A Wireless Multivariable Control Scheme for A Quadrotor Hovering Robotic Platform using IEEE® 802.15.4

Zaid Al-Khatib, Jaime Yu, Hasan Ghazi Al-Khakani, Samer Kombarji

Faculty Mentor: Dr. Amir G. Aghdam

***Abstract*—Quadrotor vehicles have seen increasing interest in the industry and academia in recent years, for their mechanical design simplicity, instability / maneuverability, and high payload capabilities. However, due to the same reasons, successful implementations of such systems were limited to those groups with extensive experience with control systems theory. Therefore, the team set a goal of developing a flexible robotic autonomous platform that abstracts the different levels of control, providing users from different backgrounds access to the functions they wish to work with. Connecting the high level controller of the robot to operator(s) is done via a flexible mesh network that is built using the IEEE® 802.15.4 and ZigBee® industrial standards. This paper discusses the application of these standards in the project, the system overview and network topology, justification for the use of these standards, implementing the networking protocols on each system component, and finally a summary of networking tests and results.**

***Index Terms*— Key Words: Robotic platform, Quadrotor, VTOL AMAV, IEEE 802.15.4, ZigBee, Mesh Network.**

I. INTRODUCTION

IN the past few years, manufactures have introduced a large number of low cost MEMS sensors and brushless motors which encouraged a large number of aerospace enthusiasts to attempt building Mirco Aerial Vehicles (MAVs). However, many were soon discouraged by the complexity of the control theory and filtering methods for sensor outputs [1-3]. Between early 2007 and up to the publishing of this paper, mid 2010, the number of university projects attempting to build quadrotors has increased significantly. This can be easily seen in the number of published papers in this time and the online videos posted by these groups. Almost all of the ones that had been able to achieve stable flight had people who are experienced with control theory and Unmanned Aerial Vehicles (UAVs). Interesting concepts and applications of using the quadrotor for mapping and other novel ideas are discouraged as a result of the difficulties of controlled flight [4-7]. This project hopes to push the MAV concept to non technical savvy users by developing a hovering robotic quadrotor platform that communicates with a main workstation and a number of controllers and peripherals. This report will discuss the wireless communication and networking aspect of the project and the application of the industry standard IEEE® 802.15.4 in doing so.

II. SYSTEM OVERVIEW

The Qx4 robotic platform system is mainly comprised of the quadrotor platform, a workstation and a number of manual controllers. These components are all nodes in a ZigBee® mesh network as shown in figure 1. This section will give a brief overview of each of these components.

A. ZigBee® Mesh Network

Connecting the Qx4 quadrotor platform, workstation, and controllers is a wireless mesh network established using XBee® RF[1] modules. The Qx4 quadrotor platform broadcasts general telemetry over this network for the workstation to display to the operator, and for peripherals that allow information display to make use of this data. This network also allows the operator to send high level commands to the platform using the workstation. Furthermore, the manual controllers can request control over the robot from the workstation, and in case permission is granted, the workstation instructs the robot to listen to and process commands from the granted controller. Essentially, the network is the main method to connect more than one person to the robotic platform.

Fig. 1. System overview – system components.

B. *Qx4 Quadrotor Robotic Platform*

The Quad-rotor vehicle, the main component of the project, a '+' shaped carbon fiber frame with a motor and propeller placed on each one of the four ends of the axes, and a speed controller attached to each one of the axes. At its center, it houses three microcontroller boards, ten proximity sensors, and three 3-axes inertia sensors. A propeller guard made out of foam and mesh wire was created as an ad-on that can surround the propellers to protect the vehicle and its surroundings in case the vehicle comes in contact with a wall or a person. The guard also serves as proximity sensors mounts. These sensors are what allows it to be operable in an indoors, GPS denied environment, allowing the implementation of such functions as mapping, obstacle detection and avoidance, and position hold.

C. *Workstation and Main Controller*

The workstation is a set of software applications implemented to run on a PC that is connected to the wireless mesh network and acts as a controller of the vehicle. It allows the operator to send commands to the vehicle and to tune control constants. It also displays real time telemetry received from vehicle. The telemetry includes but is not limited to; yaw, pitch, roll, height, errors from Proportional Integral Derivative controller (PID), and the distance of obstacles around the vehicle. The workstation also manages the wireless mesh network by granting permissions to the peripherals, such as manual controllers and other computers, to take control over the vehicle.

