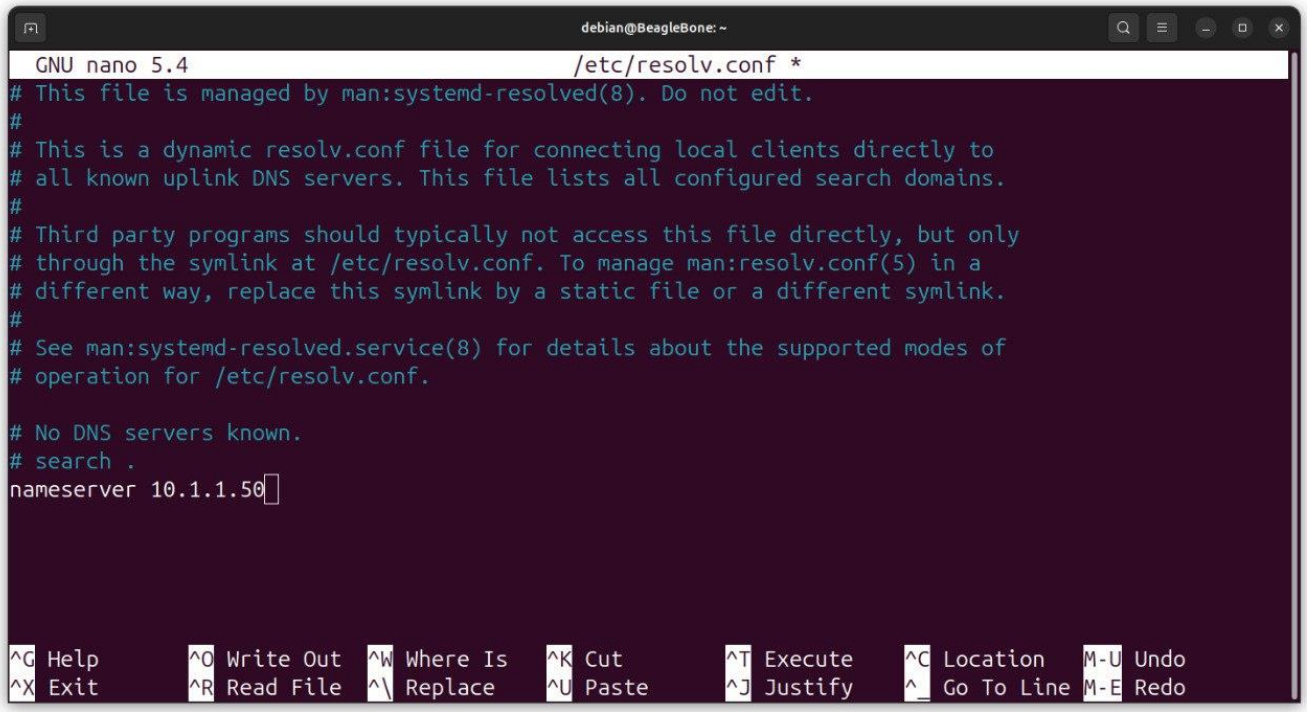


```
debian@BeagleBone:~$ sudo route add default gw 192.168.7.1
[sudo] password for debian:
```

Your computer must be connected to wired Ethernet, not Wi-Fi. If you are using PC connected to NU Ethernet (wired), you have to configure DNS in the `/etc/resolv.conf`. You can do it using `nano` (or any other text editor) as a superuser (`sudo`).



```
debian@BeagleBone:~$ sudo nano /etc/resolv.conf
debian@BeagleBone:~$ 
```



```
  GNU nano 5.4                          /etc/resolv.conf *
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients directly to
# all known uplink DNS servers. This file lists all configured search domains.
#
# Third party programs should typically not access this file directly, but only
# through the symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a
# different way, replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

# No DNS servers known.
# search .
nameserver 10.1.1.50
```

Note: using `ping/curl` might not work on campus due to NU firewall settings.
Note: gateway command and modifications to `/etc/resolv.conf` will be gone after BBB reboot, so they must be repeated on each BBB boot in case you want to establish internet connection
Include screenshots in your report showing Internet-over-USB connection. Explain the steps you have taken. Demonstrate it in the video demo as well.

