

RELIABLE DATA COLLECTION PROTOCOL FOR SELF-POWERED WIRELESS
SENSOR NETWORK IN AGRICULTURE MONITORING

By

NG CHOR SHENG

A PROPOSAL

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMMUNICATION AND NETWORKING

Faculty of Information and Communication Technology

(Perak Campus)

MAY 2013

REPORT STATUS DECLARATION FORM

Title: RELIABLE DATA COLLECTION PROTOCOL FOR SELF-POWERED WIRELESS
SENSOR NETWORK IN AGRICULTURE MONITORING

Academic Session: May 2013

I NG CHOR SHENG

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:

No 7, Jalan Hang Jebat 30,

Taman Skudai Baru,

81300 J.B.

Supervisor's name

Date: _____

Date: _____

RELIABLE DATA COLLECTION PROTOCOL FOR SELF-POWERED
WIRELESS SENSOR NETWORK IN AGRICULTURE MONITORING

By

NG CHOR SHENG

A PROPOSAL

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMMUNICATION AND NETWORKING

Faculty of Information and Communication Technology

(Perak Campus)

MAY 2013

DECLARATION OF ORIGINALITY

I declare that this report entitled “RELIABLE DATA COLLECTION PROTOCOL FOR SELF-POWERED WIRELESS SENSOR NETWORK IN AGRICULTURE MONITORING” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my supervisor, Mr Goh Hock Guan who has always given me advices, and guidance throughout the whole FYP process.

Next, I would like to thanks my family who have always motivated me when I faced the problems in FYP. With their encouragement, I was motivated to keep on finding the solutions without giving up so easily.

Last but not least, I would like to thank my fellow friend Lee Yean Sing and Chey Chun Mun who have always given me some suggestions or ideas to do this project

With the help of Mr Goh, my family as well as my fellow friends, I have successfully completed my FYP.

Once again, thank you very much to all of you.

ABSTRACT

Wireless Sensor Network (WSN) is a technology suitable for agriculture monitoring where a group of low cost, tiny sensor nodes can be deployed in outdoor field to monitor crops land. Collecting critical environmental data such as temperature, humidity, or even crops image back to farmers or agriculturalists is very important for further crops health analysis. However, due to the limitations of WSN and error prone of wireless channel transmission medium, data especially images will be dropped during data collection and therefore couldn't be arrived at farmer's site.

This work aims to design and develop a solution to collect scalar data and images using point-to-point protocol and end-to-end reliable data collection protocol. An application is built on the MICAz platform that powered by solar panels, interfaced with temperature and humidity sensors, and interfaced with VGA camera module.

First of all, some of the existing solutions have been reviewed and the way of collecting data for their agriculture applications is identified. In addition, to carry out this project, agile model is chosen as a guideline to follow throughout the development of the project. The technologies involved in this project are the MICAz platform with solar panels and sensors that had been done by previous UTAR researchers, and TinyOS is used as an Operating System for MICAz platform; while the nesC programming is the tool to mainly program the MICAz nodes. Java is used in this project to write a user interface to display the crops information in the PC, while MySQL database is responsible to store the scalar data and images.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY	i
ACKNOWLEDGEMENTS	ii
ABSTRACT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	x
LIST OF TABLES	xvi
LIST OF ABBREVIATIONS	xvi
CHAPTER 1 INTRODUCTION	1
1.1 Proposal Background	1
1.1.2 Problem Statement and Motivation	1
1.2 Project Objectives and Scope	4
1.2.1 Project Objectives	4
1.2.2 Project Scopes	4
1.3 Main Contributions of this Project	6
CHAPTER 2 LITERATURE REVIEW	7
2.1 Literature Review	7
2.1.1 COMMONSense Net: A Wireless Sensor Network for Resource-Poor Agriculture in the Semiarid Areas of Developing Countries	7

2.1.2	Practical considerations for wireless sensor networks in cattle monitoring applications	8
2.1.3	Efficient Zone-based Routing Protocol of Sensor Network in Agriculture Monitoring Systems	9
2.1.4	Experiments with Reliable Data Delivery in Wireless Sensor Networks	10
2.2	Critical remarks of previous works	11
2.2.1	Summary of advantages, disadvantages, and comments for 4 reviewed papers.	16
2.3	Summary/ Concluding Remarks	18
CHAPTER 3 PROCESS DEVELOPMENT		19
3.1	Methodology	19
3.1.1	Waterfall Model	19
3.1.2	Agile Model	20
3.1.3	Selecting a Software Process Model	21
3.2	Requirement Specifications	23
3.2.1	System Requirements	23
3.2.2	System Performance Definition	24
3.2.3	System Verification Plan	24
3.3	Timeline	26
3.4	Technology Involved	29
3.4.1	Hardware Platform	29

3.4.2	Software	31
3.5	Budget Plan	32
3.6	Summary/ Concluding Remark	35
CHAPTER 4 SYSTEM DESIGN		36
4.1	System Architecture	36
4.1.1	System Overview	36
4.1.2	Source Node	37
4.1.3	Intermediate Node	38
4.1.4	Base Station	39
4.1.5	Graphic User Interface	40
4.1.6	Overall Requirements	41
4.2	Flow Charts of Wireless Sensor Network	42
4.2.1	Flow Chart of Checking Parent's Node Coverage	43
4.2.2	Flow Chart of Layer 2 Point-to-point DAT-ACK Protocol	44
4.2.3	Flow Chart of Layer 4 End-to-end Reliable Data Collection Protocol	45
4.2.4	Flow Chart of Reading SHT11 Sensor Values	48
4.2.5	Flow Charts of Enabling C329 Camera	49
4.2.6	Flow Charts of Intermediate Node	53
4.3	Java Application Design	54
4.4	Database Design	55

4.5	Summary/Concluding Remark	55
CHAPTER 5 IMPLEMENTATION AND TESTING		56
5.1	Hardware Implementation	56
5.1.1	Overall Hardware Design	56
5.1.2	MICAz 51-pin Expansion Connector	57
5.1.3	Circuit Design of Butterfly Board	58
5.1.4	SHT11 Temperature and Humidity Sensors	60
5.1.5	C329 Camera	61
5.2	Software Implementation	62
5.2.1	Software Setup	62
5.2.1.1	TinyOS Installation	62
5.2.1.2	Driver Installation of MIB510 Programming and Interface Board	70
5.2.1.3	Verification of TinyOS Installation	74
5.2.1.5	MySQL Installation	75
5.2.1.6	Verification of MySQL Installation	82
5.2.1.7	Create Tables in MySQL	83
5.2.2	MICAz Coding	86
5.2.2.1	Node130	87
5.2.2.2	Node150	92
5.2.2.3	BaseStation Folder Files	93

5.2.2.4	java_app Folder Files	93
5.2.3	MICAz Compilation	97
5.2.4	Screenshot of the Running Program	106
5.3	Summary/Concluding Remark	111
CHAPTER 6	DISCUSSION	112
6.1	System Verification	112
6.1.1	Raw Data Packet Reliability Rate Using Point-to-point Protocol	112
6.1.2	Image Data Packet Reliability Rate Using Point-to-point and End-to-end Protocol	114
6.1.3	Total Time Delay to Receive all Image Data	117
6.2	Problems Encountered and Solutions	119
6.2.1	MICAz can't Synchronize with C329 Camera	119
6.2.2	Can't Form the Whole Image after Receiving Image Data	119
6.2.3	Flash memory can't be Accessed Simultaneously	121
6.2.4	Source Node Failed to Receive Layer 4 Control Message	121
6.3	Achievement	122
6.4	SWOT	123
6.5	Summary/Concluding Remarks	124
Chapter 7	CONCLUSION	125
7.1	Overall Conclusion	125

7.2 Future Enhancement	126
------------------------	-----

REFERENCES	128
-------------------	------------

LIST OF FIGURES

Figure 3.1.1.1: Waterfall Model (CompSci.ca, 2007)	19
Figure 3.1.2.1: Agile Model (BayAmp, 2011))	20
Figure 3.1.3.1: Project's Software Process Model	22
Figure 3.4.1.1: MICAz platform with solar panels and SHT11 sensor and C329 camera integrated.	29
Figure 3.4.1.2: Two devices from Crossbow Technology, Inc – MIB520 Mote Interface Board (left) and MICAz mote (right) (Crossbow, n.d.))	30
Figure 3.5.1.3: Layout of self-powered WSNs deployment on one block size (60m x 100m) of crops field (paddy field).	35
Figure 4.1.1.1: Overview system design of the project.	36
Figure 4.1.2.1: Simple flow diagram for the source node.	37
Figure 4.1.3.1: Simple flow diagram for the intermediate node.	38
Figure 4.1.4.1: Simple flow for the base station.	39
Figure 4.1.5.1: Simple GUI for displaying raw data and image	40
Figure 4.2.1: Main modules for source node.	42
Figure 4.2.1.1: Flow chart of Active ACK	43
Figure 4.2.2.1: Process flow of layer 2 point-to-point DAT-ACK protocol	44
Figure 4.2.3.1: Process flow of end-to-end reliable data collection protocol (Source)	45
Figure 4.2.3.1: Process flow of end-to-end reliable data collection protocol (Base Station / PC)	47
Figure 4.2.4.1: Flow diagram of reading SHT11 values	48

Figure 4.2.5.1: List of commands supported by C329-UART camera module (Electronics123.com, Inc. 2010)	49
Figure 4.2.5.2: Simple flow of using commands set. (Electronics123.com, Inc. 2010)	49
Figure 4.2.5.3: Flow chart of getting picture (Electronics123.com, Inc. 2010)	50
Figure 4.2.5.4: Flow chart of Enabling C329 camera	51
Figure 4.2.5.5: Method of storing whole image data	52
Figure 4.2.6.1: Flow chart of intermediate node	53
Figure 4.3.1: Flow chart of java application	54
Figure 4.4.2: Imag_data table design	55
Figure 4.4.1: Raw_data table design	55
Figure 5.1.1.1: Simple MICAz Platform Block Diagram	56
Figure 5.1.2.1: MICAz 51-pin expansion connector (Crossbow Technology, n.d.)	57
Table 5.1.2.1: List of needed pins and their usages	57
Figure 5.1.3.1: The layout design of butterfly board	58
Figure 5.1.4.1: Circuit design of SHT11	60
Figure 5.1.4.2: Actual implementation of SHT11	60
Figure 5.1.4.3: Layout diagram of SHT11 with breakout board	60
Figure 5.1.5.1: Layout diagram for C329 board and UART interface	61
Figure 5.1.5.2: Actual implementation of C329 camera.	61
Figure 5.2.1.1.1: Change of JDK installation directory	63

Figure 5.2.1.1.2: Change of JRE installation directory	63
Figure 5.2.1.1.3: Cygwin download source selection	64
Figure 5.2.1.1.5: Change of Local Package Directory	65
Figure 5.2.1.1.6: Edit of environment variables	66
Figure 5.2.1.1.7: Cygwin command prompt	67
Figure 5.2.1.1.8: Adding lines to “profile” file	68
Figure 5.2.1.1.9: Change destination directory for Graphviz	69
Figure 5.2.1.1.10: Installation of Crimson editor	69
Figure 5.2.1.2.1: New hardware found pop out	70
Figure 5.2.1.2.2: Selection of installation location	71
Figure 5.2.1.2.3: Selection of installation options	71
Figure 5.2.1.2.4: Warning pop out during driver installation	72
Figure 5.2.1.2.5: Pop out of successful driver installation	72
Figure 5.2.1.2.6: COMM port and PROGRAMMING port	73
Figure 5.2.1.3.1: Checking installation status for TinyOS	74
Figure 5.2.1.3.2: Status of TinyOS	74
Figure 5.2.1.5.1: First step of MySQL installation	75
Figure 5.2.1.5.2: Second step of MySQL installation	76
Figure 5.2.1.5.3: Pop out of MySQL setup finished	76
Figure 5.2.1.5.4: MySQL Configuration selection type	77
Figure 5.2.1.5.5: MySQL server selection type	77

Figure 5.2.1.5.6: MySQL database usage selection	78
Figure 5.2.1.5.7: MySQL database file path selection	78
Figure 5.2.1.5.8: MySQL database file path selection	79
Figure 5.2.1.5.9: MySQL database networking options	79
Figure 5.2.1.5.10: MySQL database default charset selection	80
Figure 5.2.1.5.11: MySQL database Windows options.	80
Figure 5.2.1.5.11: MySQL database security options.	81
Figure 5.2.1.5.12: MySQL database configuration done	81
Figure 5.2.1.6.1: Verify MySQL via Windows command prompt	82
Figure 5.2.1.6.2: MySQL command testing in command prompt	82
Figure 5.2.1.7.1: Log in to MySQL	83
Figure 5.2.1.7.2: Create database in MySQL	83
Figure 5.2.1.7.3: Create raw_data table in MySQL	84
Figure 5.2.1.7.3: Create image_data table in MySQL	85
Figure 5.2.2.2: Listed folders required in this project	86
Figure 5.2.2.1.1: Files contain in Node130 folder	87
Figure 5.2.2.1.2: Fragment part of CSFYPC.nc in Node130	87
Figure 5.2.2.1.3: Fragment part of CSFYPM.nc in Node130	88
Figure 5.2.2.1.4: Event being triggered by Timer0 in CSFYPM.nc	88
Figure 5.2.2.1.5: Data message packet structure in PacketMsg.h	89
Figure 5.2.2.1.6: Control message packet structure in PacketMsg.h	89

Figure 5.2.2.1.7: Makefile in Node130	91
Figure 5.2.2.1.7: volumes-at45db.xml in Node130	91
Figure 5.2.2.2.1: Files contain in Node150 folder	92
Figure 5.2.2.2.2: Fragment code taken from FileM.nc from Node150	92
Figure 5.2.2.3.1: Files contain in BaseStation folder	93
Figure 5.2.2.4.1: Files contain in java_app folder	93
Figure 5.2.2.4.2: Front part of PingPong class	94
Figure 5.2.2.4.3: messageReceived function in PingPong class	94
Figure 5.2.2.4.4: Fragment part of checkLostPacket function in PingPong class	95
Figure 5.2.2.4.5: Fragment code of pongReceived function in PingPong class	95
Figure 5.2.2.4.6: Front part of LoadandShowImageFromFile class	96
Figure 5.2.3.1: MICAz with 2 AA batteries	97
Figure 5.2.3.1: MICAz with programming board plug-in to PC	98
Figure 5.2.3.2: Data and programming port numbers	98
Figure 5.2.3.3: MICAz compilation command	99
Figure 5.2.3.4: Error occur during MICAz compilation	100
Figure 5.2.3.5: Reset button of programming board	100
Figure 5.2.3.6: Message of successful compilation	101
Figure 5.2.3.7: Header migration command	102
Figure 5.2.3.8: Packet structure in java format	103
Figure 5.2.3.9: Listed java files in java_app folder	103

Figure 5.2.3.10: java compilation command	103
Figure 5.2.3.11: java export command	104
Figure 5.2.3.11: java application run command	104
Figure 5.2.3.12: The whole WSNs in this project	105
Figure 5.2.4.1: GUI of this project	106
Figure 5.2.4.2: GUI of this project	107
Figure 5.2.4.3: Readings shown on Cygwin bash shell	108
Figure 5.2.4.4: SHT11 readings stored into database	108
Figure 5.2.4.5: GUI to display image captured from C329 camera	109
Figure 5.2.4.6: Image being displayed on Image tab	110
Figure 5.2.4.7: Binary image data queried from MySQL	110
Figure 5.2.4.8: Segment of images displayed in Cygwin bash shell	111
Figure 6.1.2.1: Source node failed to receive layer 4 ACK mesasge	116
Figure 6.2.2.1: Image data failed to write into flash memory	119

LIST OF TABLES

Table 3.3.1: The gantt chart showing project tasks for FYP1 in Year 3 Semester 1	27
Table 3.3.2: The gantt chart showing project tasks for FYP2 in Year 3 Semester 3	28
Table 3.5.1.1: Budget plan to carry out this project	33
Table 3.5.1.2: Budget plan for implementing the system into one block size (60m x 100m) of crops field. (e.g. paddy field).	34
Table 6.1.1.1: Single hop packet reliability rate for raw data	112
Table 6.1.1.2: Two hops packet reliability rate for raw data	113
Table 6.1.2.1: One hop packet reliability rate for image data	115
Table 6.1.2.2: Two hops packet reliability rate for image data	116
Table 6.1.2.1: Total time delay to receive whole image (single hop)	117
Table 6.1.2.2: Total time delay to receive whole image (two hops)	117

LIST OF ABBREVIATIONS

ACK	Acknowledgement
NACK	Negative Acknowledgement
PC	Personal Computer
UTAR	Universiti Tunku Abdul Rahman
WSN	Wireless Sensor Network
USB	Universal Serial Bus

CHAPTER 1 INTRODUCTION

1.1 Proposal Background

1.1.2 Problem Statement and Motivation

Agriculture plays an important role in human civilization. Agriculture is extremely sensitive towards climate and environment changes, the crop yields are highly dependent on environment conditions. In traditional agriculture, farmers observe the relevant environmental data based on their own knowledge and experiences. This may lead to inappropriate use of resources such as water, pesticide as well as fertilizer. Nonetheless, because the size of the agricultural land can be as large as few hundred hectares, in order to monitor as such a big agricultural land, high labor power is required (Goh, et al., 2007).

Wireless Sensor Network (WSN) is suitable for agriculture monitoring where a collection of low cost, tiny sensor nodes can be deployed in outdoor fields to help monitor the crops land. The sensing parameter can be air temperature and relative humidity, water level and so on. Besides, the inexpensive CMOS camera can attach with sensor nodes to capture the crops images in daily basis so that farmers do not need to observe and monitor the growth of crops at their land physically.

Sensor nodes usually powered by batteries which have limited lifespan and energy. The communication between sensor nodes may drain up the battery power quickly. Replacing or recharging the exhausted batteries in agricultural field is inconvenient and the incurring cost of buying batteries is usually expensive than buying low-cost sensor nodes. Hence, the use of renewable power source such as solar panels is suitable to be integrated to the node depending on the appropriateness of the environment where the sensor is deployed. With this kind of self-powered sensor nodes, instead of spending the cost on batteries, farmers can invest more on

low-cost sensor nodes to be deployed on large-scale farmland resulting in more crops can be monitored and being treated properly.

In order for the scalar data and crop images obtained from the sensor nodes to be represented as useful information, data and images must be communicated back to the farmers. However, image data has several unique characteristics as compared with scalar data. First, image data is usually large in size whereas the scalar data such as temperature and humidity are usually represented as numerical values and the size is relatively small (usually one or two bytes) in which it is enough to be fit into one packet. For instance, because the maximum packet size for IEEE 802.15.4 standards can only be up to 128 bytes in size (Shelby & Bormann, 2009, p. 192), a crops image capture from the image sensor compressed with jpeg format normally would requires 5 Kbytes or more depending on the quality of the image file. This means that many packets are needed to be transmitted across sensor network back to the farmer.

In addition, image data is very sensitive to loss (one small portion of data loss will corrupt the entire image) whereas the scalar data can tolerate some loss in some situation. Transferring data through wireless channel may suffer from packet loss or error due to variety of environment factors such as multi-path fading, interference and so on. More seriously, as the number of hops (sensor node) increase, the probability of packet error rates will be increased exponentially and hence resulting in higher chance of packets being dropped when passing through multiple intermediate sensor nodes (Hu & Cao, 2010). Because of the unreliable packet transmission in wireless medium, it is hard to ensure that the image data which comprised of several packets being able to communicate back to farmer and to transform the data into useful environmental information to do further crops health analysis. Hence, a data

collection protocol should provide proper reliable data transport in agriculture monitoring.

However, existing data collection protocol in WSNs in agriculture monitoring only apply hop-to-hop acknowledgement and retransmission mechanism to reduce packet loss or error rate (Verma, et al., 2010), but not end-to-end.

Therefore, in order to cope with aforementioned problems, a reliable data collection protocol which provides end-to-end solution is needed to be developed to cater for transferring huge scalar data and crops images in a large-scale sensor network in agriculture monitoring. With this solution, the farmers or experts can get these critical environmental data and crops images from sensor network and utilize these data to diagnose the growth of the crops, yield and pests. Accurate decision use of water and fertilization can be made as well. Besides, the lifetime of the sensor nodes could be prolonged by not overwhelming the whole sensor network traffic caused by the data collection. Continuing monitor the agriculture could greatly save the labor cost while maximizing the production of crops. This could increase the economics of the country and benefit society as a whole.

1.2 Project Objectives and Scope

1.2.1 Project Objectives

The objectives of this project are:

- To enable and activate the functionalities of temperature and humidity as well as VGA camera sensors on MICAz platform.
- To design and implement reliable data collection protocol suitable for self-powered WSNs in agriculture monitoring.
- To collect the sensed data and image from MICAz using the protocol installed.
- To set up a database to store the sensed data and image file.
- To create user interface to display the environmental information and crops image.
- To carry out a performance study in terms of data packet reliability rate according to the number of hops.

1.2.2 Project Scopes

Agriculture monitoring can be classified into two categories which is crop monitoring (e.g. paddy, oil palms, plants, grains etc.) and livestock monitoring (e.g. to monitor the health of the cow) where different types of agriculture monitoring have different agricultural parameters to be sensed. Besides, the sensor nodes are usually deployed in a grid topology as it is very suitable for agriculture monitoring (Goh, et al., 2007). In this project, the scopes only focus on crop monitoring where the topology of the sensor nodes is usually static without constantly change.

Aforementioned under the section “Problem Statement and Motivation”, the final aim of this project is to come out with a solution which can properly collect and aggregate the scalar data and crops images in a reliable way in agriculture monitoring.

Therefore, in this paper, an application with reliable data collection protocol is developed based on the MICAz hardware platform which had been done by previous UTAR researchers. The MICAz platform consists of temperature and humidity sensors as well as color VGA camera module integrated and the platform is powered by solar energy. The end product in this project should be able to allow the MICAz platform to perform the sensing and collect the sensed data (temperature, humidity and crops image) through multi-hop communication in a reliable manner and send back to user.

At first, since the hardware (MICAz platform with solar panel and sensors) had already been done by previous researchers, therefore, the first part in this project focus on writing a program to enabling the functionalities of the sensors (temperature, humidity and VGA camera) to be used in that platform so that the MICAz is able to trigger the sensors to sense the data and send them back to user.

For the second part, in order to allow the sensed data to be communicated back to farmer, a reliable data collection protocol is designed and implemented to collect and aggregate the sensed data from sensor nodes. At first, the protocol is designed with parent and child relationship so that sensor nodes will know where it should route the data to when the sensed data is available.

Besides, in order to ensure data especially image file can be successfully arrived at receiver end, the combination of hop-to-hop recovery and end-to-end reliability mechanism such as ACK or NACK have to be design and implemented in the application. Moreover, as mentioned in “Problem Statement and Motivation” where there is a need to transmit several packets which composed of the entire image due to the limitation of WSNs, therefore, the segmentation mechanism is needed to be programmed in this application as well.

When the scalar data and segmented image data arrive at the sink node (the base station), the sink node is then forward them back to PC with database installed. However, the PC is required to check for any missing packet. The end-to-end ACK or NACK is therefore come into play to ask the sender to retransmit the missing packet. At the end, the PC is required to assemble those fragment packets into a single image file and store the data into database for further data analysis. User can view the information such as the crops image, temperature and humidity on the PC screen as well.

1.3 Main Contributions of this Project

The main contributions of this project are highlighted as below:

- 1a. Interfaced temperature and air humidity sensors using the existing mechanism which is done by the previous researcher in UTAR
- 1b. Interfaced embedded camera using owned developed mechanism that captures data stream from the camera part by part in RAM and then stores in flash memory.
- 2a. Developed new cross-layer (layer 2 and 4) protocol to allow transmission of image from a source to a base station.
- 2b. Developed DAT-ACK protocol for simple and raw data transmission though point-to-point.
3. Calibrated, processed and displayed the data and image.
4. Configured a database to store image and temperature and humidity values.
5. Developed a GUI to display image and temperature and humidity values.
6. Carried out performance study on packet reliability rates based on number of hops.

CHAPTER 2 LITERATURE REVIEW

2.1 Literature Review

In this section, four related papers were selected to be reviewed in details. The four papers are listed as follow:

- (1) COMMONSense Net: A Wireless Sensor Network for Resource-Poor Agriculture in the Semiarid Areas of Developing Countries (Panchard, et al., 2007)
- (2) Practical Considerations for Wireless Sensor Networks in Cattle Monitoring Applications (Kwong, et al., 2012)
- (3) Efficient Zone-based Routing Protocol of Sensor Network in Agriculture Monitoring Systems (Karim, et al., 2011)
- (4) Experiments with Reliable Data Delivery in Wireless Sensor Networks (Marin-Perianu & Havinga, 2005)

2.1.1 COMMONSense Net: A Wireless Sensor Network for Resource-Poor Agriculture in the Semiarid Areas of Developing Countries

This journal article aims to design and implement WSNs for agriculture monitoring targeting in developing countries (Panchard, et al., 2007). The data collection scheme that they used is through multi-hop transmission. The nodes are responsible to relay the data received from the other nodes and send them towards base station. As no mobility involved in the network, a tree construction algorithm is being used in this paper. The tree construction is based on the link quality and hop count to the base station.

This article has also pointed out that multihop routing algorithm from TinyOS caused the topology changes very frequently even the nodes is static. Besides, the excessive control messages are exchanged between nodes caused the batteries being used up very quickly.

2.1.2 Practical considerations for wireless sensor networks in cattle monitoring applications

In this journal article, Kwong, et al. (2012) presented both non-real-time and real-time data collection schemes where the former is applied for free ranging beef cattle and the latter is designed for loose housed dairy cattle.

For non-real-time data collection scheme, one of the methods is to use a data collector (mobile/portable) to download data from the nodes (the collar attached to the cattle) and relay back to the sink. The data collection is done in two phases which is pickup and dispatch phases. The collector will first handshake with nearby nodes within transmission range and start the pickup phase to download the data. When the source node receives an ACK message from collector, the source node will remove the sent data from its memory to free up memory spaces. During transmission, the pickup phase will also be triggered to broadcast the received data in memory to base station. If the collector doesn't receive ACK message from the base station, it will defer the current data transmission for T sec and retransmission again.

The other method is to deploy numbers of router within the farm to extend the transmission coverage of the base station. In this method, the two phases (pickup and dispatch) also need to be included in the process of data gathering just that it use the unicast to relay the data to next router or base station. Because of the pre-defined location between the routers and the base station that make the radio link robust enough, this method can then eliminate the needs of deferring the transmission for T sec.

Apart from the two methods mentioned above, an Implicit Routing Protocol (IRP) was proposed by the paper's authors to facilitate the real-time data collection. This routing protocol consists of two phases: configuration phase and data forwarding

phase. The configuration phase is happened when base station periodically floods the TIER message (contains base station's ID field and hop count field) throughout the network to maintain the information in the network. Initially, the hop count field will start from zero before base station start broadcasting, once the TIER message is being received by other collars, it increases the hop count value by 1 and the collars will help advertising the TIER message throughout the network. That TIER message is actually used to determine how many h -th tier away from base station. As for the data forwarding phase, when the collar wanted to transfer the data to base station, the collar will form a packet comprising its current TIER ID and sensed data before broadcasting to other collars. If the collar received the packet, an ACK packet will be generated and send to the source. Meanwhile, the collar will also inform other collars to discard the ACK packet as well. However, only collars with lower TIER IDs are responsible to receive and process the packet, the equal or higher TIER IDs will discard the packet. The collars with lower TIER IDs are then continuing broadcast the packet through multi-hop communication until it reaches at the base station.

