HARDWARE-VIRTUAL ENVIRONMENT INTEGRATION

F. Y. Annaz and H. K. Wazir

Institut Teknologi Brunei, Electrical & Electronic Engineering Department, Faculty of Engineering, Jalan Tungku Link, Gadong, BE 1410, Bandar Seri Begawan, Brunei Darussalam, Email: fawaz.annaz@itb.edu.bn, hassamkhanwazir@yahoo.com

Keywords: UAV, virtual environment, robot navigation, object orientation

Abstract

Testing Unmanned Aerial Vehicles (UAVs) while ensuring safety and still maintaining efficiency seems to be an important issue for researchers due to the unstable nature of the flying platforms. Computer simulations are a popular choice, but they are unable to provide accurate real-time hardware data. Thus, researchers have to rely on the software equivalents of approximate hardware models.

This paper introduces the concept of real rotary-winged UAV hardware testbed integration with a Virtual Environment (VE), where navigation within the virtually created environment is possible. During navigation, Inertial Measurement Unit (IMU) data is sent wirelessly to the VE in real-time. The orientation data from the IMU is processed by a microcontroller to calculate the Pitch, Roll and Yaw angles. The data is transmitted wirelessly using a pair of RF modules to the virtual environment created using the Unity engine, running on a computer. The data can then be used to navigate a virtual UAV inside the virtual environment.

The testbed will help researchers to emulate flying in different geographical scenarios and test algorithms to their limits without the risk of damaging UAV.

1 Introduction

Evaluation of monitoring and control algorithms onto Unmanned Ground and Ariel Vehicles (UGVs and UAVs) can be costly, particularly when random host environments are considered. Although pure computer simulations are popular alternatives, they naturally rely on approximating hardware models that may compromise accuracy. Here, we propose Hardware-Virtual-Environment Coupling (HVEC) as a better alternative, as it does not compromise accuracy, yet allows for efficient evaluation of monitoring and control algorithms in randomly dynamic varying environments. The authors in [1, 2, 3] have implemented this approach and have developed virtual environments to navigate ground robots as well as UAVs inside the virtually created maps.

This approach of using virtual environments for hardware testing has been used in studies related to autonomous and remote controlled helicopter and multi-copter tracking. The studies range from obtaining real-time orientation and position data from the Unmanned Aerial Vehicles (UAVs) to building more accurate UAV models in order to create learning tools and virtual environments for UAV platform testing and flight training.

In this paper, real-time 3D object orientation sensing was integrated to a virtually created environment, which was custom built using the Unity game engine [4]. The VE can be exported to multiple platforms, including Windows, Mac OS X, and Linux.

2 TestBed Hardware Architecture

Robot tracking and control algorithms evaluation can be taxing, complex and costly when using conventional methods. Furthermore, it can be potentially dangerous when very big robots and UAVs are considered. To minimize these risks, different methods were used to provide controlled environments to test for algorithms functionality and robot deployments in random environments. Examples of such new methods are those that considered UGVs [1, 2, 3], where mobile robots and there motion were coupled with VEs. As for UAVs, testbeds were used to provide controlled test environment to observe physical behaviour [5, 6, 7]. The testbeds allowed for manoeuvres in restricted space, which enabled researchers the ability to observer the movement of the actual hardware instead of approximate equivalents.

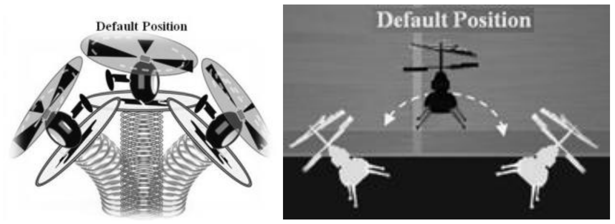


Figure 1. Proposed hardware-VE design

The study here expands on the previously mentioned UGVs navigation in a 2D-VE, by considering UAV navigation in 3D-VE, and proposes a new testbed platform that is wirelessly connected to the 3D-VE, as proposed in Figure 1. The UAV is mounted on top of the testbed, which allows for limited (yet unrestricted) 3D physical motion that is mapped in real-time onto a VE. The VE is a user created 3D environment depicting real environments or structures, in which the real-time hardware manoeuvres are mapped. Thus the end result is a HVEC where the UAV show realistic movement as it navigates through various types of created topographies.

