DEVELOPMENT & IMPLEMENTATION of an QoS-AWARE ROUTING in WIRELESS SENSOR MESH AND MULTI-HOP NETWORKS

by

Yu Yang Long

Dissertation

Dissertation is submitted in partial fulfillment of
the requirement for the
Bachelor of Engineering (Hons)
(Electrical and Electronics Engineering)

December 2013

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

DEVELOPMENT & IMPLEMENTATION of an QoS-AWARE ROUTING in WIRELESS SENSOR MESH AND MULTI-HOP NETWORKS

by

Yu Yang Long

A dissertation submitted to the
Electrical & Electronic Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirements for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONIC ENGINEERING)

Approved by,

____

(DR MICHEAL DRIEBERG)
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
December 2013

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except in the references and acknowledgments, and that the original work contained herein have been undertaken or done by unspecified sources or persons.

____

YU YANG LONG

ABSTRACT

Wireless Sensor Network (WsN) is contributing as one of the most important roles in communication and data transfer nowadays. With the high demand in providing real time application in WSNs, quality of service (QoS) became the top priority in designing a real reliable, energy efficient, priority based and delay guarantee routing protocol. This paper emphasize on the selection of suitable routing protocol and implementation of the selected routing which leads to improvement on the selected routing protocol. In this project, the author will look into the various WsN routing protocol such as Sequential Assignment Routing (SAR), Message-initiated Constrained-based Routing (MCBR), Multi-Path and Multi-SPEED Routing (MMSPEED) and Energy Efficient and QoS Multipath Routing (EQSR) in order to choose the suitable routing protocol to be implemented. The selection of suitable routing protocol is purely based on the QoS metric where data priority, reliability, end to end delay, energy efficiency and network lifetime is taken into consideration. Before the implementation of selected routing protocol, the author will try and implement Ad-hoc On Demand Vector (AODV) routing protocol so that author can familiarize himself with the software and hardware that is used in this project and from there author will do some modification so that the running AODV routing protocol can have the selected routing protocol behavior. All the results in shown in graphs and tables.

ACKNOWLEDGEMENT

I would like to take this opportunity to express my gratitude toward my supervisor, Dr. Micheal Drieberg for this great supervision, continuous support, previous guidance and understanding which truthfully help me throughout this project.

My gratitude is extended to Mr. Mamta for his kind assistant and sharing in enhancing my technical knowledge on the current project.

Special thanks to my colleagues, Muhammad Syafiq bin Samsuddin and Mohamad Afiq Bin Azman for their cooperation to help me during my difficulties throughout this project.

I would like to convey my appreciation to all the lecturers from Electrical and Electronic Engineering Department for imparting the engineering knowledge to us throughout the five years of studies, which provide me the skills needed to develop my final year project.

Last but not the least, I wish to express my love and gratitude to my beloved families and friends, for their understanding and endless love, through the duration of my studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Project Background

In this revolutionary and full of advanced electronic devices world, wireless sensor network had gained a quite important role in communication between devices where intelligent, and capable of making decision by itself become the main focus in it. In wireless sensor network (WSN), there are many kinds of network topology which bring can perform according to the pros and cons of the network. One of the network topology is mesh network which the author is going to look into the mesh multihop network. Wireless mesh multihop network (WNN) is a network where all the nodes were connected to each other and each of the nodes can take the function to become master nodes. The data transfer from a sensor node to base station or receiver station will be based on the shortest route available in the network. The design of the route will be based on the routing algorithm that is suitable for the network. There a few types of routing algorithm which is introduced in this paper which is SAR, SPEED and SPAN. This routing algorithm will be chosen according to the QoS-aware of this project where energy conservation on data transmission and priority of data transfer will be the main focus in this paper. In order to perform the algorithm in the network, IRIS mote wireless sensor will be the main part in the network design where it will be run using software called TinyOS. Data will be collected and analysis on each attempt in performing the types of algorithm to find the most suitable algorithm according to the QoS.

1.2 Problem Statement

Nowadays wireless sensor network is been widely used in our daily life either in data transferring or communication. In a typical wireless sensor network, all generated data will be delivered with the same priority. However, in certain application, some type of data should be given priority (e.g. high temperature which signals a fire event when compared to humidity in a home monitoring application). In this project, a Quality of Service (QoS)-Aware routing algorithm for wireless sensor mesh and multi-hop network will be developed and implemented. The system should be able to detect and prioritize the routing for the important sensor data collected and send it to the allocated receiver station. The low transmission power will be one of the aspects to design the system so that it can be energy-efficient and provide longer life span. The effectiveness of this system will be evaluated in terms of its responsiveness, robustness and network lifetime.

1.3 Objectives

To compare different QoS-Aware routing algorithms for wireless sensor mesh and multi-hop network

- To compare different QoS-Aware routing algorithms for wireless sensor mesh and multihop network
- To implement exisiting QoS-Aware routing algorithms
- To develop an improved QoS-Aware routing algorithms

1.4 Scope of study

In this project, the main subject consists of research and analysis works. The research is vital for better understanding for fundamental principle and theory applied to achieve this technology. In order to achieve the objectives, the followings are scope of work of this project that was carried out;

- Understanding the basic concept of multihop mesh network routing algorithms
- Utilize and familiarize with TinyOS software
- Implementation of existing routing protocol
- Improvement on the existing routing protocol

CHAPTER 2

LITERATURE REVIEW

2.1 Wireless Sensor Network

The growth of wireless sensor networks (WSNs) have gained worldwide attention in recent years particularly in 'smart' energy management application or data transfer, including home and building automation. According to a recent data by research firm IDTechEx which forecasts a near ten-fold growth of the WSNs market, to $1.8B by 2019 [1]. The main impact of these growths has led to the development of smart sensors which the sensors are smaller, cheaper and intelligent. The sensors are equipped with wireless interfaces with which they will be able to sense, measure and gather information from the environment and based on some local decision process and communicate with one another to form a network so that they can transmit the sensed data to the user.

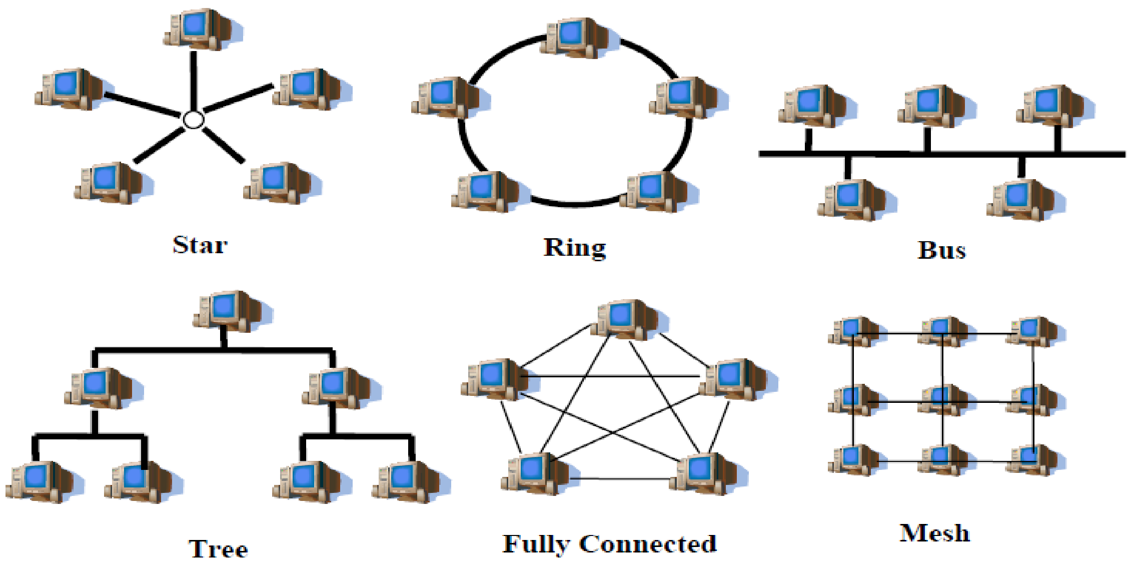


Figure 1: Basic Network Topologies [1]

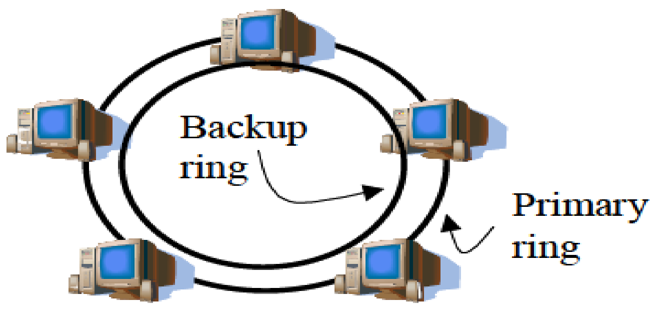


Figure 2: Self-Healing Ring [1]

Wireless sensor network basically consist of 6 types of networking topologies which is point-to-point, star (point-to-multipoint), ring, mesh, bus, tree and fully connected which is shown in Figure 1.

Point-to-point is just simply a link between two points while star networks is an multipoint to a master node which act as an interface in controlling and managing a fix number of points in the data transferring and communication between nodes to nodes. The master nodes can be link with other master nodes of star topologies which sometimes is called as cluster-tree network. Unfortunately one of the disadvantages of the star topologies is that if a master node of a network failed or malfunctioned, the entire sub-network fails.

Ring network connected each node to form a circle where there are no any master nodes and data transfer is from one node to another node in one direction. The disadvantages of the ring network is that when the connection between two nodes in the network were cut off or disconnected, the network itself will fail as the data cannot be send through the network. Thus, a self-healing ring Figure 2 is introduced to overcome this problem where the connection between nodes will be redundant. If either one of the communication line between nodes failed then the backup communication line will take the role in transferring data between nodes.

Meanwhile for bus network, it is commonly used for receiving data where every data send from the master nodes will be send to each nodes in the network without any needs of those nodes to resend the data. So basically bus network is mainly for monitoring work for example the monitoring room in a power plant control room.

For a fully connected network, the biggest problem it faced was with the routing protocol where for each node which is added into the network the number of links increased too [1]. This leads to the complex routing algorithm need to connect each of the nodes especially for a huge network. Followed by the tree network, a master node will control and give command to each node in data transmitting and receiving. The disadvantage of this network is when the master nodes failed to respond or malfunctioned, the whole network will failed.

In mesh network, every node has multiple pathways to every other node, providing the most resiliency and support routing [2]. Most of the mesh network utilizes a type of pseudo-mesh with peer-to-peer communication links that support routing where message will be transfer in the network using a multi-hop routing algorithm that can be optimized for the lowest latency or lowest power. This result in the memory requirements and processing overhead required at each node in mesh network are higher as each node in the mesh network must maintain knowledge of other nodes in the network routing table. Since there are multiple routes for nodes to take to send a data, thus this made this network is almost impossible to fail. To add on, one of the advantages of the mesh network are that when the master nodes are malfunctioned or stop responding, the other nodes will replace the master nodes and take over the function.