2.1.3 Efficient Zone-based Routing Protocol of Sensor Network in Agriculture Monitoring Systems

This paper proposed an energy efficient zone-based routing protocol suitable for agriculture monitoring in developing countries where minimum amount of sensor nodes are deployed in the field.

In this protocol, the sensor nodes will first be localized by using the RELMA localization algorithm. The nodes will then be assigned into zone based on number of hops a node is away from base station (Karim, et al., 2011). Besides, active and passive nodes are also be chosen where the formal is used to sense the data and form

the shortest path to the base station while the later remain in sleep mode to save energy.

There are two path establishments being proposed in this paper. Both of the methods are established based on actual distance (meter) and energy level. Once the path has been established, the data can send through the shortest path.

Besides, fault tolerance can be achieved by using subscription paradigm and special packet as well. The base station will subscribe an event to all of the active nodes to indicate that sensed data need to be collected from particular active node. The active nodes either send the sensed data to uplink or a special packet being sent to indicate that it is still alive. The uplink can base on the received special packet to determine the status of the node and can exclude the failed node from its routing table if no special packet is received. The base station is then able to assign another active node among the alternative nodes in that zone.

2.1.4 Experiments with Reliable Data Delivery in Wireless Sensor Networks

This paper carried out experiments study regarding the use of cross layer approach with transport layer to solve the reliable data delivery problem. Three reliable data delivery solutions listed as below (Marin-Perianu & Havinga, 2005):

- (1) End-to-end acknowledgement for every packet
- (2) End-to-end, window-based acknowledgement
- (3) Same as 2, plus intermediary caching

In the first method, the acknowledgment message is required for every packet received by the receiver during transmission whereas in second method, a selective NACK and completed ACK are in used. If the receiver doesn't receive the packet, the receiver will generate an NACK with binary bit to indicate which packet is lost and

ask for sender to advance its window based on the binary bit and retransmit the packet again. The last method is similar to second method but to create additional buffer (same with window size) on every intermediate node along the path where the acknowledgement and retransmission of a packet is done by link layer. The node can also detect failure of the next node based on the information retrieved from link layer and new path can be created by broadcasting the current packet until the packet reach at receiver end. The ACK issued by receiver along the path can therefore determine a new path.

2.2 Critical remarks of previous works

In this section, critical remarks of these four papers' literature reviews papers are done:

Paper 1 - COMMONSense Net: A Wireless Sensor Network for Resource-Poor Agriculture in the Semiarid Areas of Developing Countries

Advantage:

1. Simplicity: Tree-structure with parent and child relationship is easier to be set up.

Disadvantage:

1. Unreliability: No reliable mechanism is introduced in this paper.
2. Bottleneck in upper layer: A bunch of data relaying to upper layer may cause the bottleneck in upper layer.

Comment:

Tree-structure with parent and child relationship is suitable for static topology and easier to implement.

Paper 2 - Practical considerations for wireless sensor networks in cattle monitoring applications

Advantages:

1. Increase network performance: In non-real-time data collection scheme, the data pickup phase only occur in between particular collar with router or collector, and the data is then directly broadcast to base station or other routers help relaying the data back to base station. Other collars are of no interest in transmitting the data will not be involved in the communication without overload the network and hence resulting in increasing the overall network performance.
2. Reduce packet delivery delay: Since the IRP uses the tier concept where the packet is broadcast to the collars, the collars will only accept the packet only if the tier ID is lower than its own tier ID. Due to this implicit tier relationship, IRP therefore doesn't require creating and maintaining an explicit routing path which result in shorter packet delivery delay.
3. Scalability: The IRP is proven to suit for loose housing of dairy cattle in which the cows do not move so frequently. Besides, based on the experimental result done, the IRP is also proven to be most effective for farms stocking a large number of cows. The more the cows in the farm, the easier the cows can establish the viable links in between them which resulting in increasing the received packet rate.
4. Header Simplicity: The message header for IRP protocol is rather simple and straightforward. It only contains the base station's ID and hop count for TIER

message. The node can then transmit the data by using the hop count value back to a base station.

Disadvantages:

1. Additional cost of installation and farm management: For router scheme, in order to further extend the radio coverage of a base station, the additional cost and management is needed to buy the extra routers, install and maintain them in the farm.
2. Excessive broadcast message: Base station periodic broadcast of the configuration message and also the needs of informing the same TIER IDs' nodes to discard the ACK message to the source will overwhelm the network and might result in network congested and battery will be drained more quickly.

Comments:

The router scheme is applicable for small-scale livestock farmland where only few routers are required to be installed.

The uplink and downlink in IRP is simple and straightforward and hence the concept of this protocol can be considered to be designed and implemented in this project. Since this project is focus is on crops monitoring where the nodes will always be static, the needs of periodic broadcast of configuration message can be eliminated. Besides, by using unicast instead of broadcast during data forwarding phase can eliminate the needs of informing same TIER IDs' nodes to discard the ACK message back to source.

Paper 3 - Experiments with Reliable Data Delivery in Wireless Sensor Networks

Advantages:

1. In approach 1, this reliable mechanism is very simple and straightforward, easier to implement as well.
2. In approach 2, since ACK or NACK message will be issued based on the window size, the control message being sent back and forth in a network will be lesser and hence reduce the overhead.
3. In approach 3, since every drop in packets is repaired locally, the needs of end-to-end control messages can be reduced resulting in less overhead.

Disadvantages:

1. In approach 1, the experiment result shows that this approach doesn't perform well and may incur overhead when the source is multiple hops away from sink.
2. In approach 2, Longer delay when the window size is large.
3. In approach 3, it needs to consume extra memory to buffer the packets. Latency will also be incurred as the nodes need to schedule the packets in the buffer before transmit them.

Comments:

The first approach is applicable only for small size network whereas the second approach is suitable when the error rate of the link is less. The third approach is good especially transmitting data over multiple hops.

Paper 4 - Efficient Zone-based Routing Protocol of Sensor Network in Agriculture Monitoring Systems

Advantages:

1. Shorter delay transmission: The shortest path is pre-calculated before data transmission takes place; therefore, sensed data can be delivered to base station in shorter delay time.
2. Reduce energy consumption: Only active nodes are required to sense the data within pre-calculated sensing range and transmit the data to other active nodes in other zone. The alternative nodes in the sensing range can therefore be turned off to save the power.
3. Fault tolerance: The failure of a node can be detected by using special packets.

Disadvantages:

1. High processing power and memory spaces is required in base station: The base station requires to setting up the zone, assigning/reassigning actives and passive nodes, determine the shortest path based on many route request messages received. When the network grows, base station may heavily load.

Comments:

This routing protocol is good in small-scale agriculture field. Large-scale network will make the network complicated as many procedure and steps needed to be done (such as zone creation and selecting shortest path) and could hardly be maintained as well.

2.2.1 Summary of advantages, disadvantages, and comments for 4 reviewed papers.

Paper	Advantages	Disadvantages	Comments
Paper 1	<ul style="list-style-type: none"> • Simplicity 	<ul style="list-style-type: none"> • Bottleneck may occur in upper layer • Unreliability 	<ul style="list-style-type: none"> • Tree-structure is suitable for static topology • Easier to implement
Paper 2	<ul style="list-style-type: none"> • Increase network performance • Reduce packet delivery delay • Scalability • Header Simplicity 	<ul style="list-style-type: none"> • Additional cost of installation and farm management • Excessive broadcast message 	<ul style="list-style-type: none"> • The router scheme is applicable for small-scale livestock farmland • The excessive broadcast message is required because of the mobility • This gives the idea of using unicast instead of broadcast for static topology

CHAPTER 2

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

<p>Paper 3</p>	<ul style="list-style-type: none"> • Approach 1: Simple and straightforward • Approach 2: Lesser control message being sent across the network • Approach 3: Hop-to-hop recovery reduce the chance of accumulating the packet error rate across multiple hops End-to-end control message can be reduced. 	<ul style="list-style-type: none"> • Approach 1: May incur overhead when the source is far away from sink • Approach 2: Longer delay when the window size is large • Approach 3: Extra memory is required in every node Latency will occur due to the needs of packet scheduling. 	<ul style="list-style-type: none"> • The first approach is applicable for small size network. • The second approach is suitable when the error rate of the link is less • The third approach is good for transmitting data over multiple hops
<p>Paper 4</p>	<ul style="list-style-type: none"> • Shorter delay transmission • Reduce energy consumption • Fault tolerance 	<ul style="list-style-type: none"> • High processing power and memory spaces is required in base station 	<ul style="list-style-type: none"> • This routing mechanism is good in small-scale agriculture field

2.3 Summary/ Concluding Remarks

By analyzing and studying the existing approaches done by previous researchers, the author decided to come out with a solution which combining the tree like structure with parent and child relationship for routing message purpose and hop-to-hop recovery for the scalar data whereas end-to-end reliability with segmentation mechanism for image data which can't tolerate to loss when the link is bad.

CHAPTER 3 PROCESS DEVELOPMENT

3.1 Methodology

In this section, two software process models which is traditional heavyweight methodology Waterfall Model and modern lightweight methodology Agile Model will be discussed and a software process models will be chosen that is suitable for this project.

3.1.1 Waterfall Model

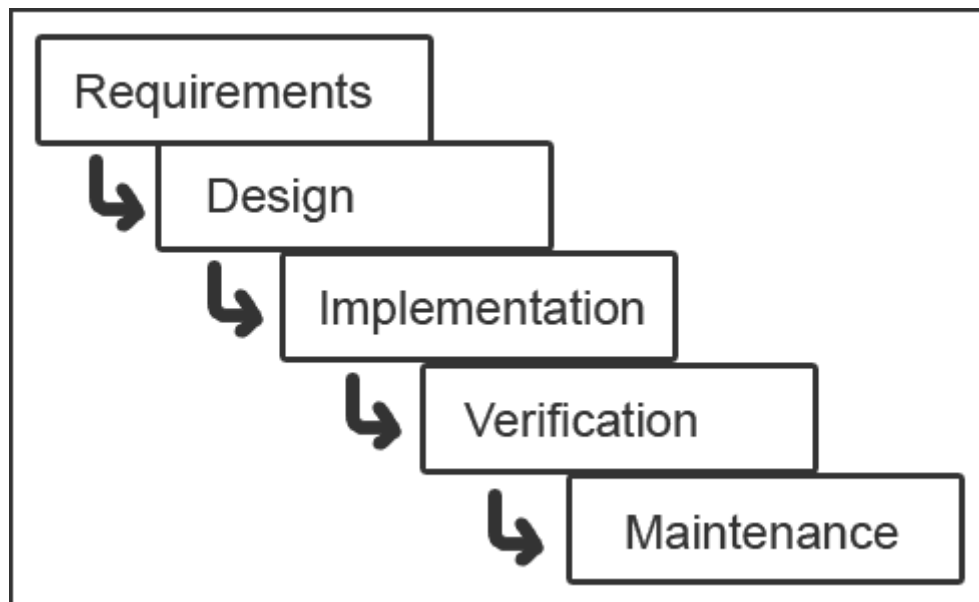


Figure 3.1.1.1: Waterfall Model (CompSci.ca, 2007)

Waterfall is a traditional heavyweight software process development model and it is a sequential development approach in which the phases, as can be seen in figure 3.1.1.1, are sequentially moving downwards throughout the whole phases. Each of the phases have its own deliverable and this deliverable is important for the next phase so that the process can be continue throughout the end of the life cycle.

This model is easy to understand and implement. However, each of the phases can't be overlapped, if one of the phases goes wrong, the next phase can't be returned back to the previous phase and fine tune it.

3.1.2 Agile Model

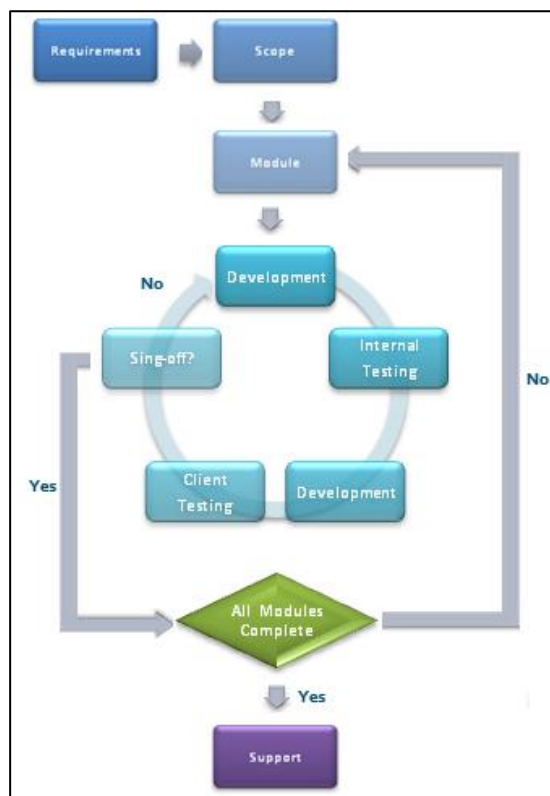


Figure 3.1.2.1: Agile Model (BayAmp, 2011))

Williams (2007) said that agile model is a kind of iterative and incremental methods which is based on iterative enhancement and opportunistic development process. Agile model emphasizes on prioritizing the project task into a 'to do' list and each of the list item can be moving into iteration phases. Each of the iterations is a self-contained, mini-project with activities that span analysis, development, testing. Once

one of the iteration is done, a small sub-system will be released and the sub-system is ready to integrate with other sub-system which is done by another iterative process. The duration of iterative process is about one to four weeks. The small duration is to allow having feedback from user and changes can be enhancement and additional requirement can be added for the next iteration. Once there are no tasks or additional changes from customer or scope, all the sub-system is integrated and a full system can then be ready to deliver to customer.

The advantages of adopting this model is having shorter iterations can reduce complexity and risk, better feedback from customer and higher productivity and success rates. The disadvantages of this model are the difficulty of handling large team members and the plans or requirements changed often, it may lead to project can't be able to complete within timeline.

3.1.3 Selecting a Software Process Model

After considering both model as discussed in section 3.1.1 and 3.1.2, Agile Model will be chosen as a software development life cycle throughout the whole project lifetime. The reason of choosing this model is because it can divide the task stated in the scope into several mini and feasible tasks, and each subsystems is moving into the "looping state" until the subsystems is met the requirements from the user. In this project, as can be seen in figure 3.1.3.1, the project will start with determining the requirements of the project, defining the scope and divide the scope into separate tasks.

In general, this project will be divided into three main tasks, which is enabling the functionalities of the sensor on MICAz platform, designing and implementing the protocol that is suitable for agriculture monitoring as well as designing and implementing database and user interface. Supervisor and moderator will be our main

customer and mentor throughout the project. Once each of the tasks is almost completed, the end result will be given to them to see if there is any flaw in the sub system and enhancement can be made through iteration again. Once the listed tasks are completed, integration among the tasks will be performed, in this stage, the project is almost finished and support phase is where the possible maintenance or future enhancement can be made by either the author or future FYP student.

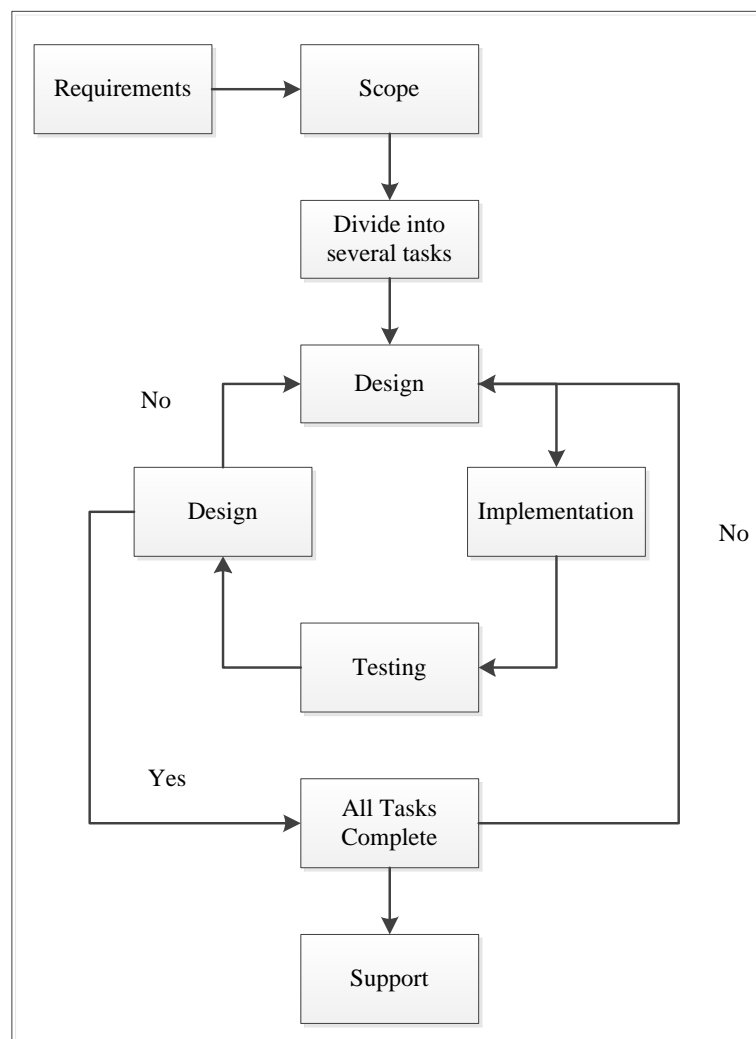


Figure 3.1.3.1: Project's Software Process Model

3.2 Requirement Specifications

As previously mentioned under the section objectives and scope, this project is aiming to develop an application with reliable data collection protocol based on the MICAz platform with solar panels. Therefore, in this section, system requirements, system performance definition as well as system verification plan will be listed.

3.2.1 System Requirements

The system requirements which comprised of hardware and software components are listed as follow:

Hardware

- MICAz platform with solar panels and SHT11 sensors and C329 camera integrated
- MICAz Motes
- MIB520 Mote Interface Board
- Personal Computer which support TinyOS

Software

- TinyOS
- MySQL database
- Windows or Linux operation system

For the hardware requirement, The MICAz platform will be mainly used in this project. The MICAz platform and some MICAz Motes (which act as intermediate nodes) are needed to be programmed so that they are able to perform sensing and relaying data. MIB520 Mote Interface Board is therefore allow the MICAz Motes to be programmed by interfacing the MICAz Mote with its 51-pin expansion connector and connects to PC through USB port. Besides, MIB520 Mote Interface Board is

needed to connect with 1 MICAz Mote to act as base station so that any data can be sent and received over the USB connection to an attached PC.

As for the software requirements, TinyOS will be the main part in this project. TinyOS is an operating system which is specifically designed for WSNs. It allows programmer to be able to develop an application and install it on a sensor network with relatively low cost and small size. Besides, Java is needed in this project so that the MIB520 Mote is able to interact with computer and therefore data is able to be shown on PC screen as well. Therefore, the software development kit for Java is needed to be installed as well. TinyOS can be run on either Windows or Linux operating system platform, if Windows were to be chosen to run TinyOS, Cygwin is therefore required to be installed in Windows so that it can emulate the Unix-like environment to allow TinyOS to use some UNIX tools. In addition, native compilers and nesC compiler are required to be installed as well. For database, MySQL is chosen to install in the PC connected with Java so that the received data can be store into database.

3.2.2 System Performance Definition

There are 3 parameters which define the system performance for this project. The first parameter is to determine the data packet reliability rate according to the number of hops, while the second one is to determine the time delay to receive image data and the last one will be to determine how many duplicate packets receive by the receiver.

3.2.3 System Verification Plan

To verify the 3 parameters as mentioned under system performance definition, 3 of the parameters will be tested individually starts from 1 hop up to 2 hops and the distance between each node is about 10 meter away.

For data packet reliability rate, the source node will start transmitting both scalar data as well as crop image data packets with each packets assigned with unique consecutive sequence number, after the transmission is done, the receiver can then base on the gap in between sequence number to calculate the reliability rate from hops to hops. Another way is to calculate based on the amount of end-to-end NACK or ACK sent by the receiver to sender node to ask for retransmission.

For total time delay to receive whole image data, the author will calculate the time start from the first segment image received, then the time end when all segments of image have transmitted to the PC. This test will also be conducted according to number of hops (maximum 2 hops) as well.

The last verification plan would be to calculate the total amount of duplicate packets received by the receiver. This test can be realized by summing the total amount of the packets which has the same sequence number shown at the receiver's PC.

3.3 Timeline

Project Task	Project Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Determine FYP title	█	█												
Data Collection			█	█										
Determine Project Scope and Objectives				█	█									
Doing Literature Review					█	█	█							
Determine the technologies involve							█							
Finalize a complete preliminary report and submit on week 9							█	█						
Determine software process model to be used in this project									█					
Do some enhancement on final report based on the mistake found in preliminary report											█			

CHAPTER 3

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

Determine the system requirement specification and sketch out timeline and budget																		
FYP 1 documentation																		
FYP 1 Presentation																		

Table 3.3.1: The gantt chart showing project tasks for FYP1 in Year 3 Semester 1.

Project Task	Project Week													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Enabling the sensors and camera's functionalities on the MICAz platform and test it using single hop communication.														
Analysis, design and implement reliable data collection protocol in MICAz platform and test it using some test data through multihop communication.														

CHAPTER 3

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

Integrate the sensors and camera's functionalities coding with the reliable data collection protocol and do some multihop communication data collection testing.														
Design and implement database and User Interface.														
Carry out performance study as stated in requirement specification plan.														
FYP 2 Documentation														
FYP 2 Presentation														

Table 3.3.2: The gantt chart showing project tasks for FYP2 in Year 3 Semester 3.

3.4 Technology Involved

3.4.1 Hardware Platform

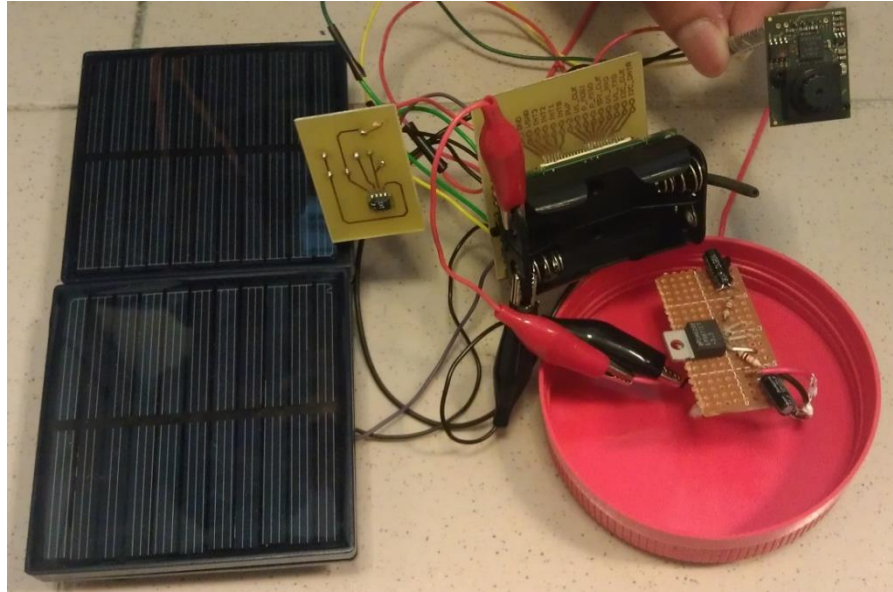


Figure 3.4.1.1: MICAz platform with solar panels and SHT11 sensor and C329 camera integrated.

Figure 3.4.1.1 shows the whole MICAz platform with solar panel done by previous researchers. In this project, the author will implement the application in this MICAz platform where the MICAz is powered by 2 AA rechargeable batteries and a solar panel which can recharge the batteries as long as the solar panel has enough current voltage and ampere. Besides there is an interface board attached with MICAz which can allow the SHT11 sensor and C329 camera to be integrated, The SHT11 sensor from SENSIRION product (SENSIRION, n.d.) is capable of sensing the temperature and humidity and C329 VGA camera from Electronics123 product (Electronics123.com, Inc. n.d.) is able to capture image in 640 x 480 resolution JPEG format.

MICAz



Figure 3.4.1.2: Two devices from Crossbow Technology, Inc – MIB520 Mote Interface Board (left) and MICAz mote (right) (Crossbow, n.d.)

The MIB520 Mote Interface Board has an USB port where it can use as programming board as well as communication board. The MICAz mote is mainly used in this project where it acts as sensing nodes (the MICAz with solar panels) and intermediate nodes.

The MICAz is a tiny wireless measurement system with 2.4 GHz IEEE802.15.4 which offering high data rate radio (250kbps). The expansion connector can be attached with variety type of sensor such as light, temperature, acceleration and so on (Crossbow, n.d.). The MICAz can be functioned as base station by attaching it with the MIB520 where the data will be transferred through the USB interface onto the server.

Personal Computer

Personal computer is needed in this project so that the scalar data as well as images can be processed and store them into database. User can see the parameters and crop image on pc screen as well. The minimum requirements are listed as follow:

- Support TinyOS, Java, and MySQL database
- 2 USB ports
- 1 GB RAM, 512 harddisk space.

- At least Core-duo dual processors

3.4.2 Software

TinyOS

TinyOS is an open source, embedded operating system written in nesC programming— a dialect of the C language designed specifically for WSNs. TinyOS supports event-driven concurrency model based on split-phase interfaces, asynchronous events, and deferred computation called task. The applications are usually small in size to optimize memory, therefore it is very suitable for WSNs (Philip et al., 2004).

TinyOS also supports Java and this allows the author to write the application to handle the raw data and image files received from the base station. The Java can log the data and image into a database as well. In this project, Java JDK 1.5 version will be chosen as it is compatible and stable for the TinyOS and can be supported in most end-user platforms.

Database

MySQL is an open source relational database management system (RDBMS). MySQL is very easy to use; provides high performance as well as high reliability (Oracle Corporation, 2012). MySQL supports many programming languages with its APIs to connect the MySQL database and access the data. Besides, MySQL also supports JDBC driver for Java in which both technologies can be easily connected. In this project, the MySQL will be responsible for storing and retrieving both raw data and image files.

3.5 Budget Plan

The budget plan for this project is shown in table 3.5.1.1. As mentioned earlier where the hardware platform had already done by UTAR researchers, therefore, no additional items are required to buy in this project. Besides, total 10 MICAz Motes are required to carry out this budget. (1 MICAz will connect to the platform which includes solar panels, and sensors, while another MICAz is required to connect with MICAz Programming Board to act as base station so that data can be forwarded to receiver end. The additional 8 MICAz Motes are required to act as intermediate nodes for data relaying purpose.). As for the batteries usage, only one pair of Eneloop AA rechargeable batteries will be used on MICAz with solar panels platform. The intermediate nodes will be using normal AA batteries.

In addition, table 3.5.1.2 shows the budget plan for the purpose of real field deployment in crops field such as paddy field. For one block size (around 6000 square meters) of paddy field, it requires 15 units of sensor nodes to be covered the whole area. As shown in figure 3.5.1.3, the sensor nodes can be deployed in a grid topology and the distance in between two nodes is about 10 meter away. In order to ensure that the sensor nodes are able to communicate with each other, the transmission range must be set no lesser than 20 meter. Besides, solar panels can be equipped with sensor nodes which have the high traffic load, sensor nodes which near to base station indeed require solar panels to recharge. In this budget plan, to make it simplicity in calculation, all of the sensor nodes will be equipped with solar panels, SHT11 sensor and C329 camera. There could be cellular network coverage in between user and base station so that the user can remotely control the whole sensor nodes. However, the budget plan does not include the price of setting up and subscribing the network services. Again, please be noted that this budget plan is only an estimated price and the price might subject to change especially for MICAz Motes.