D. *Secondary Manual Controller – Wii Nunchuck*

The team created a manual controller using a Wii Nunchuck for the project which is used primarily as a proof of concept. The Wii Nunchuck shows how easy it is to program peripherals for the system. It is used to control the quadrotor. The manual controller is also used to show the maneuverability of the quadrotor.

E. *Other Controllers Adapter Box*

To increase the time required to develop software that can interface with the robot, the team created an external controller adapter box. It consists of a microcontroller and a wireless module which gives the developer a simple serial interface with very simple commands that it can receive and translate into data packets that are understandable to the mesh network. The intent of this box is to speed up the development third party components that could serve as a controller for the quadrotor.

III. WHY IEEE® 802.15.4 AND ZIGBEE® FOR THE NETWORK

In selecting the wireless communication technology multiple factors had to be considered. The main use of the communication is to relay telemetry and high level commands, therefore high data rates are not critical. However multi-node support and low power consumption are important factors in the decision. Figure 2 shows some of the main wireless technologies often considered for indoors wireless requirements [9]. Furthermore, table 1 shows the comparison

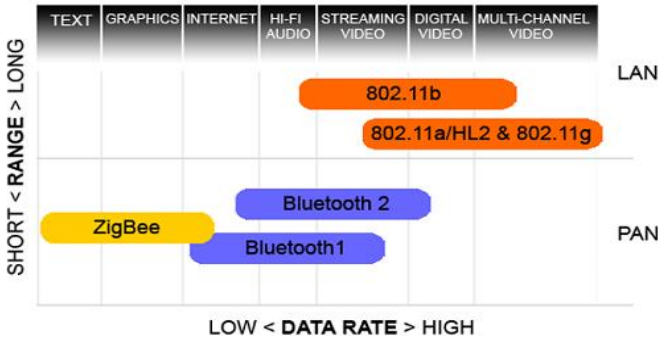

Fig. 2. **Comparison between different wireless technologies** [8]

of these technologies in addition to the cellular network (2.5G and/or 3G).

As shown in the comparison, ZigBee® is most appropriate technology for this application. That is because of its inherent support of multi-node networking, reliable communication, and very low energy consumption that the PHY and MAC implementation using the IEEE® 802.15.4 standard provides. ZigBee®, which is based on the above-mentioned standard, gives further levels of reliable communication and support for mesh networking which is desired. Even though ZigBee® fails to compete with the other technologies such as in range, throughput and data rate, it still is favorable because these aspects are not needed for telemetry and commands. In case an operator decided to add a sensor that requires high data rates, it is recommended that a second dedicated wireless module to be added with it. This is done to increase reliability and safety by ensuring that the mesh network's bandwidth is dedicated to real time telemetry and controls.

The ZigBee modules used do not emit high power RF signals. This is good for battery life and for the reliability of radio frequency sensitive electronics. The most affected of those is the state estimation controller. Tests showed the state outputs were seriously corrupted when any wireless transmitter gets to a few centimeters away from it. The results of the testing showed that a minimum distance of 4 centimeters was required between the state estimation controller and the ZigBee for the proper functioning of the state estimation controller.

TABLE I
ALTERNATIVES EVALUATION FOR THE WIRELESS CONNECTION

Aspects	***Weight***	*Bluetooth®*	*ZigBee®*	*Wifi®*	*Cellular*
Multi-node network support	**100**	5	10	10	10
Throughput	**60**	7	6	8	3
Data rate	**60**	7	6	10	10
Range	**50**	6	5	7	10
Ease of implementation	**50**	6	8	6	4
Power consumption	**-80**	6	2	8	6
Cost	**-100**	5	3	7	8
Total		460	910	390	200

Fig. 3. xBee® wireless ZigBee® [9]

The ZigBee® wireless modems used in the project are the xBee® modules from Digi [9]. The xBee® hardware consumes an average 50 mA and can support up to 115200 bps[2]. It was desired because of its reasonable price, low power, small footprint, light weight, and hardware upgradability. A higher power model can be easily dropped in the place of the current model. This also gives the user the ability to customize the quadrotor's range for different missions by easily replacing the wireless module. The module used is shown in figure 3.

IV. The Network

The wireless mesh network uses ZigBee® hardware which is based on the IEEE® 802.15.4 standard for wireless personal area network. The xBee® modules provided the team with the ability to broadcast and receive packets from the network using simple UART serial communication protocol. However, the serial protocol is not enough to send the amount of data required to the robot and back. Because the type of commands to the vehicle had to identify the source, a command action and value, the 1 –Byte communication provided by xBee® was not enough.