2.2 Mesh and MultiHop Network

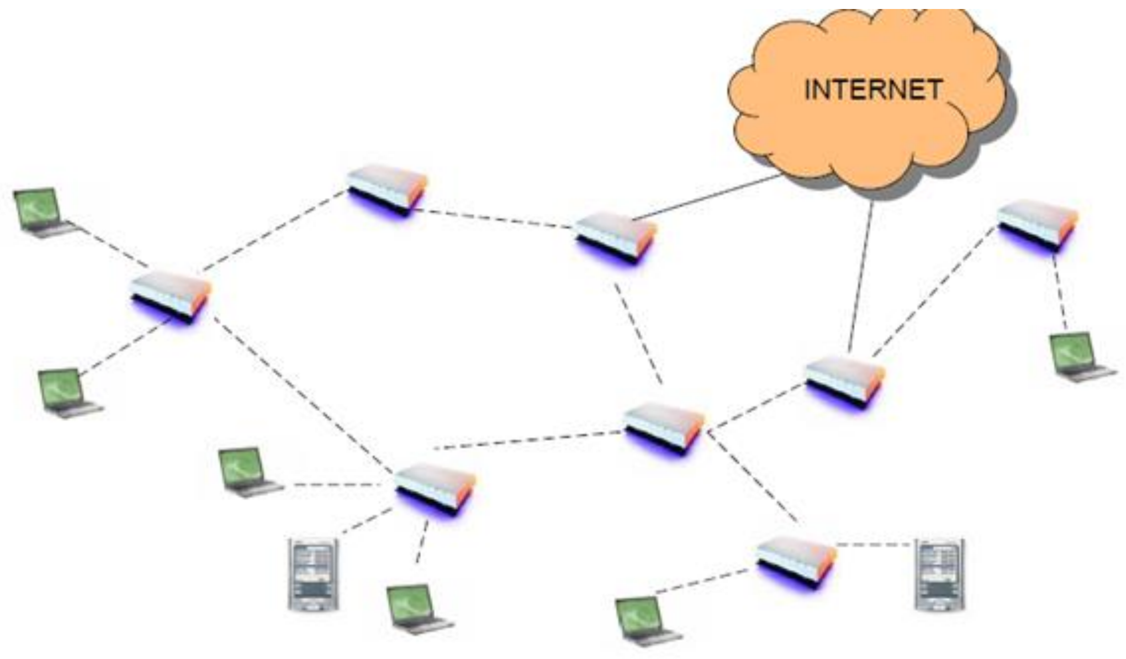


Figure 3: Wireless Mesh Topology [2]

A wireless mesh network (WMN) is a communication network made up of nodes organized in mesh topology which is shown Figure3. It consists of two main components in this network which is mesh router and mesh clients. Mesh clients are mainly computers, laptops, cell phones and other wireless devices while mesh routers is the interface between wireless devices to gateway. Among mesh router and mesh clients, the function of gateway in the WMN helps the network to connect to other network by supplying the nodes with additional network interface cards (NIC). Thus, user with NIC supported devices can connect to the network easily for example, Ethernet. Ashish Raniwala stated that a mesh nets can be a good models for large-scale networks of wireless sensors that are distributed over a geographic region, e.g. personnel or vehicle security surveillance systems [2].

Meanwhile the capability of self-organizing and self –configuration has made WMN able to deployed incrementally, one node at a time, as needed [3]. With the increasing of nodes in the network will result in the increase of reliability and connectivity of user accordingly [4].

Technique such as routing and flooding can be used in designing a MWN. In addition, to ensure the availability of all the routes or path, self-healing algorithm is applied in the MWN where the routing network will be able to allow continuous connection and reconfiguration around broken or blocked paths. With the self-healing capability, this will provides continuous connection between nodes although there is nodes break down or when a connection goes bad. This proves the WMN is actually a reliable network as there will be not only one path or routes for nodes to communicate with another node although there is a broken or malfunctioned node in the network.

According to Jian Tang, Guoliang Xue and Weiyi Zhang in their research paper, there a few advantages in choosing WMN such as low up-front costs, easy incremental deployment, and easy maintenance [4]. The cost of building up a large wireless mesh network is lower comparing to various networks as the WMN only need a few connections to the wired backbone. The point-to-point line configuration makes maintenance of faulty nodes easier and faster while privacy and security in this network can be trusted as each message is send through a specific line directly to the receiver. If there is any fault link between two nodes in the WMN, only the specific link will be affected. Meanwhile, one of the biggest disadvantages of WMN is that is the network is bigger or huge, the amount of money invested in building it will be huge due to the amount of cabling and the hardware ports it requires.

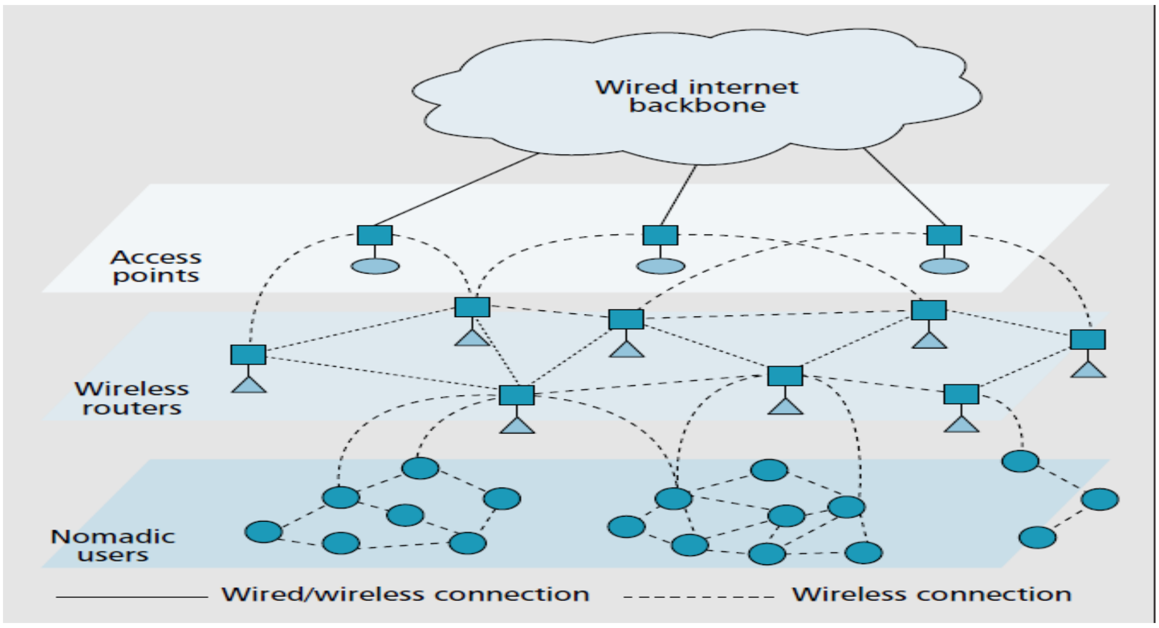


Figure 4: Mesh Multi Hops Network [2]

Mesh networks that contain multiple hops become one of the reason MWN is growing rapidly nowadays in undertaking the problems such as bandwidth degradation, radio interference and network latency [4]. Multi-hop wireless network is similar to Moblie Ad hoc Networks (MANET) but nodes in MWN are relatively fixed [5].

The multi-hop wireless network is divided into two categories; which is relay or mesh. Relay is a tree based topology, where one end of the path is the base station. Meanwhile mesh topology is using multiple connections among users which routing is done by carrier owned devices or equipment. The benefit of multi-hop network is that it is easy to provide coverage in hard-to-wire areas which there is little spaced in deployment of wireless devices. Under some specific condition, the network may be able to extend coverage due to multi-hop forwarding, enhance throughput due to shorter hops and extend battery life due to low power transmission [6].

2.3 WsNs Routing Protocol

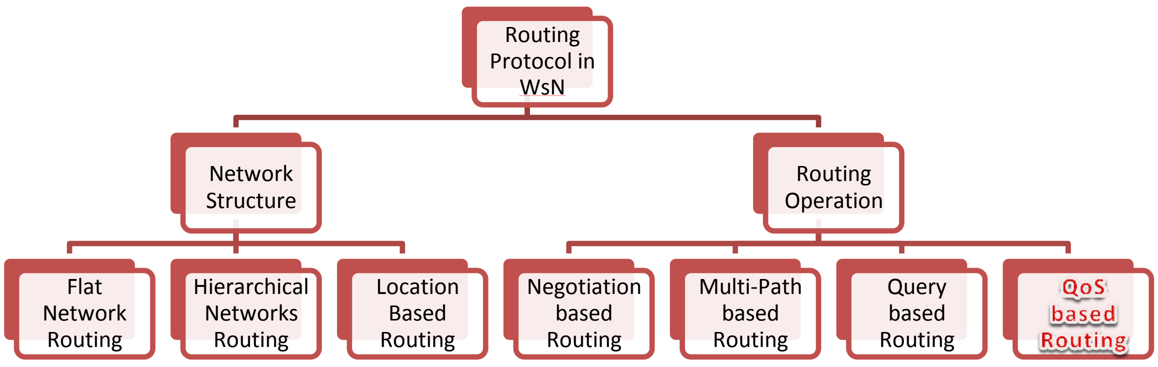


Figure 5: Wireless Sensor Network Routing Protocols [15]

According to the source from [7], routing protocols in WSN can be divided into two parts which is network structure and protocol operation as shown in Figure5. In Figure 5, it is shown that under network structure, WSN can be divided into three categories which are flat networks routing, hierarchical networks routing and location based routing. While for protocol operations, there is 5 classes in it which is negotiation based routing, multi-path based routing, query based routing, QoS based routing and coherent based routing.

The difference between these three network routing structure is that for flat network routing all of the nodes will have the same function and doing the same work. While for hierarchical network routing, each node will have different function in the network. Meanwhile in location based routing, each node is set to a position where it will be only managing the data for the selected position in the network. For routing protocols, it can be classified into three different categories which is proactive, reactive and hybrid [8].

Proactive protocols can be defined as where all the routing is being computed before they are needed. Where else for reactive protocol, the routes are computed on demand. For hybrid protocol, it is the combination of reactive and proactive protocol which is energy consuming as for a static nodes whenever the node need to send a data, it will require a routing to be drawn which require a specific amount of energy to be used[9]. In this paper the author will look in QoS based routing where the network designed is based on the energy saving and priority of data which must be balanced in the design.

2.4 QoS-Aware

Quality of Service (QoS) for networks is a standard used to ensure devices or applications produced by industry are in high quality and meet the expectation. Conceptually the quality of networks can be define as all the network traffic is treated equally and as the result was that all the network traffic received the network's best effort, with no guarantees for reliability, delay, variation in delay, or other performance characteristics [10]. Nowadays more and more routing protocols had been introduced in order to enhance and satisfied the application and users requirement in wireless sensor network. Low transmission power or energy saving system became the main objective in designing new routing algorithm to produce a system with have a longer life span. Meanwhile the author in this paper focuses on the data priority in data transmission from nodes to a receiver station where the low transmission power of the system will also be taken in consideration.

The concept of QoS is to produce an applications or devices that meet the requirement of sufficient bandwidth, controlling latency and jitter, and reducing data loss [11]. The following table describes these network characteristics.

Table 1: Network Characteristics

Reliability	• the ratio of the total number of unique RT data packets received by the sink within the bounded time to the number of packets generated by the source nodes. The higher the value is, the better the efficiency of the protocol.
End-to-End Delay	• Is the time used to send a packet of data from a location to the receiver station. The lower the delay the better the performance.
Energy efficiency	• average amount of energy consumption for the successful transmission of initiated RT traffic flow. If the energy equal to amount of energy used by other protocol in transferring data = reliability high
Network Lifetime	• this is defined as the length of time from network deployment to the first intermediate node drains out of its energy.

According to the source from [11], QoS provides the following benefits:

- Gives administrators control over network resources and allows them to manage the network from a business, rather than a technical, perspective.
- Ensures that time-sensitive and mission-critical applications have the resources they require, while allowing other applications access to the network.
- Improves user experience.
- Reduces costs by using existing resources efficiently, thereby delaying or reducing the need for expansion or upgrades.

By referring to the QoS-Aware specification in this paper, the author is going to focus on routing protocol that delivers messages or data according to the criticality of the data or messages. Each data will be separated into different classes for example data with higher priority and lower priority. Those data with higher priority will be those which the sensor will be sending to the receiver frequently that the author will look into in this project. For example, in a plant when there is a fire the smoke detector(sender) will send signal or messages to the computer (receiver) frequently to inform that the detector sensed smoke from its sensor and it will keep sending the signal until someone on the alarm panel acknowledge the alarm. The routing algorithm for the signal sent from the sensor to the receiver is another aspect which the author will look into thus determine how the data with higher priority is going to be send out first when there is emergency for example or in a normal operation of a system.

2.5 QoS-Aware Routing Protocol

This section will be focusing on summarizing the research which is already been published where we will go into few types of QoS based routing protocol.

Sequential Assignment Routing (SAR) according to the source [12] is considering one of the first routing protocols for wireless sensor network that introduce ‘notion’ in the routing decision making. In SAR routing system it merely depends on three factors which is energy resources, QoS on each path, and the priority level of each packet [12]. In order to prevent any failure in a single route path, multi route path is introduced which a tree rooted on the source node to receiver station is built. Kumar et al. also mentioned that such SAR is table-driven multi-path protocol that mainly focuses on energy efficiency and fault tolerance [13].

Meanwhile a message-initiated constrained-based routing (MCBR) is composed of explicit specifications of constraint-based destinations, route constraints and QoS requirements for messages, and a set of QoS aware meta-strategies [13]. The Meta routing strategies is where for a specific data, the routing from the sensor to the receiver is depends solely on the QoS requirements of the data that needed to be transferred. In another word, the MCBR is allowing the message to discover and learn their routes on their way to the destinations [13].