CHAPTER 3

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

Item	Quantity	Price (Per Unit)	Price
Temperature and Humidity Sensor (SHT11)	1	Supplied by UTAR	-
C329 VGA Camera	1	Supplied by UTAR	-
MICAz Mote (MPR2400)	10	Supplied by UTAR	-
MICAz Programming Board (MIB520)	1	Supplied by UTAR	-
AA Batteries (per pair)	9	Supplied by UTAR	-
Eneloop AA Rechargeable Batteries (per pair)	1	Supplied by UTAR	-
Solar Panels (per pair)	1	Supplied by UTAR	-
Other Components and PCB	1	Supplied by UTAR	-
Personal Computer	1	Supplied by UTAR	-
MySQL	1	Freeware	-
TinyOS, Java, nesC	1	Freeware	-
Total:			RM 0.00

Table 3.5.1.1: Budget plan to carry out this project.

CHAPTER 3

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

Item	Quantity	Price (Per Unit)	Price
SHT11 Temperature and Humidity Sensor	15	RM 75.00	RM 75.00
C329 VGA Camera	15	RM 150.00	RM 150.00
MICAz Mote (MPR2400)	16	RM 650.00	RM 9,750.00
MICAz Programming Board (MIB520)	1	RM 450.00	RM 450.00
Enloop AA Rechargeable Batteries (per pair)	15	RM 89.00	RM 1,335.00
Solar Panels (per pair)	15	RM 40.00	RM 600.00
Other Components and PCB	15	RM 50.00	RM 750.00
Personal Computer	1	RM 1,500.00	RM 1,500.00
MySQL	1	Freeware	-
TinyOS, Java, nesC	1	Freeware	-
Total:			RM 14,610.00

Table 3.5.1.2: Budget plan for implementing the system into one block size (60m x 100m) of crops field. (e.g. paddy field).

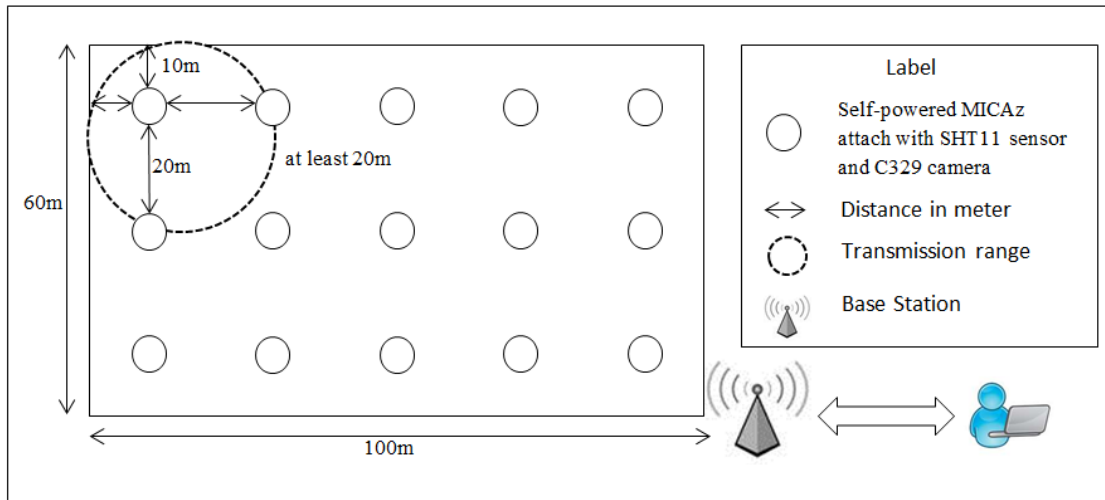


Figure 3.5.1.3: Layout of self-powered WSNs deployment on one block size (60m x 100m) of crops field (paddy field).

3.6 Summary/ Concluding Remark

In a nut shell, agile model is chosen to be used throughout the whole FYP life cycle. The system requirement specifications are stated so as to allow the author to prepare for the FYP2. Project timeline and budgets are also listed to ensure the project can be delivered in time while controlled the budget plan listed in this chapter.

CHAPTER 4 SYSTEM DESIGN

4.1 System Architecture

4.1.1 System Overview

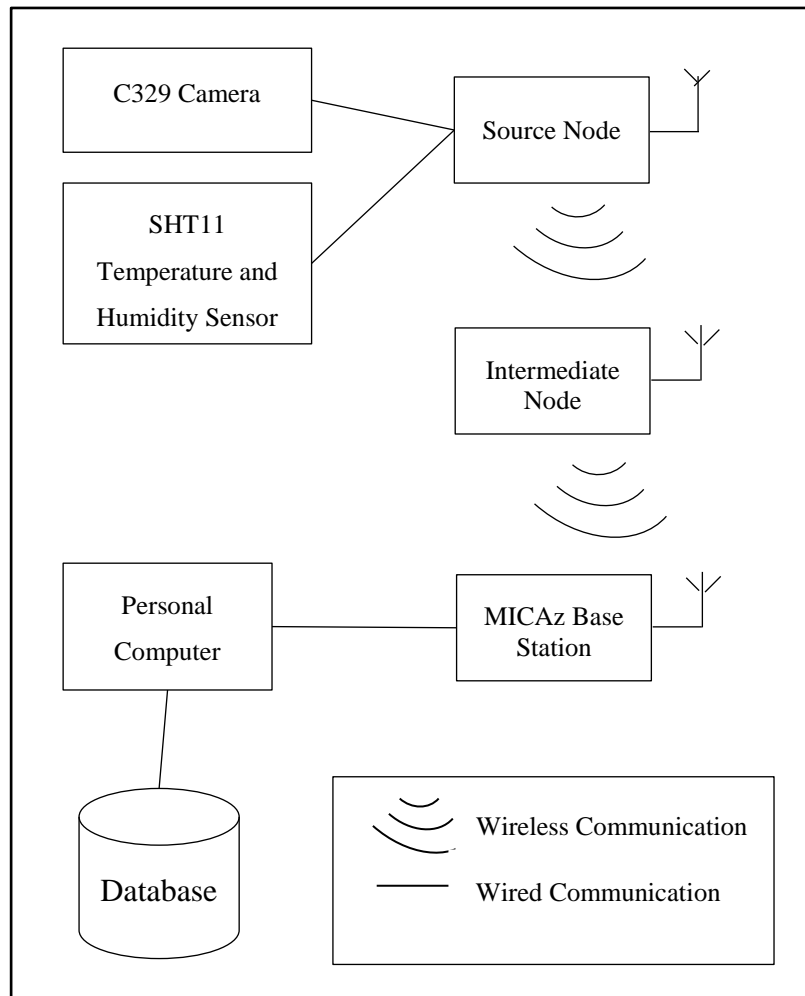


Figure 4.1.1.1: Overview system design of the project.

Figure 4.1.1.1 shows the overall system design of the project. The source node which equipped with C329 camera and SHT11 sensor will do the sensing and send the data back to end user. The data gathered from source node are passed through intermediate node to help relaying the data back to base station. The base station is connected with MySQL database and all the data are stored into database automatically. The user can view the raw and image data on the screen after the processing is done.

4.1.2 Source Node

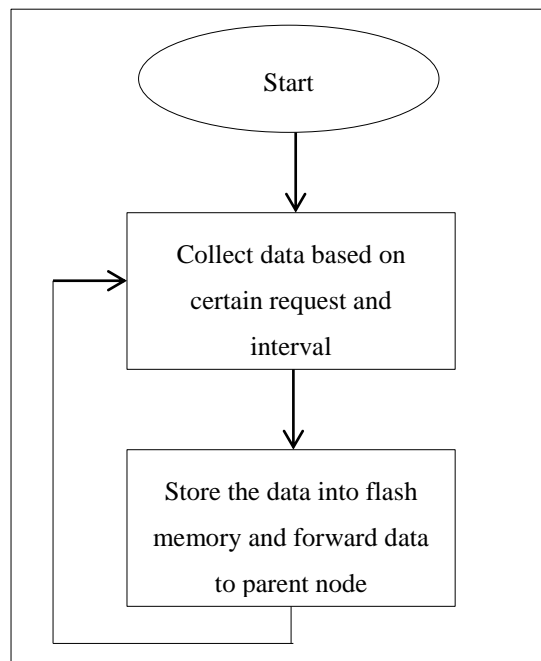


Figure 4.1.2.1: Simple flow diagram for the source node.

As mentioned earlier, the source node is equipped with SHT11 sensor and C329 camera to do the sensing. The design of temperature and humidity readings is based on push-based protocol where SHT11 sensor is triggered and generate

readings periodically whereas the image capture is based on pull-based protocol where image can be queried by user. After the sensing is done, all the data are stored into flash memory to overcome the limit size of MICAz RAM. Finally, the source node read the data from flash memory and forwards them to intermediate node (parent node).

4.1.3 Intermediate Node

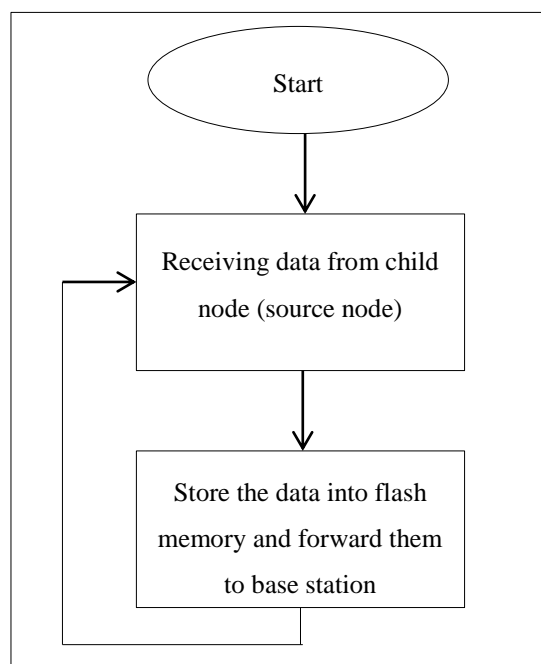


Figure 4.1.3.1: Simple flow diagram for the intermediate node.

The intermediate acts as a relay node where it just helps relaying data whenever it receives data from source node. After receiving data, it will first store them into flash memory before sending them to the base station.

4.1.4 Base Station

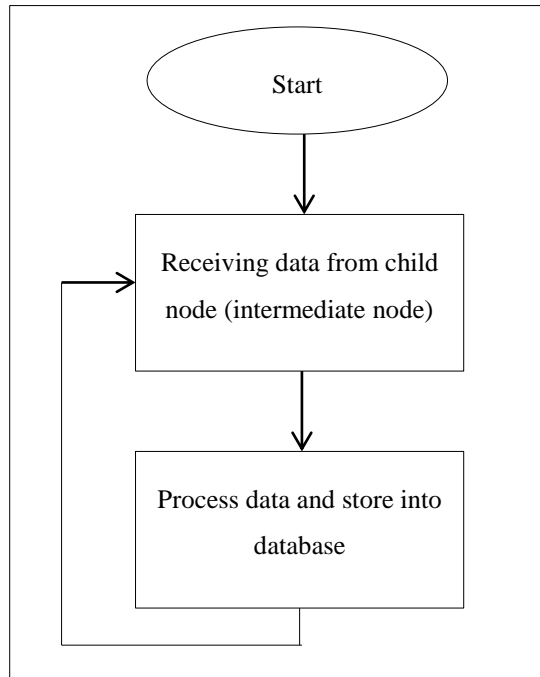


Figure 4.1.4.1: Simple flow for the base station.

In this project, base station is connected directly to PC with database installed. There can be cellular network in between base station and PC for remote access purpose; however, it is not covered in this project.

The base station keeps listening on the channel to receive packets. It is also responsible for processing, storing the data that receive from intermediate node into database for future retrieve purpose.

4.1.5 Graphic User Interface

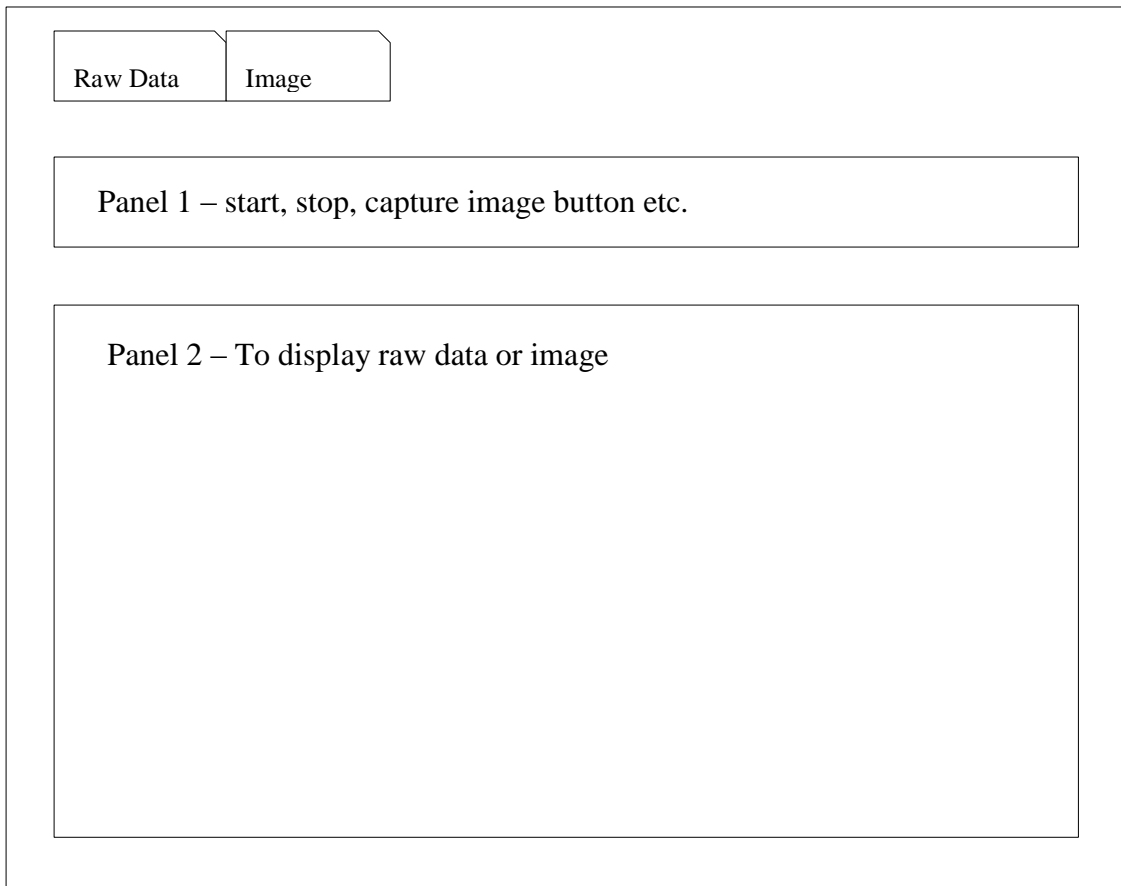


Figure 4.1.5.1: Simple GUI for displaying raw data and image

Figure 4.1.5.1 shows that the simple GUI design for this project. In this design, there will be two tabs which are raw data and image on top of the GUI. User can start and stop getting raw data or capture image from the button on panel 1. The default tab will be raw data tab and all the raw data such as temperature, humidity will be shown on the panel 2 text area whereas the second tab is designed particularly for showing image purpose.

4.1.6 Overall Requirements

The C329 camera is able to trigger and snapshot as per user request while the SHT11 sensor is able to trigger and start the environment readings periodically.

All sensor nodes (source, intermediate and base station) should form a parent and child relationship so that sensor node could know where to pass the data when data is ready. All the readings and image data collect from source node are able to transmit through multi-hop to a base station and forward to PC.

All the data readings and images must be stored into flash memory before sending out. The child node, for instance source node, should check the parent node's (intermediate node) connectivity before reading data from flash memory and sending out the data.

To tackle packets lost during transmission, the sender node should be able to receive acknowledge message from next hop to indicate the data has been received. The whole image should be sending segment by segment to overcome the limit MTU size (128 Bytes) of WSNs. The receiver end should be able to check for any losses of packet and query missing packets from the sender. The receiver should reconstruct the whole image back when receiving all segmented images.

The base station should forward all the data to PCs and store them into database. The user can view the temperature and humidity readings as well as querying image from source node.

4.2 Flow Charts of Wireless Sensor Network

Section 4.1 only shows the simple flow for each individual node. In this section, all the sensors node operations and the design of reliable data collection protocol were discussed in more detail.

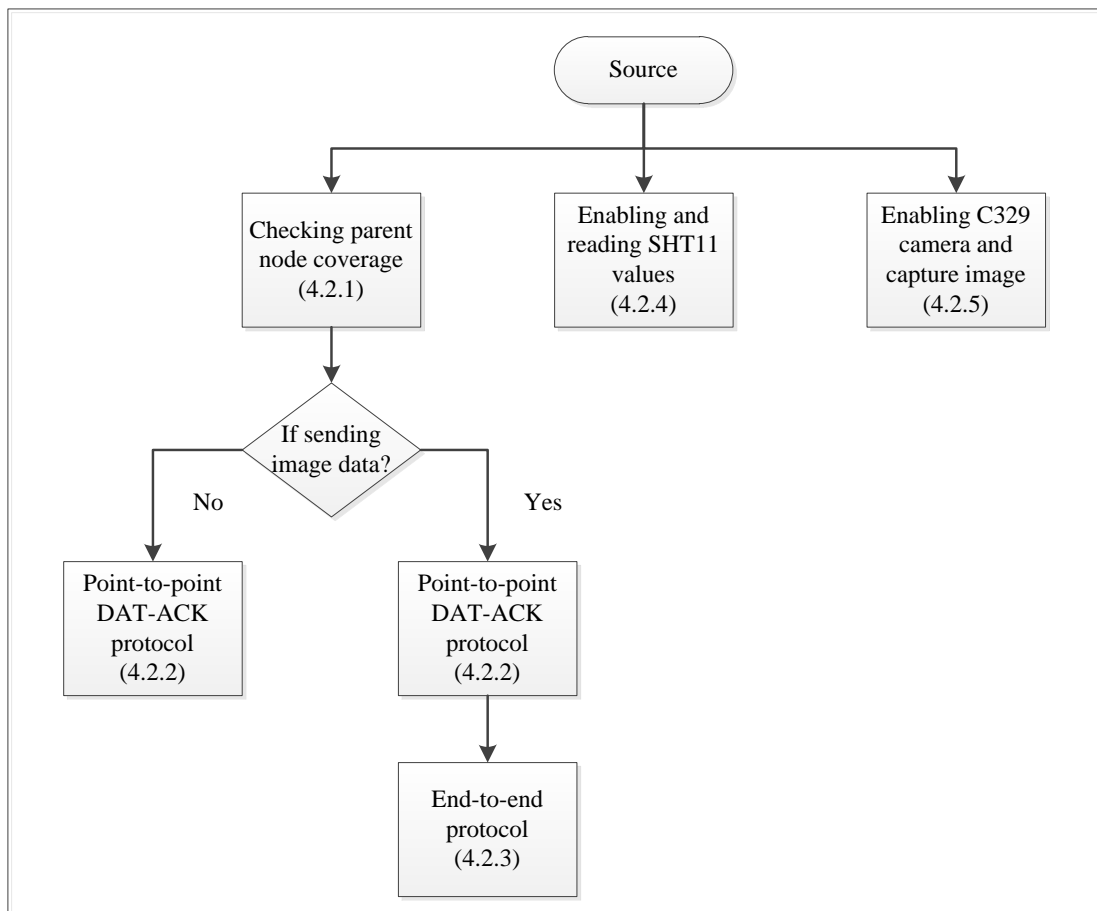


Figure 4.2.1: Main modules for source node.

Figure 4.2.1 shows the main modules included in the source node designed. There are three different modules which are checking parent node's coverage, enabling and reading SHT11 sensor as well as enabling C329 camera and capturing

image. There are two protocols involved in this project which is point-to-point and end-to-end protocols. The point-to-point protocol is only dealing with sending raw data (temperature and humidity readings) whereas point-to-point and end-to-end protocols are combined to deal with sending image data. The details process flow of each module is explained in the following sub-sections.

4.2.1 Flow Chart of Checking Parent's Node Coverage

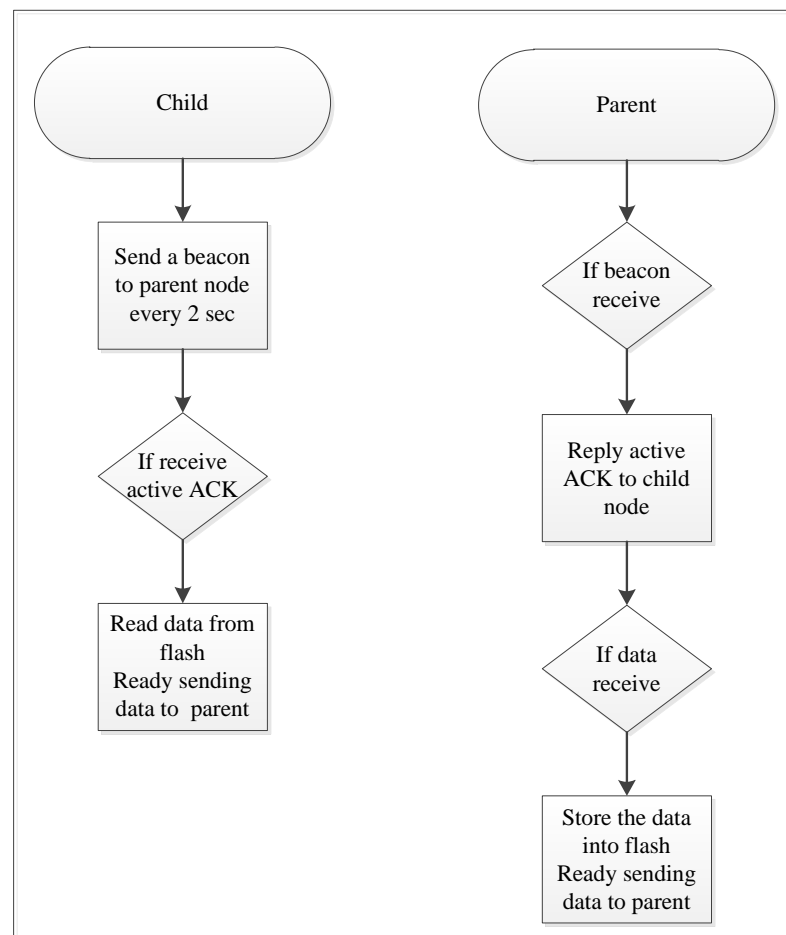


Figure 4.2.1.1: Flow chart of Active ACK

Figure 4.2.1.1 shows the flow of checking's parent node coverage by sending a small beacon message to parent. The checking of parent's coverage should have done before the actual sending data have made. If parent reply active ACK message to child node, this means that the parents is in coverage area, the child node can then read the data from flash and send the data out to the radio.

4.2.2 Flow Chart of Layer 2 Point-to-point DAT-ACK Protocol

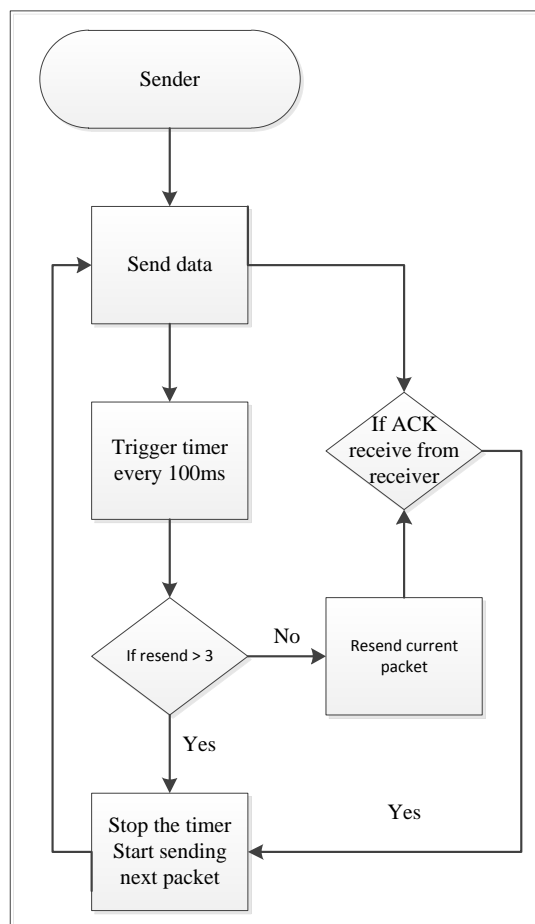


Figure 4.2.2.1: Process flow of layer 2 point-to-point DAT-ACK protocol

Figure 4.2.2.1 shows the design of the point-to-point DAT-ACK protocol. Sender triggers the timer with 100ms interval whenever sending data out. If timeout occur, the data is being resent up to 3 times. If an ACK message is received from receiver, the sender will stop the timer and proceed to sending next packet. However, if ACK message never received and the current packet has been sending up to 3 times, the sender will give up sending the current packet and proceed to sending next packet.

4.2.3 Flow Chart of Layer 4 End-to-end Reliable Data Collection Protocol

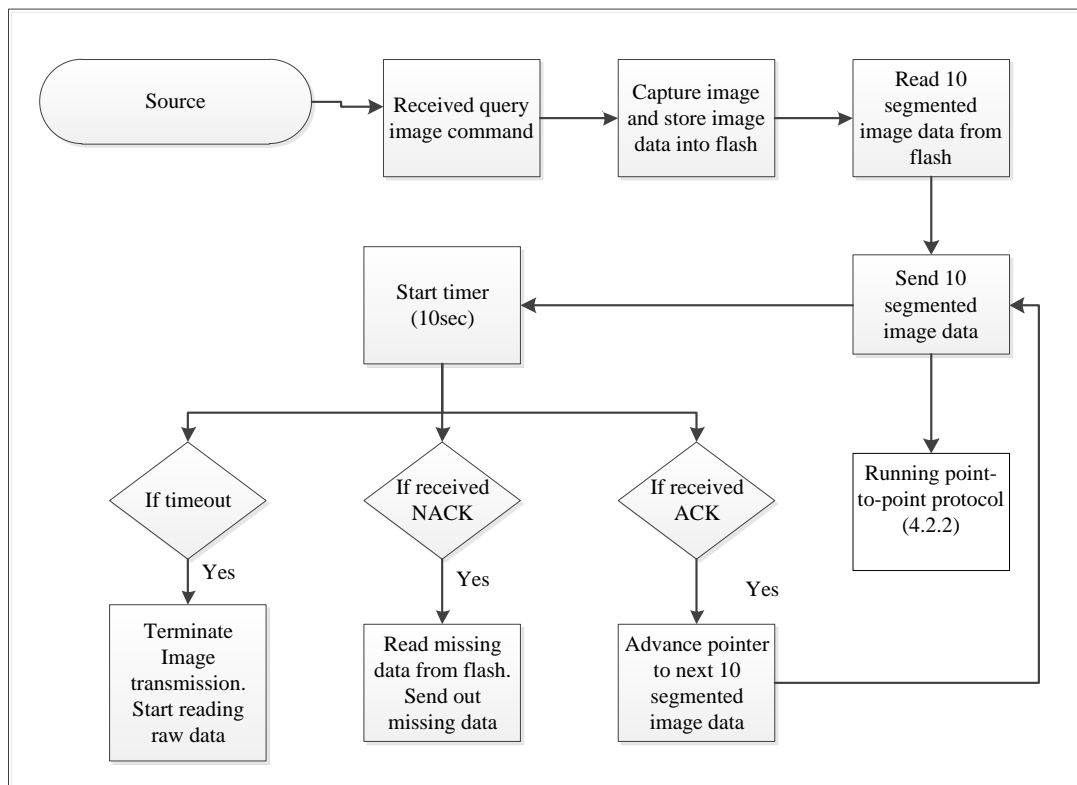


Figure 4.2.3.1: Process flow of end-to-end reliable data collection protocol (Source)

Figure 4.2.2.1 shows the design of end-to-end reliable data collection protocol for the source which is source node in this project. This protocol is designed for transferring image data. When the source node received image query command from base station, the source node will capture the image using C329 camera and store the image data into flash memory. The source will read 10 segments of image per round (1 segment = 70bytes) before sending the segments out. All the packets are required to go through the layer 2 point-to-point DAT-ACK protocol as mentioned in 4.2.2 as well.