This paper primarily focuses on the hardware side of the testbed development, including the various associated sensors, the calculated motion and orientation and the way they are wirelessly transmitted and mapped onto the VE.

The hardware used (shown in Figure 2) includes an Arduino board (on the right), which is solely used to power the other following circuits:

- Inertial Measurement Unit (IMU): This is a Razor 9DOF IMU, which consists of a tri-axis digital Accelerometer, a tri-axis digital Gyroscope and a tri-axis digital Magnetometer [8]. It is used to determine the orientation of the testbed by processing the sensor data inside the on-board microcontroller.
- A microcontroller: The microcontroller calculated the Pitch, Roll and Yaw orientation data.
- A Radio Frequency (RF) transceiver module: a pair of XBee Pro S1 RF transceivers are used to serially transfer the orientation data to a computer,
- Computer: This can be any recent desktop/laptop model that hosts and maps the real-time manoeuvre onto the running VE.

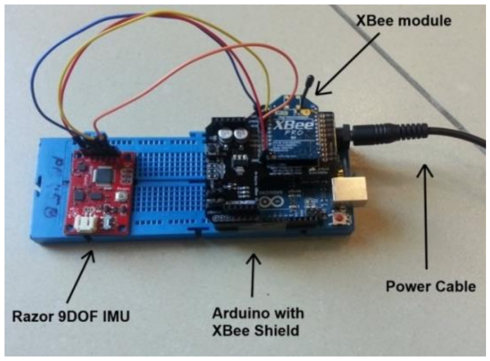


Figure 2. The hardware design

3 Sensor Fusion

Although gyroscopes or accelerometers individually calculate the orientation of rigid bodies, however, over time:

- Gyroscopes drift, resulting in error accumulation and the inaccuracies in the final output values; and
- Accelerometers provide fairly reliable output data, however, they are prone to noise distortion, thus give noisy output signals that affect precision. Furthermore, since accelerometer rotations around the x-axis and y-axis are calculated in relation to the z axis, accelerometers cannot be used to calculate the rotation around the z axis.

Therefore, IMUs (Attitude and Heading Reference System, AHRS) used to determine the orientation and heading of a UAV can comprise of different combinations of accelerometers, gyroscopes, magnetometers and Global Positioning System (GPS) modules. The IMU used in this study consists of tri-axes accelerometer, gyroscope and magnetometer. Figure 3 shows the IMU data acquisition process.

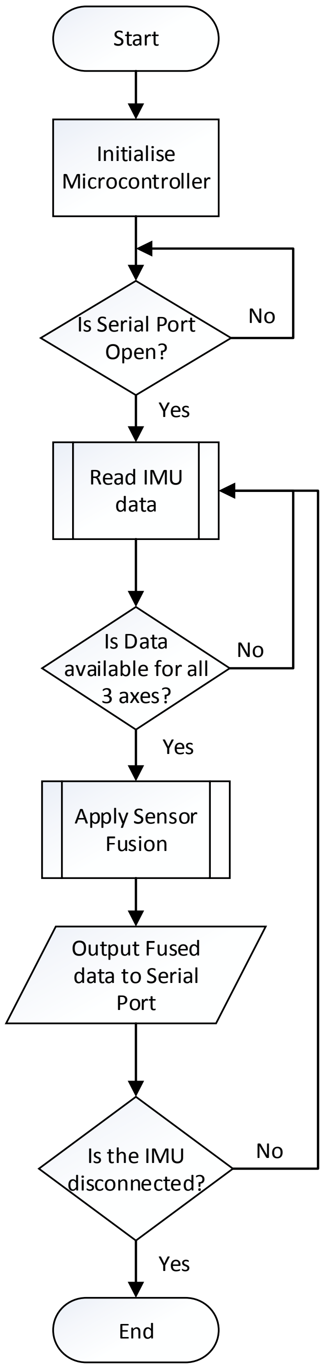


Figure 3. The process of IMU data acquisition

In this research, the Premerlani and Bizard approach is adopted, using the Direction Cosine Matrix (DCM) to calculate corrected sensors data by correcting the gyro drift through the accelerometer value and determine the heading using the magnetometer [9]. The approach is based on a modified version of the Robert Mahony's complimentary filter design and incorporates sensor fusion into the hardware architecture [10]. The filter uses the rotation matrices that describe the orientation of one coordinate system with respect to another, as shown in Figure 4.