7. For most of the tasks, you will not need Internet. You may need it in certain cases to download additional packages and files, especially, in the following projects with BBB. You can also do file transfer to BBB over SSH (pp. 35 – 36). Show an example of a file transfer from host PC to BBB via SSH.

```
sftp debian@192.168.7.2

debian@192.168.7.2's password: temppwd

sftp> put file

sftp> bye
```

TASK 2. BBB PROGRAMMING PRACTICE (15 POINTS)

Related reference textbook material:

- *Chapter 5. Practical BeagleBone Programming*

1. Read pages 167-197 of the reference textbook. Run helloworld.c, helloworld.cpp, pointers.c, cppstrings.cpp, makeLEDs.cpp on your BBB. Include demos in your report and video.
   You may download the source codes from the textbook GitHub page:
   https://github.com/derekmolloy/exploringBB

TASK 3. (OPTIONAL) CROSS-COMPILATION AND THE ECLIPSE IDE (0 POINTS)

Related reference textbook material:

- *Chapter 7. Cross-compilation and the Eclipse IDE*

1. Study Chapter 7 of the textbook (pages 251-260) and update the Linux OS on your host computer by running:

```
sudo apt install crossbuild-essential-armhf
```

2. Follow the instruction from the ***Testing the Toolchain*** section on pages 254 – 255 of the textbook. Cross-compile on host PC makeLEDs.cpp program (from Task 2) and transfer it to BBB. Run the executable and observe the LED flashing pattern.
3. Follow the ***Emulating the armhf Architecture*** section on pages 258 – 260 of the textbook. Cross-compile the makeLEDs.cpp program using –static flag and run the executable on host PC.
4. Following the instructions in the ***Cross-Compilation Using Eclipse*** section from page 260, create C++ project with toolchain cross-GCC.
5. Following the ***Remote System Explorer*** section on pages 318 – 322 (2nd edition!) of the textbook install the Remote System Explorer and the TM Terminal View RSE add-in in Eclipse IDE. Choose the following options:
   Work with: All available sites
   General Purpose Tools → Remote System Explorer User Actions
   General Purpose Tools → Remote System Explorer End-User Runtime
   General Purpose Tools → TM Terminal View Remote System Explorer (RSE) add-in
6. Cross-compile the makeLEDs program and copy/paste its executable and run it on BBB board in Eclipse.

TASK 4. IMU INTERFACING (20 POINTS)

Related reference textbook material:

- *Chapter 6. Interfacing to the BeagleBone Inputs/Outputs*
- *Chapter 8. Interfacing to the BeagleBone Buses*

Your group will interface an inertial measurement unit (IMU) to the BBB board using I^2C synchronous communication bus.

Do you want to detect the collision of a robot or to build a self-balancing robot? Or you have plans to build a drone? All these robots need sensors such as accelerometers, gyroscopes, magnetometers and IMUs. These small components are embedded into the robot to generate information about the different mechanical phenomenon such as acceleration, vibration, tilt, orientation in space, angular velocity, pitch or rotation.

These types of sensors with capabilities to measure the acceleration, tilt, angular velocity, and other mechanical phenomena are used in different devices including smartphones, gaming consoles or toys.

If an accelerometer sensor is designed to measure the acceleration and tilt, the gyroscopic sensor measures the angular velocity and orientation. The IMU sensor is a special one designed to combine the features of an accelerometer and gyroscope in order to display complete information about the acceleration, position, orientation, speed, etc. for a robot.

The **accelerometer** sensor measure acceleration in two different units including meters per second squared, or when the acceleration felt like a weight, in G-forces. The advantages of the accelerometer sensor include a high accuracy in applications with noises, as well the acceleration measurement down to zero Hertz. The biggest disadvantage of this sensor is the limited high frequency where the sensor works.

The **gyroscope** sensor is inexpensive and measures in degrees per second or revolutions per second the angular velocity. It's frequently used in robotic applications to measure the balancing and send corrections to motors or drones to stabilize the flight.