The timer is required to run when the source has sent out 10 segments of image. If an ACK message is received before timer exceed, this means that the base station has received all the 10 segment images in the current round. The source can then advance the pointer the point to the next 10 segment image and ready to be transferred. If NACK received, this means that the base station has discovered lost segment and request for retransferring it. If timeout occur, the source will just terminate the current image's transmission and start proceeding to reading raw data. This can avoid the source node from waiting for layer ACK message and hanging forever.

Figure 4.2.2.2 shows the protocol design for the base station. At first, the base station (host) sends an image query command to source node and start the timer waiting for receiving image segment. For every 10 segments received or timeout occur, the base station will invoke checking missing packet to check for missing segments. If no lost is found for the current round (10 segments per round), the base station will just send a layer 4 ACK to source node as an indication of no segment lost is found. If lost is found, NACK which carrying the missing lost packet's ID will be sent to source node. All the ACK and NACK packets are gone through layer 2 point-to-point protocol as well to minimize the lost packet from base station to source.

Another timer will be invoked so that when source failed to resend back missing packet, the user will be notified and ask for requesting to capture another new picture. If the base station received the last packet, the base station is required to reconstruct back the whole image.

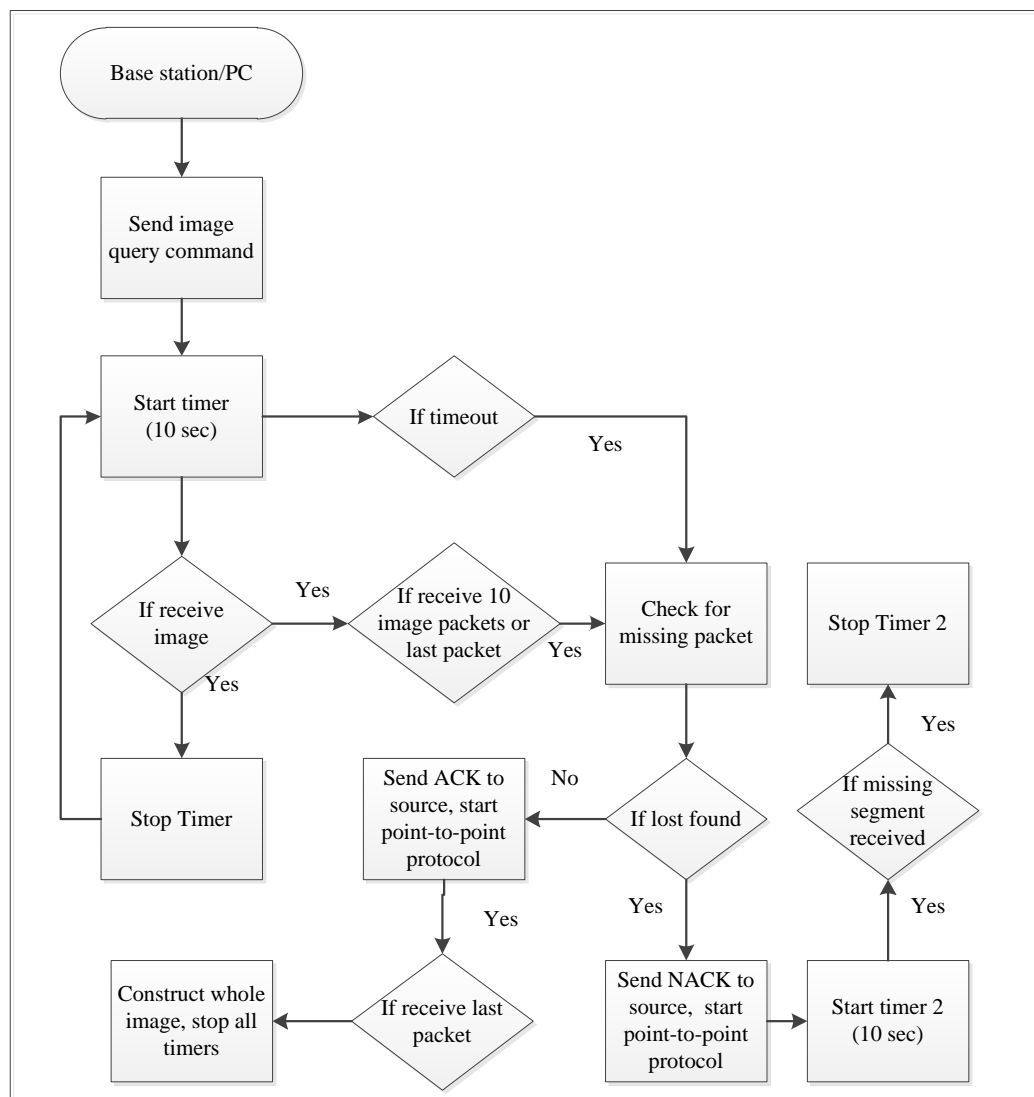


Figure 4.2.3.1: Process flow of end-to-end reliable data collection protocol (Base Station / PC)

4.2.4 Flow Chart of Reading SHT11 Sensor Values

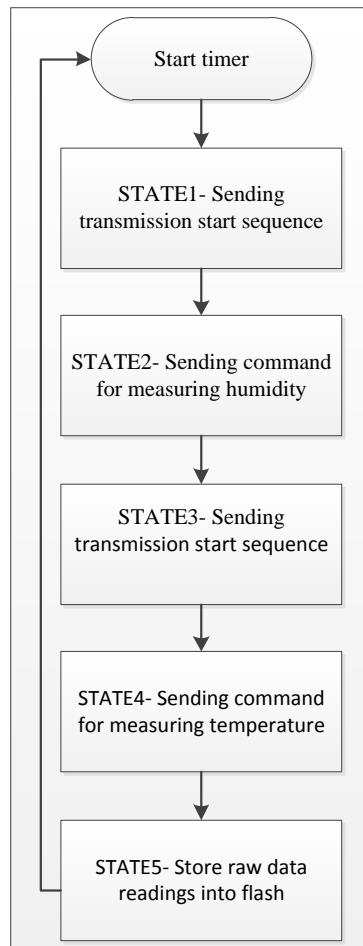


Figure 4.2.4.1: Flow diagram of reading SHT11 values

Figure 4.2.4.1 shows the flow of starting up SHT11 and reading both temperature and humidity data. The transmission start sequence has to be signalled before the actual reading can be made. The first reading would be humidity value then goes to the temperature value. After both reading are done, the data are stored into flash memory and the timer is then started again.

4.2.5 Flow Charts of Enabling C329 Camera

Command	Cmd Token	Parameter 1	Parameter 2	Parameter 3	Parameter 4
INITIAL	FFFFFF01h	Baud Rate	Colour Type	Preview Res.	Compres. Res
GET PIC	FFFFFF04h	Pic. Type	00h	00h	00h
SNAPSHOT	FFFFFF05h	Snapshot Type	00h	00h	00h
RESET	FFFFFF08h	Reset Type	00h	00h	00/FFh
POWER OFF	FFFFFF09h	00h	00h	00h	00h
DATA	FFFFFF0Ah	Data Type	Length Byte 0	Length Byte 1	Length Byte 2
SYNC	FFFFFF0Dh	00h	00h	00h	00h
ACK	FFFFFF0Eh	Cmd Token	ACK counter	00h	00h
NAK	FFFFFF0Fh	00h	NAK counter	Err. Number	00h
QUALITY	FFFFFF10h	Quality Level	00h	00h	00h

Figure 4.2.5.1: List of commands supported by C329-UART camera module
(Electronics123.com, Inc. 2010)

C329 camera uses UART serial interface to communicate with external embedded device. To enable C329 camera, the host has to send some commands supported by C329 camera. Figure 4.2.5.1 shows the list of commands that can be sent to/from embedded device and C329 camera. Each of the commands is 8 continuous bytes.

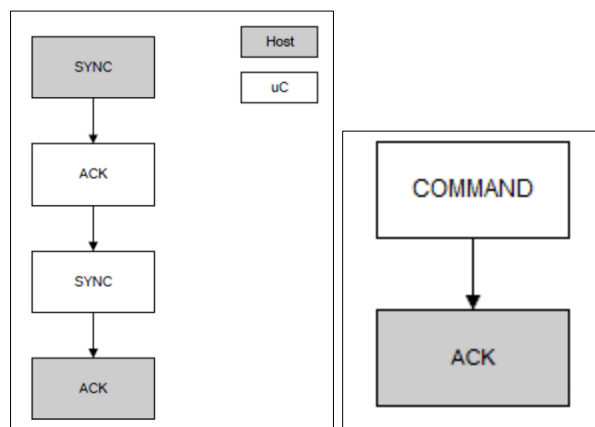


Figure 4.2.5.2: Simple flow of using commands set. (Electronics123.com, Inc. 2010)

Figure 4.2.5.2 shows how the host can communicate with C329 camera by sending the necessary commands. At first, host has to be sending SYNC command to synchronize with C329 camera baud rate. After successfully synchronized, the host can then send the commands to choose the options and set the quality of the image. Figure 4.2.5.3 below shows the get picture command being sent by host and get the whole image data.

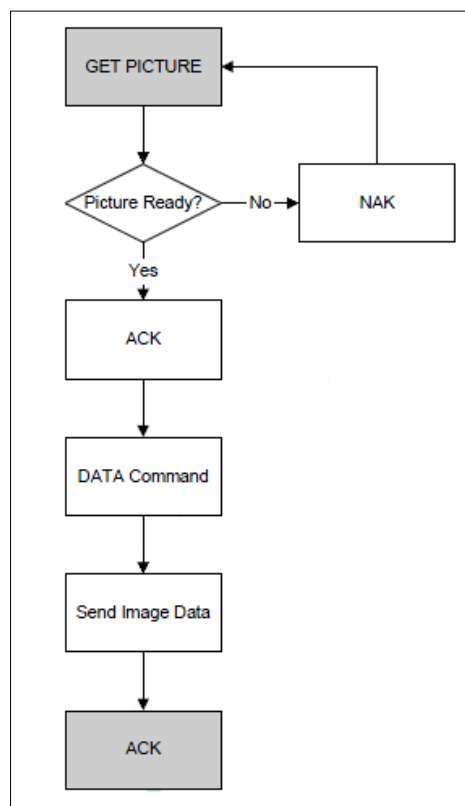


Figure 4.2.5.3: Flow chart of getting picture (Electronics123.com, Inc. 2010)

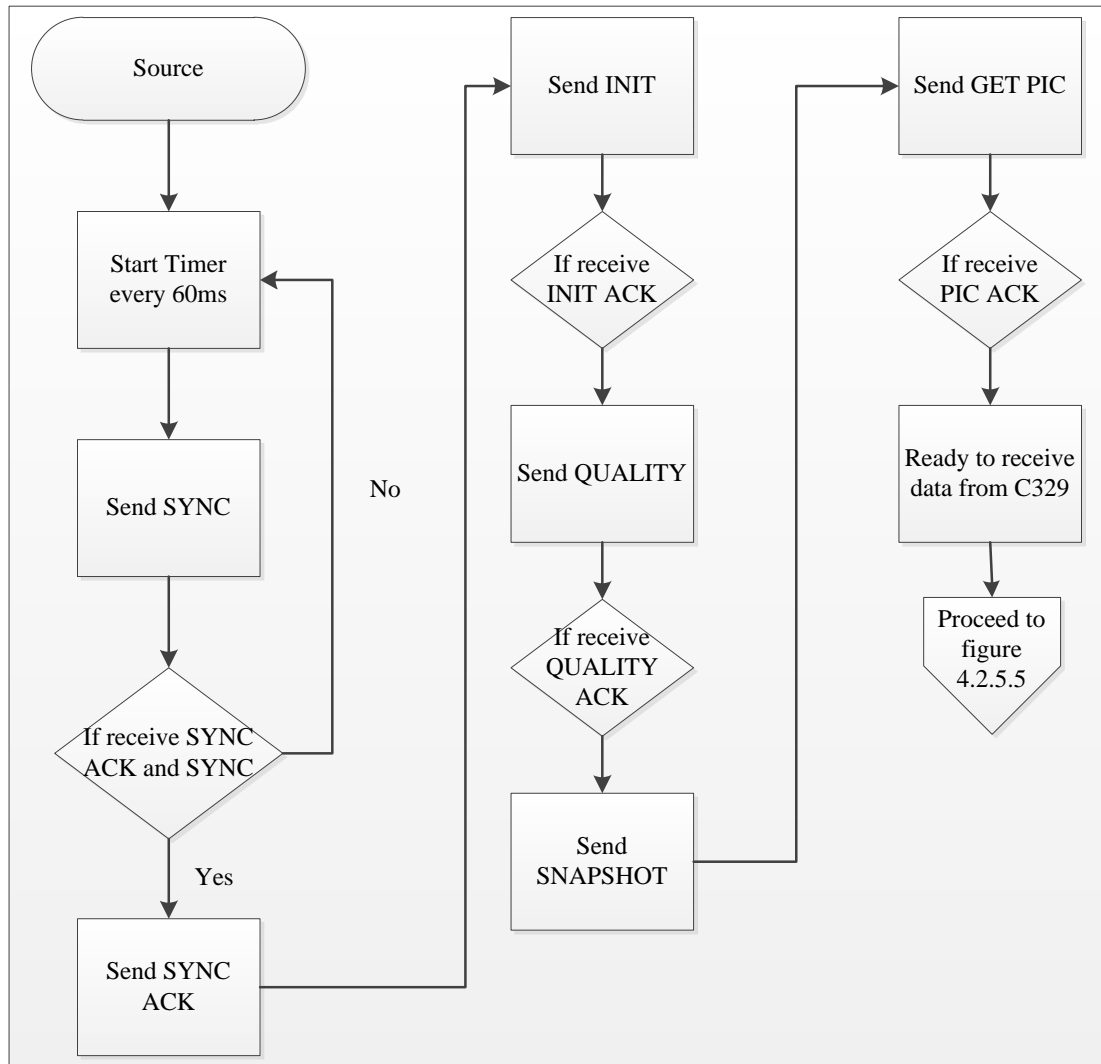


Figure 4.2.5.4: Flow chart of Enabling C329 camera

Figure 4.2.5.4 shows the process flow of enabling C329 camera. At first, a timer will start 60ms periodically to synchronize the camera. After C329 replied, the host can start sending the commands listed in the flow chart.

After all the necessary commands being sent and received, the host can then be ready to receive whole image data which come in one shot.

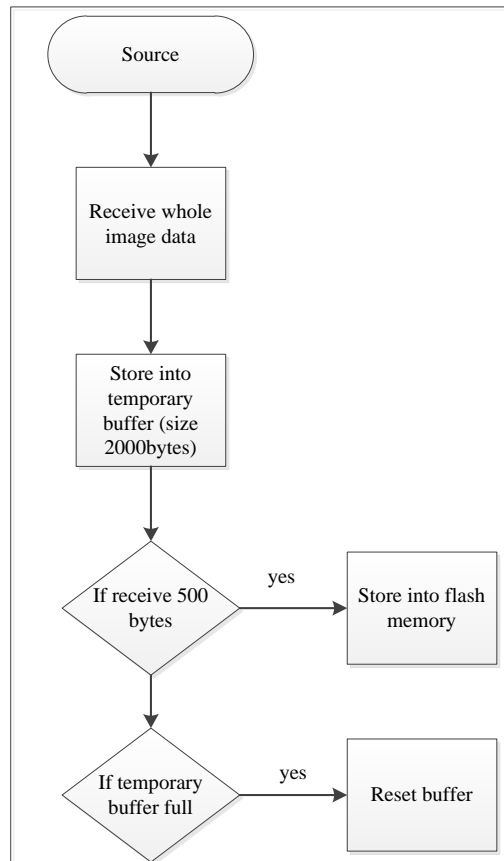


Figure 4.2.5.5: Method of storing whole image data

Figure 4.2.5.5 shows the method of storing whole image into flash. The continuous image data is stored into a temporary buffer (maximum 2000 bytes), once it received 500 bytes of image data, the source will store them into flash memory. If it reached 2000 bytes which is the maximum size of buffer, the source will based on the concept of circular buffer to reset the pointer to point to beginning of the buffer again.

4.2.6 Flow Charts of Intermediate Node

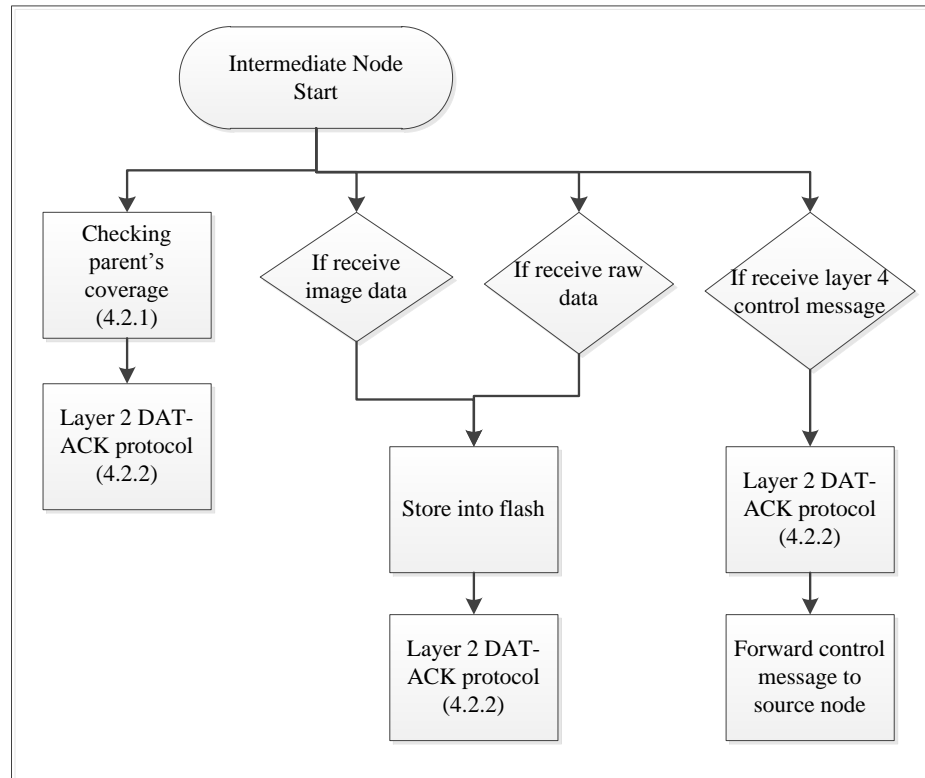


Figure 4.2.6.1: Flow chart of intermediate node

Figure 4.2.6.1 shows the flow of intermediate node. The main task of intermediate node is to listen on the channel, waiting for image or raw data come in, and store them into flash memory. It also needs to check its parent's coverage which is base station before sending out data. The intermediate node will also receive some control message such as layer 4 ACK/NACK control message, it then reply layer ACK to the end and forward the control message to the source.

4.3 Java Application Design

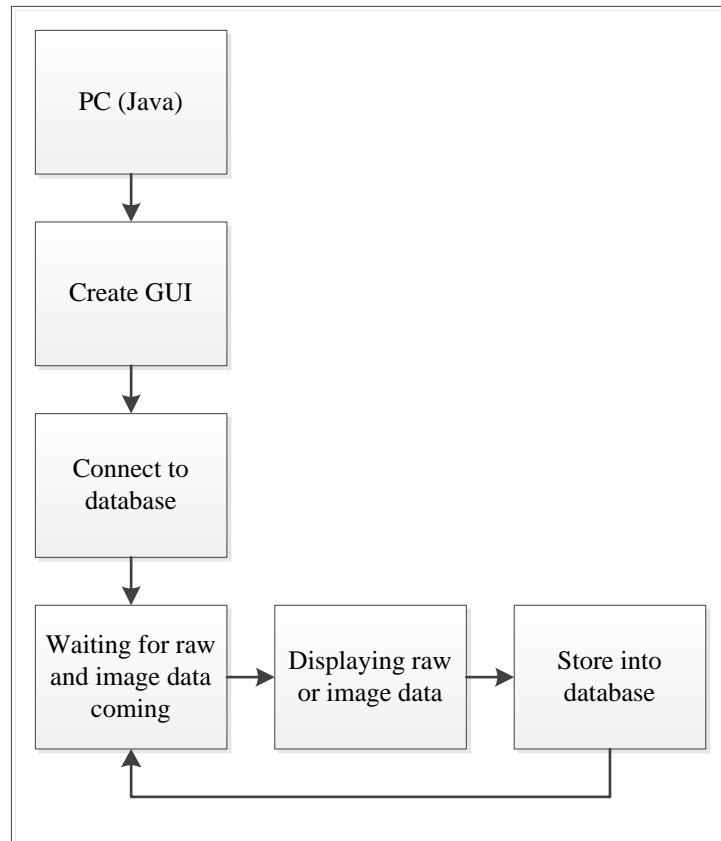


Figure 4.3.1: Flow chart of java application

Figure 4.3.1 shows the flow diagram of java application that is developed in the user PC with database installed. At first, GUI is created whenever the application is started. The database will also be connected after creating GUI. After that, the application will keep listening on the message listener to wait for any available data (raw or image data) come in. The data will be displayed out on GUI and finally being stored into database.

4.4 Database Design

Field	Type
pc_timestamp	datetime
node_id	integer
packet_id	long
temp	double
humi	double

Figure 4.4.1: Raw_data table design

Field	Type
pc_timestamp	datetime
node_id	integer
name	varchar(25)
photo	blob

Figure 4.4.2: Imag_data table design

Figure 4.3.1 and figure 4.3.2 show that the design of the two databases table for storing the raw data readings and images. The pc_timestamp field is needed so that user knows exactly the receiving time of the raw data and image. The node_id field indicates that which node is the source that sending the data. The packet_id is the identifier for each particular packet. The temp and humi with type double is the field to store the temperature and humidity reading whereas image data will be stored as binary object type in database.

4.5 Summary/Concluding Remark

In summary, all the basic flow of sensor nodes including source, intermediate and base station are explained. The details design of reliable data collection protocols such as point-to-point and end-to-end are included in this section as well. The process of enabling SHT11 sensor and C329 camera and how the data are being transferred, displayed and stored into database are included as well.

CHAPTER 5 IMPLEMENTATION AND TESTING

5.1 Hardware Implementation

This section describes the actual implementation of the main hardware components and how they are being interfaced and communicated as well.

5.1.1 Overall Hardware Design

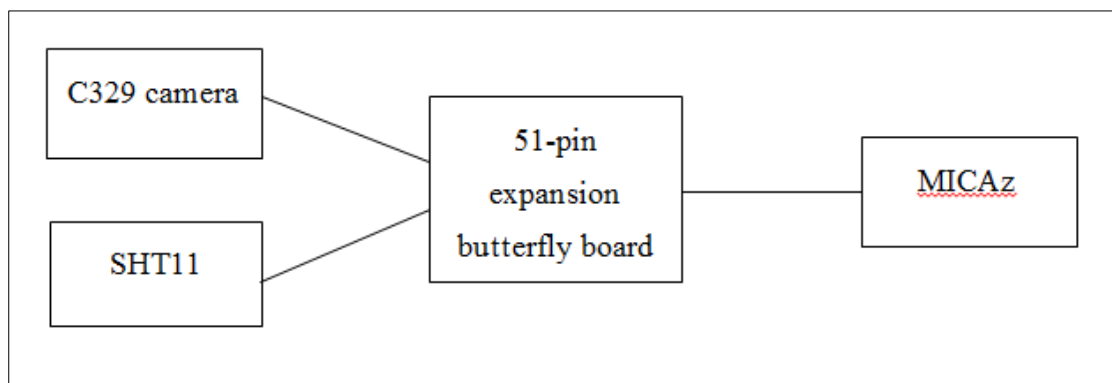


Figure 5.1.1.1: Simple MICAz Platform Block Diagram

Figure 5.1.1.1 shows the simple block diagram for MICAz platform. Both C329 embedded camera and SHT11 sensor are soldered to 51-pin expansion butterfly board. Then, the soldered butterfly board is connected with MICAz 51-pin connector. In this project, only source node is attached with the butterfly board soldered with those sensors.

5.1.2 MICAz 51-pin Expansion Connector

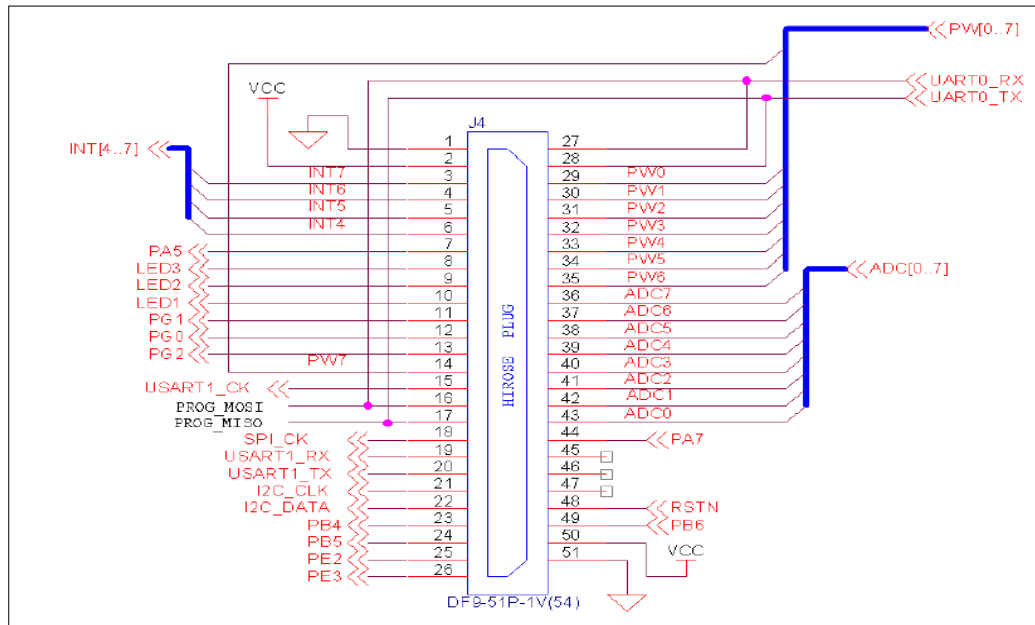


Figure 5.1.2.1: MICAz 51-pin expansion connector (Crossbow Technology, n.d.)

In figure 5.1.2.1, we can see that MICAz has 51 pins that can be expanded out for different purpose. However, there are only few pins needed to be expanded out in this project. Table 5.1.2.1 below shows that the list of needed pins and their usages.