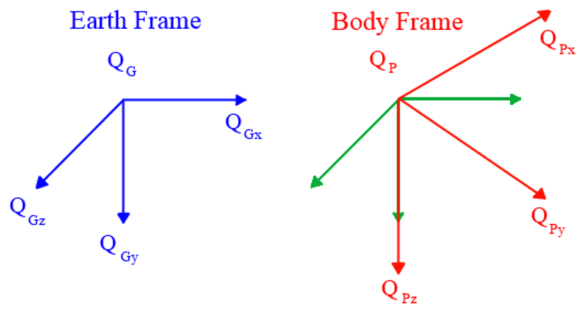


Figure 4. Earth and body frame of references

4 Direction Cosine Matrix

Vectors such as accelerations, velocities, translations and directions can be transformed from one frame of reference to another by multiplying them with a 3x3 matrix. Here, we are mainly concerned with the UAV body frame of reference and the earth frame of reference. The transformation between the two frames maybe explained by considering the following vectors:

$$\boldsymbol{Q} = \begin{bmatrix} Q_x \\ Q_y \\ Q_z \end{bmatrix} = \text{the direction, velocity or acceleration vector}$$

$$\boldsymbol{Q_P} = \text{Q being measured in the body frame of reference(BFOR)}$$
$$\boldsymbol{Q_G} = \text{Q being measured in the earth frame of reference(EFOR)}$$

$$\boldsymbol{R} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} \text{ is the transformation matrix} \qquad (1)$$

$$\text{Therefore, } \boldsymbol{Q_G} = \boldsymbol{RQ_P}$$

The Euler angles are related to the direction cosine matrix as follows:

$$\boldsymbol{R} = \begin{bmatrix} cos\theta\cos\psi & \sin\varphi sin\theta\cos\psi - cos\theta sin\psi & \cos\varphi sin\theta\cos\psi + sin\varphi sin\psi \\ cos\theta sin\psi & sin\varphi sin\theta sin\psi + \cos\varphi cos\psi & cos\varphi sin\theta sin\psi - sin\varphi cos\psi \\ -sin\theta & sin\varphi cos\theta & \cos\varphi cos\theta \end{bmatrix} \qquad (2)$$

Equation (1) is the direction cosines of the rotation vector, measured in the body frame of reference with respect to the earth frame of reference. Equation (2) describes the same situation in terms of Euler angles. To sum up the previous information, equation 1 can be restated as follows:

$$\left.\begin{aligned} \boldsymbol{Q}_{Gx} &= \boldsymbol{r}_{xx}\boldsymbol{Q}_{Px} + \boldsymbol{r}_{xy}\boldsymbol{Q}_{Py} + \boldsymbol{r}_{xz}\boldsymbol{Q}_{Pz} \\ \boldsymbol{Q}_{Gy} &= \boldsymbol{r}_{yx}\boldsymbol{Q}_{Px} + \boldsymbol{r}_{yy}\boldsymbol{Q}_{Py} + \boldsymbol{r}_{yz}\boldsymbol{Q}_{Pz} \\ \boldsymbol{Q}_{Gz} &= \boldsymbol{r}_{zx}\boldsymbol{Q}_{Px} + \boldsymbol{r}_{zy}\boldsymbol{Q}_{Py} + \boldsymbol{r}_{zz}\boldsymbol{Q}_{Pz} \end{aligned}\right\} \qquad (3)$$

Since the rate of change of a rotating vector due to its rotation is given by:

$$\frac{d\boldsymbol{r}(t)}{dt} = \boldsymbol{\omega}(\mathrm{t}) \times \mathbf{r}(\mathrm{t}) \qquad (4)$$

$$\text{Where, } \left\{\begin{aligned} &\boldsymbol{\omega}(\mathrm{t}) = \text{rotation rate vector} \\ &\frac{d\boldsymbol{r}(t)}{dt} = \boldsymbol{\omega}(\mathrm{t}) \times \mathbf{r}(\mathrm{t}), \text{ and} \\ &\boldsymbol{\omega}(\mathrm{t}) = \text{rotation rate vector} \end{aligned}\right.$$

If the initial conditions of a rotation vector are known, along with its time history, equation 3 can be numerically integrated to track the rotating vector:

$$\boldsymbol{r}(t) = \boldsymbol{r}(0) + \int_0^t d\boldsymbol{\theta}(\tau) \times \mathrm{r}(\tau) \qquad (5)$$