The **magnetometer** sensor is finding increasing use as compasses in consumer devices such as mobile phones and tablet computers

The **IMU or Inertial Measurement Unit** is a sensor that hosts three types of sensors.

The term **Internet of Things (IoT)** is widely used to describe the network of devices such as vehicles, and home appliances that contain electronics, software, actuators, and connectivity which allows these things to connect, interact and exchange data. The IoT involves extending Internet connectivity beyond standard devices, such as desktops, laptops, smartphones and tablets, to any range of traditionally dumb or non-internet-enabled physical devices and everyday objects. Embedded with technology, these devices can communicate and interact over the Internet, and they can be remotely monitored and controlled. In this project, you will use the BBB board and an IMU sensor to create an embedded IoT device for communicating sensor orientation information over network.

The AltIMU-10 v5 https://www.pololu.com/product/2739 is an inertial measurement unit (IMU) and altimeter that features the LSM6DS33 gyro and accelerometer, the LIS3MDL magnetometer, and the LPS25H digital barometer (not used in this assignment). An I²C interface accesses ten independent pressure, rotation, acceleration, and magnetic measurements that can be used to calculate the sensor's altitude and absolute orientation. The board operates from 2.5 to 5.5 V and has a 0.1″ pin spacing.

I²C Communication

The LSM6DS33's gyro and accelerometer, the LIS3MDL's magnetometer, and the LPS25H's barometer can be queried and configured through the I²C bus. Each of the four sensors acts as a slave device on the same I²C bus. A detailed explanation of the protocols used by each device can be found in the LSM6DS33 and LIS3MDL datasheets (also available on Moodle).

The LSM6DS33 and LIS3MDL each have separate slave addresses on the I²C bus. The following table shows the slave addresses of the sensors:

Sensor	Slave Address (default)
LSM6DS33 (gyro and accelerometer)	1101011b (0x6B)
LIS3MDL (magnetometer)	0011110b (0x1E)

To connect the IMU sensor to BBB:

1. Study principles of general-purpose input/output (GPIO) interfacing to BBB in Chapter 6, pp. 201-203.
2. Study relevant section of Chapter 8, pp. 275 - 290 and connect the IMU to the BBB board in a similar way as in Figure 8-1 (p. 277) using the following connections:

   SCL – P9_19
   SDA – P9_20
   GND – P9_1
   VIN – disconnected
   VDD – P9_3

   Note: The textbook is using a different IMU sensor, ADXL345, as a reference. We have AltIMU-10 v5. Physical connection to BBB remains the same, however some of the register addresses are different. Please refer to the relevant datasheets to get correct register addresses.

3. Follow the textbook (pp. 280-283) and test i2c-tools: **i2cdetect** and **i2cdump** in the BBB terminal.

4. Observe that the IMU gyro, accelerometer and magnetometer sensors are all off by default and do not provide orientation measurements using **i2cget** command (pp. 283-285). Use sensor datasheets to learn correct data register addresses, e.g., 8-bit registers **OUTX_L_G (0x22)** and **OUTX_H_G (0x23)** provide 16-bit X-axis gyro measurement data (LSM6DS33 sensor). The register reading commands examples are **i2cget -y 2 0x6B 0x22** and **i2cget -y 2 0x6B 0x23**. Answer in the report why is value 2 used in the above commands?
5. To turn the IMU sensor modules on, configuration registers must be set to the normal mode of operation. This can be done by sending control words to the corresponding configuration registers as given in the table below using **i2cset** command (p. 285).

LSM6DS33		**LIS3MDL**	
Register (address)	Control word	Register	Control word
CTRL1_XL (0x10)		CTRL_REG1 (0x20)	
CTRL2_G (0x11)		CTRL_REG2 (0x21)	
CTRL7_G (0x16)			
CTRL8_XL (0x17)			
CTRL9_XL (0x18)			

Fill in the table above and provide control words in the report. Learn the meaning of the control words using sensor datasheets and modify if necessary, the corresponding accelerometer, magnetometer and gyroscope sensor control registers and set the sensors to following settings:

	Gyroscope	**Accelerometer**	**Magnetometer**
Data output frequency	50Hz	50Hz	6.25Hz
Range of output	± 245 deg/sec	$\pm 2g\ (2\ x\ 9.81 m/s^2)$	$\pm 4\ gauss\ (2x10^{-4} Tesla)$

6. Use the program example from Listing 8-1 on pp. 286 - 287 and write a program for setting the configuration registers and reading measurement data from x, y and z axes of all of the IMU onboard sensors (gyroscope, accelerometer, magnetometer). Note that the 16-bit data word for each measurement axis is obtained by combining readings from two 8-bit data registers (low and high).
7. Using the IMU sensor datasheets you are required to prepare a register map similar to Table 8-2 on p. 283 of the textbook containing information about configuration and data registers. Please explain the meaning of the sensor control words in the report.

TASK 5. IMU SIGNAL PROCESSING (20 POINTS)

1. Study, download, and implement an open-source IMU sensor measurement fusion algorithm in C code developed by Sebastian Madgwick from http://www.x-io.co.uk/open-source-imu-and-ahrs-algorithms/
   The algorithm outputs orientation estimations in the quaternion form. Study the self-study lecture on quaternions in Moodle. More information about quaternions can be found in Internet, for example at http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-17-quaternions/
2. Study the Magwick IMU signal fusion algorithm carefully and define in what physical values the algorithm accepts the input sensor data. In your IMU sensor read programs **convert the sensor raw measurements to physical values using scaling factors defined based on the maximum measurements' values and the IMU sensor settings set above**. Write your calculations in the report.
3. Interface your IMU sensor measurement data program with the algorithm code and obtain orientation estimations in the terminal window when running on the BBB.

TASK 6. IMU GUI DEVELOPMENT (20 POINTS)

Related reference textbook material:

- *Chapter 10. The Internet of Things*
- *Chapter 11. BeagleBone with a Rich User Interface*

In this task you will develop a client/server application to visualize an IMU sensor orientation on your laptop/PC. The BBB with connected IMU sensor acts as a server and sends processed orientation data to your laptop/PC (client). The client runs a Qt GUI for visualizing the IMU sensor 3D orientation. Please implement the following:

1. Study the **C++ Client/Server** section on pages 412-415.
2. Study the Qt GUI application development in Chapter 11 of the textbook**.** Focus on the **Remote UI Application Development** section **(Fat-Client Qt GUI Application** subsection) on pages 455-462. Example program codes from Chapters 10 and 11 can be downloaded from the textbook GitHub page.
3. Implement on the **server side (BBB)** the code with the following functions:
   - Reading the IMU raw sensor measurement data and converting it to physical values as explained in Task 4.
   - Processing IMU sensor data with the open-source Magwick IMU sensor measurement fusion algorithm and sending the output sensor orientation quaternion data to the server socket communication for reading by the client side. **Please make the quaternion component normalization before sending the data to the algorithm (Task 5-2).**

     The textbook describes the use of the Apache web server embedded into the BBB Debian image. You may use alternative socket communication servers such as NodeJS, etc. if preferable.

4. On the **client side (your laptop/PC)** you will implement GUI application in the Qt Creator which handles:
   - the client socket communication. It reads the data.
   - runs the data processing and visualizes IMU orientation.

Please develop the IMU sensor orientation visualization in the Qt GUI using the provided on Moodle **test-opengl** framework based on OpenGL and QTthread classes. You may use alternative visualization tools, e.g. QtCanvas3D or others, if preferable.

You may follow the online C++ GUI with Qt Tutorial videos #28 to #35 (if deemed necessary) from https://www.youtube.com/watch?v=JaGqGhRW5Ks for learning how to work with QThread class in the Qt.

GRADING CRITERIA

The total grade for all implemented assignments described will constitute 90%.

The bonus 10% of the grade will be awarded to the groups showing outstanding coding experience in the form of modifications/added complexity of Task 6, and project report completeness and quality, etc.

Please prepare a detailed report with program code and submit it to the project folder in Moodle by the end of Sunday, 2 March.

This project evaluation may be done using individual grading depending on the level of participation and understanding of the project assignments.

Late submission penalty – 10% per day