Pin	Usage
VCC	Supply voltage to sensors
GND	Connection to the ground
PW3	Linkage of SCK clock pin of SHT11
PW4	Linkage of DATA pin of SHT11
UART0_RX	Linkage of UART receive pin of C329 camera
UART0_TX	Linkage of UART transmit pin of C329 camera

Table 5.1.2.1: List of needed pins and their usages

5.1.3 Circuit Design of Butterfly Board

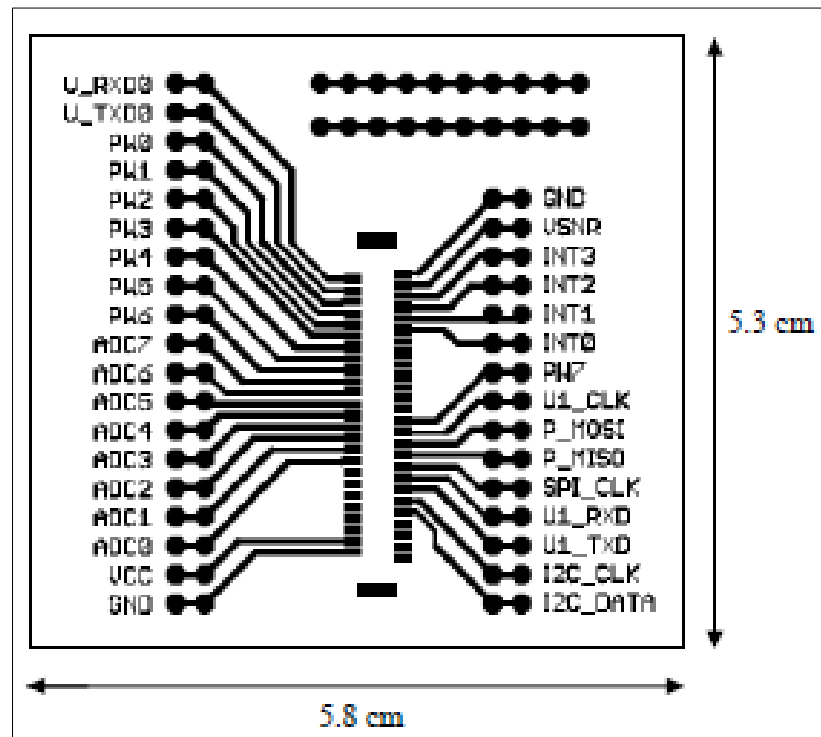


Figure 5.1.3.1: The layout design of butterfly board

Figure 5.1.3.1 shows the layout design of butterfly board. This board can be expanded by many sensors and attach to other external embedded device. In this project, it was used to attach onto the MICAz 51-pin expansion connector so that sensors can communicate with MICAz. The actual implementation of butterfly board can be seen in figure 5.1.3.2 as below.

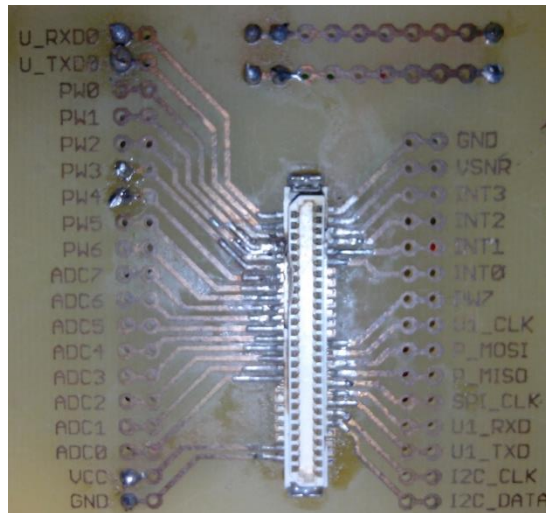


Figure 5.1.3.2: The actual implementation of butterfly board

From figure 5.1.3.2, we can see that there are only 6 pins being expanded out in this project. However, the rest of the pins can be expanded out as well when adding more sensors for future.

There must be additional 2 wires to be soldered from the bottom of VCC and GND to the top two rows of the holes so that MICAz can supply voltage to the sensors. In this project, solar panels and power regulator were the source for supplying voltage up to 3.3V to the sensors and MICAz.

5.1.4 SHT11 Temperature and Humidity Sensors

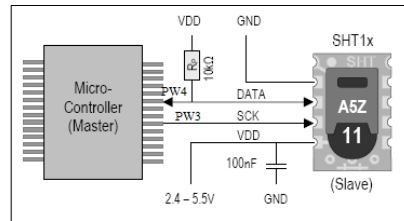


Figure 5.1.4.1: Circuit design of SHT11

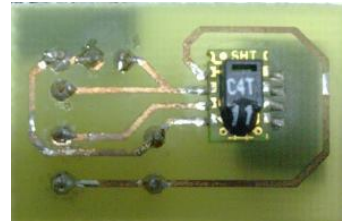


Figure 5.1.4.2: Actual implementation of SHT11

Figure 5.1.4.1 shows how the SHT11 sensor can be linked with external micro-controller. There are only 4 pins needed to trigger the SHT11 which are VDD, GND, DATA, and SCK. Figure 5.1.4.3 shows the details design layout diagram for SHT11 and breakout board. The GND and VDD pins from SHT11 were linked to MICAz GND and VDD pins. The DATA pin was linked to the PW4 pin while the SCK pin was linked to the PW3 pin of MICAz. The actual implementation of soldered SHT11 with breakout can be views in figure 5.1.4.2 as above.

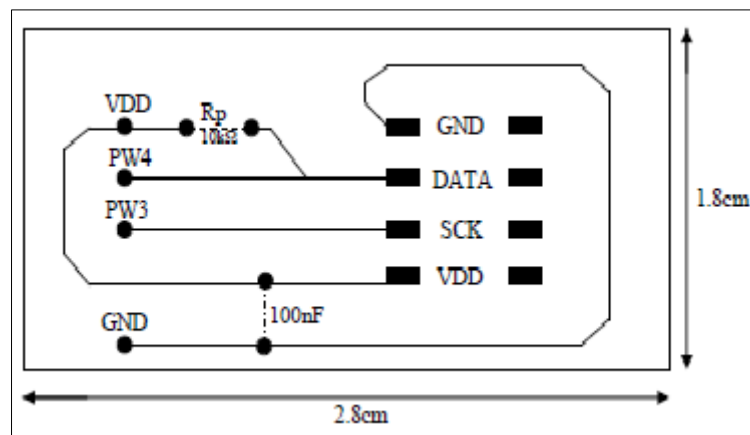


Figure 5.1.4.3: Layout diagram of SHT11 with breakout board

5.1.5 C329 Camera

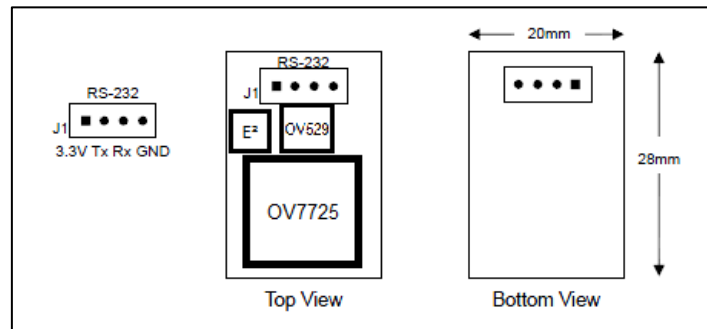


Figure 5.1.5.1: Layout diagram for C329 board and UART interface

C329 camera use UART serial to communicate with external embedded device. Therefore, the four pins (3.3V, Tx, Rx, and GND) needed to link to MICAz via butterfly board. The Tx transmit pin of C329 camera should be soldered to U_RXD0 receive of the butterfly board and vice versa. This is because when C329 sending command via Tx pin, the MICAz should use U_RXD0 pin to receive the command and process it. Figure 5.1.5.2 shows the actual implementation of C329 camera soldered to butterfly board. The yellow color wire linking between Tx transmit pin of C329 to U_RXD0 hole of butterfly board while the green color wire linking between Rx pin of C329 to U_TXD0 hole of butterfly board.

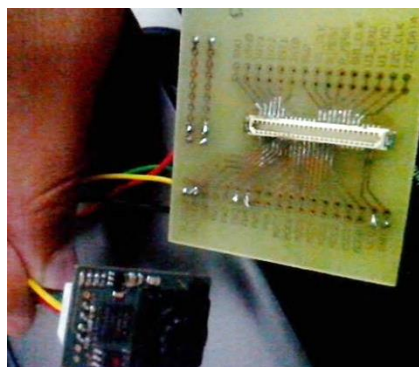


Figure 5.1.5.2: Actual implementation of C329 camera.

5.2 Software Implementation

5.2.1 Software Setup

This section includes all the necessary steps to setup the software needed in this project. It includes TinyOS and MySQL installation. Before proceed to installation, user needs to have a PC or laptop with Windows XP 32-bit version OS in order to successfully install the software.

5.2.1.1 TinyOS Installation

The first main software in this project is TinyOS. The following are the required steps to install TinyOS and the author has already included the installation software in CD as well.

Step 1:

Create a folder in C:\ directory and named it as “tinyos”. After that, put the “tos2_cygwin_install” to C:\tinyos\

Step 2:

Go to” tos2_cygwin_install\tinyos202\” folder, double click “jdk-1_5_0_12-windows-1586-p.exe” to install Java 1.5 JDK. During the installation, change the jdk installation directory to C:\tinyos\jdk1.5.0_12 while the jre installation directory to C:\tinyos\jre1.5.0_12

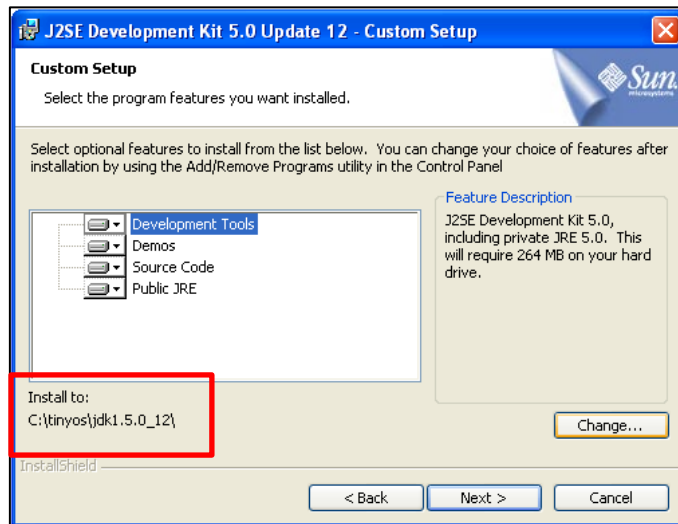


Figure 5.2.1.1.1: Change of JDK installation directory

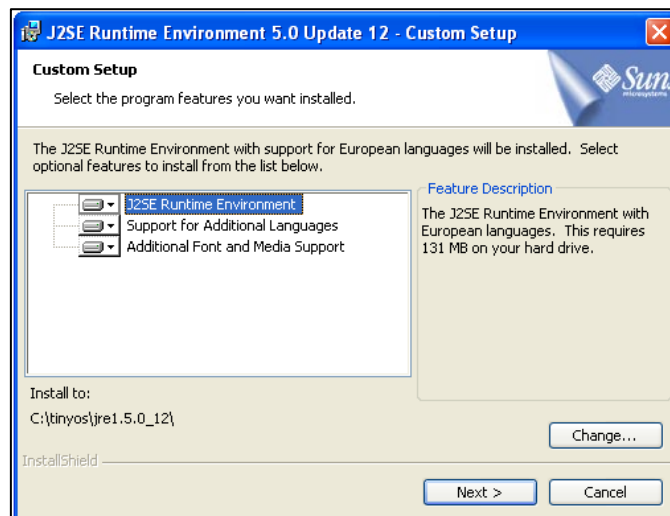


Figure 5.2.1.1.2: Change of JRE installation directory

Step 3:

Go to “tos2_cygwin_install\tinyos202\cygwin_tinyos_install” and then double click cygwin_setup.exe to install Cygwin Unix-like environment for Windows.

When the installation requires for choosing a download source, choose “Install from Local Directory”. After that, insert the installation directory to “C:\tinyos\cygwin”.

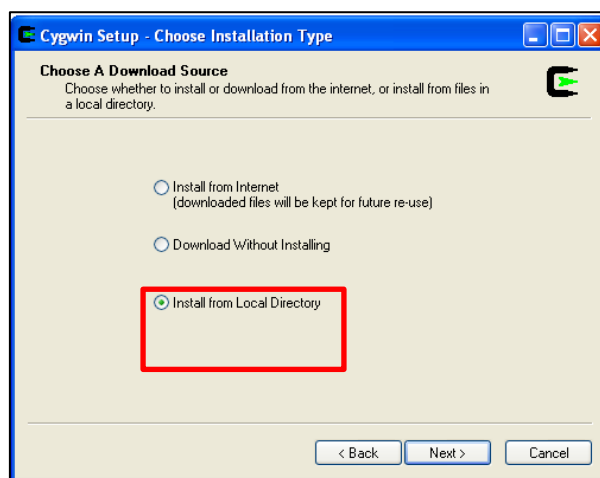


Figure 5.2.1.1.3: Cygwin download source selection

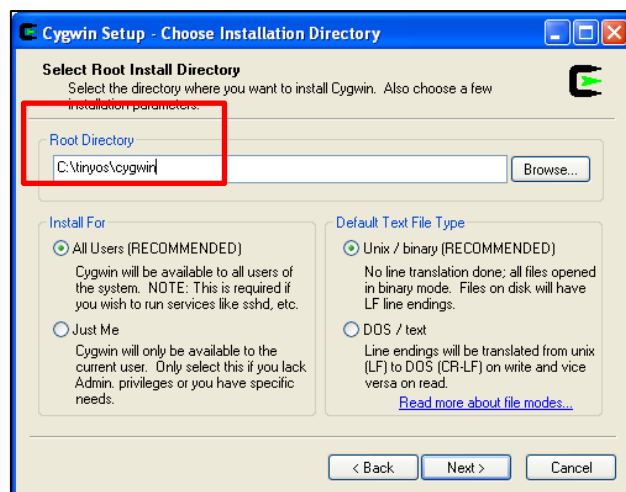


Figure 5.2.1.1.4: Installation directory of Cygwin

Step 4

Go to “tos2_cygwin_install\tinyos202\cygwin_tinyos_update” and double click cygwin_update.exe to update Cygwin version.

During the installation, user needs to choose “Install from Local Directory” and its installation directory to “C:\tinyos\cygwin” same as previous step as well.

After that, set the local package directory to “C:\tinyos\tos2_cygwin_install\tinyos202\cygwin_tinyos_update”

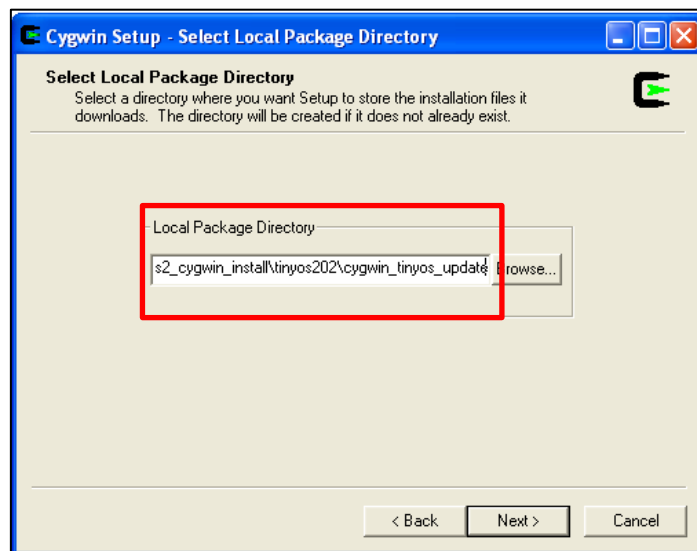


Figure 5.2.1.1.5: Change of Local Package Directory

During the Cygwin setup process, several windows will be prompt out. Just click OK to proceed. After few minutes, a pop out appeared to indicate installation complete.

Step 5

Go to Control Panel -> System -> Advanced -> Environment Variables -> Path -> Edit

Insert this line “C:\tinyos\jdk1.5.0_12\bin;” to the front line of variable value.

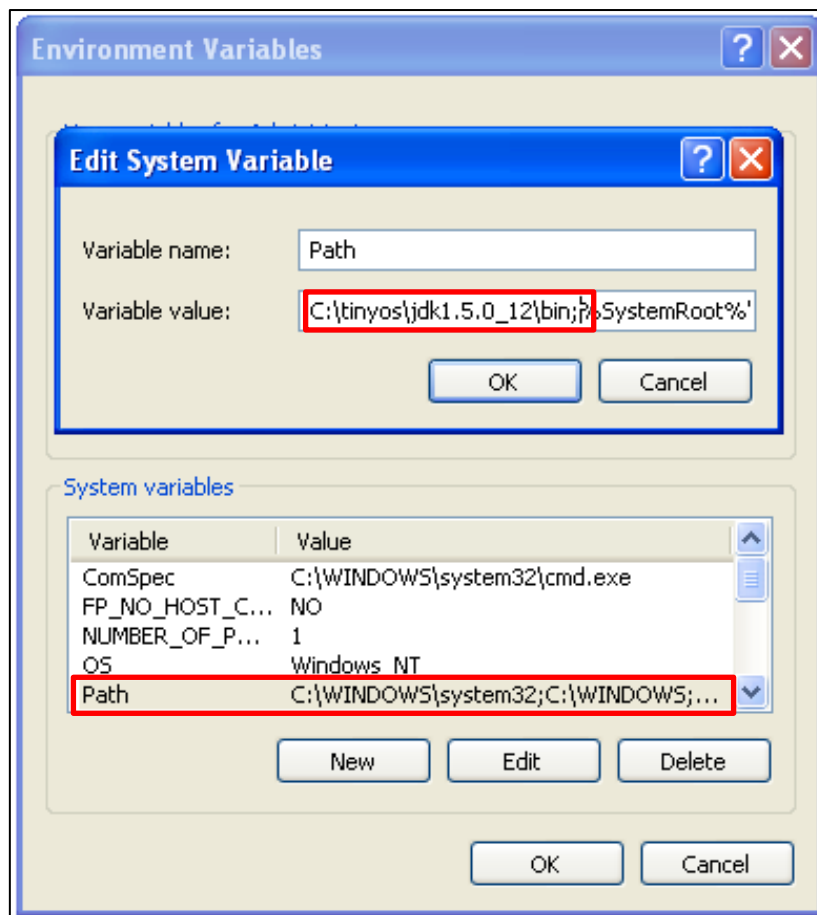
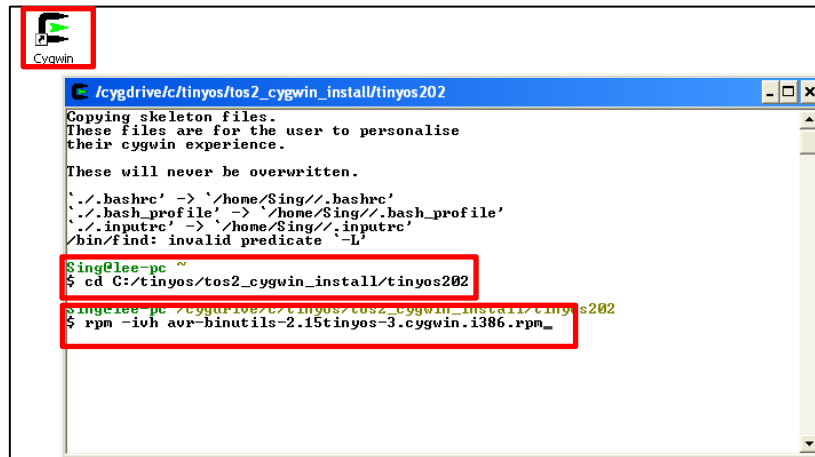


Figure 5.2.1.1.6: Edit of environment variables

Step 6

Go to desktop, and run the shortcut icon Cygwin, a unix-like command prompt will prompt out. Change the directory to “C:/tinyos/tos2_cygwin_install/tinyos202”



```

Cygwin
/cygdrive/c/tinyos/tos2_cygwin_install/tinyos202
Copying skeleton files.
These files are for the user to personalise
their cygwin experience.

These will never be overwritten.
'./bashrc' -> '/home/Sing//.bashrc'
'./bash_profile' -> '/home/Sing//.bash_profile'
'./inputrc' -> '/home/Sing//.inputrc'
/bin/find: invalid predicate '-l'

Sing@lee-pc ~
$ cd C:/tinyos/tos2_cygwin_install/tinyos202
Sing@lee-pc /cygdrive/c/tinyos/tos2_cygwin_install/tinyos202
$ rpm -ivh avr-binutils-2.15tinyos-3.cygwin.i386.rpm

```

Figure 5.2.1.1.7: Cygwin command prompt

After that, key in all the 8 commands to install the tools. The commands are:

1. rpm -ivh avr-binutils-2.15tinyos-3.cygwin.i386.rpm
2. rpm -ivh avr-gcc-3.4.3-1.cygwin.i386.rpm
3. rpm -ivh avr-libc-1.2.3-1.cygwin.i386.rpm
4. rpm -ivh avarice-2.4-1.cygwin.i386.rpm
5. rpm -ivh avr-insight-6.3-1.cygwin.i386.rpm
6. rpm -Uvh nesc-1.2.8b-1.cygwin.i386.rpm
7. rpm -ivh --ignoreos tinyos-tools-1.2.4-2.cygwin.i386.rpm
8. rpm -ivh --ignoreos tinyos-2.0.2-2.cygwin.noarch.rpm

Step 7:

Go to “C:\tinyos\cygwin\etc” and using WordPad to open a file called “profile”

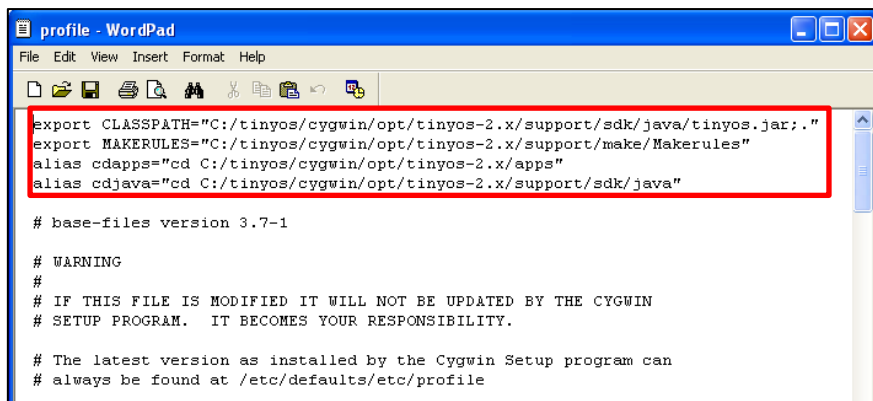
Add the following lines on top of the file and save over.

```
export CLASSPATH="C:/tinyos/cygwin/opt/tinyos-2.x/support/sdk/java/tinyos.jar;."
```

```
export MAKERULES="C:/tinyos/cygwin/opt/tinyos-2.x/support/make/Makerules"
```

```
alias cdapps="cd C:/tinyos/cygwin/opt/tinyos-2.x/apps"
```

```
alias cdjava="cd C:/tinyos/cygwin/opt/tinyos-2.x/support/sdk/java"
```



```
profile - WordPad
File Edit View Insert Format Help
export CLASSPATH="C:/tinyos/cygwin/opt/tinyos-2.x/support/sdk/java/tinyos.jar;."
export MAKERULES="C:/tinyos/cygwin/opt/tinyos-2.x/support/make/Makerules"
alias cdapps="cd C:/tinyos/cygwin/opt/tinyos-2.x/apps"
alias cdjava="cd C:/tinyos/cygwin/opt/tinyos-2.x/support/sdk/java"

# base-files version 3.7-1

# WARNING
#
# IF THIS FILE IS MODIFIED IT WILL NOT BE UPDATED BY THE CYGWIN
# SETUP PROGRAM. IT BECOMES YOUR RESPONSIBILITY.

# The latest version as installed by the Cygwin Setup program can
# always be found at /etc/defaults/etc/profile
```

Figure 5.2.1.1.8: Adding lines to “profile” file

CHAPTER 5

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

Step 8:

Go back to “tos2_cygwin_install\tinyos202” folder, double click graphviz-1.10.exe to install Graphviz. Change the destination folder to “C:\tinyos\ATT” and click next to finish installation.

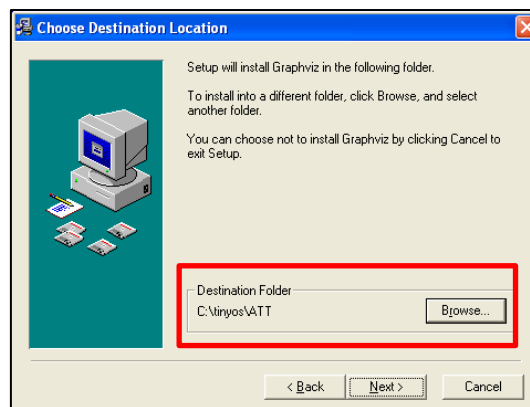


Figure 5.2.1.1.9: Change destination directory for Graphviz

Step 9:

Go to “tos2_cygwin_install\tinyos202”, install Crimson editor by just double click CrimsonSetup.exe.

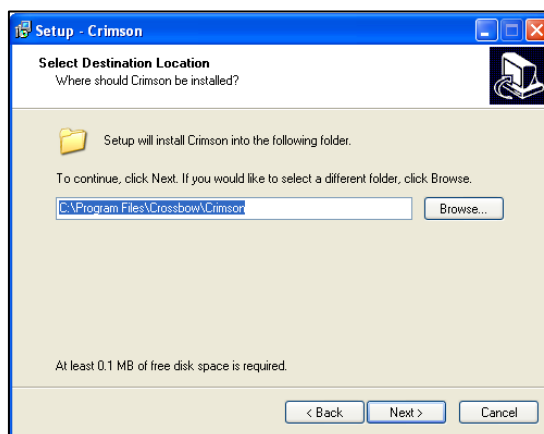


Figure 5.2.1.1.10: Installation of Crimson editor

5.2.1.2 Driver Installation of MIB510 Programming and Interface Board

After TinyOS 2.0.2 is successfully installed into Windows XP, the next thing is to install the driver for MIB510 programming board. This board is needed because it is used to program the mote and can be used as a base station when interface with a MICAz mote. The steps to install driver are listed as below:



Figure 5.2.1.2.1: New hardware found pop out

When first plug the programming board to PC through a USB cable, a pop up will prompt user to install the driver. Click “No, not this time” and proceed to next step.

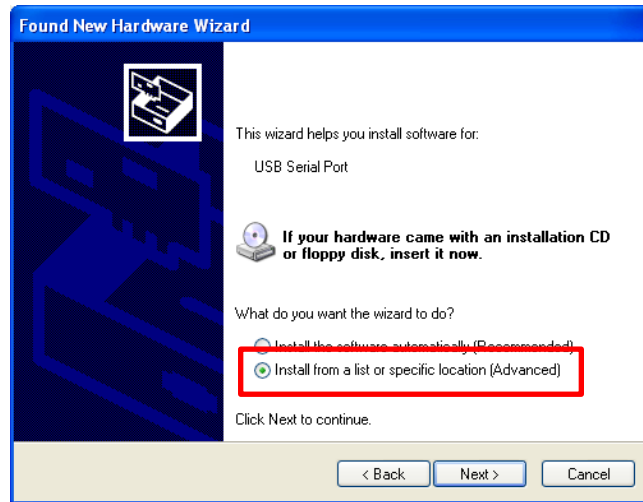


Figure 5.2.1.2.2: Selection of installation location

Choose “install from a list or specific location (Advanced)” and proceed to next step.