Another protocol which provides self-real-time end to end guarantees is MMSPEED (Multi-Path and Multi-SPEED Routing) which is an upgraded version of the SPEED protocol. This upgraded version of SPEED enables the protocol to provide probabilistic QoS differentiation with respect to timeliness and reliability domains [14]. In MMSPEED each of the data packets is provided with different types of delivery speed where each packet will be queue up according to it speed and the packet with higher speed are served followed by the next highest speed and so on. So the data with higher priority will be provided with higher speed compared to the lower priority data. Thus, this made the protocol a scalable and adaptable to large network [14]. The one and only disadvantages of this protocol is that the energy metric or energy consumptions of the protocol are not taking into consideration in organizing this protocol.

Meanwhile EQSR (Energy Efficient and QoS Multipath Routing) is one of the recent proposed routing protocols [15], which provides service differentiation by giving real-time traffic absolute preferential treatment over the non-real-time traffic [15]. One of the strength of this protocol is that it uses the Forward Error Correction (FEC) technique together with multi-path diagram to recover from nodes failure without affecting the network-wide flooding for path discovery. . A.R.Rezaie and M. Mirnia stated in their research [15] that EQSR uses the residual energy, node available buffer size, and Signal-to-Noise Ratio (SNR) to predict the best next hop in the process of building the path for selected data. In order to preserve the reliability of this protocol, EQSR separated the transmitted message into a few parts or segment with an equal size, add correction codes, and transmitted it through the multipath so that the percentage of data which will reach the receiver will be higher without any delay.

From [16], a table is built to show the ranking of each protocol according to the QoS-Aware metric is formed. The protocol with the best ranking is to be implemented in this project which is the energy efficient and QoS Aware Multipath based Routing (EQSR).

Table 2: Ranking of each protocol according to QoS-Aware Metric

	SAR	**EQSR**	**MCBR**	**MMSPEED**
Data Priority	2	1	3	4
Reliability	4	1	2	3
End to End Delay	3	1	4	2
Energy Efficiency	4	1	3	2
Network Lifetime	2	1	3	4

Where, Ranking: 1-4 where; 1= Highest and 4= Lowest

2.6 Ad hoc On-Demand Vector (AODV) Routing Protocol

Apart from this the most common routing protocol or which we consider as basic routing is the AODV (Ad hoc On-Demand Distance Vector) Routing which is a routing protocol commonly used for mobile ad hoc networks (MANETs) and other wireless ad hoc networks. It is mainly for networks that contain of thousands of nodes. The AODV is a reactive routing protocol [17] which indicated that it only establishes a route to a destination purely on demand. In other words, this protocol is silent until connection is needed [18]. Each node acts as both a host and routing node where each node must maintain a routing table that contains information about known destination nodes and all of the addressing is done by using IP addressing. Thus, each entry in the routing table contains four fields which are:

- Destination address
- Next hop address
- Destination sequence number
- Hop count

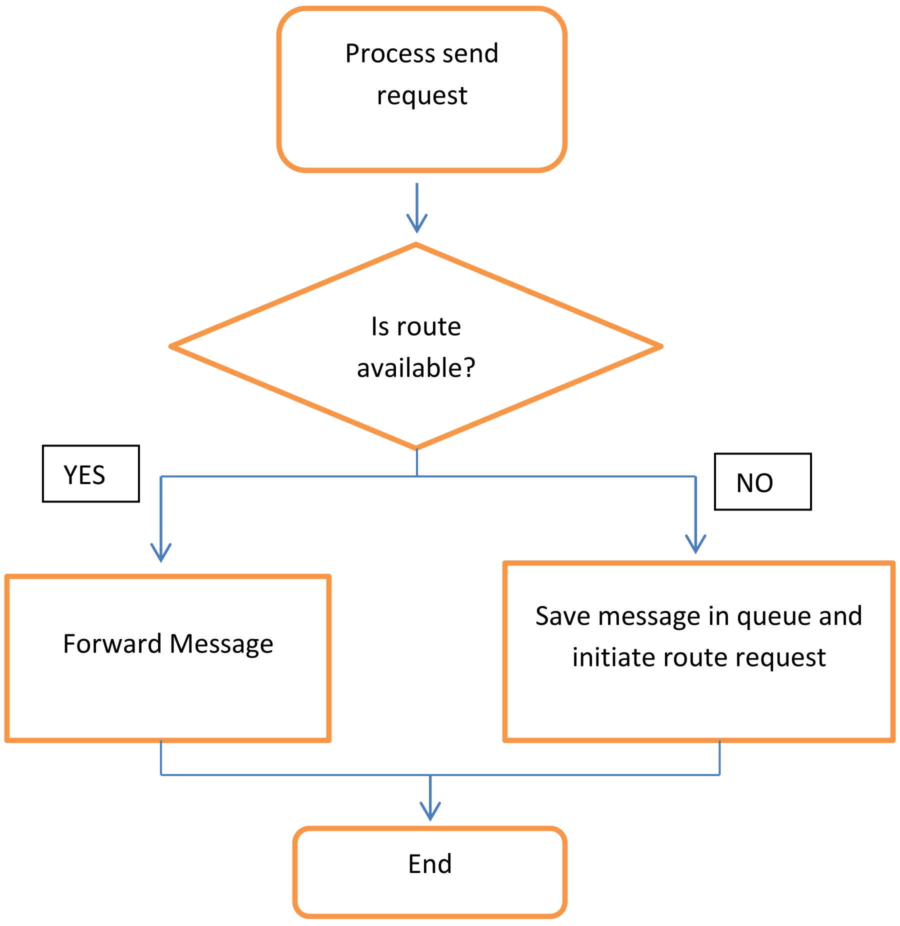
Spring mentioned that in addition to the destination node IP address, the fields contain routing information and information that relates to the qualitative state of the route for maintenance purposes [18]. Unlike some other protocols, AODV only maintains information on the next destination (hop) in the route, not the entire routing list. This result least memory consumption and lowers computational overhead for route maintenance. Apart from this, it also contains information enabling the host to share information with other nodes when link states change. This can be explain for example when a network node that needs a connection, it will broadcast a request for connection.

The other AODV nodes forward this message to the others nodes in the network and record it which created an explosion of temporary routes back to the needy route [19]. When there is a node which suits the route of the desired node, it will send a message backwards through the temporary route to the requesting node and then the needy node will begin to use the route which has the least number of hops to reach the specific nodes. The process will repeat if a links fails where a routing error will be send back to the transmitting node. The design of this protocol is also to lower the number of messages stored in the network thus conserve the capacity of the network.

For example, each request for routes will have a sequence number where the nodes will use this sequence number so that there will be no repeated routing request as the request is done. Other than that, the AODV routing protocol also features the route request to have a 'time to live' number that limits how many times a message can be transmitted [20].

In other words, if a route exists, the router simply forwards the message to the next hop. Otherwise, it saves the message in a message queue, and then it initiates a route request to determine a route. Below is a flow chart to illustrate this process:

Figure 6: AODV process



Upon receipt of the routing information, it will update its routing table and sends the queued message. There are four types of messages that are used by AODV nodes which are Route Request (RREQ) and Route Reply (RREP) which is used for route discovery. Meanwhile Route Error (RERR) and HELLO messages are used for route maintenance.

2.6.1 AODV Route Discovery

When a node needs to determine a route to a destination node, it will flood the network with a Route Request (RREQ) message. The message will be forwarded from each node to another until there is a node replying with the Route Reply (RREP) message. To prevent cycles, each node remembers recently forwarded route requests in a route request buffer. As these requests spread through the network, intermediate nodes store reverse routes back to the originating node. Since an intermediate node could have many reverse routes, it always picks the route with the smallest hop count. When a node receiving the request either knows of a “fresh enough” route to the destination (see section on sequence numbers), or is itself the destination, the node generates a *Route Reply (RREP) message*, and sends this message along the reverse path back towards the originating node. As the RREP message passes through intermediate nodes, these nodes update their routing tables, so that in the future, messages can be routed though these nodes to the destination.

2.6.2 Route Request Buffer

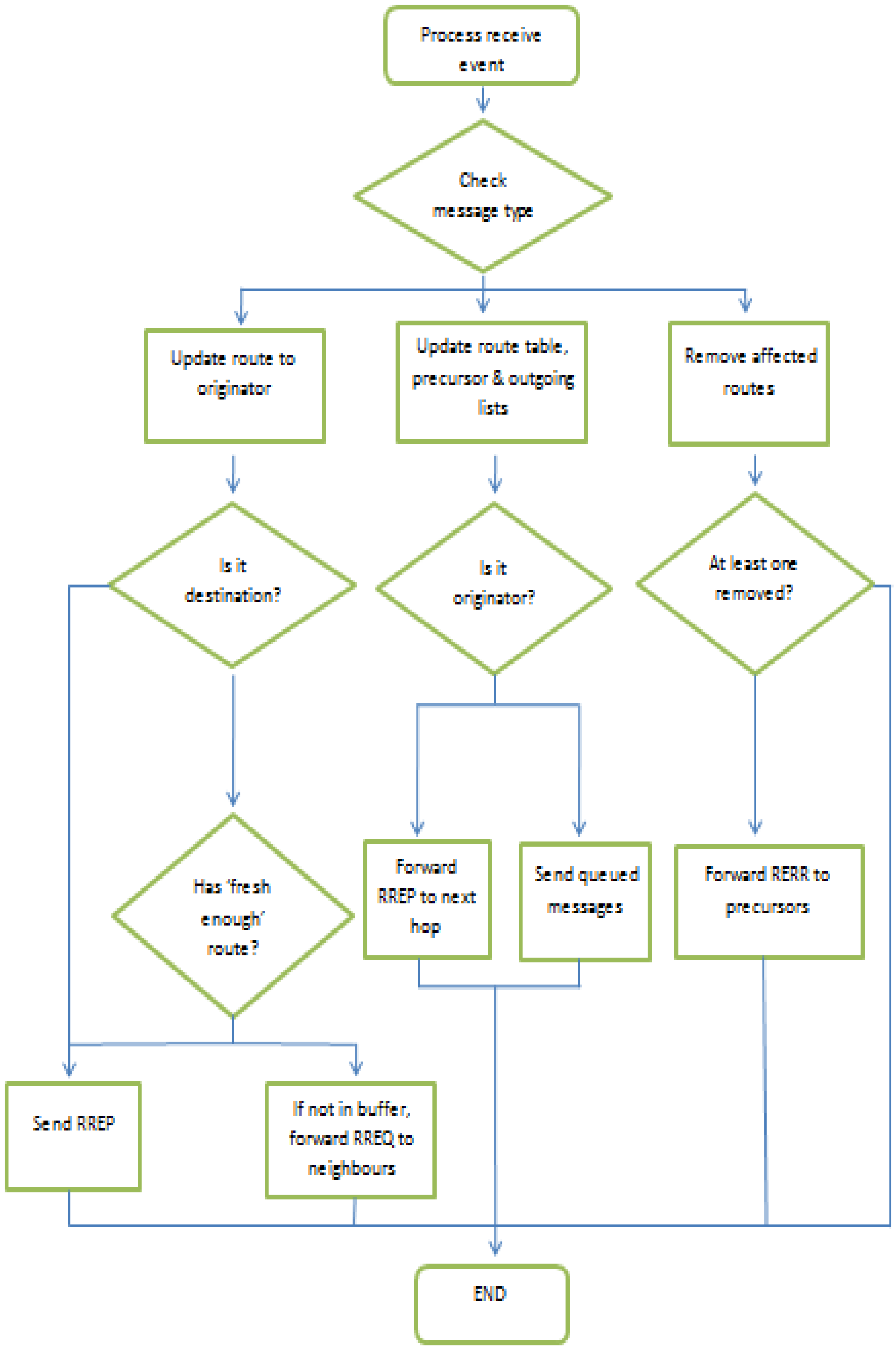
In a large network where there are a lot of nodes, when a node originates or forwards a route request message to its neighbors, the node will likely receive the same route request message back from its neighbors. Route request buffer is pretty useful when it comes to preventing nodes from resending the same Route Request (RREQ) where it had a list of recently broadcasted route requests. Thus, every time before a node forward a RREQ message, it will always check the buffer to make sure the request is not forwarded yet. The message on the list will expires and removed after a certain period of time to prevent the route request buffer list become packed.