Since in the OFOR, the EF of rotating equal and opposite to the rotation of the object in the EFOR, the earth axes can be tracked as seen in the OF by flipping the sign of the gyro signals. For convenience, the sign can be flipped back and the factors in the cross product can be interchanged:

$$\boldsymbol{r}_{earth}(t) = \boldsymbol{r}_{earth}(0) + \int_0^t \boldsymbol{r}_{earth}(\tau) \times d\boldsymbol{\theta}(\tau) \qquad (6)$$

$$\text{Where, } r_{earth}(t) \text{ is an earth axes, viewed from the object}$$

Taking the Mahony approach, equation 6 can be written back in its differential form.

$$\left.\begin{aligned} r_{earth}(t+dt) = r_{earth}(t) + r_{earth}(t) \times d\theta(t) \\ d\theta(t) = \omega(t)dt \end{aligned}\right\} \qquad (7)$$

Finally, the correction rotation rate that comes out of the proportional plus integral drift compensation feedback controller can be added to the gyro output, to produce the best estimate of the true rotation rate.

$$\left.\begin{aligned} \boldsymbol{\omega}(t) = \boldsymbol{\omega}_{gyro}(t) + \boldsymbol{\omega}_{correction}(t) \\ \boldsymbol{\omega}_{gyro}(t) = \text{three axis gyro measurements} \end{aligned}\right\} \qquad (8)$$

When we repeat equation 6 for each of the earth axes, the result can be put into a convenient matrix form:

$$\boldsymbol{R}(t+dt) = \boldsymbol{R}(t) \begin{bmatrix} 1 & -d\theta_z & d\theta_y \\ d\theta_z & 1 & -d\theta_x \\ -d\theta_y & d\theta_x & 1 \end{bmatrix} \qquad (9)$$

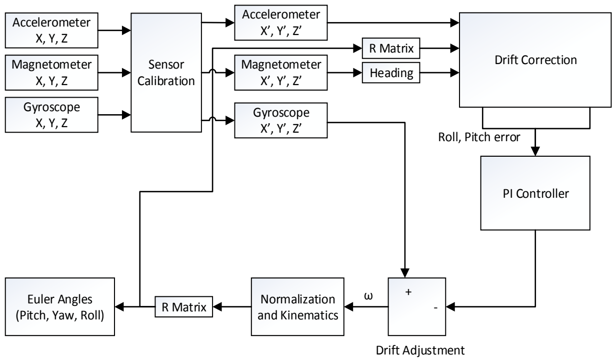


Figure 5. Block diagram explaining the filter

5 Filter Overview

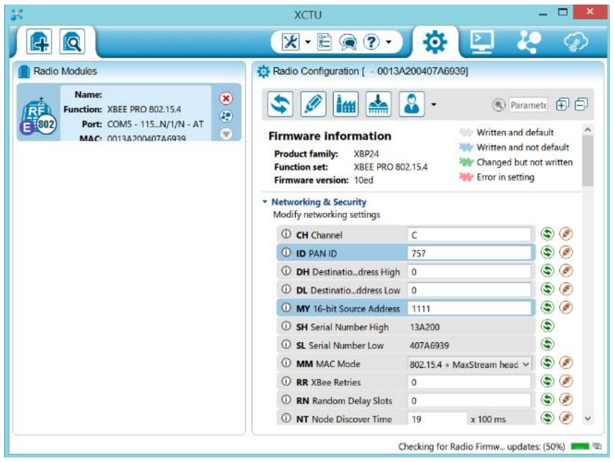
The filter (shown in Figure 5) considers the gyroscope sensor measurements as the primary source of data orientation due to its fairly accurate pitch and roll values. Accelerometer and magnetometer are used to calculate a reference vector which can be compared to the orientation vectors of the object in order to determine the object's orientation with respect to the earth frame of reference. Since the gyro readings suffer from drift over time, the X and Y directions accelerometer readings can be used as a reference. The magnetometer is used to calculate the heading of the object. The pitch and roll readings obtained from the accelerometer coupled with the yaw reading from the magnetometer are used to calculate a correction matrix. The algorithm, later on, uses a proportional plus integral feedback controller on the correction matrix to compensate for the gyro drift.

The corrected gyro readings denoted by ω, are then fed to the "Normalisation and Kinematics" block which converts the vectors in the rotation matrix to unit vectors before calculating the kinematics. Once this is accomplished, the gyroscope along with the previous rotation matrix is used to calculate the current rotation matrix (R matrix) by using equation 9. Finally, the Euler angles are calculated from the updated rotation matrix.