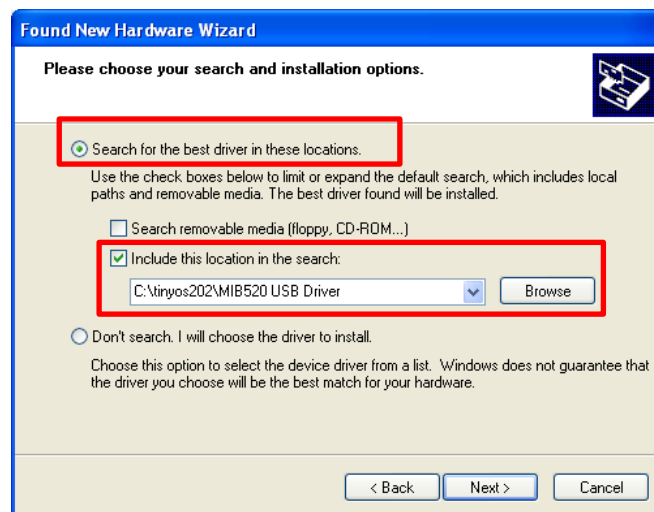


Figure 5.2.1.2.3: Selection of installation options

Click the “search for the best driver in these locations” and checked “Include this location in the search” to select where the driver is located.

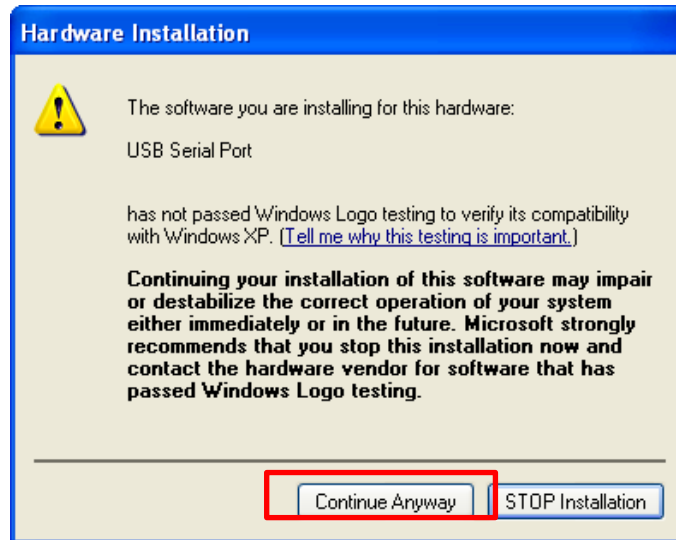


Figure 5.2.1.2.4: Warning pop out during driver installation

When the windows pop out this warning message, click “Continue Anyway” and you are required to repeat install the driver (up to 3 times). The installation steps are same as figures above. Figure 5.2.1.2.5 shows that the pop out of successful installed driver.

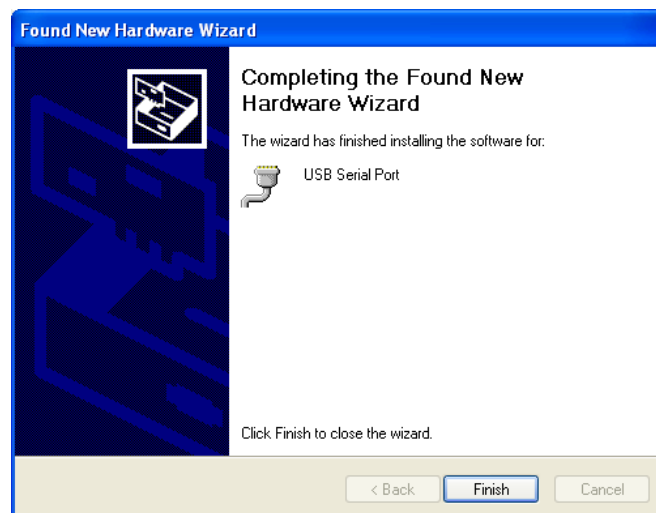


Figure 5.2.1.2.5: Pop out of successful driver installation

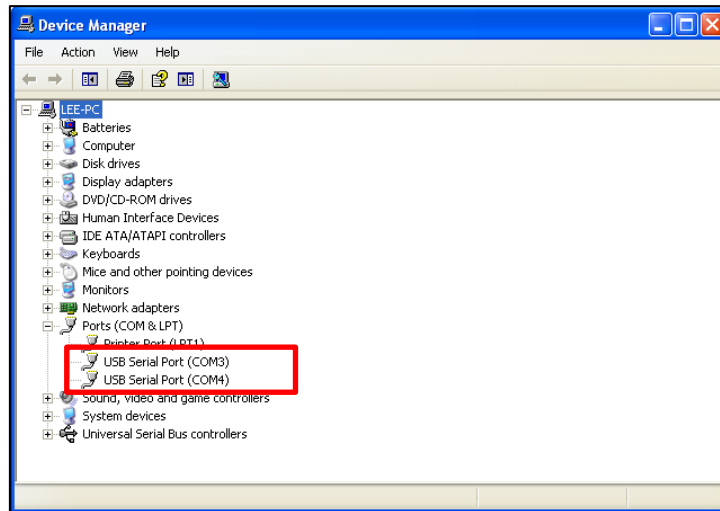
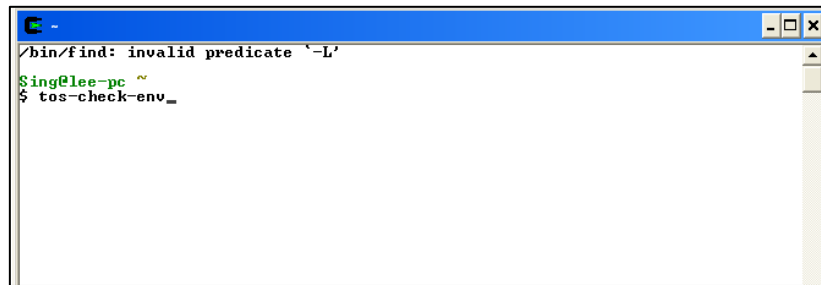


Figure 5.2.1.2.6: COMM port and PROGRAMMING port

To verify whether the COM port has been detected, go to Control Panel -> System -> Advanced -> Hardware -> Device Manager and two ports will be displayed under the “Ports” sections. The first port which is “COM3” port is the programming port while the second port which is “COM4” is the data port used for forwarding data to PC.

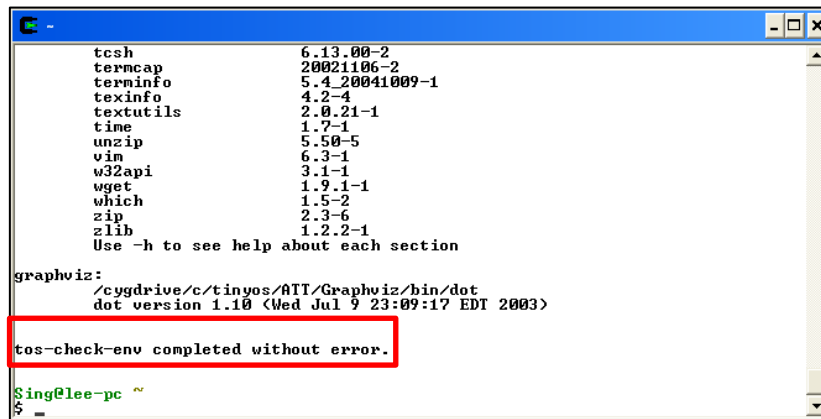
5.2.1.3 Verification of TinyOS Installation

To verify whether TinyOS has been installed successfully, open the Cygwin bash shell, and type “tos-check-env” command.



```
~/bin/find: invalid predicate '-L'  
$ing@lee-pc ~  
$ tos-check-env_
```

Figure 5.2.1.3.1: Checking installation status for TinyOS



```
tcsh                6.13.00-2  
termcap             20021106-2  
terminfo            5.4.20041009-1  
texinfo             4.2-4  
textutils           2.0.21-1  
time                1.7-1  
unzip               5.50-5  
vin                 6.3-1  
w32api              3.1-1  
wget                1.9.1-1  
which               1.5-2  
zip                 2.3-6  
zlib                1.2.2-1  
Use -h to see help about each section  
  
graphviz:  
/cygdrive/c/tinyos/ATT/Graphviz/bin/dot  
dot version 1.10 (Wed Jul 9 23:09:17 EDT 2003)  
  
tos-check-env completed without error.  
  
$ing@lee-pc ~  
$
```

Figure 5.2.1.3.2: Status of TinyOS

Figure 5.2.1.3.2 shows that no error was made during the installation. Therefore, the TinyOS was successfully installed and is ready to use.

5.2.1.5 MySQL Installation

In this section, the author listed down the steps on how to install and configure MySQL version 5.1 database on Window XP platform. Before the installation guide starts, user needs to download MySQL Windows 32-bit installer. The installer link is “<http://dev.mysql.com/downloads/mysql/5.1.html#windows32>”. After finished download, unzip the file, double click the installation file and then follow the following steps.

Step 1:

Click the “Next” button



Figure 5.2.1.5.1: First step of MySQL installation

Step 2:

Tick the “Typical” radio button and click “Next”. (Only common MySQL features are required in this project). After that, click “finish” button as shown in figure 5.2.1.5.3.



Figure 5.2.1.5.2: Second step of MySQL installation



Figure 5.2.1.5.3: Pop out of MySQL setup finished

Step 3:

The configuration wizard will prompt out. Tick “Detailed Configuration” and move on to “Next”. After that, tick the “Developer Machine” as shown in 5.2.1.5.5.

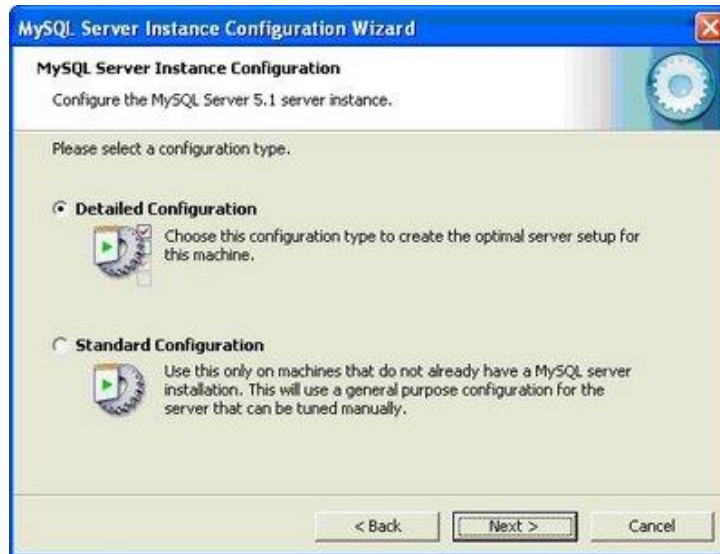


Figure 5.2.1.5.4: MySQL Configuration selection type



Figure 5.2.1.5.5: MySQL server selection type

Step 4:

Among three of the database usages, choose “Multifunctional Database”.



Figure 5.2.1.5.6: MySQL database usage selection

Next, select the file path and its space where the files in database will be stored.



Figure 5.2.1.5.7: MySQL database file path selection

Step 5:

Choose the “Decision Support (DSS)/OLAP” as we do not need many connections to server. After that click “Next”. A networking options will be prompted, leave the default port number “3306” and click “Next”



Figure 5.2.1.5.8: MySQL database file path selection

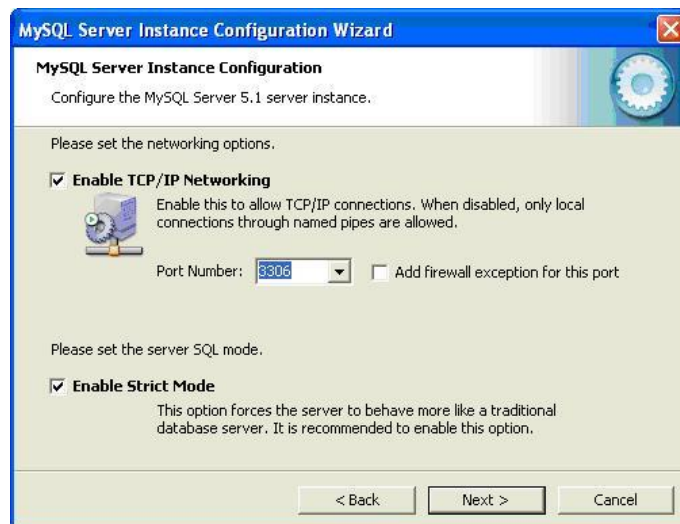


Figure 5.2.1.5.9: MySQL database networking options

Step 6:

Choose “Standard Character Set” as default charset.



Figure 5.2.1.5.10: MySQL database default charset selection

Tick “Install As Windows Service”.



Figure 5.2.1.5.11: MySQL database Windows options.

Step 7:

Then, key in root password. “The author has chosen “1234” as root password.



Figure 5.2.1.5.11: MySQL database security options.

After that, you will see the pop out as shown in figure 5.2.1.5.12 showing that configuration is successful. Click “Finish” and the MySQL is ready to use.



Figure 5.2.1.5.12: MySQL database configuration done

5.2.1.6 Verification of MySQL Installation

To verify the accessibility of MySQL database, fire up the Windows command prompt and type “mysql -u root -p”, and you will be prompted to enter the password, insert “1234” as root password and hit enter. After that, as shown in figure 5.2.1.6.2, type “show databases;” MySQL command to test its accessibility.

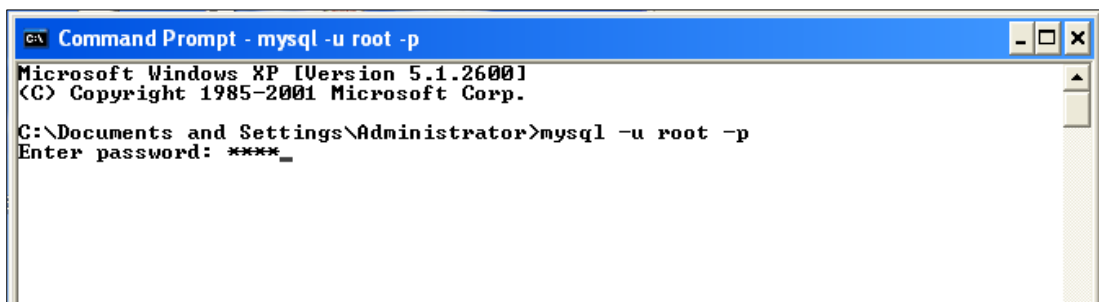


Figure 5.2.1.6.1: Verify MySQL via Windows command prompt

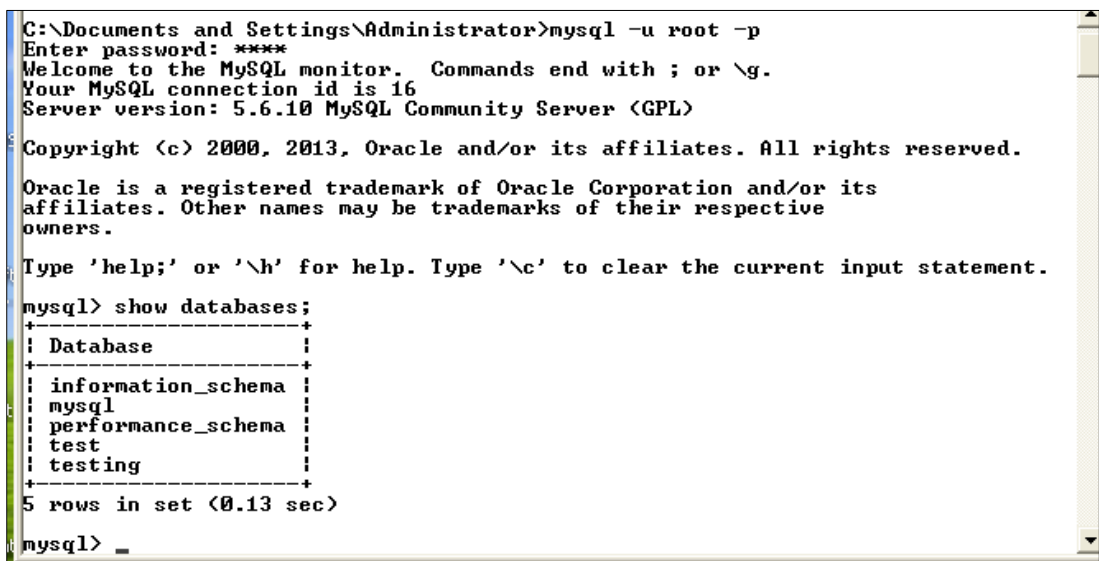


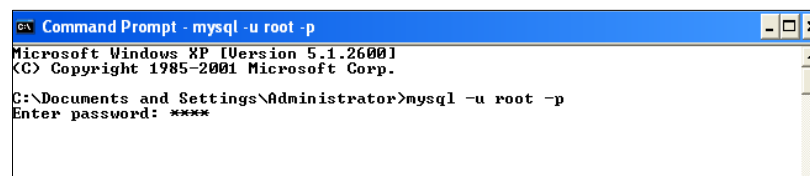
Figure 5.2.1.6.2: MySQL command testing in command prompt

5.2.1.7 Create Tables in MySQL

This session will guide user on creating MySQL tables that will be used in this project. The two tables are “raw_data” and “image_data” and all the records have been defined in chapter 4.4.

Step 1:

Run the command prompt, and type `mysql -u root -p` followed by key in password “1234”.

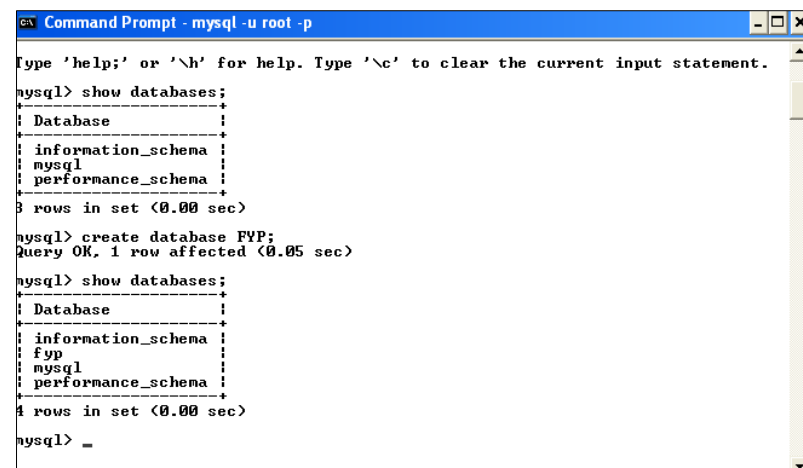


```
Command Prompt - mysql -u root -p
Microsoft Windows XP [Version 5.1.26001
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>mysql -u root -p
Enter password: ****
```

Figure 5.2.1.7.1: Log in to MySQL

Step 2:

Create database named as “FYP”. The sql command is “create database FYP”. You will see a database being created by typing the command “show databases;”



```
Command Prompt - mysql -u root -p
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
+-----+
3 rows in set (0.00 sec)

mysql> create database FYP;
Query OK, 1 row affected (0.05 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| fyp |
| mysql |
| performance_schema |
+-----+
4 rows in set (0.00 sec)

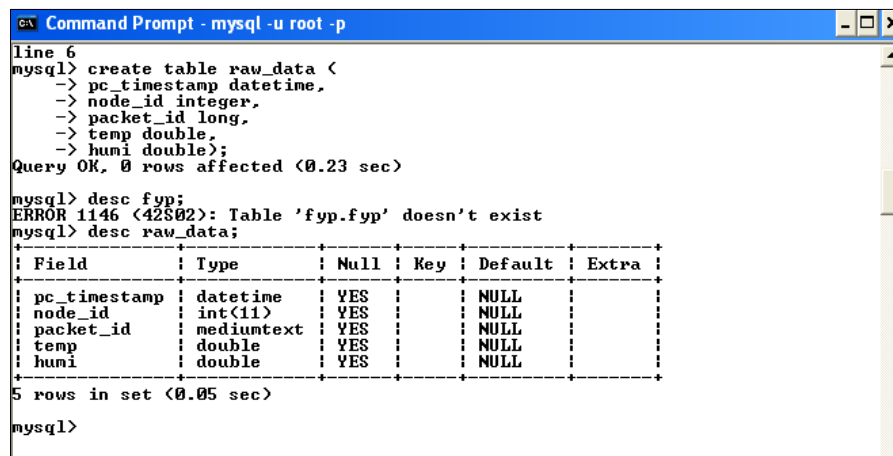
mysql> _
```

Figure 5.2.1.7.2: Create database in MySQL

Step 3:

This step is to create the “raw_data” table and insert its records into database. This table will be used for storing raw data records later on. The MySQL command is listed as follow:

```
Create table raw_data (
    pc_timestamp datetime,
    node_id integer,
    packet_id long,
    temp double,
    humi double);
```



```
Command Prompt - mysql -u root -p
line 6
mysql> create table raw_data (
  -> pc_timestamp datetime,
  -> node_id integer,
  -> packet_id long,
  -> temp double,
  -> humi double);
Query OK, 0 rows affected (0.23 sec)

mysql> desc fyp;
ERROR 1146 (42S02): Table 'fyp.fyp' doesn't exist
mysql> desc raw_data;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| pc_timestamp   | datetime     | YES  |     | NULL    |       |
| node_id        | int(11)      | YES  |     | NULL    |       |
| packet_id      | mediumtext   | YES  |     | NULL    |       |
| temp           | double       | YES  |     | NULL    |       |
| humi           | double       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.05 sec)

mysql>
```

Figure 5.2.1.7.3: Create raw_data table in MySQL

After successfully created table, type “desc raw_data;” and you will see the description of raw_data table.

Step 4:

The last step is to create image_data table and its records to database. This table is mainly used for storing imaging purpose. The commands are listed as below:

```
Create table image_data (
    pc_timestamp datetime,
    node_id integer,
    name varchar(25),
    photo BLOB);
```

```
mysql> create table image_data (
  -> pc_timestamp datetime,
  -> node_id integer,
  -> name varchar(25),
  -> photo BLOB);
Query OK, 0 rows affected (0.11 sec)

mysql> desc image_data;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| pc_timestamp | datetime | YES | | NULL | |
| node_id | int(11) | YES | | NULL | |
| name | varchar(25) | YES | | NULL | |
| photo | blob | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
1 rows in set (0.00 sec)

mysql>
```

Figure 5.2.1.7.3: Create image_data table in MySQL

Again, type “desc image_data;” and the metadata of this table will be shown.

5.2.2 MICAz Coding

This section includes the implementation of MICAz coding/programming based on the design that explained in chapter 4. There are 4 folders needed to run in this project which is:

- “Node130” folder which keep all the files for compiling source node. (Source node is given “130” as its unique node ID)
- “Node150” folder, which keep all the files for compiling intermediate node. (Intermediate node is given “150” as its unique node ID)
- “BaseStation” folder, which keep all the files for compiling base station. (Base station is given “1” as its unique node ID)
- “java_app” folder, which keep all the files for compiling and running front end application.

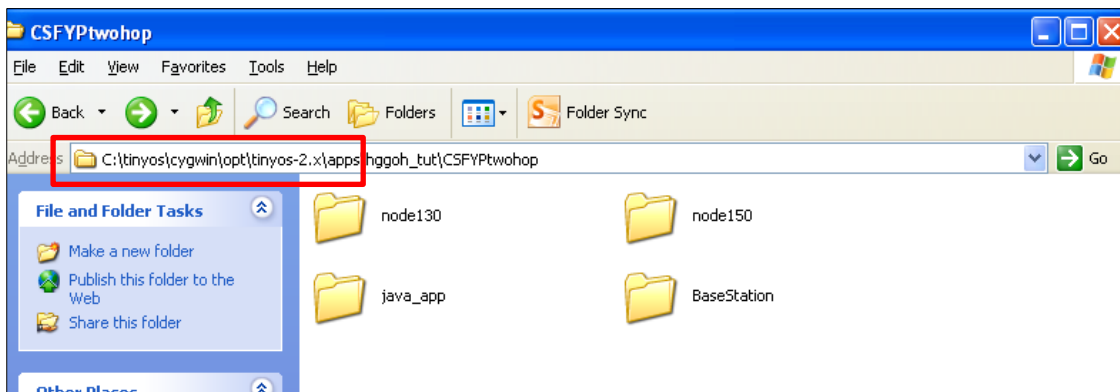


Figure 5.2.2.2: Listed folders required in this project

All the file are required to be put under “C:\tinyos\cygwin\opt\tinyos-2.x\apps” so as to ensure the library file can be retrieved and the code can be run and compiled properly.