Therefore the team set out to create a communication protocol that ensures that data sent to the robot arrives accurately in clustered packets. That is to make sure that all parts of the command packet are delivered all together, or, in case one byte is dropped, the whole cluster is disregarded. The communication protocol's packet to send data to the robot is defined by a cluster of 6 bytes cluster of packets as shown in table 2.

The definition of each byte in the cluster is described as follows:

1) Every valid communication packet start with the `start byte`. The start byte is set as the ASCII character '*' to provide legibility when debugging the communication protocol by eye.
2) The source ID refers to the ID of the sender. The workstation has the unique ID of the ASCII character 'a'. This is used by the robot to determine whether the workstation or another device is sending data to it. The source ID must fit within a byte of data, which limits the network to 256 unique devices that the platform can communicate with.
3) The checksum is a byte of data that contains the count of digital 1's that the entire package contains, omitting the 1's in the checksum and start, and end bytes. This provides an added level of confidence that the data that arrives to the AMAV is the correct data. In case the robot could not verify the checksum, it considers the packet as invalid and sends a notification indicating that fact. Note that the checksum is only a byte wide which means there can be a maximum of 255 digital 1's in the package. Since the package contains at most 40 digital 1's, the checksum is large enough to accommodate this.
4) Command ID is the byte that determines the type of action required of the robot. A summary of commands implemented at the moment are listed in table 3. The system is implemented with 26 commands, with the integer 0 not used. There is a maximum of 255 commands due to the size of the byte but only 26 commands are currently needed.
5) Command value byte is the byte that stores the value needed for the commands. As shown in table 3, some commands like take-off to height x, require the source to indicate the height. The units of the values have to be predefined on both the source and the robot.

TABLE 3
List of command IDs used in the network packet clusters

Command ID [in decimals]	*Command*
1	Halt
2	Take-off [to height x inches][2]
3	Land
4	Go-up/down [to height x inches][1]
5	Go-front [for x ms][2]
6	Go-right [for x ms] [2]
7	Go-back [for x ms] [2]
8	Go-left [for x ms] [2]
9	Rotate-clockwise [for x degrees] [2]
10	Rotate-counter-clockwise [for x degrees] [2]
11	Emergency land [attempt an immediate land without verifying landing location]
12	Emergency Halt [breaking all motors immediately]
[13 – 25]	Update PID constant [n] [1] to value [x (multiplied by][2]
26	Listen to controller ID x[2], [only valid if received from the workstation.]

TABLE 2
Components of a packets cluster

Byte[0]	**Byte[1]**	**Byte[2]**	**Byte[3]**	**Byte[4]**	**Byte[6]**
Start byte	Source id	Checksum	Command id	Command value	End byte

A. *The Qx4 Platform and Networking*

The Qx4 robotic platform is required to perform a number of hard real-time control functions. Those include filtering state estimation data from multiple sources at a high frequency, state control using PID controllers that command the four motors, and processing commands received from the mesh network. The team decided to use three microcontrollers onboard with distinct roles assigned to each as shown in the block diagram in figure 4. The control loops are divided amongst them as shown in figure 5.

As seen from these figures, the high frequency loops were assigned dedicated controllers to perform them. However, for the other computations such as obstacle avoidance and networking, a high level controller is used.

In the high level controller, the platform receives packets from the network, verifies their validity, stores them into a first-in-first-out queue, then interprets them one at a time and commands the low level controller accordingly. It is programmed to obey all valid commands received from the workstation, identified by the source ID. It can also serve commands received from the other controller, if and only if the workstation had commanded it to do so using the command number 26.

B. *The Workstation and Networking*

As the AMAV is fully autonomous, it does not need an operator to control its every move. The workstation provides the operator a simple set of buttons that will make the robot conduct an action such as move forward and move backwards. This is done for cases where manual control is desired. It also allows the operator to take control in cases where the platform behaves in an abnormal way.

1) *Administrator Panel*

The administrator panel is the primary controller of the robot, and the main interface on the work station. It is designed using Java to be cross platform, as long as Java is supported by the host machine. Figure 6 shows a screen shot of the administrator panel. It is clear that it provides users the ability to control the robot manually. It also works in the background to relay the telemetry from the robot over an Ethernet network to peripherals and secondary applications.