6 Data Transmission

The hardware is connected to the VE through a RF link using a pair of XBee modules [11]. The XBee modules transmit and receive data, based on the source and destination addresses programmed into its firmware, using the XCTU software provided by Digi, Figure 6. Once the firmware is programmed and the baud rate of the RF modules matches that of the serial port of the microcontroller, the data can be transmitted wirelessly from the hardware circuit to the computer containing the VE.

Once the data is received by the RF module (that is connected to the computer), the VE can parse the IMU data by reading the serial port that can be configured inside its Graphical User Interface (GUI). By default, the data will only be parsed properly if the serial data stream coming from the hardware follows a certain sequence. However, if the data sequence from the hardware is not correct, this problem can be corrected by modifying the data parsing sequence inside the virtual environment. By default, the virtual environment accepts Roll, Yaw and Pitch value separated by a comma. An example of serial data stream can be seen in the Figure 7.



Name:
Function: XBEE PRO 802.15.4
Port: COM5 - 115...N/1/N - AT
MAC: 0013A200407A6939

Firmware information
Product family: XBP24
Function set: XBEE PRO 802.15.4
Firmware version: 10ed

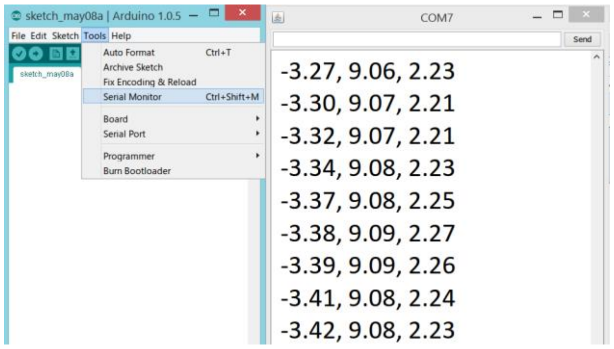
Written and default
Written and not default
Changed but not written
Error in setting

Networking & Security
Modify networking settings

CH Channel	C
ID PAN ID	757
DH Destinatio...dress High	0
DL Destinatio...ddress Low	0
MY 16-bit Source Address	1111
SH Serial Number High	13A200
SL Serial Number Low	407A6939
MM MAC Mode	802.15.4 + MaxStream head
RR XBee Retries	0
RN Random Delay Slots	0
NT Node Discover Time	19 x 100 ms

Checking for Radio Firmw... updates: (50%)

Figure 6. XBee module firmware information in XCTU



```
-3.27, 9.06, 2.23
-3.30, 9.07, 2.21
-3.32, 9.07, 2.21
-3.34, 9.08, 2.23
-3.37, 9.08, 2.25
-3.38, 9.09, 2.27
-3.39, 9.09, 2.26
-3.41, 9.08, 2.24
-3.42, 9.08, 2.23
```

Figure 7. Sensor data obtained from the IMU observed through serial monitor of the Arduino IDE

7 Hardware-Software Integration

The GUI of the virtual environment provides the user with connectivity options such as Port selection and enabling/disabling hardware connection. A simplified version of the system architecture of the virtual environment is shown in Figure 8.

Once the hardware is properly set up, the orientation data calculated by the microcontroller using sensor fusion, which is transmitted to the computer via the RF modules. On the software side, once a map is generated (either built using the build menu or imported), and the COM port settings are configured (in the options menu), the user may enters the "Navigate" mode and starts navigating the UAV through the VE after pressing the "connect" button. This opens the COM port allowing serial data stream to enter the software, where it is interpreted and converted into Euler angles (roll, yaw and pitch rotation) of the virtual UAV inside the VE. As the IMU is rotated around, the orientation of the virtual UAV is updated in real-time, giving the user direct control over the flight of the UAV.