5.2.2.1 Node130

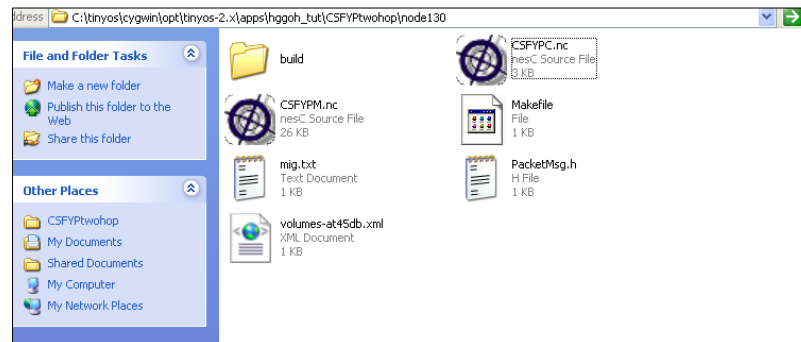


Figure 5.2.2.1.1: Files contain in Node130 folder

Figure 5.2.2.1.1 shows all the necessary files contain in Node130 folder. Again, you can find all the files inside attached CD. The CSFYPM.nc is the module file while the CSFYPC.nc is the configuration file for Node130. Figure 5.2.2.1.2 below show a fragment part of CSYPC.nc configuration code. This configuration included the components being declared and the wiring of different components together.

```

1 #include "StorageVolumes.h"
2
3
4 configuration CSFYPC {
5 }
6
7 implementation {
8   components CSFYPM, MainC, LedsC, MicaBusC;
9   components ActiveMessageC;
10  components new TimerMilliC() as Timer0;
11  components new TimerMilliC() as Timer1;
12  components new TimerMilliC() as Timer2;
13  components new TimerMilliC() as Timer3;
14  components new TimerMilliC() as Timer4;
15  components new TimerMilliC() as Timer5;
16  components new TimerMilliC() as Timer6;
17  components new TimerMilliC() as Timer7;
18  components new TimerMilliC() as Timer8;
19  components new BlockStorageC(VOLUME_BLOCKTEST);
20  components new AMSenderC(AM_PACKETMSG) as SendDAT;
21  // components new AMReceiverC(AM_PACKETMSG) as ReceiveDAT;
22  components new AMSenderC(AM_PACKETRMSG) as SendACK;
23  components new AMReceiverC(AM_PACKETRMSG) as ReceiveACK;
24  components new AMSenderC(AM_PACKETMSG) as CheckCon;
25  components new AMReceiverC(AM_PACKETMSG) as ReceiveCon;
26  components new AMSenderC(AM_PACKETMSG) as SendIMG;
27  components new AMSenderC(AM_PACKETQMSG) as SendSACK;
28  components new AMReceiverC(AM_PACKETQMSG) as ReceiveSACK;
29  components new AMSenderC(AM_PACKETDMSG) as SendTYPE;
30  components new AMReceiverC(AM_PACKETDMSG) as ReceiveTYPE;
31  components Hp1Atm128UartC;
32  components CC2420PacketC;

```

Figure 5.2.2.1.2: Fragment part of CSFYPC.nc in Node130

```

1 #include "PacketMsg.h"
2
3 #define SIZE1 5
4 module CSFYPM {
5     uses {
6         interface Boot;
7         interface Leds;
8         interface Timer<TMilli> as Timer0;
9         interface Timer<TMilli> as Timer1;
10        interface Timer<TMilli> as Timer2;
11        interface Timer<TMilli> as Timer3;
12        interface Timer<TMilli> as Timer4;
13        interface Timer<TMilli> as Timer5;
14        interface Timer<TMilli> as Timer6;
15        interface Timer<TMilli> as Timer7;
16        interface Timer<TMilli> as Timer8;
17        interface GeneralIO as Pin_PW3;
18        interface GeneralIO as Pin_PW4;
19        interface HplAtm128Uart as Uart0;
20        interface StdControl as Uart0xCtrl1;
21        interface StdControl as Uart0xCtrl1;
22        interface SplitControl as AMControl;
23        interface Packet as Packet1;
24        interface AMSend as AMSend1;
25        interface Packet as Packet2;
26        interface AMSend as AMSend2;

```

Figure 5.2.2.1.3: Fragment part of CSFYPM.nc in Node130

CSFYPM.nc is the file that defines how the source node would behave. All the logic and design flows in chapter 4 are being coded in this nesC file. Figure 5.2.2.1.3 shows the front part of CSFYPM.nc module file. It shows some of the interface being declared to be used. For example in line 8, Timer<TMilli> is being declared and named as Timer0 to be used later. Figure 5.2.2.1.4 shows the example of code which an event is triggered after Timer0 timeout.

```

750
751 //SHT11 state control
752 event void Timer0.fired() {
753     atomic { //humidity
754         if(state==STATE1) {
755             post Signalling1();
756         }
757         else if(state==STATE2) {
758             post ReadData1();
759         }
760         else if(state==STATE3) { //temperature
761             post Signalling2();
762         }
763         else if(state==STATE4) {
764             post ReadData2();
765         }
766         else if(state==STATE5) {
767             post WriteToFlash();
768         }
769     }
770 }
771 }
772

```

Figure 5.2.2.1.4: Event being triggered by Timer0 in CSFYPM.nc

```

1  enum {
2    AM_PACKETMSG = 21,
3    AM_PACKETRMSG = 22,
4    AM_PACKETMSG = 23,
5    AM_PACKETQMSG = 24,
6    AM_PACKETMSG = 25,
7    AM_PACKETMSG = 26
8  };
9
10 typedef nx_struct PacketMsg {
11   nx_uint16_t nodeid;
12   nx_uint32_t pktid;
13   nx_uint16_t humi_data;
14   nx_uint16_t temp_data;
15 } PacketMsg;
16
17 typedef nx_struct PacketIMsg {
18   nx_uint16_t nodeid;
19   nx_uint32_t pktid;
20   nx_uint8_t endbit;
21   nx_uint8_t resendbit;
22   nx_uint8_t imageData[70];
23 } PacketIMsg;

```

Figure 5.2.2.1.5: Data message packet structure in PacketMsg.h

Figure 5.2.2.1.5 shows two packet structure defined in PacketMsg.h. These two packet structures are used for data message purpose whereas in figure 5.2.2.1.6 below, these four structures used for control message purpose.

```

25 typedef nx_struct PacketRMsg {
26   nx_uint16_t nodeid;
27   nx_uint32_t pktid;
28   nx_uint8_t resend;
29   nx_uint8_t type;
30 } PacketRMsg;
31
32 typedef nx_struct PacketCMsg {
33   nx_uint16_t nodeid;
34 } PacketCMsg;
35
36 typedef nx_struct PacketQMsg {
37   nx_uint16_t nodeid;
38   nx_uint8_t endBit;
39   nx_uint8_t queryData;
40   nx_uint16_t trace_pkt;
41   nx_uint8_t ack;
42 } PacketQMsg;
43
44
45 typedef nx_struct PacketDMsg {
46   nx_uint16_t nodeid;
47   nx_uint8_t type;
48 } PacketDMsg;
49

```

Figure 5.2.2.1.6: Control message packet structure in PacketMsg.h

The details are the description of each member in the packet structures.

PacketMsg (Raw data)

- nodeid (2 bytes) : ID of the mote
- pktid (4 bytes) : Packet ID

CHAPTER 5

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

- humi_data (2 bytes): Humidity data
- temp_data (2 bytes): Temperature data

PacketIMsg (Image data)

- nodeid (2 bytes) : ID of the mote
- pktid (4 bytes) : Packet ID
- endBit (1 byte) : To indicate end of segmented image
- resendbit (1 byte): To indicate it is a resend image packet
- imageData[70] (70 bytes): Image data

PacketRMsg (for layer 2 ACK control message)

- nodeid (2 bytes) : ID of the mote
- pktid (4 bytes) : Packet ID
- resend (1 byte): To indicate it is a ACK message for missing segmented image
- type (1 byte): To indicate whether is it raw data or image data

PacketCMsg (for checking parent's coverage control message)

- nodeid (2 bytes) : ID of the mote

PacketQMsg (for sending/receiving layer 4 ACK/NACK control message)

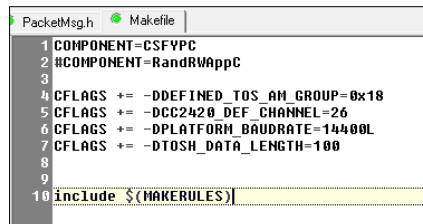
- _nodeid (2 bytes) : ID of the mote
- endBit (1 byte) : To indicate end of image transmission
- queryData (1 byte): Missing segmented image identifier (NACK)
- trace_pkt (1 byte): Tracing segmented image
- ack(1 byte): layer 4 ACK message

PacketDMsg (type of data)

- nodeid (2 bytes) : ID of the mote
- type (1 byte): type of data

Next is the “Makefile” file shown in figure 5.2.2.1.7, there are some commands need to be added in Makefile in order to run this project. First, the channel and AM group need to be same as intermediate node and base station so they can communicate with each other. In this project, “0x18” is set as AM group while “26” is set as channel. The default baud rate of MICAz has to change to 14400 so that MICAz can communicate with C329 camera which its default baud rate is 14400. Lastly, the data length also needs to change to 100 to accommodate the packet size of image data.

Last file is “volumes-at45db.xml”. This file to declare the volume of flash memory will be used in this mote. 262144 bytes which equivalent to 256KB has chosen to use in this mote

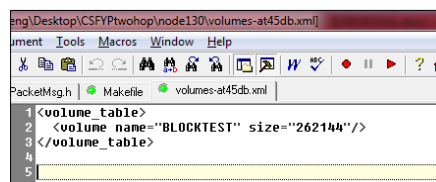


```

1 COMPONENT=CSFYPC
2 #COMPONENT=RandRMAppC
3
4 CFLAGS += -DDEFINED_TOS_AM_GROUP=0x18
5 CFLAGS += -DCC2420_DEF_CHANNEL=26
6 CFLAGS += -DPLATFORM_BAUDRATE=14400L
7 CFLAGS += -DTOSH_DATA_LENGTH=100
8
9
10 include $(MAKERULES)

```

Figure 5.2.2.1.7: Makefile in Node130



```

1 <volume_table>
2   <volume name="BLOCKTEST" size="262144"/>
3 </volume_table>
4
5

```

Figure 5.2.2.1.7: volumes-at45db.xml in Node130

5.2.2.2 Node150

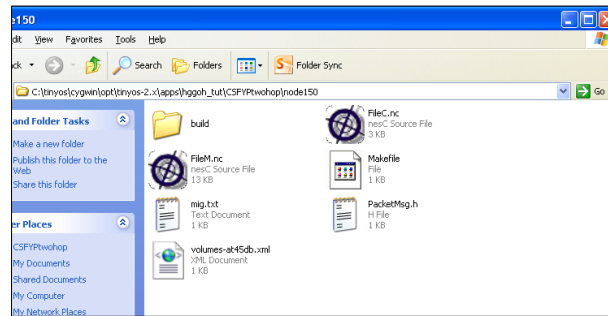


Figure 5.2.2.2.1: Files contain in Node150 folder

Node150 as mentioned earlier, contain the files to build and run the intermediate node. The files contain in this folder is same to Node130 but the code in nesC file is different from the Node130. Figure 5.2.2.2 shows a fragment code taken from FileM.nc.

```

423
424 //receive image data
425 event message_t* Receive4.receive(message_t* msg, void* payload, uint8_t len) {
426
427     PacketIMsg *ipkt = (PacketIMsg* )(call Packet4.getPayload(msg, NULL));
428
429     if (ipkt->nodeid == 130) {
430         call Leds.led00n();
431         receivedpkt = *msg;
432         atomic {
433             image_busy = TRUE;
434             nodeId=ipkt->nodeid;
435             pktId=ipkt->pktid;
436             resendbit=ipkt->resendbit;
437
438             //length=len;
439             post ImageWriteToFlash();
440         }
441     }
442     return msg;
443 }
444

```

Figure 5.2.2.2.2: Fragment code taken from FileM.nc from Node150

The fragment code above is the example code of receiving image data from source node, and post a task named as “ImageWriteToFlash ()” to write the image into flash memory. Again, the source code of intermediate node can be found in attached CD as well.

5.2.2.3 BaseStation Folder Files

In “BaseStation” folder, there are three files which are BaseStationC.nc, BaseStationP.nc and Makefile in the folder. These three files are taken from TinyOS sample code which can be found in “C:\tinyos\cygwin\opt\tinyos-2.x\apps\BaseStation”.

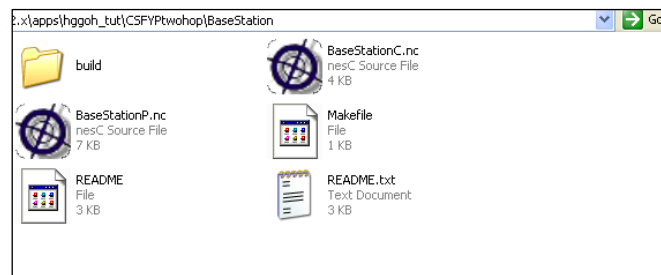


Figure 5.2.2.3.1: Files contain in BaseStation folder

5.2.2.4 java_app Folder Files

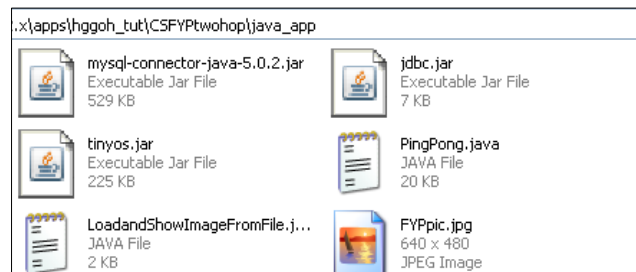


Figure 5.2.2.4.1: Files contain in java_app folder

Figure 5.2.2.4.1 shows the files in java_app folder. The .jar files are needed for connecting the java to the MySQL database. The PingPong.java file is the main java while the LoadandShowImageFromFile.java is the java file that drawing and repainting image on GUI.


```

1 import net.tinyos.message.*;
2 import net.tinyos.packet.*;
3 import net.tinyos.util.*;
4 import java.io.*;
5 import java.awt.image.*;
6 import javax.imageio.*;
7 import java.awt.*;
8 import java.awt.event.*;
9 import javax.swing.*;
10 import javax.swing.border.*;
11 import javax.swing.table.*;
12 import java.util.*;
13 import java.util.Date;
14 import java.text.ParseException;
15 import java.text.SimpleDateFormat;
16 import java.text.DecimalFormat;
17 import java.text.DateFormat;
18 import java.util.Timer;
19 import java.util.TimerTask;
20 import java.sql.*;
21
22
23 public class PingPong extends JPanel implements MessageListener, Runnable{
24
25     public Timer timer = null;
26     public Timer timer1 = null;
27     public Timer timer_retry = null;
28     FileOutputStream fms;
29     FileInputStream fis;
30     File image;
31     //public static BufferedImage image;
32     HotelIF note;

```

Figure 5.2.2.4.2: Front part of PingPong class

Figure 5.2.2.4.2 shows the front part of PingPong.java class while figure 5.2.2.4.3 below shows the small function “messageReceived ()”. This function is called when there is any packet being forwarded from base station to PC. As can be seen that the packet message will be compared with different type of packet structure object, for example, if the message is in PacketMsg object type, pongReceived function will be involved to do some data processing.

```

544
545 //packet received
546 public void messageReceived(int dest_addr, Message msg) {
547     long pktid = 0;
548     PacketIMsg imsg;
549     if (msg instanceof PacketMsg) { //raw data
550         pongReceived(dest_addr, (PacketMsg)msg);
551     }
552     else if (msg instanceof PacketCMsg) { //active ack
553         replyAck(dest_addr, (PacketCMsg)msg);
554     }
555     else if (msg instanceof PacketIMsg) { //image data
556         timer.cancel();
557         timer_retry.cancel();
558         attempt = 0;
559         imsg = (PacketIMsg)msg;
560         pktid = imsg.get_pktid();
561         if (pktid == 1) {
562             startTime = System.currentTimeMillis();
563             System.out.println("starttime is "+startTime);
564         }
565         imageReceived(dest_addr, (PacketIMsg)msg);
566     }
567     else if (msg instanceof PacketDMsg) { //layer 2 ack received
568         timer_retry.cancel();
569         attempt = 0;
570     }
571 }
572
573

```

Figure 5.2.2.4.3: messageReceived function in PingPong class

```

322 //check lost image segment
323 public void checkLostPacket() {
324
325     int bit = 0, missing_bit = 0;
326     msg = new PacketQMsg();
327
328     for (int i = 1; i <= 10 ; i++) {
329
330         if ((imagePkt[i%10].get_pktid()%10) == i%10 && imagePkt[i%10].get_nodeid() != 0) { //check for in sequence packet
331
332             if (imagePkt[i%10].get_endbit() == 1 ) {
333                 bit = 1;
334                 break;
335             }
336             else {
337                 if(i == 10) {
338                     previous_data = imagePkt[i%10].get_pktid();
339                     flag = true;
340                 }
341                 continue; //no lost found, continue next segment
342             }
343         }
344         else { //missing packet found
345
346             if (i == 1)
347                 missing_pkt = previous_data + 1;
348             else
349                 missing_pkt = imagePkt[i-1%10].get_pktid()+1;
350
351             System.out.println("Missing packet found. Missing packet ID is: "+missing_pkt);
352             flag = false;
353         }
354     }

```

Figure 5.2.2.4.4: Fragment part of checkLostPacket function in PingPong class

Figure 5.2.2.4.4 shows the fragment part of checkLostPacket function of PingPong class. Basically, this function is to check the lost segment of image. There is a for loop condition to check the lost packet. If no lost found, the loop will continue, else, it will note down the lost packet and send an NACK to source. If there is no lost in the current round (10 segment of image data). Those segmented images will be written into a .jpg format file.

```

583 //calibration of raw data
584 humi_data = -2.0468 + 0.0367*humi_data + -1.5955E-6*humi_data*humi_data;
585 temp_data = -39.6 + 0.01*temp_data;
586 System.out.print(f_2.Format(humi_data)+"% "+f_2.Format(temp_data)+"C ");
587
588 System.out.println();
589
590 textarea.append("[ "+DFormat.Format(startTime)+" ] "+
591 " - "+msg.get_nodeid()+" "+msg.get_pktid()+" "+f_2.Format(humi_data)+"% "+f_2.Format(temp_data)+"C ");
592 textarea.append("\n");
593
594 dat_seq_raw = (int)msg.get_pktid();
595
596 pingSend(msg.get_nodeid(), msg.get_pktid(), 0, 2); //reply layer 2 ACK
597
598 try {
599
600     // Execute SQL statements
601     String sqlTable = "INSERT INTO Raw_Data VALUES ('"+
602         DFormat.Format(startTime)+"', '"+
603         msg.get_nodeid()+"', '"+
604         msg.get_pktid()+"', '"+
605         f_2.Format(temp_data)+"', '"+
606         f_2.Format(humi_data)+"')";
607
608     stat.execute(sqlTable); //data insertion

```

Figure 5.2.2.4.5: Fragment code of pongReceived function in PingPong class

Figure 5.2.2.4.5 shows the fragment code of pongReceived function. At line number 684 to 685, these are the two calibration formular to calibrate the received raw data into temperature and humidity values. At line number 700 to 708 is the code that inserting those values into MySQL database.

```

1 import java.awt.Dimension;
2 import java.awt.Graphics;
3 import java.awt.event.*;
4 import java.awt.image.*;
5 import java.io.*;
6 import javax.imageio.ImageIO;
7 import javax.swing.*;
8
9 public class LoadandShowImageFromFile extends JPanel {
10
11     BufferedImage image;
12     private static JPanel imagePanel;
13     private static JFrame frame;
14
15     public LoadandShowImageFromFile() {
16
17     }
18
19
20     //Draw our image on the screen with Graphic's "drawImage()" method
21     public void paint(Graphics g) {
22
23         try{
24             image = ImageIO.read(new File("FYPpic.jpg"));
25         } catch (IOException e) {
26
27             System.out.println("Error reading dir: " + e.getMessage());
28         }
29         if(image != null)
30             g.drawImage(image, 0, 0, null);
31
32
33

```

Figure 5.2.2.4.6: Front part of LoadandShowImageFromFile class

Figure 5.2.2.4.6 show the fragment part of LoadandShowImageFromFile java class. This class is used to drawing and repainting the panel on GUI whenever image is forwarded to PC.

5.2.3 MICAz Compilation

This section includes the necessary steps to compile MICAz. Figure 5.2.3.1 below show the MICAz mote powered by 2AA batteries. Be reminded that damage will occur if the MICAz hasn't turned off after interfacing to the programming board.



Figure 5.2.3.1: MICAz with 2 AA batteries

CHAPTER 5

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

Step 1:

Interface the MICAz 51-pin connector to the programming board. After that, connect to PC/Laptop via a USB cable. Go to Device Manager to check the port number. Figure 5.2.3.2 shows that the (COM12) is the programming port while the (COM13) is the data port.



Figure 5.2.3.1: MICAz with programming board plug-in to PC

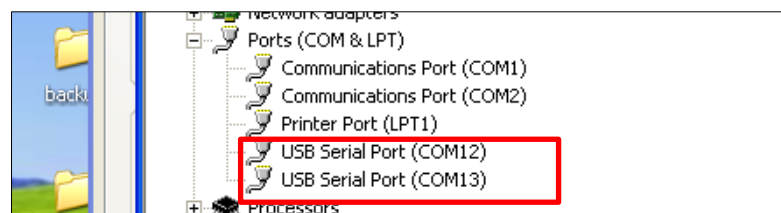
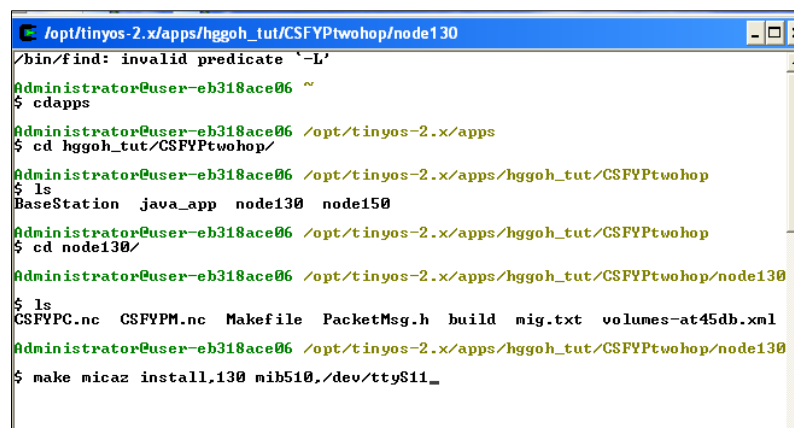


Figure 5.2.3.2: Data and programming port numbers

Step 2:

Go to desktop and double click the “Cygwin.exe”. Type “cdapps” to direct go into “/opt/tinyos-2.x/apps/” directory. After that type “cd hggoth_tut/CSFYPTwohop/” which is the directory that putting the 4 four folders needed in this project. Then type “ make micaz install, 130 mib 510,/dev/ttyS11” to start the compiling. The 130 is the unique node ID for the mote. The node ID in the command needed to be changed if compiling node150 (node ID =150) and Base Station (nodeID = 1). The ttyS11 is the programming port number where “11” is the number got from programming port number 12 minus 1. After that, hit enter button to start compiling.



```
Administrator@User-eb318ace06 ~
$ cdapps
Administrator@User-eb318ace06 /opt/tinyos-2.x/apps
$ cd hggoth_tut/CSFYPTwohop/
Administrator@User-eb318ace06 /opt/tinyos-2.x/apps/hggoth_tut/CSFYPTwohop
$ ls
BaseStation  java_app  node130  node150
Administrator@User-eb318ace06 /opt/tinyos-2.x/apps/hggoth_tut/CSFYPTwohop
$ cd node130/
Administrator@User-eb318ace06 /opt/tinyos-2.x/apps/hggoth_tut/CSFYPTwohop/node130
$ ls
CSFYPC.nc  CSFYPM.nc  Makefile  PacketMsg.h  build  mig.txt  volumes-at45db.xml
Administrator@User-eb318ace06 /opt/tinyos-2.x/apps/hggoth_tut/CSFYPTwohop/node130
$ make micaz install,130 mib510,/dev/ttyS11_
```

Figure 5.2.3.3: MICAz compilation command

```

administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPTwohop/node130
make micaz install.130 mib510./dev/ttyS11
kdir -p build/micaz
compiling CSFYPC to a micaz binary
cc -o build/micaz/main.exe -Os -finline-limit=100000 -Wall -Wshadow -Wnesc-all
target=micaz -fnesc-cfile=build/micaz/app.c -board=micasb -Ibuild/micaz -DDEFIN
D_TOS_AM_GROUP=0x18 -DCC2420_DEF_CHANNEL=26 -DPLATFORM_BAUDRATE=14400L -DTOSH_D
TA_LENGTH=100 -DIDENT_PROGRAM_NAME=\"CSFYPC\" -DIDENT_USER_ID=\"administrator\"
-DIDENT_HOSTNAME=\"user-eb318ace06\" -DIDENT_USER_HASH=0x0347c4f2L -DIDENT_UNIX
TIME=0x521977eal -DIDENT_DID_HASH=0xdad24e35L -fnesc-dumpwiring -fnesc-dump'i
nterfaces(<abstract>)' -fnesc-dump='referenced(interfacedefs, components)' -fne
c-dumpfile=build/micaz/wiring-check.xml CSFYPC.nc -lm
SFVPM.nc:541: warning: 'BlockWrite.write' called asynchronously from 'Uart0.rxD
ne'
SFVPM.nc:441: warning: 'Timer7.stop' called asynchronously from 'Uart0.rxDone'
SFVPM.nc:403: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
xDone'
SFVPM.nc:492: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
xDone'
SFVPM.nc:484: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
xDone'
SFVPM.nc:476: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
xDone'
SFVPM.nc:468: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
xDone'
SFVPM.nc: in function 'CSFYPM$WriteToFlash$runTask':
SFVPM.nc:714: warning: unused variable 'i'
compiled CSFYPC to build/micaz/main.exe
21910 bytes in ROM
2691 bytes in RAM
vr-objcopy --output-target=srec build/micaz/main.exe build/micaz/main.srec
vr-objcopy --output-target=ihex build/micaz/main.exe build/micaz/main.ihex
writing TOS image
os-set-symbols build/micaz/main.srec build/micaz/main.srec.out-130 TOS_MODE_ID=
30 ActiveMessageAddressC$addr=130
installing micaz binary using mib510
-wr fuse_h=0xd9 -dpart=A1mega128 --wr_f
se_eff --erase --upload if=build/micaz/main.srec.out-130
firmware Version: 1.1
ailed to enter programming mode.
ake: *** [program] Error 2

```

Figure 5.2.3.4: Error occur during MICAZ compilation

As can be seen in figure 5.2.3.4, error might occur during the compilation. This error can be solved by plug-out the MICAZ mote and reconnect with programming board again. After that press the reset button as shown in figure 5.2.3.5. After reset and compile again, you will see the output of successful compilation as in figure 5.2.3.6.



Figure 5.2.3.5: Reset button of programming board

```

/opt/tinyos-2.x/apps/hggoh_tut/CSFYTwohop/node130
nkdir -p build/micaz
compiling CSFVPC to a micaz binary
ncc -o build/micaz/main.exe -O3 -finline-limit=100000 -Wall -Ushadow -Wnesc-all
-target=micaz -fnesc-cfile=build/micaz/app.c -board=micasb -Ibuild/micaz -DDEFIN
ED_TOS_AM_GROUP=0x18 -DCC2420_DEF_CHANNEL=26 -DPLATFORM_BAUDRATE=14400L -DTOSH_D
ATA_LENGTH=100 -DIDENT_PROGRAM_NAME="CSFVPC" -DIDENT_USER_ID="Administrator"
-DIDENT_HOSTNAME="user-e318acc86a" -DIDENT_USER_HASH=0x0347c42L -DIDENT_UNI
_TIME=0x52197813L -DIDENT_UID_HASH=0xf60bffeL -fnesc-dump=wiring -fnesc-dump'i
nterfaces(!abstract())' -fnesc-dump='referenced(interfacedefs, components)' -fne
sc-dumpfile=build/micaz/wiring-check.xml CSFVPC.nc -lm
CSFVPM.nc:541: warning: 'BlockWrite.write' called asynchronously from 'Uart0.rxD
one'
CSFVPM.nc:441: warning: 'Timer7.stop' called asynchronously from 'Uart0.rxDone'
CSFVPM.nc:403: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
txDone'
CSFVPM.nc:492: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
rxDone'
CSFVPM.nc:484: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
rxDone'
CSFVPM.nc:476: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
rxDone'
CSFVPM.nc:468: warning: 'Timer8.startOneShot' called asynchronously from 'Uart0.
rxDone'
CSFVPM.nc: In function 'CSFVPM$writeToFlash$runTask':
CSFVPM.nc:714: warning: unused variable 'i'
compiled CSFVPC to build/micaz/main.exe
2198 bytes in ROM
2691 bytes in RAM
avr-objcopy --output-target=srec build/micaz/main.exe build/micaz/main.srec
avr-objcopy --output-target=ihex build/micaz/main.exe build/micaz/main.ihex
writing TOS image
tos-set-symbols build/micaz/main.srec build/micaz/main.srec.out-130 TOS_NODE_ID=
130 ActiveMessageAddressC$addr=130
installing micaz binary using mib510
uisp -dprog=mib510 -dserial=/dev/ttyS11 --wr_fuse_h=0xd9 -dpart=ATmega128 --wr_f
Firmware Version: 1.8
Atmel AVR ATmega128 is found.
Uploading: flash
Fuse High Byte set to 0xd9
Fuse Extended Byte set to 0xff
avr-objcopy -f build/micaz/main.exe.out-130 build/micaz/main.srec.out-130

```

Figure 5.2.3.6: Message of successful compilation

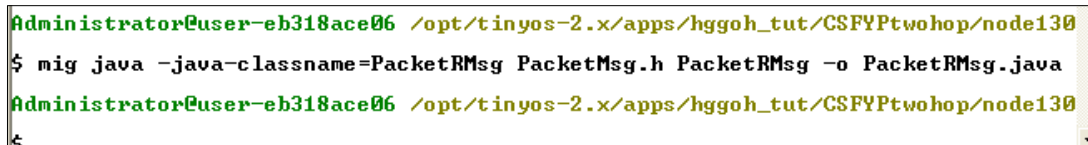
Step3:

In step 2, the author only compiled the node130 which is source node, however, in this step, the base station and node150 are also required to be compiled as well. Go to the respective folder and type the same install command which is “ make micaz install, 130 mib 510,/dev/ttyS11”. Be reminded that the “130” has to be changed to its respective node ID before installing.