The administrator panel saves all telemetry data received from the AMAV in the comma separated file format which is natively supported by numerous spreadsheet applications, such as Microsoft Office's Excel. This helps in debugging and testing the system.

2) *Telemetry Graphing Application*

This application is written to accesses the telemetry data sent from the administration panel over the Ethernet. It then displays select telemetry data in figures in real-time. This is used to track the robot's telemetry over time and for testing new settings.

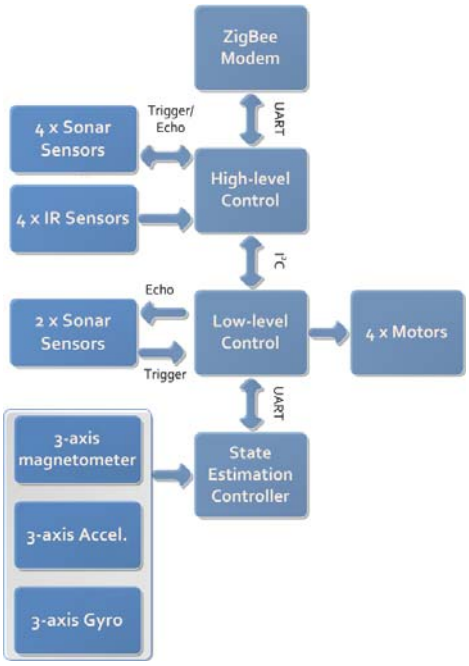

Fig. 4. Qx4 quadrotor robotic platform block diagram.

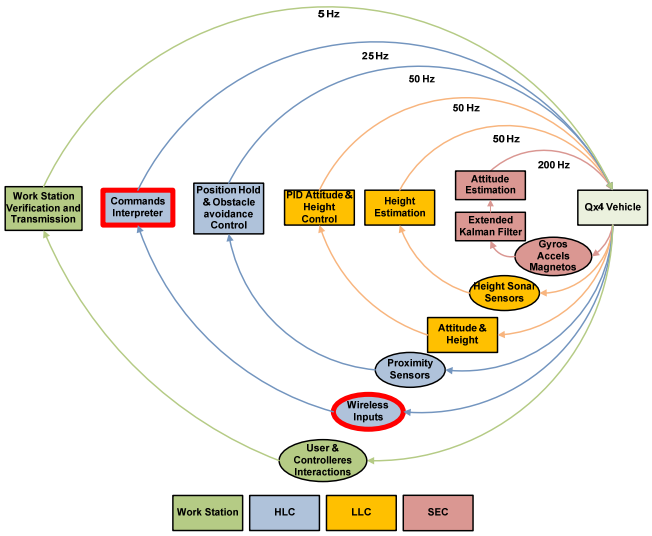

Fig. 5. Control loops as divided between the controllers, highlighting the wireless control loop

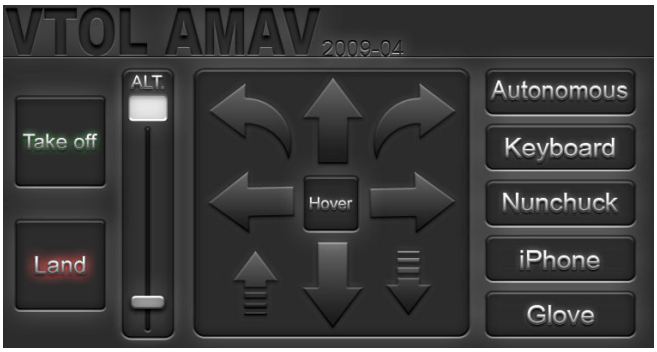

Fig. 6. Administration panel screenshot.

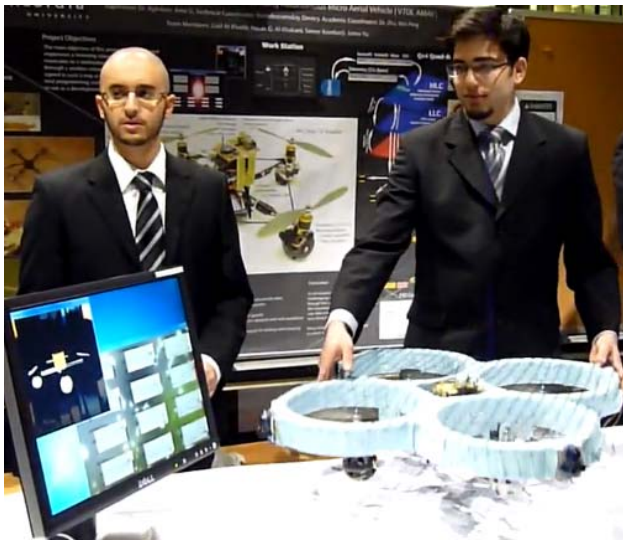

Fig. 7. Qx4 3D orientation and telemetry graphing application demonstration.