2.6.3 Link Monitoring & Route Maintenance

There is a precursor list and an outgoing list which each node will keep track of it. A precursor list is a set of nodes that route through the given node. The outgoing list is the set of next-hops that this node routes through. Each node will send HELLO messages to its precursors from time to time for example a node will only send a HELLO message to its precursors only if no message has been send to that precursors recently. Alternatively, each node will expect to receive a HELLO message from time to time from its outgoing nodes. If there is no message receive from an outgoing node for some period of time, then the nodes is assumed unreachable. Thus, whenever a node determines one of its next hops is unreachable, it removes all affected route entries, and generates a Route Error (RERR) message. This RERR message will include the route which is unreachable which is send to its precursors. Like a ‘’domino effect’’ these precursors will update their routing tables and forward the message to their precursors.

To summarize it the flow chart below shows the action of AODV node when processing an incoming message.

Figure 7: Overall AODV process flow chart



Chapter 3

METHODOLOGY

3.1 Research Methodology

This work is suggested to be carried out in the following phase:

i. Literature Review

   The very first step in this project is to get an insight and understanding of what is wireless sensor network followed by mesh multi-hop sensor network. With the understanding of the network the author will proceed to types of routing algorithm and what is QoS-Aware end by the types of wireless sensor that is used in this project and the software that is use.

ii. Implementing QoS-Aware

   In this project, the main purpose is to meet the requirement of QoS-Aware in data transmission where some types of data from a sensor node should be given priority in sending it to the receiver station without consuming much energy in generating routes and sending the data.

iii. Implementing existing routing algorithm

   The author will choose one out of various routing algorithm to be implement so that author will get an idea of how a routing algorithm works. In this section, the IRIS wireless sensor will be used with its software TinyOS.

iv. Improvement on the existing routing algorithm

   Try to modify or improvise the existing routing algorithm according to the project objectives.

v. Data Collection & Analysis

Collect and do discussion on all the data that if obtained from try and error.

3.2 Key Milestones

Several key milestones must be achieved in order to complete the project.

i. Implementing QoS-Aware

In this project, the main purpose is to meet the requirement of QoS-Aware in data transmission where some types of data from a sensor node should be given priority in sending it to the receiver station without consuming much energy in generating routes and sending the data.

ii. Implementing existing routing algorithm

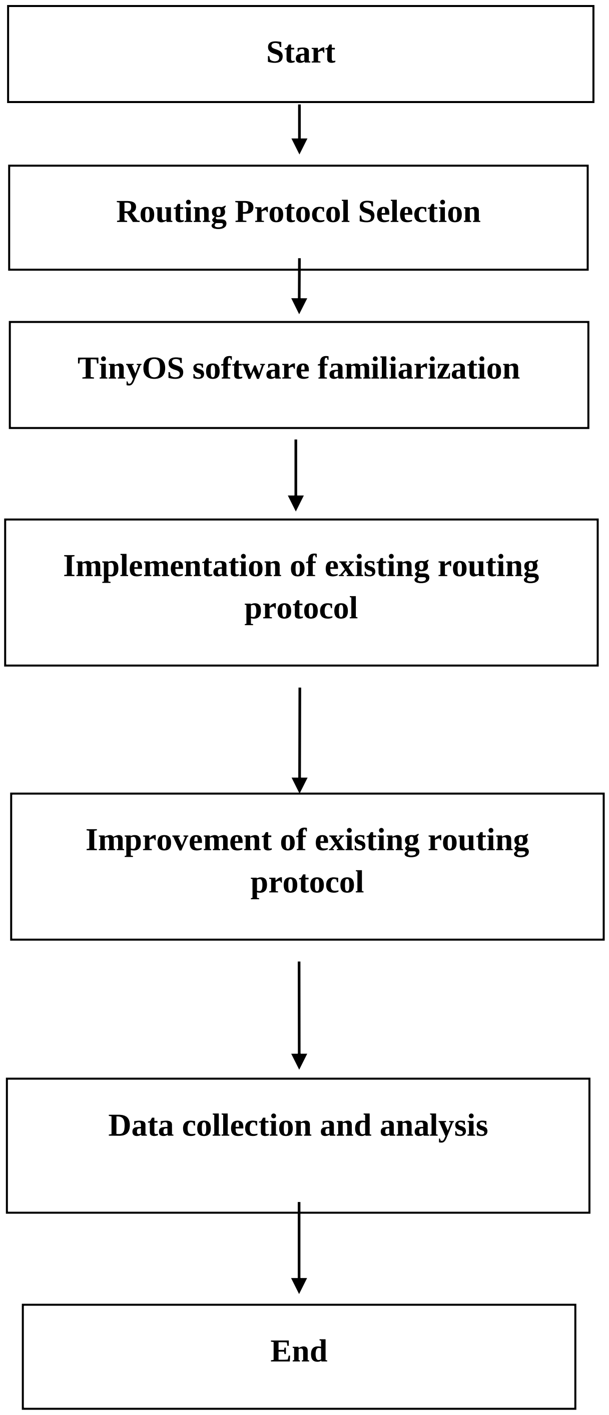
The author will choose one out of various routing algorithm to be implement so that author will get an idea of how a routing algorithm works. In this section, the IRIS wireless sensor will be used with its software TinyOS.

iii. Improvement on the existing routing algorithm

Try to modify or improvise the existing routing algorithm according to the project objectives.

3.3 Project Flowchart for FYP I and FYP II

Figure 8: Project Flow in FYP I and FYP II


Start
Routing Protocol Selection
TinyOS software familiarization
Implementation of existing routing protocol
Improvement of existing routing protocol
Data collection and analysis
End

3.4 Gantt Chart for FYP I

No	Topic WEEK	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Finalized Selection Project Topic for FYP	🟥	🟥												
2	Research Work and Literature Review for Project		🟩	🟩	🟩	🟩									
3	Attend FYP Talk	🟦	🟦	🟦	🟦	🟦									
4	Decision on which routing protocol to be implement				🟩	🟩	🟩	🟩	🟩	🟩					
5	Identify hardware and software for the project				🟪	🟪	🟪	🟪	🟪	🟪	🟪				
6	Research on hardware and software used in the project				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	
8	Online Submission of Extended Proposal Defence						🟪								
9	Preparation for Oral Proposal Defence							🟩	🟩						
10	Viva: Proposal Defence and Progress Evaluation									🟦					
11	Detailed Literature Review								🟪	🟪	🟪	🟪	🟪		
12	Preparation of Interim Report			🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦		
13	Submission of Interim Draft Report													🟩	
14	Online Submission of Interim Final Report														🟥

3.5 Gantt Chart for FYP II

No	Topic WEEK	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	TinyOS software familiarization	🟪	🟪	🟪	🟪	🟪									
2	Implementation on the existing routing algorithm (AODV)		⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛	⬛
3	Writing Progress Report	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩						
4	Debugging on errors from TinyOS software				🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5	Try out other software in AODV implementation • Cygwin • Motework • Moteview				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
6	Writing of final draft report								🟨	🟨	🟨	🟨	🟨	🟨	
7	Completion of dissertation (soft bound)												🟩	🟩	
8	Writing technical paper												⬛	⬛	
9	Preparation for Oral Presentation													🟦	🟦
10	Completion of dissertation (hard bound)														⬛

3.6 Hardware and Software

In this project the author will using IRIS wireless sensor mote which is produced by MEMSIC. It is designed based on the requirement of low power devices which supports microprocessors ranging from 8-bit architectures. With a well-defined set of APIs for application programming, these APIs provide access to the computing capabilities of sensor node [21]. By using these features, data on sensor nodes can be preprocessed and optimize both network and battery consumption avoiding irrelevance sends and receives data process.

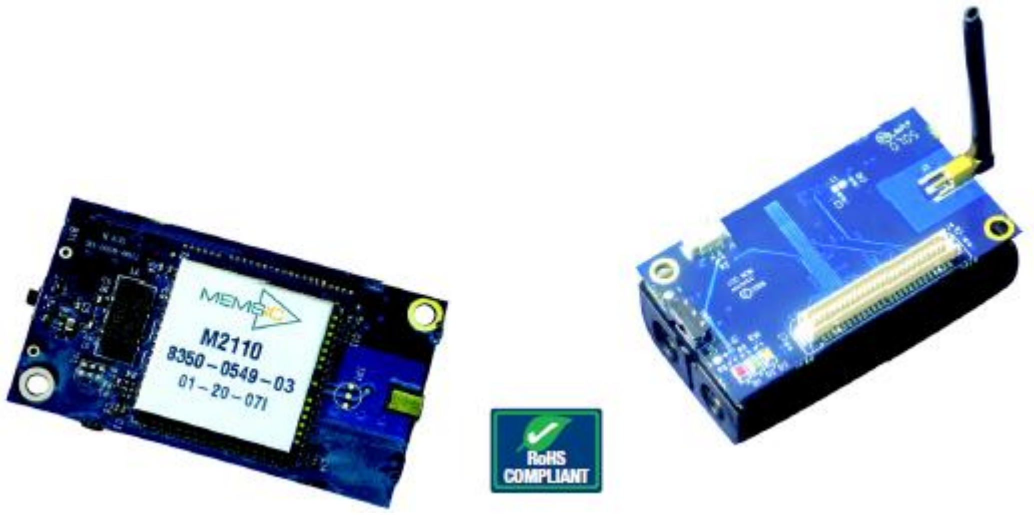


Figure9: IRIS Mote Sensor front and back view [10]

The IRIS is a 2.4GHz Mote used for enabling low-power wireless sensor networks [21]. It is designed especially for deeply embedded sensor network with 250kbps, high data rate radio. With the high level functional integration designed, the module is able to optimize the addition of wireless mesh networking technology to various sensing application providing up to three times improved radio range and twice the program memory [21]. The module is commonly used for indoor building monitoring specifically security, or large scale sensor network. With the 500 meters ranges between nodes without amplification, the module also can directly sequence spread spectrum radio which is resistant to RF interference and provides internet data security. Figure 6 shows the features of the IRIS wireless sensor module.

Apart from this the author also uses MIB520CA USB Interface Board which provides USB connectivity to the MICA family of Motes for communication and in-system programming. The board will become the interface between the nodes and PC via USB which act as a base station for the nodes.

Figure 10: MIB520CA USB Interface Board

Last but not least is the sensor mote which in this project the author uses the MDA300CA which is manufactured by MEMSIC Inc. that can be interfaced with the IRIS Mote mentioned above to function as a sensor node. The multiple function built on this board made it applicable to variety applications.

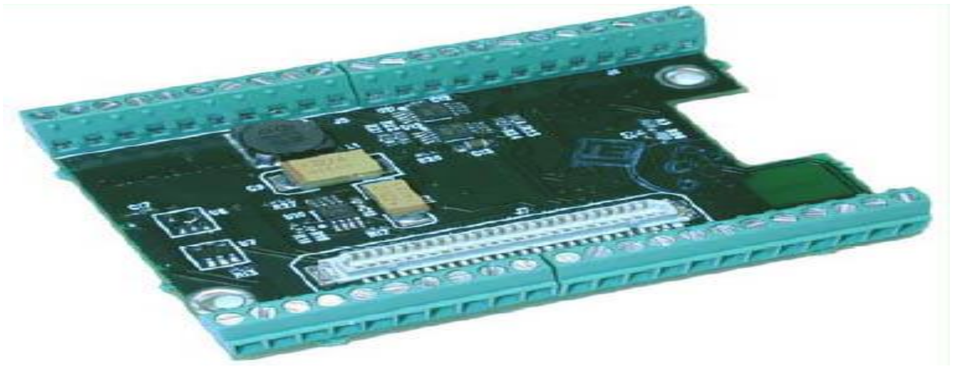


Figure 11: MDA300CA Sensor Board

Figure 12: TinyOS Logo

Meanwhile the software that is used in this project is mainly the TinyOS which is a open source software with based operating system and platform targeting wireless sensor network (WSNs). TinyOS is an embedded operating system written in the nesC programming language as a set of cooperating tasks and process which is intended to be incorporated into smartdust [21]. The TinyOS applications are written in nesC, a dialect of the C language mainly for the memory limits of sensor networks and its supplementary tools are mainly in form of Java and shell script frontends [22]. Other than that, TinyOS also provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage [22].

The XubunTOS is an operating system based on Ubuntu where the TinyOS software is built in inside the operating system. In this project the author is going to use the UBUNTU operating system for easier installation and operating of TInyOS later on.