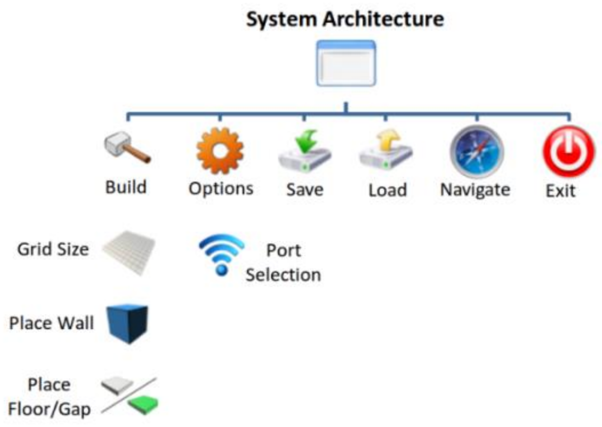
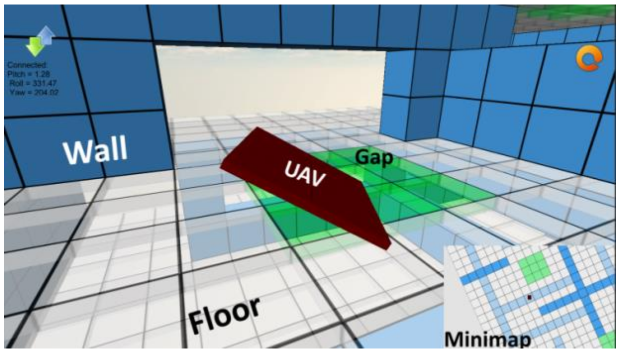


Figure 8. Virtual environment GUI

The authors in [12] give a full account of the virtual environment development and hardware navigation within the environment. During navigation, real-time data exchange between the hardware and the virtual environment takes place, as shown in Figure 9.



```
Connected
Pitch = 1.28
Roll = 331.47
Yaw = 204.02
```

Figure 9. UAV navigation inside the virtual environment

8 Conclusion

The concept of real-time UAV navigation inside a VE was introduced. The paper mainly focused on the implemented hardware and explained the different mechanisms used to read, interpret and apply the data acquired from the sensors and incorporate in the virtual environment.

The hardware was simplified with the view that it can easily be modified so that it is easily integrated with various testbed designs. The circuit has very low power consumption with real-time data transfer ability.

The study focused on the prospect of rapid prototyping and efficiency to hardware and software development. The fact that a custom built VE was used, opens up new future possibilities, as it will be easier to modify and improve the software according to researchers needs.

References

[1] F. Y. Annaz, "Path-Whispering in a Virtual Environment", *International Review of Mechanical Engineering (IREME),* **Vol. 5**, No. 5, 2011.

[2] F. Y. Annaz, "A Mobile Robot Solving a Virtual Maze Environment", *International Journal of Electronics, Computer and Communications Technologies, IJECCT,* **Vol. 2**, No. 2, pp. 1-7, Jan 2012.

[3] F. Y. Annaz, "Real Time Robot Navigation in Virtually Created Environments", *International Journal of Electronics, Computer and Communications Technologies, IJECCT,* **Vol. 3**, No. 4, pp. 7-13, Jul 2013.

[4] "Unity Game Engine", Unity Technologies, *https://unity3d.com/.*

[5] C. Cheron, A. Dennis, V. Semerjyan and Y. Chen, "A multifunctional HIL testbed for multirotor VTOL UAV actuator", *Mechatronics and Embedded Systems and Applications (MESA),* IEEE/ASME International Conference, 2010.

[6] T. Oliveira, G. Cruz, E. Marques and P. Encarnaçao, "A test bed for rapid flight testing of UAV control algorithms", *RED-UAS*, 2011.

[7] P. Twu, R. Chipalkatty, J.-P. d. l. Croix, T. Ramachandran, J. Shively, M. Egerstedt, A. Rahmani and R. Young, "A hardware testbed for multi-uav collaborative ground convoy protection in dynamic environments", *AIAA Modeling and Simulation Technologies Conference*, 2011.

[8] "9 Degrees of Freedom - Razor IMU", *Sparkfun, [Online]. https://www.sparkfun.com/products/10736.*

[9] W. Premerlani and P. Bizard, "DCM IMU Theory: First Draft", [Online]. *http://diydrones.com/profiles/blogs/dcm-imu-theory-first-draft.*

[10] R. Mahony, T. Hamel and J.-M. Pflimlin, "Complementary filter design on the special orthogonal group SO(3)", *44th IEEE Conference on Decision and Control, and the European Control Conference* 2005, Seville, Spain, 2005.

[11] "RF Modules", [Online]. *http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/.*

[12] H. K. Wazir and F. Y. Annaz, "Using Unity for 3D Object Orientation in a Virtual Environment", *Proceeding of Brunei International Conference on Engineering and Technology, Institut Teknologi Brunei, November 1-3*, Brunei Darussalam, 2014.