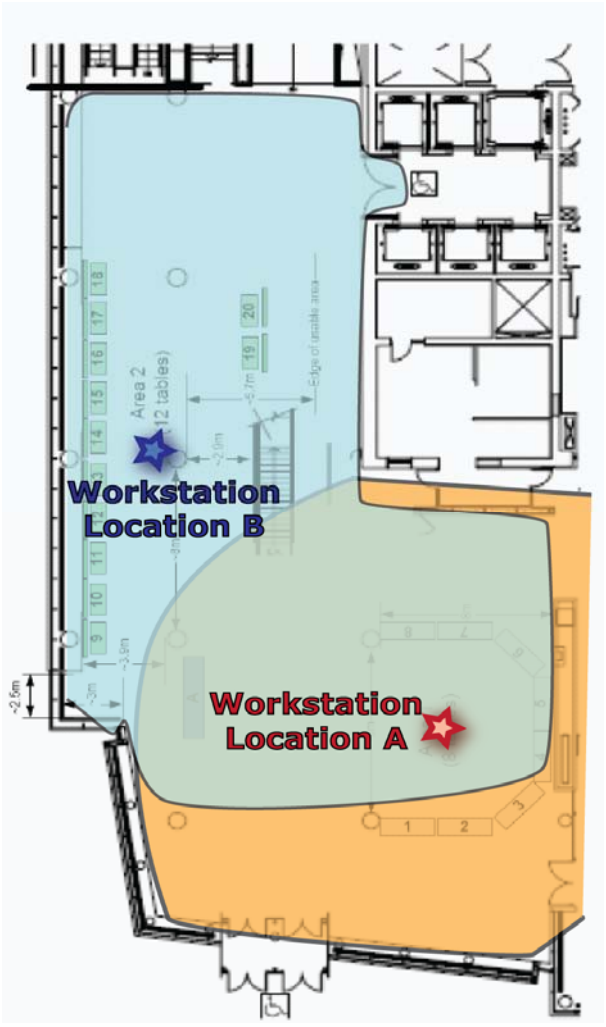

Fig. 8. Range test – the areas in which the wireless coverage met the requirements of the system [10].

3) The orientation Application

This application is written in to accesses the telemetry data over the Ethernet as well to show, in real time, the orientation of the quadrotor while in flight using a 3D graphical model. The application displays the model oriented in the same way the vehicle is oriented. This is done using the state telemetry broadcasted from the robot. Other peripherals that are part of this network can also use this data and display them in various formats. Figure 7 gives a picture taken from the demonstration day showing the team demonstrating this application by forcing the robot to tilt, with the model showing that orientation as well as the new motor speeds to counter that force and restore the state to a stable one.

4) The PID Application

This application is written to give the operator a tool to fine tune the control constants through a graphical interface over the wireless network. It issues packet-clusters that have a command action between 13 and 25 alongside a new value for a given constant. The robot uses this information to update its PIDs in real time over the wireless network while in hover mode. This application is also used to track and graph the errors used in the PIDs.

V. Network Testing and Results

A number of tests were performed for the overall system and robot. This section however will discuss the main tests performed on the wireless link and network.

A. Range Test

The first test the team implemented once the network was established is a range test in the building where the final demonstration will be held. Knowing the possible places for the demonstration are two main clusters of tables, the team examined the range around each group. A workstation and a laptop with an xBee® module each are used for the test. For each place, the workstation was positioned stationary in the middle of the group of tables, while a team member walked around with a laptop computer measuring the number of received and dropped packets. The goal is to find the areas in which the packets were delivered at least 99% of the time.

The areas found with this result were as expected from the specs of the wireless modules. That is about 30 meters for line of sight and about 10 meters for when obstacles and fading were present. The areas found are mapped as shown in figure 8.