Apart from this in order to program these IRIS XM2110, USB Interface Board MIB520 and the MDA300CA sensor board, the author need to install Moteworks and Moteview. Moteworks is software which enables the user to compile code and build it into hardware. It comes with all the libraries which is need in burning a program or code into a mote or board. Same function as TinyOS, it uses Nesc compiler which is the extension of C language. Meanwhile Moteview is an interface between a user and a deployed network of wireless sensor. The biggest or the disadvantages of this Moteview and Motework are that it only runs on Windows XP and Windows 2000 which is problematic to the author.

This project is started by assembling all the tools or hardware required which is mainly based on plug-and-play method according to the hardware manual. The software installation comes after that where the author will install virtual machine (Oracle VM) so that it can run in Linux background. As mentioned in the section, the hardware which is used in this project is the processor, RF module (mote), the sensor boards and the interface board.

3.6.1 Installing software (XubunTOS) on Windows [17]

The author is using windows operating system, thus author chooses to use virtual machine which is named as Oracle Virtual Machine. Below are the steps in the installation of the VirtualBox.

i. Download **VirtualBox**. Make sure you choose the right version (in this case **Virtualbox x.y.z for Windows Hosts**)!
ii. Download **VirtualBox x.y.z Oracle VM VirtualBox Extension Pack** from the same site. (We need it for proper USB connections.)
iii. Run **VirtualBox-x.y.z-_____-Win.exe**.
iv. Click on **Next >**.
v. You can select the components to be installed. (The default selection should be fine.) Click on **Next >**.
vi. Select the shortcuts you need and click on **Next >**.
vii. Say **Yes** on the Warning about installing Network Interfaces.
viii. Click on **Install** and wait.
ix. You can start **Oracle VM VirtualBox** now.
x. Import the XubunTOS Virtual Machine. Click on **File -> Import Appliance... (CTRL+I)**.
xi. Click on **Open appliance...** and select the XubunTOS.ova file that you've downloaded.
xii. Tick the **Reinitialize the MAC address of all network cards** checkbox and click on **Import**.
xiii. Click on **Agree** and wait.
xiv. If the import was successful you can now Power On the virtual machine. Click on **Start**.
xv. You may get some notification about screen resolution, mouse/keyboard capture, etc. Not to worry, that's normal, just say **OK**.
xvi. If you forgot to install the **Oracle VM Virtualbox Extension Pack** you will get an error message. Just click **OK**, install it, and try to start the virtual machine again.

3.6.2 Ubuntu Installation Guide [18]

i. Before you begin installing Xubuntu (or any other operating system) I would always recommend backing up any important data from your Windows 7 install just in case things don't go as planned.

ii. Insert the disc into your CD/DVD drive and restart your computer, if the CD doesn't boot then we will need to change some settings in the BIOS to change the boot order, you may wish to consult your computer or motherboard manual. The BIOS can usually be accessed by pressing ESC, DEL, F1, F2 or another key when booting. Many laptops also have a feature where you can press F12 when you turn the laptop on to display a onetime boot menu.

iii. Once the disc has booted you will be prompted with the following screen, on the left panel you can choose the language file to load. On the right you can choose to boot Xubuntu from the CD (a live system), this can be useful for testing that the operating system works with you chosen computer, or to Install Xubuntu.

iv. You will now be prompted with the following screen. This lets you know if you have enough available space on your computer, it is plugged into a power source (if it's a laptop) and wether or not it is connected to the internet.

v. If you use wireless networking you will be asked to connect to your chosen network. Select it from the list and type your password into the box below.

vi. You can now choose the type of install you wish to do. Either select install alongside Windows or replace Windows (this option will permenantly delete Windows, all its installed programs and any documents you have)

vii. Now you can choose how much space you wish to use for your Xubuntu install and Windows 7 install. Slide the bar in the middle to adjust the amount of space for each operating system. The space on the left is for Windows 7, the space on the right is for Xubuntu. Click 'Install Now' and 'Continue' on the confirmation box to proceed. Your computer may appear to not be doing anything for 5 or more minutes, this is normal as it is re-sizing partitions and copying system files to the new Xubuntu partition.

viii. You will be prompted with the following screen where you can now choose your desired time zone from the map.

ix. Choose your keyboard layout from the list or click 'Detect Keyboard Layout' to attempt to automatically select the correct option.

x. On the following screen you can input your details to create a new user account with Xubuntu and decide whether the computer should automatically log in on start up or not.

xi. Xubuntu will now install itself on your computer; this may take a little while depending on the computer it is being installed on.

xii. You will now be prompted with the following dialogue box. Click 'Restart Now' to proceed. Remember to remove the Xubuntu CD from the CD drive before it reboots.

xiii. When your computer reboots you should see a menu similar to the following one, you can choose to either boot 'Ubuntu' or 'Windows 7' from the menu by selecting it and pressing 'Enter'. Select 'Xubuntu' and press 'Enter' to proceed

Figure 13: Xubuntu boot page

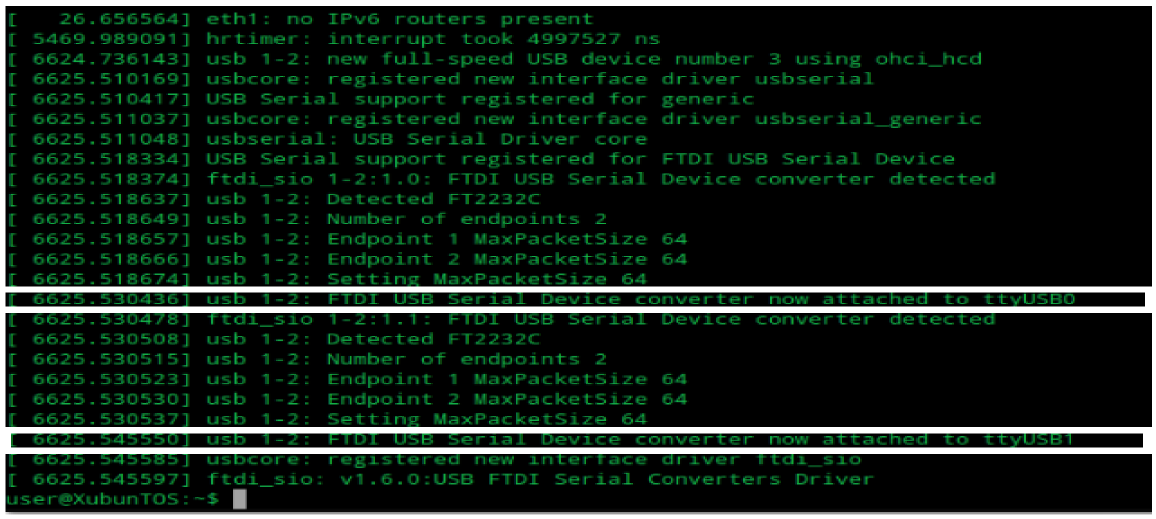
3.7 Programming the TinyOS in Linux Enviroment

TinyOS in Linux environment become one of the options for programmer that wants to compile the nesC into Java which is written usually in C language. This is applicable especially in applications that are communication based.

In this project, author used XubunTOS which is an Ubuntu operating system which comes with a built in TinyOS that runs on desktop or windows. Ubuntu is a complete desktop Linux operating system which is an open source program available free of charge from the internet. Meanwhile TinyOS is interface or medium in communication between motes with a PC through serial or USB port which uses Java in its compilation. Upon finishing installing the Ubuntu operating system, the author tested the TinyOS software itself in XubunTOS. The author plug in the MIB520 interface board which have a USB-to-Serial converter and try to invoke **motelist** which ended up resulting ‘no device found’. As a solution, in order to check whether the MIB520 interface board is plug in or not, the author open the terminal and key in command:

~$ dmesg

The results of **~$ dmesg** in shown in Figure 14.



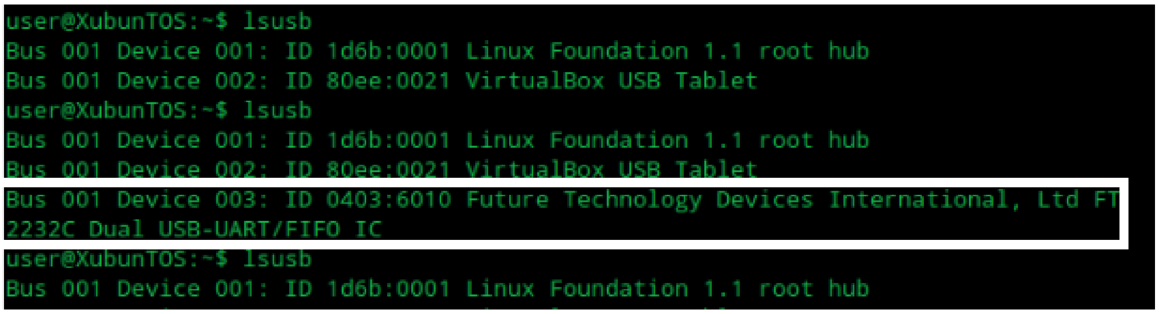
```
[   26.656564] eth1: no IPv6 routers present
[ 5469.989091] hrtimer: interrupt took 4997527 ns
[ 6624.736143] usb 1-2: new full-speed USB device number 3 using ohci_hcd
[ 6625.510169] usbcore: registered new interface driver usbserial
[ 6625.510417] USB Serial support registered for generic
[ 6625.511037] usbcore: registered new interface driver usbserial_generic
[ 6625.511048] usbserial: USB Serial Driver core
[ 6625.518334] USB Serial support registered for FTDI USB Serial Device
[ 6625.518374] ftdi_sio 1-2:1.0: FTDI USB Serial Device converter detected
[ 6625.518637] usb 1-2: Detected FT2232C
[ 6625.518649] usb 1-2: Number of endpoints 2
[ 6625.518657] usb 1-2: Endpoint 1 MaxPacketSize 64
[ 6625.518666] usb 1-2: Endpoint 2 MaxPacketSize 64
[ 6625.518674] usb 1-2: Setting MaxPacketSize 64
[ 6625.530436] usb 1-2: FTDI USB Serial Device converter now attached to ttyUSB0
[ 6625.530478] ftdi_sio 1-2:1.1: FTDI USB Serial Device converter detected
[ 6625.530508] usb 1-2: Detected FT2232C
[ 6625.530515] usb 1-2: Number of endpoints 2
[ 6625.530523] usb 1-2: Endpoint 1 MaxPacketSize 64
[ 6625.530530] usb 1-2: Endpoint 2 MaxPacketSize 64
[ 6625.530537] usb 1-2: Setting MaxPacketSize 64
[ 6625.545550] usb 1-2: FTDI USB Serial Device converter now attached to ttyUSB1
[ 6625.545585] usbcore: registered new interface driver ftdi_sio
[ 6625.545597] ftdi_sio: v1.6.0:USB FTDI Serial Converters Driver
user@XubunTOS:~$
```

Figure 14: dmesg Terminal Results

Another method is by using the command:

~$ lsusb

The results of the command **~$ lsusb** is shown in Figure 12.



```
user@XubunTOS:~$ lsusb
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet
user@XubunTOS:~$ lsusb
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 80ee:0021 VirtualBox USB Tablet
Bus 001 Device 003: ID 0403:6010 Future Technology Devices International, Ltd FT
2232C Dual USB-UART/FIFO IC
user@XubunTOS:~$ lsusb
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

Figure 15: lsusb Terminal Results

Before plug in the MIB520 interface board the terminal results in detecting 2 devices which are shown in Figure. After plug in the MIB520 interface board a third devices appear in the terminal.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Data Gathering and Analysis

The first analysis is to test the MIB520 interface board with the IRIS mote attached to it which is shown in Figure16.

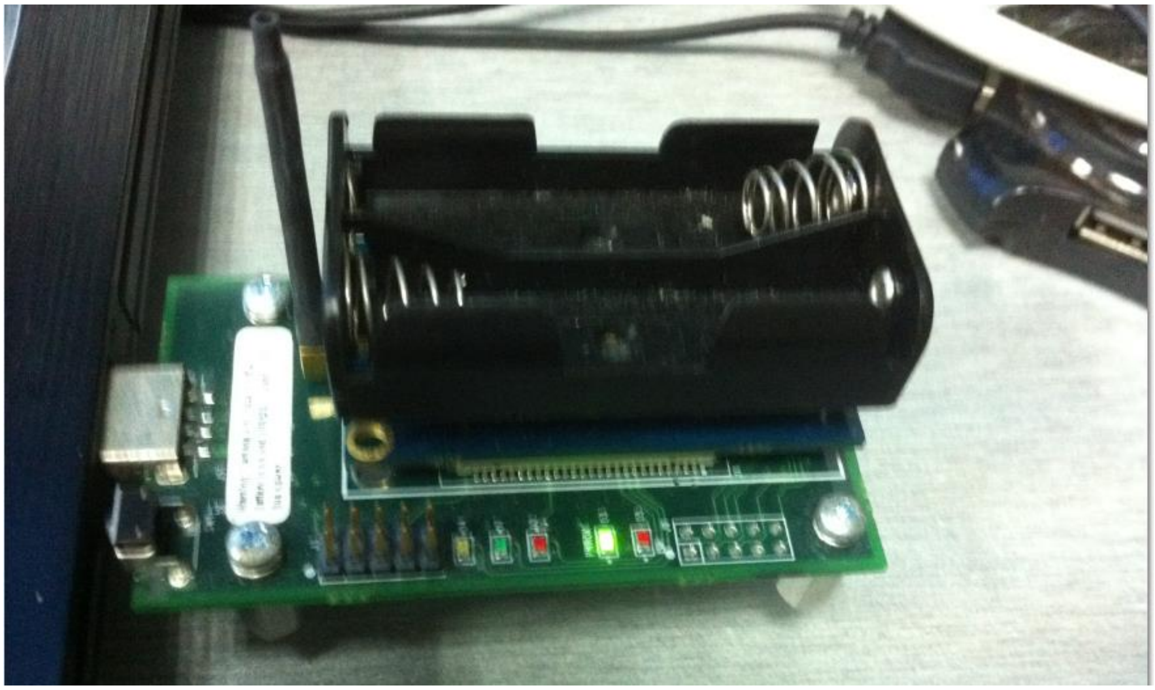


Figure 16: Assembled MIB520 Interface board with IRIS mote

By using the terminal the author will invoke the command:

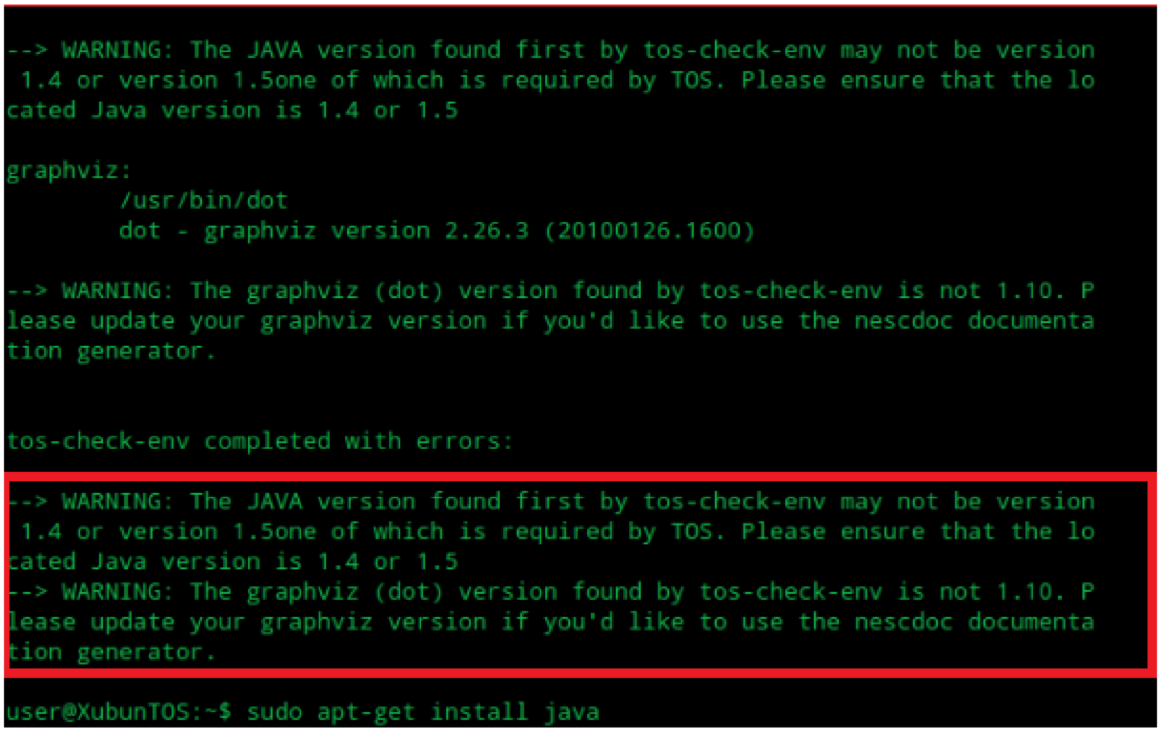
```
~$ cd top/t2_cur/tinyos-2.x/apps/Blink
```

The result of this command will lead to blinking red LED which will prove that the IRIS mote is communicating with the MIB520 interface board.

Unfortunately when invoking the command, the author receives no respond from the interface board thus after some research and analysis, the author found that the version of TinyOS that was in the Ubuntu contained outdated version of Java and graphviz with the command:

~$ tos-check-env

The result is shown in the Figure 17.



```
--> WARNING: The JAVA version found first by tos-check-env may not be version
 1.4 or version 1.5one of which is required by TOS. Please ensure that the lo
cated Java version is 1.4 or 1.5

graphviz:
        /usr/bin/dot
        dot - graphviz version 2.26.3 (20100126.1600)

--> WARNING: The graphviz (dot) version found by tos-check-env is not 1.10. P
lease update your graphviz version if you'd like to use the nescdoc documenta
tion generator.


tos-check-env completed with errors:

--> WARNING: The JAVA version found first by tos-check-env may not be version
 1.4 or version 1.5one of which is required by TOS. Please ensure that the lo
cated Java version is 1.4 or 1.5
--> WARNING: The graphviz (dot) version found by tos-check-env is not 1.10. P
lease update your graphviz version if you'd like to use the nescdoc documenta
tion generator.

user@XubunTOS:~$ sudo apt-get install java
```

Figure 17: tos-check-env Terminal Result

Upon finding out the problem with the TinyOS software, the author did some research and discussion which come to a conclusion that the java and graphviz will need to be updated with the help of [22]. After that the IRIS motes and MIB520 Interface Board will need to be programmed by using MoteWork which will result in the motes to be able to communicate with each other.

4.2 Transmission Power Testing

An experiment was conducted to check on the transmission range of the IRIS motes radio. This experiment is done in a large empty area with no obstacle in between the base station and the iris mote.

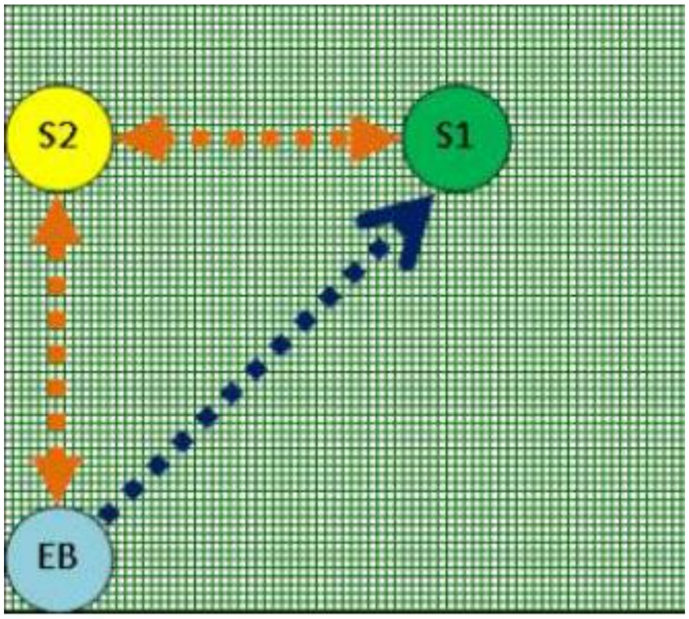


Figure 18: Open field testing

The experiment is performed by executing the command:

```
~$ cd top/t2_cur/tinyos-2.x/apps/BlinkToRadio
~$ cd top/t2_cur/tinyos-2.x/apps/BlinkToRadio make iris install
5.mib520,/dev/ttyUSB0
```

The execution of this command will enable the IRIS mote to receive the signal send from base station and followed by an acknowledgement signal sent back to the base station from the IRIS mote. Upon testing a Table 4 was formed where the transmission power in mili decibel (dBm) and the range of transmission is taken into consideration. The results are shown in Table 3.

Table 3: Transmission Power Test Result

TRANSMISSION POWER	RANGE
-18	16
-14	28
-12	34
-9	45
-7	56
-5	70
-4	76
-3	74
-2	77
-1	79
0	81
0.7	85
1.2	103
1.8	124
2.6	184
2.9	196
3.5	243
3.7	250

The result of the experiment is shown in the graph in Figure 19.

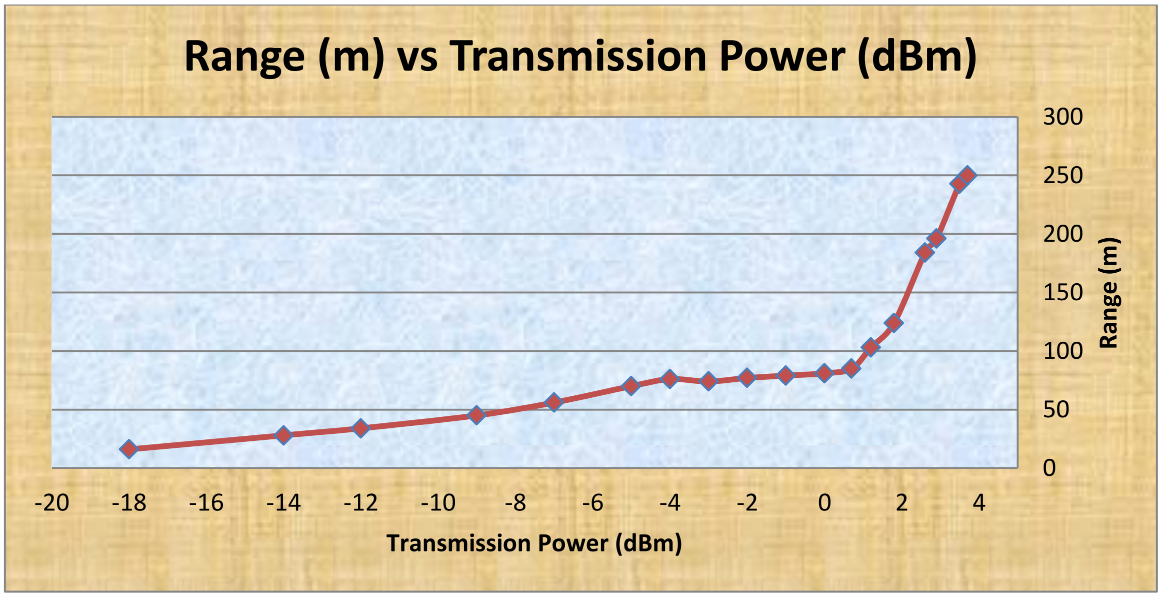
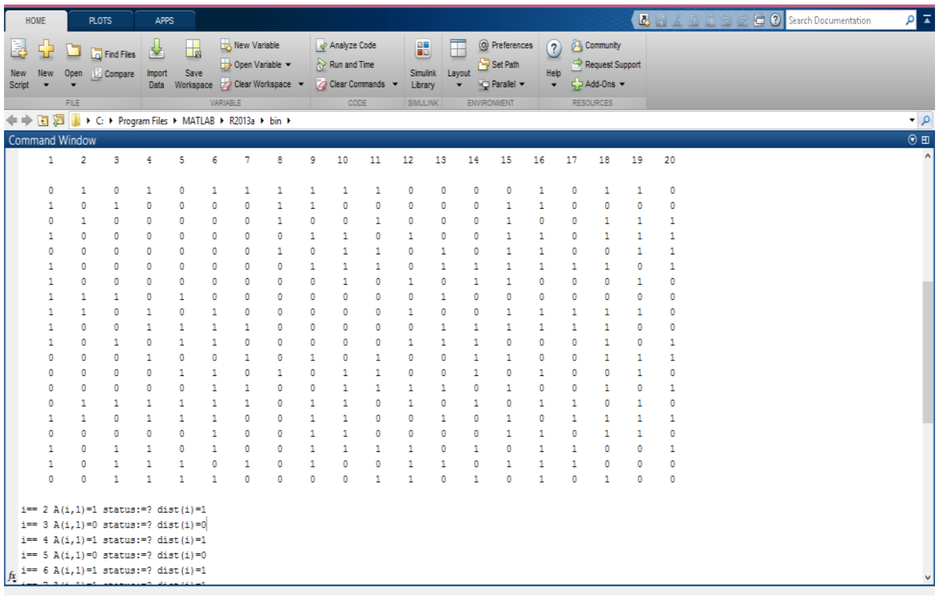


Figure 19: Range vs Transmission Power (dBm) graph.

The purpose of this experiment is to test the power consumption of the IRIS mote when the range is increased. This experiment is tested in an open field where no obstacle is in between the base station and IRIS mote. Thus the value of the transmission power will differ if other disturbance such as obstacle, height and etc. is taken into consideration.

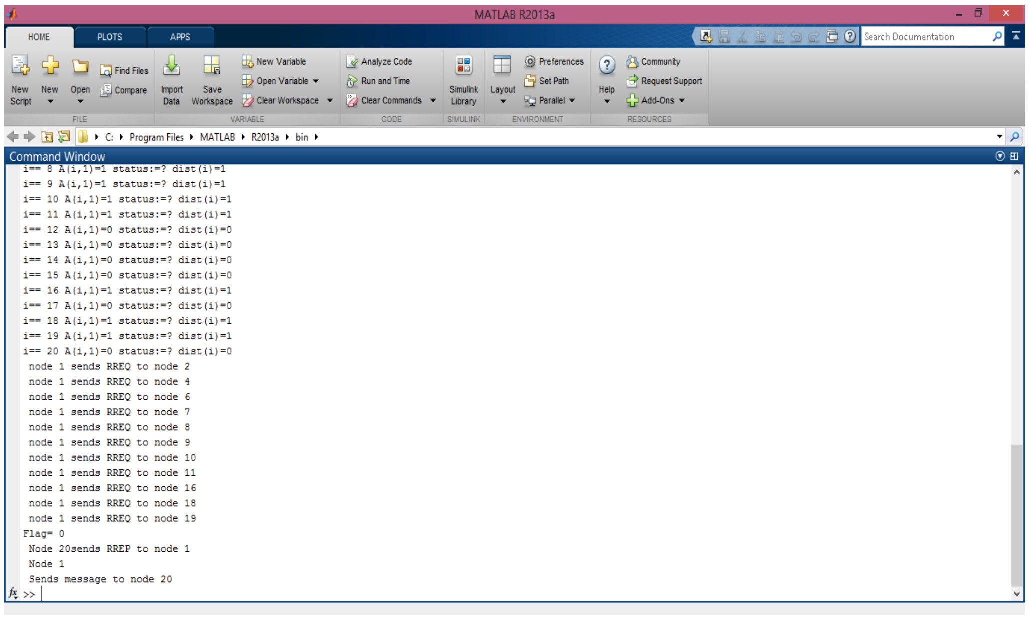
4.3 AODV Implementation using Matlab

In order to understand the working of the AODV routing protocol the author tried to implement AODV by using matlab. The source code for the matlab m-file can be found in appendice. In this implementation the author is trying to send a packet of data from node 1 to node 20. In figure , it shows the matrix table 20x20 of all the nodes available where 1 indicate the nodes is active while 0 is inactive. The result of this implementation is shown in figure 20 and figure 21.



```
     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20

     0     1     0     1     0     1     1     1     1     1     1     0     0     0     0     1     0     1     1     0
     1     0     1     0     0     0     0     1     1     0     0     0     0     0     1     1     0     0     0     0
     0     1     0     0     0     0     0     1     0     0     1     0     0     0     1     0     0     1     1     1
     1     0     0     0     0     0     0     0     1     1     0     1     0     0     1     1     0     1     1     1
     0     0     0     0     0     0     0     1     0     1     1     0     1     0     1     1     0     0     1     1
     1     0     0     0     0     0     0     0     1     1     1     0     1     1     1     1     1     1     0     1
     1     0     0     0     0     0     0     0     0     1     0     1     0     1     1     0     0     0     1     0
     1     1     1     0     1     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0
     1     1     0     1     0     1     0     0     0     0     0     1     0     0     1     1     1     1     1     0
     1     0     0     1     1     1     1     0     0     0     0     0     1     1     1     1     1     1     0     0
     1     0     1     0     1     1     0     0     0     0     0     1     1     1     0     0     0     1     0     1
     0     0     0     1     0     0     1     0     1     0     1     0     0     1     1     0     0     1     1     1
     0     0     0     0     1     1     0     1     0     1     1     0     0     1     0     1     0     0     1     0
     0     0     0     0     0     1     1     0     0     1     1     1     1     0     1     0     0     1     0     1
     0     1     1     1     1     1     1     0     1     1     0     1     0     1     0     1     1     0     1     0
     1     1     0     1     1     1     0     0     1     1     0     0     1     0     1     0     1     1     1     1
     0     0     0     0     0     1     0     0     1     1     0     0     0     0     1     1     0     1     1     0
     1     0     1     1     0     1     0     0     1     1     1     1     0     1     0     1     1     0     0     1
     1     0     1     1     1     0     1     0     1     0     0     1     1     0     1     1     1     0     0     0
     0     0     1     1     1     1     0     0     0     0     1     1     0     1     0     1     0     1     0     0

i== 2 A(i,1)=1 status:=? dist(i)=1
i== 3 A(i,1)=0 status:=? dist(i)=0
i== 4 A(i,1)=1 status:=? dist(i)=1
i== 5 A(i,1)=0 status:=? dist(i)=0
i== 6 A(i,1)=1 status:=? dist(i)=1
```

Figure 20: AODV matrix 20x20



```
i== 8 A(i,1)=1 status:=? dist(i)=1
i== 9 A(i,1)=1 status:=? dist(i)=1
i== 10 A(i,1)=1 status:=? dist(i)=1
i== 11 A(i,1)=1 status:=? dist(i)=1
i== 12 A(i,1)=0 status:=? dist(i)=0
i== 13 A(i,1)=0 status:=? dist(i)=0
i== 14 A(i,1)=0 status:=? dist(i)=0
i== 15 A(i,1)=0 status:=? dist(i)=0
i== 16 A(i,1)=1 status:=? dist(i)=1
i== 17 A(i,1)=0 status:=? dist(i)=0
i== 18 A(i,1)=1 status:=? dist(i)=1
i== 19 A(i,1)=1 status:=? dist(i)=1
i== 20 A(i,1)=0 status:=? dist(i)=0
 node 1 sends RREQ to node 2
 node 1 sends RREQ to node 4
 node 1 sends RREQ to node 6
 node 1 sends RREQ to node 7
 node 1 sends RREQ to node 8
 node 1 sends RREQ to node 9
 node 1 sends RREQ to node 10
 node 1 sends RREQ to node 11
 node 1 sends RREQ to node 16
 node 1 sends RREQ to node 18
 node 1 sends RREQ to node 19
Flag= 0
 Node 20sends RREP to node 1
 Node 1
 Sends message to node 20
fx >>
```

Figure 21: AODV implementation on Matlab

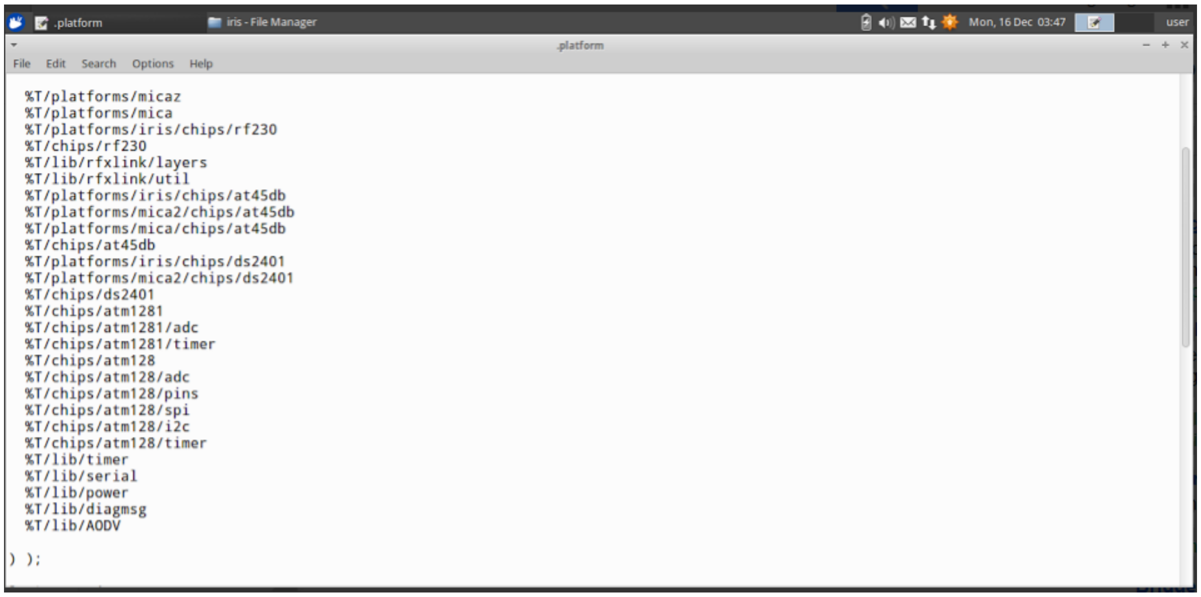
As we can see in Figure , the node 1 will send a route request message to each node which is active. When the RREQ reach until node 20 which is the message destination, the node 20 will send back to node 1 a route reply. Thus, node 1 can send data to node 20. The implementation of AODV using Matlab clearly shows the working protocol of that algorithm.

4.4 Ad-Hoc On-Demand Vector (AODV) Implementation using TinyOS

The method author used in the implementation of this routing protocol is by using Tinyos-2.x which runs in linux environment with the help of Oracle virtual machine. With the documentation that is provided by a PhD student in [23] that is under GNU license, the documents were copy into the Tinyos.2x folder which is

$tinyos-2.x/root/apps and $tinyos-2.x/root/tos.

After that inside the sub-folder platform which is in tos folder, the author needs to add another line in the platform file which is shown in Figure 22.



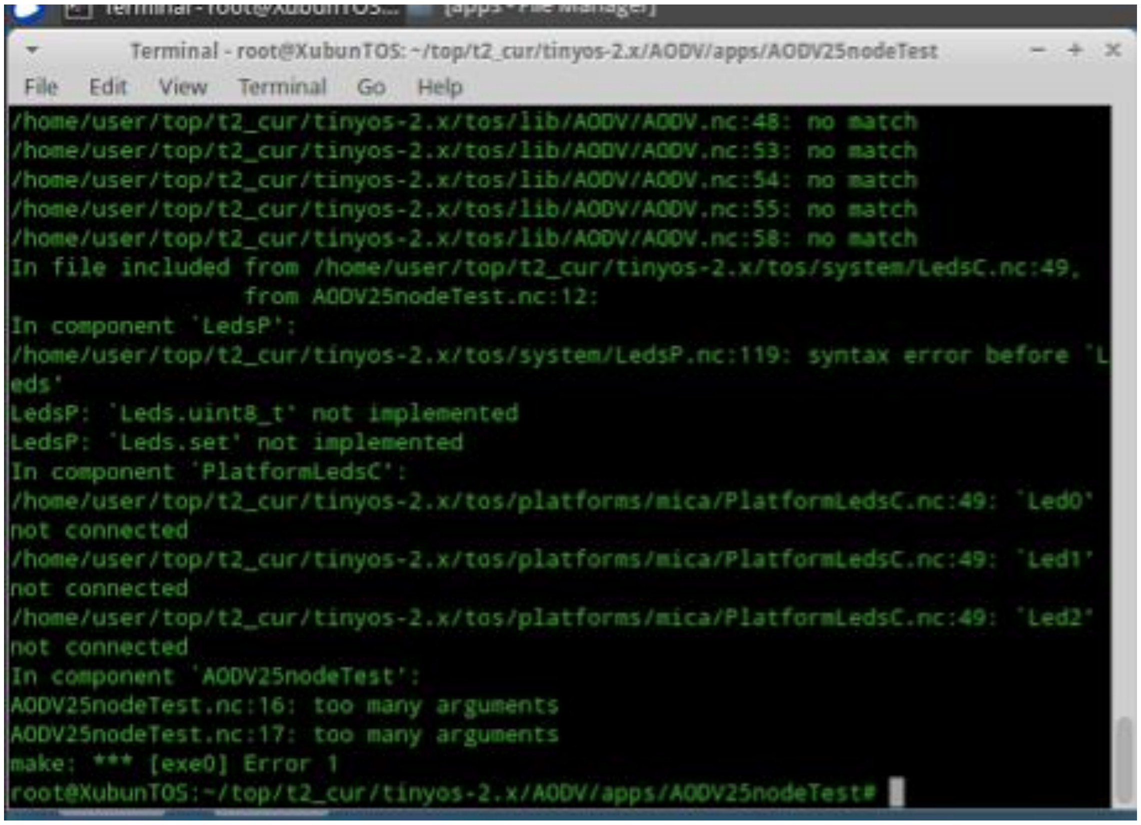
```
  %T/platforms/micaz
  %T/platforms/mica
  %T/platforms/iris/chips/rf230
  %T/chips/rf230
  %T/lib/rfxlink/layers
  %T/lib/rfxlink/util
  %T/platforms/iris/chips/at45db
  %T/platforms/mica2/chips/at45db
  %T/platforms/mica/chips/at45db
  %T/chips/at45db
  %T/platforms/iris/chips/ds2401
  %T/platforms/mica2/chips/ds2401
  %T/chips/ds2401
  %T/chips/atm1281
  %T/chips/atm1281/adc
  %T/chips/atm1281/timer
  %T/chips/atm128
  %T/chips/atm128/adc
  %T/chips/atm128/pins
  %T/chips/atm128/spi
  %T/chips/atm128/i2c
  %T/chips/atm128/timer
  %T/lib/timer
  %T/lib/serial
  %T/lib/power
  %T/lib/diagmsg
  %T/lib/AODV