B. Wireless Programming

The team originally wanted to implement wireless programming while it is in flight. The xBee module came with digital input-output pins that the group used to assert the reset bit of the microcontroller. The platform's multiprocessor architecture allowed it to be reprogrammed in flight while ensuring stability. It was also used to reduce time spent reprogramming the high level controller onboard since connecting it to cables can be cumbersome when the rotor guards are placed on. However, even though the team was able to program the high level controller wirelessly, this process was not reliable. The team ran a test to show its

TABLE 4
WIRELESS PROGRAMMING TEST RESULTS

Test number	*Result*	*Progress before sync error*
1	Fail	63%
2	Fail	45%
3	Pass	100%
4	Fail	89%
5	Fail	28%
6	Pass	100%
7	Pass	100%
8	Fail	91%
9	Fail	53%
10	Fail	72%

unreliability, the results of which are shown in table 4.

The table shows that wireless programming would only work 3/10 times. This proved the validity of the concept, however, the team decided to drop this feature due to time constraints.

VI. CONCLUSION

In conclusion, the team was able to build the system and meet all of the objectives of the project in a very timely fashion. The use of the industry standard IEEE® 802.15.4 wireless mesh networking lowered the amount of time it took for the team to complete the project. It further illustrated the importance of using industrial international standards in that it eases compatibility of systems designed separately, which is a very important practice. The IEEE® 802.15.4 and ZigBee standards were preferred early in the project for their low power, high reliability, small footprint and light weight as perfect candidate to base the network on. As expected, the standards lived up to the task when the team received the xBee® modems developed by Digi. The modules performed well in the target location, giving the system a very good range to demonstrate the application with very high reliability.

ACKNOWLEDGMENT

First and foremost, we would like to acknowledge our supervisor, Dr. Amir G. Aghdam for putting his trust in us on this very interesting and challenging project. His encouragement, advice and motivation helped us make it through the tough times during this project. We want to also acknowledge Mr. Dmitry for his time, vital encouragement and support

Also, we would like to acknowledge the following people who have been there for us for expert advice, assistance and support; Dr. Wei-Peng Zhu, Mr. Jeffry Landry, Mr. Dave Chu, Mohannad Al-Khatib, and Nick Major

Last but not least, we want to thank all of our family members, friends, colleagues, and loved ones for their warm support and enthusiasm throughout the development of the Qx4 project.

REFERENCES

[1] B. Cole, J. Cook, J. Forest, S. Johnson, E. Massie, and C. Rogers N. Carlos, "IARC Team Quadrotor," Virginia Tech University, Virginia, Project Report 2009.

[2] D. Korff, E. Gjioni, and H. Yang R. AbouSleiman, "The Oakland University Unmanned Aerial Quadrotor System," Oakland University, Oakland, Competition Report 2008.

[3] F. Lewis E. Stingu, "Design and Implementation of a Structured Flight Controller for a 6DoF Quadrotor Using Quaternions," *17th Mediterranean Conference on Control & Automation*, vol. 1, no. 17, pp. 1233-1238, June 2009.

[4] P. James, and J. Taylor E. Altug, "Control of a Quadrotor Helicopter Using Dual Camera Visual Feedback," *The International Journal of Robotics Research*, no. 24, pp. 329-341, 2005.

[5] G. Gremillion, B. Ranganathan, and J. S. Humbert J. Conroy, "Implementation of wide-field integration of optic flow for autonomous quadrotor navigation," *Springer Science+Business Media*, February 2009.

[6] T. S. Stirling, J. Zufferey, and D. Floreano J. F. Roberts, "Quadrotor Using Minimal Sensing For Autonomous Indoor Flight," *MAV07*, 2007.

[7] J. Chin, S. Mehrabian, L. Montejom, and H. Thompson C. Canetta, "Quad-rotor Unmanned Aerial Vehicle," Columbia University, Columbia, Project Final Report 2007.

[8] University of Luxembourg, SECAN-Lab. (2008, June) SECAM-Lab. [Online]. http://wiki.uni.lu/secan-lab/ZigBee+technology+in+sensor+network.html

[9] Digi International, *XBee®/XBee-PRO® RF Modules - IEEE® 802.15.4 RF Modules*. Minnetonka, MN: Digi International Inc., 2009.

[10] Department of Electrical and Computer Engineering - Concordia University, "Capstone Project Manual," Concordia University, Montreal, Canada, 2009.