) );
```

Figure 22: AODV's 'platform' modification

With this, the implementation can be continuing by executing the command:

~$ cd top/t2_cur/tinyos-2.x/AODV/apps/AODV25nodeTest

When executing the command the author receives different result where such errors like in Figure 23 pop out.


Terminal - root@XubunTOS: ~/top/t2_cur/tinyos-2.x/AODV/apps/AODV25nodeTest
File Edit View Terminal Go Help

```
/home/user/top/t2_cur/tinyos-2.x/tos/lib/AODV/AODV.nc:48: no match
/home/user/top/t2_cur/tinyos-2.x/tos/lib/AODV/AODV.nc:53: no match
/home/user/top/t2_cur/tinyos-2.x/tos/lib/AODV/AODV.nc:54: no match
/home/user/top/t2_cur/tinyos-2.x/tos/lib/AODV/AODV.nc:55: no match
/home/user/top/t2_cur/tinyos-2.x/tos/lib/AODV/AODV.nc:58: no match
In file included from /home/user/top/t2_cur/tinyos-2.x/tos/system/LedsC.nc:49,
                 from AODV25nodeTest.nc:12:
In component `LedsP':
/home/user/top/t2_cur/tinyos-2.x/tos/system/LedsP.nc:119: syntax error before `L
eds'
LedsP: `Leds.uint8_t' not implemented
LedsP: `Leds.set' not implemented
In component `PlatformLedsC':
/home/user/top/t2_cur/tinyos-2.x/tos/platforms/mica/PlatformLedsC.nc:49: `Led0'
not connected
/home/user/top/t2_cur/tinyos-2.x/tos/platforms/mica/PlatformLedsC.nc:49: `Led1'
not connected
/home/user/top/t2_cur/tinyos-2.x/tos/platforms/mica/PlatformLedsC.nc:49: `Led2'
not connected
In component `AODV25nodeTest':
AODV25nodeTest.nc:16: too many arguments
AODV25nodeTest.nc:17: too many arguments
make: *** [exe0] Error 1
root@XubunTOS:~/top/t2_cur/tinyos-2.x/AODV/apps/AODV25nodeTest#
```

Figure 23: Errors from executing the program

The author did some debugging with the help from some website on internet but the problem still remains the same. Thus, the author sent an email to the TinyOS community forum to enquire about this problem which resulted in most of the people encounter the same problem from the comment they leave in on that topic. Some of the experts in the forum assume that it maybe perhaps because of the different version of tinyos that is being used by the author and the PhD student who provide the source code.

As a result, the author is currently trying to implement the AODV routing protocol by using the software Cygwin with the help of one of the people from the TinyOS community forum as he mentioned he was able to implement it by using Cygwin.

CHAPTER6

CONCLUSION AND FUTURE WORK

Up until now, the author was able to indicate which routing protocol will be implementing in his project. The outcome of this project will be determined by the successfulness of the implementation of the routing algorithm and improvement of the routing. Progress of this project is up until the implementation stage where the author is stuck on the implementation an existing routing protocol AODV routing by using the TinyOS software. The TinyOS software that is used in this project results in various kind of error when executing a command.

The delay of the progress of this project is mainly due to the time consuming in debugging the error that produce when trying to compile or execute a command. Software such as MoteWork and Moteview contains some problems concerning Java-TinyOS compilation. Thus, XubunTOS is the best option in performing this project. But more research on debugging the error produce by the TinyOS software need to be done before going into the implementation of routing protocol in order to meet the timeline of this project. Although XubunTOS is the best option, due to those error which the author get from executing the program which the author had send some email to CROSSBOW TINYOS community forum to enquire about it. While it is still under debugging, the author will try to implement the AODV routing protocol by using Motework. This project is considered 50% complete as the author only manages to achieve one out of three of the objective of this project. In the author opinion this project should be done by more than one person where more research can be done on debugging the error resulted by the TinyOS software before going into the implementation of routing protocol in order to meet the timeline of this project.

REFERENCES

[1] Akkaya, K. and M. Younis (2005). "A survey on routing protocols for wireless sensor networks." Ad Hoc Networks 3(3): 325-349.

[2] Akyildiz, I. F., et al. (2005). "Wireless mesh networks: a survey." Computer Networks 47(4): 445-487.

[3] Al-Karaki, J. N. and A. E. Kamal (2009). "Routing Techniques in Wireless Sensor Networks: A Survey."

[4] Uthra, R. A. and S. V. K. Raja (2012). "QoS routing in wireless sensor networks—a survey." ACM Computing Surveys 45(1): 1-12.

[5] B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, \SPAN: an energy-e±cient coordination algorithm for topology maintenance in ad hoc wireless networks", Wireless Networks, Vol. 8, No. 5, Page(s): 481-494, September 2002.

[6] Qian Zhang, S. M. I. and F. I. Ya-Qin Zhang (January 2008). "Cross-Layer Design forQoS Support in Multihop Wireless Networks." 96(1).

[7] Yick, J., et al. (2008). "Wireless sensor network survey." Computer Networks 52(12): 2292-2330.

[8] Braginsky and D. Estrin, \Rumor Routing Algorithm for Sensor Networks," in the Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Atlanta, GA, October 2002.

[9] K. Sohrabi, J. Pottie, "Protocols for self-organization of a wireless sensor network", IEEE Personal Communications, Volume 7, Issue 5, pp 16-27, 2000.

[10] Y. Xu, J. Heidemann, D. Estrin,\Geography-informed Energy Conservation for Ad-hoc Routing," In Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking 2001, pp. 70-84.

[11] Jian Tang, Guoliang Xue andWeiyi Zhang, "Interference-Aware Topology Control and QoS Routing in Multi-Channel Wireless Mesh Networks", 2005.

[12] A.B. Bagula, K.G. Mazandu, Energy constrained multipath routing in wireless sensor networks, in:The Proceedings of the 5th International Conference on Ubiquitous Intelligence and Computing, UIC-2008, Oslo, Norway, June, 23-25, 2008, pp.453-467.

[13] E. Felemban, C.G. Lee, E. Ekici, MMSPEED: multipath multispeed protocol for QoS guarantee of reliability and timelines in wireless sensor networks, IEEE Transactions on Mobile Computing 5 (6) (2006) 738-754.

[14] X. Huang, Y. Fang, Multiconstrained QoS multipath routing in wireless sensor networks, Journal of Wireless Networks 14 (4) 465-478 2008. [25] Y. Zhang, M. Fromherz, Message-initiated constraint-based routing for wireless adhoc sensor networks,in: The Proceedings of the First IEEE Consumer Communication and Networking Conference, CCNC-2004, LasVegas, Nevada, USA, January, 5-8, 2004, pp.648-650.

[15] M. Tarique, K.E.Tepe, S.Adibi, S.Erfani (JULY 2009). “Survey of multipath routing protocols for mobile ad hoc networks”

[16] K. Akkaya, M. Younis, An energy aware QoS routing protocol for wireless sensor networks, in: The Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops, Providence, RI, USA, May, 19-22, 2003, pp.710-715.

[17] (2013, March 5) [Online], Available: http://tinyos.stanford.edu/tinyoswiki/index.php/Installing_XubunTOS_(with_TinyOS_from_tp-freeforall/prod_repository)_in_VirtualBox

[18] Thevdm. (2012, December 18).Linux-Distro UK-Open Source Heaven [Online], Available: http://blog.linux-distro.com/index.php/2012/12/18/installing-xubuntu-12-10-amd64-along-side-windows-7-on-dell-xps-15-l502x

[19] D. Chakeres, Elizabeth M. Belding-Royer (2010). “AODV Routing Protocol Implementation Design”

[20] Ali Ranjide Rezaie and Mirkamal Mirnia(MAY 2012). ‘’CMQ: Clustering based Multipath routing algorithm to improving QoS in wireless sensor networks’’ in: IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 1,May 2012

[21] NesCT: A language translator, “Example Application: TinyAODV” [Online], Available: http://nesct.sourceforge.net/tinyaodv.html#Code_Generation

[22] (2007, April 8).’’Lesson 1: Getting Started with TinyOS and nesC ‘’ [Online], Available: http://www.tinyos.net/dist-2.0.0/tinyos-2.0.0/doc/html/tutorial/lesson1.html

[23] Junseok KIM, (2011, April 8). “AODV Implementation on Tinyos-2.x” [Online], Available: http://www2.engr.arizona.edu/~junseok/AODV.html

[24] Vassiliadis V.,(2012, March 11). “AODV Implementation for TINYOS” [Online], Available: http://vasiliadisv.wordpress.com/2012/03/11/aodv-implementation-for-tinyos/

APPENDIX

MATLAB-M files

```
%Code : AODV Routing.
x=1:20;
s1=x(1);
d1=x(20);
clc;
A=randint(20);
% Making matrix all diagonals=0 and A(i,j)=A(j,i),i.e. A(1,4)=a(4,1),
% A(6,7)=A(7,6)
for i=1:20
        for j=1:20
                if i==j
                    A(i,j)=0;
                else
                    A(j,i)=A(i,j);
                end
        end
end
disp(A);
t=1:20;
disp(t);



 disp(A);
 status(1)='!';
% dist(1)=0;
dist(2)=0;
 next(1)=0;
```

```
for i=2:20

    status(i)='?';
    dist(i)=A(i,1);
    next(i)=1;
  disp(['i==  '  num2str(i)  '  A(i,1)='  num2str(A(i,1))  '  status:='  status(i)  '  dist(i)='
         num2str(dist(i))]);
end

flag=0;
for i=2:20
      if A(1,i)==1
          disp([' node 1 sends RREQ to node ' num2str(i)])
              if i==20 && A(1,i)==1
                     flag=1;
              end
      end
end
disp(['Flag= ' num2str(flag)]);
while(1)

   if flag==1
          break;
   end

   temp=0;
   for i=1:20
       if status(i)=='?'
           min=dist(i);
           vert=i;
```

```
            break;
        end
    end

    for i=1:20
        if min>dist(i) && status(i)=='?'
            min=dist(i);
            vert=i;
        end
    end
    status(vert)='!';

    for i=1:20
        if status()=='!'
            temp=temp+1;
        end
    end

    if temp==20
        break;
    end
end

i=20;
count=1;
route(count)=20;

while next(i) ~=1
    disp([' Node ' num2str(i) 'sends RREP message to node ' num2str(next(i))])
    i=next(i);
    %disp(i);
    count=count+1;
    route(count)=i;
    route(count)=i;
end

disp([ ' Node ' num2str(i) 'sends RREP to node 1'])
disp(' Node 1 ')
for i=count: -1:1
    disp([ ' Sends message to node ' num2str(route(i))])
end
```