Step 5:

Go back to “node130” folder, type the following commands.

- `mig java -java-classname=PacketMsg PacketMsg.h PacketMsg -o PacketMsg.java`
- `mig java -java-classname=PacketRMsg PacketMsg.h PacketRMsg -o PacketRMsg.java`
- `mig java -java-classname=PacketIMsg PacketMsg.h PacketIMsg -o PacketIMsg.java`
- `mig java -java-classname=PacketQMsg PacketMsg.h PacketQMsg -o PacketQMsg.java`
- `mig java -java-classname=PacketDMsg PacketMsg.h PacketDMsg -o PacketDMsg.java`
- `mig java -java-classname=PacketCMsg PacketMsg.h PacketCMsg -o PacketCMsg.java`



```
Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPtwohop/node130
$ mig java -java-classname=PacketRMsg PacketMsg.h PacketRMsg -o PacketRMsg.java
Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPtwohop/node130
$
```

Figure 5.2.3.7: Header migration command

These commands are used to migrate the nesC language type of header structure into Java type packet structure so that it can be compatible and run in Java application. After that, go to the node130 folder, some .java files will appear in the folder, cut and paste those .java file into “java_app” folder.

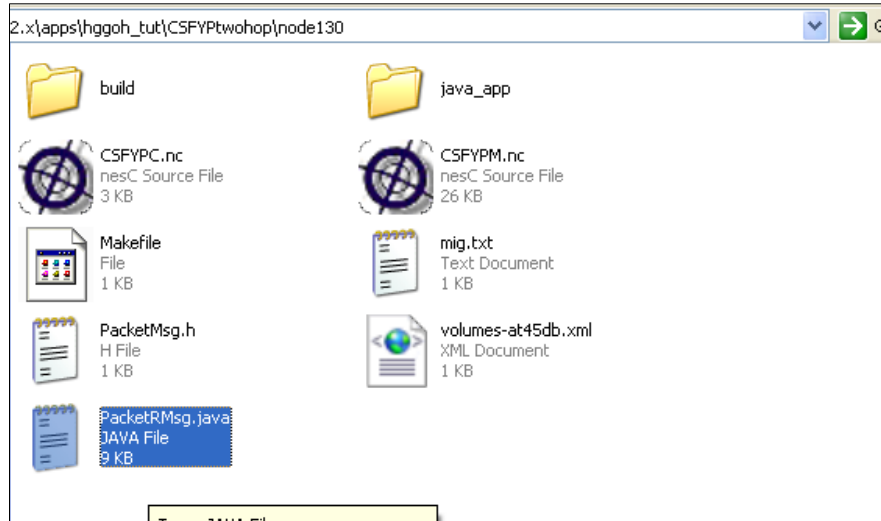


Figure 5.2.3.8: Packet structure in java format

```
Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPtwohop
$ cd java_app/

Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPtwohop/java_ap
$ ls *.java
LoadandShowImageFromFile.java  PacketIMsg.java  PacketRMsg.java
PacketCMsg.java                PacketMsg.java   PingPong.java
PacketDMsg.java                PacketQMsg.java
```

Figure 5.2.3.9: Listed java files in java_app folder

After moving all the packet structures into java_app folder, type “ls *.java” and you will see all the java files inside the folder. After that, type “javac *.java” to compile all the java files.

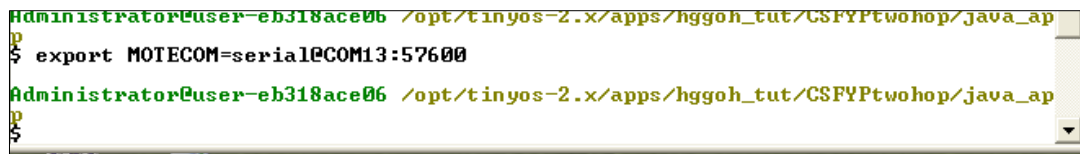
```
Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPtwohop/java_ap
$ javac *.java

Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPtwohop/java_ap
```

Figure 5.2.3.10: java compilation command

Step 6:

Type “export MOTECOM=serial@COM13:57600” command into Cygwin bash shell. The COM13 is the data communication port that forwarding all the data from base station to PC.

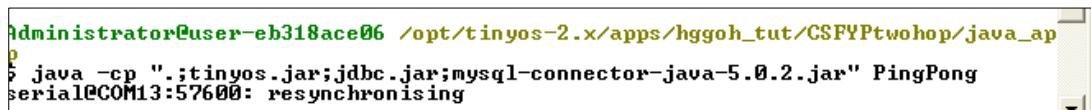


```
Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPTwohop/java_ap
$ export MOTECOM=serial@COM13:57600
Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPTwohop/java_ap
$
```

Figure 5.2.3.11: java export command

Step 7:

Finally, to run the java application, type “java -cp “.;tinyos.jar;jdbc.jar;mysql-connector-java-5.0.2.jar” PingPong”.



```
Administrator@user-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFYPTwohop/java_ap
$ java -cp ".;tinyos.jar;jdbc.jar;mysql-connector-java-5.0.2.jar" PingPong
serial@COM13:57600: resynchronising
```

Figure 5.2.3.11: java application run command

CHAPTER 5

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring



Figure 5.2.3.12: The whole WSNs in this project

Figure 5.2.3.12 shows the picture of source node interfaced with sensors, intermediate node and base station. The application is ready to run and some screenshot is shown in next section.

5.2.4 Screenshot of the Running Program

After done all the necessary compilation steps which have been described in section 5.2.3, a java GUI is appeared on Windows. Figure 5.2.4 shows the simple GUI of this project. There are three buttons on top panel. To start the application, user have to press “Start” button while “Stop” button is used to stop listening on message. “Clear” button is used to clear out the data on text area on panel below.

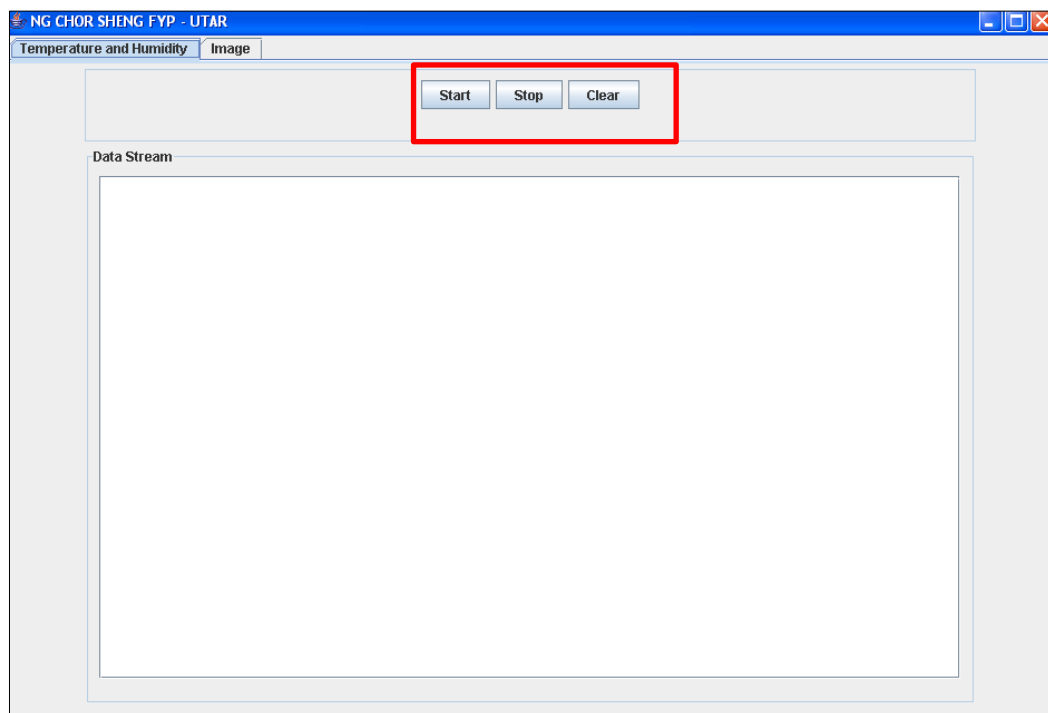


Figure 5.2.4.1: GUI of this project

CHAPTER 5

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

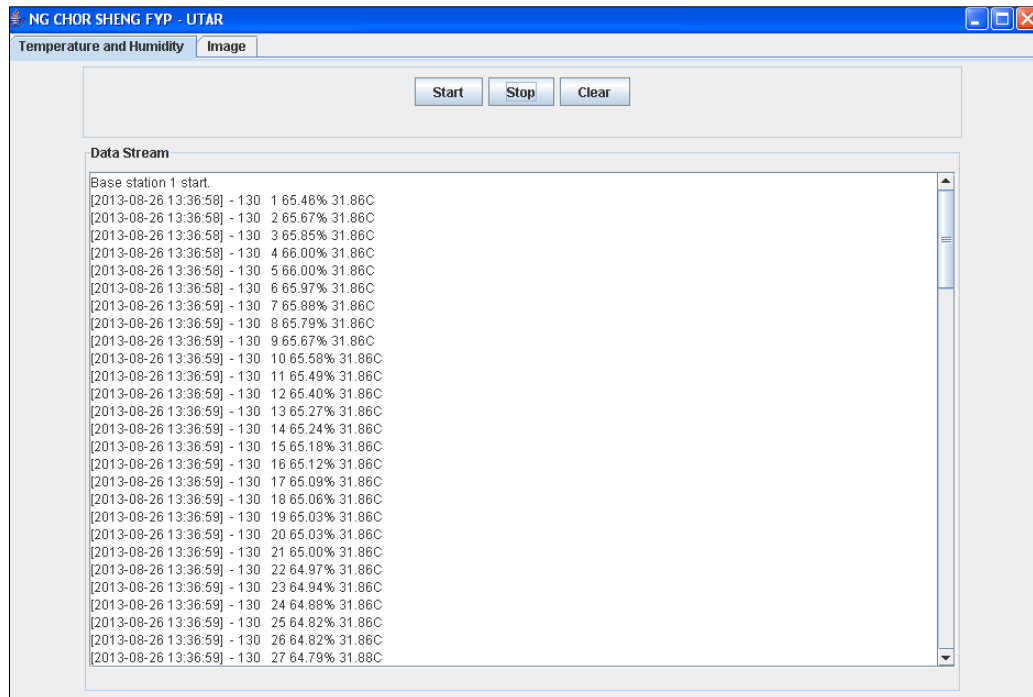


Figure 5.2.4.2: GUI of this project

Figure 5.2.4.2 shows the some readings on Data Stream panel. The format of the readings is described as below:

[Date and time] [Node ID] [Packet ID] [Humidity value] [Temperature value]

```
Administrator@User-eb318ace06 /opt/tinyos-2.x/apps/hggoh_tut/CSFPtwohop/java_ap
p
$ java -cp ".:tinycos.jar;jdbc.jar:mysql-connector-java-5.0.2.jar" PingPong
serial@COM13:57600: resynchronising
Driver loaded
Database connected
Base station 1 start.
2013-08-26 13:36:58 130 1 65.46% 31.86C
2013-08-26 13:36:58 130 2 65.67% 31.86C
2013-08-26 13:36:58 130 3 65.85% 31.86C
2013-08-26 13:36:58 130 4 66.00% 31.86C
2013-08-26 13:36:58 130 5 66.00% 31.86C
2013-08-26 13:36:58 130 6 65.97% 31.86C
2013-08-26 13:36:59 130 7 65.88% 31.86C
2013-08-26 13:36:59 130 8 65.79% 31.86C
2013-08-26 13:36:59 130 9 65.67% 31.86C
2013-08-26 13:36:59 130 10 65.58% 31.86C
2013-08-26 13:36:59 130 11 65.49% 31.86C
2013-08-26 13:36:59 130 12 65.40% 31.86C
2013-08-26 13:36:59 130 13 65.27% 31.86C
2013-08-26 13:36:59 130 14 65.24% 31.86C
2013-08-26 13:36:59 130 15 65.18% 31.86C
2013-08-26 13:36:59 130 16 65.12% 31.86C
2013-08-26 13:36:59 130 17 65.09% 31.86C
2013-08-26 13:36:59 130 18 65.06% 31.86C
2013-08-26 13:36:59 130 19 65.03% 31.86C
2013-08-26 13:36:59 130 20 65.03% 31.86C
2013-08-26 13:36:59 130 21 65.00% 31.86C
2013-08-26 13:36:59 130 22 64.97% 31.86C
2013-08-26 13:36:59 130 23 64.94% 31.86C
2013-08-26 13:36:59 130 24 64.88% 31.86C
2013-08-26 13:36:59 130 25 64.82% 31.86C
2013-08-26 13:36:59 130 26 64.82% 31.86C
2013-08-26 13:36:59 130 27 64.79% 31.88C
2013-08-26 13:36:59 130 28 64.73% 31.88C
2013-08-26 13:36:59 130 29 64.70% 31.88C
2013-08-26 13:37:00 130 30 64.64% 31.88C
2013-08-26 13:37:00 130 31 64.58% 31.88C
2013-08-26 13:37:00 130 32 64.53% 31.88C
```

Figure 5.2.4.3: Readings shown on Cygwin bash shell

Figure 5.2.4.3 shows some sample readings of SHT11 values on Cygwin bash shell. Figure 5.2.4.4 shows the sample data in raw_data table which stored all the SHT11 readings into the database.

```
Command Prompt - mysql -u root -p
mysql> select *from raw_data
-> ;
+-----+-----+-----+-----+-----+
| pc_timestamp | node_id | packet_id | temp | humi |
+-----+-----+-----+-----+-----+
| 2013-08-26 13:36:58 | 130 | 1 | 31.86 | 65.46 |
| 2013-08-26 13:36:58 | 130 | 2 | 31.86 | 65.67 |
| 2013-08-26 13:36:58 | 130 | 3 | 31.86 | 65.85 |
| 2013-08-26 13:36:58 | 130 | 4 | 31.86 | 66 |
| 2013-08-26 13:36:58 | 130 | 5 | 31.86 | 66 |
| 2013-08-26 13:36:58 | 130 | 6 | 31.86 | 65.97 |
| 2013-08-26 13:36:59 | 130 | 7 | 31.86 | 65.88 |
| 2013-08-26 13:36:59 | 130 | 8 | 31.86 | 65.79 |
| 2013-08-26 13:36:59 | 130 | 9 | 31.86 | 65.67 |
| 2013-08-26 13:36:59 | 130 | 10 | 31.86 | 65.58 |
| 2013-08-26 13:36:59 | 130 | 11 | 31.86 | 65.49 |
| 2013-08-26 13:36:59 | 130 | 12 | 31.86 | 65.4 |
| 2013-08-26 13:36:59 | 130 | 13 | 31.86 | 65.27 |
| 2013-08-26 13:36:59 | 130 | 14 | 31.86 | 65.24 |
| 2013-08-26 13:36:59 | 130 | 15 | 31.86 | 65.18 |
| 2013-08-26 13:36:59 | 130 | 16 | 31.86 | 65.12 |
| 2013-08-26 13:36:59 | 130 | 17 | 31.86 | 65.09 |
| 2013-08-26 13:36:59 | 130 | 18 | 31.86 | 65.06 |
| 2013-08-26 13:36:59 | 130 | 19 | 31.86 | 65.03 |
| 2013-08-26 13:36:59 | 130 | 20 | 31.86 | 65.03 |
| 2013-08-26 13:36:59 | 130 | 21 | 31.86 | 65 |
| 2013-08-26 13:36:59 | 130 | 22 | 31.86 | 64.97 |
| 2013-08-26 13:36:59 | 130 | 23 | 31.86 | 64.94 |
| 2013-08-26 13:36:59 | 130 | 24 | 31.86 | 64.88 |
| 2013-08-26 13:36:59 | 130 | 25 | 31.86 | 64.82 |
| 2013-08-26 13:36:59 | 130 | 26 | 31.86 | 64.82 |
```

Figure 5.2.4.4: SHT11 readings stored into database

CHAPTER 5

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

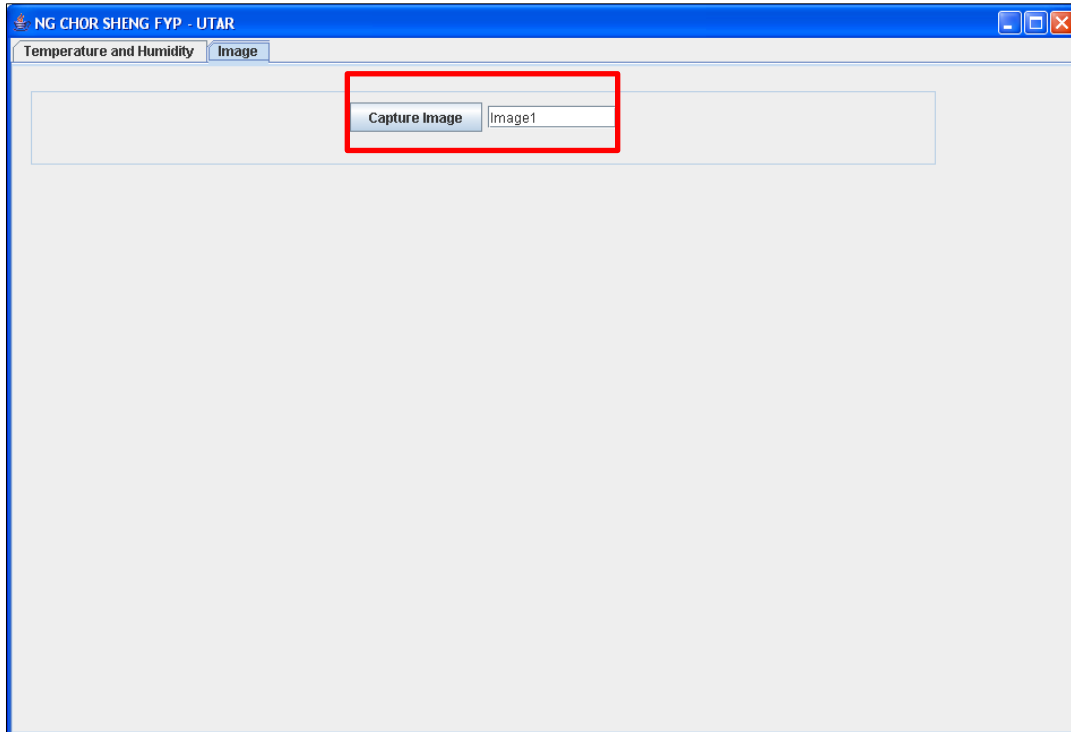


Figure 5.2.4.5: GUI to display image captured from C329 camera

Figure 5.2.4.5 shows the second tab of GUI application which is used to display image that captured from C329 camera. To start capturing, click the “Capture Image” button, user can provide a name for the captured image and store into database for further retrieve purpose.

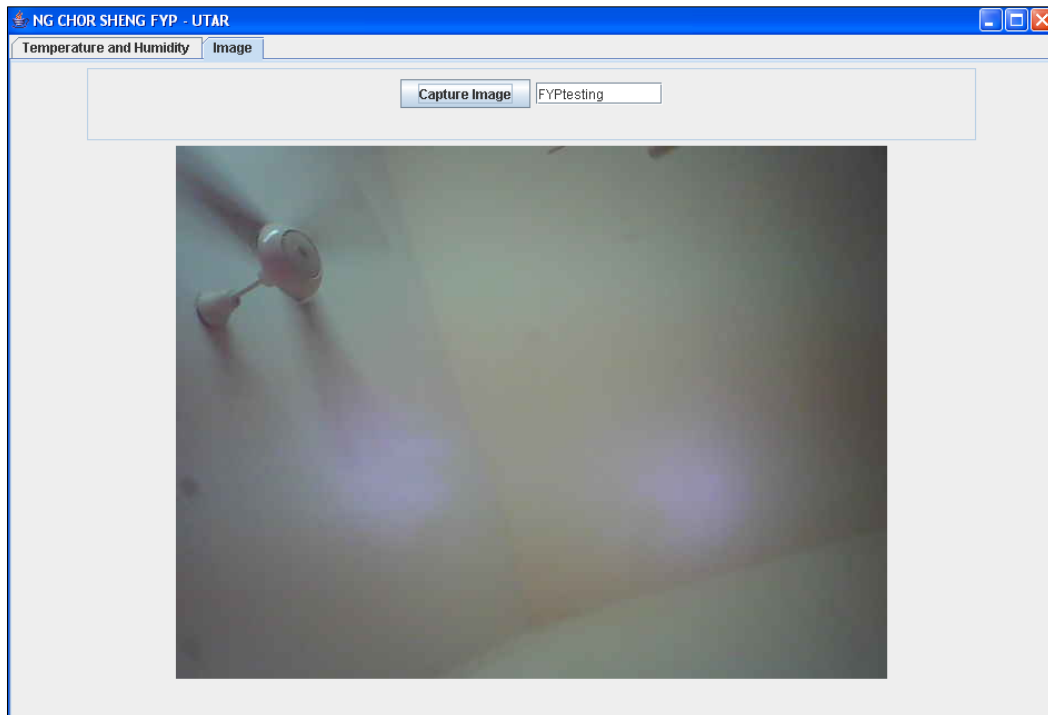


Figure 5.2.4.6: Image being displayed on Image tab

Figure 5.2.4.6 shows a sample image being captured from source and displayed on the GUI Image tab. Figure 5.2.4.7 below shows a fragment part of binary image data queried from MySQL.

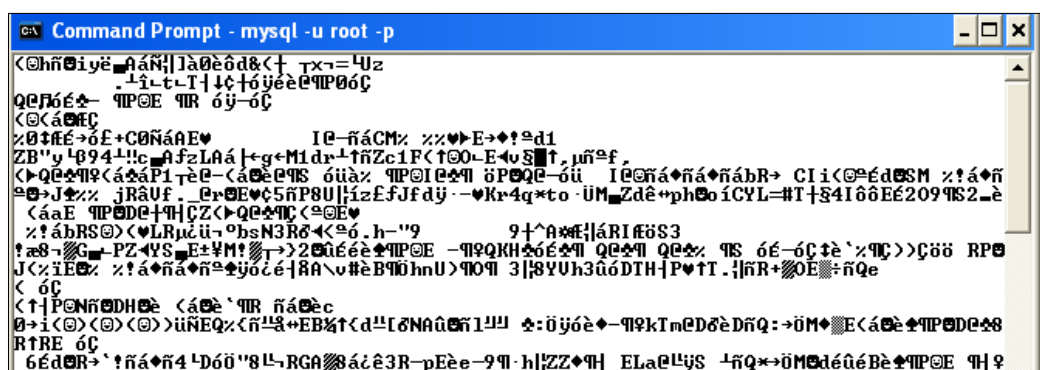
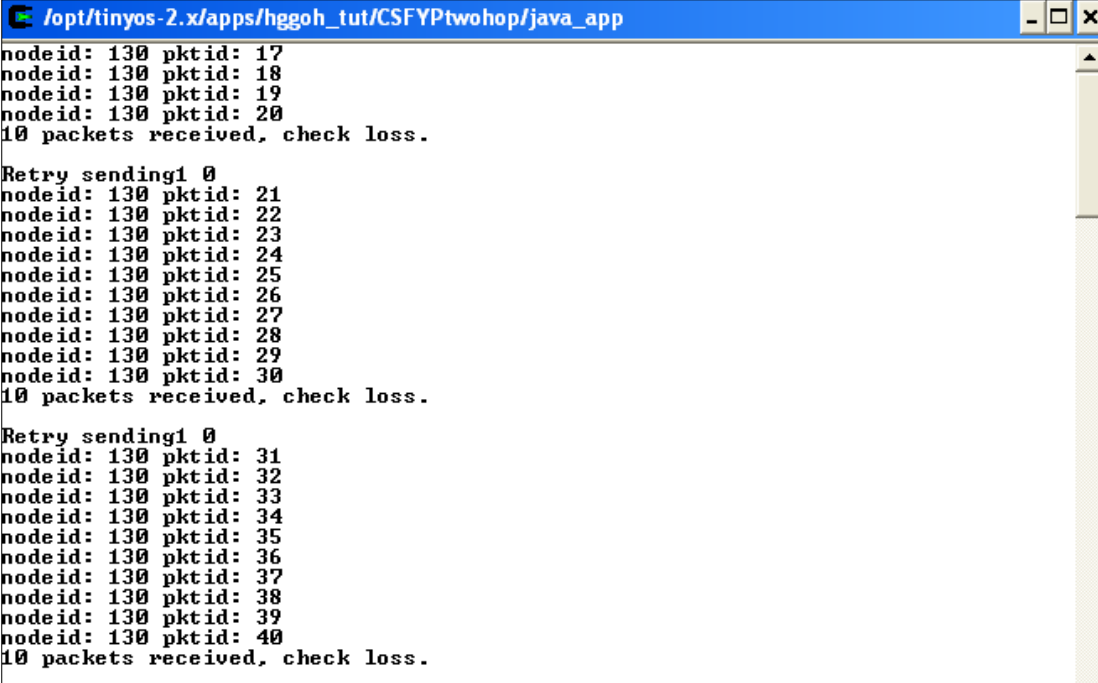


Figure 5.2.4.7: Binary image data queried from MySQL

A screenshot of a Cygwin bash shell window. The title bar reads "/opt/tinyos-2.x/apps/hggoh_tut/CSFYPtwohop/java_app". The terminal output shows a sequence of network data: "nodeid: 130 pktid: 17", "nodeid: 130 pktid: 18", "nodeid: 130 pktid: 19", "nodeid: 130 pktid: 20", followed by "10 packets received, check loss.". This is repeated for three more groups of packets, each starting with "Retry sending1 0" and ending with "10 packets received, check loss.". The packet IDs in each group are 21-30, 31-40, and 31-40 respectively.

```
/opt/tinyos-2.x/apps/hggoh_tut/CSFYPtwohop/java_app
nodeid: 130 pktid: 17
nodeid: 130 pktid: 18
nodeid: 130 pktid: 19
nodeid: 130 pktid: 20
10 packets received, check loss.

Retry sending1 0
nodeid: 130 pktid: 21
nodeid: 130 pktid: 22
nodeid: 130 pktid: 23
nodeid: 130 pktid: 24
nodeid: 130 pktid: 25
nodeid: 130 pktid: 26
nodeid: 130 pktid: 27
nodeid: 130 pktid: 28
nodeid: 130 pktid: 29
nodeid: 130 pktid: 30
10 packets received, check loss.

Retry sending1 0
nodeid: 130 pktid: 31
nodeid: 130 pktid: 32
nodeid: 130 pktid: 33
nodeid: 130 pktid: 34
nodeid: 130 pktid: 35
nodeid: 130 pktid: 36
nodeid: 130 pktid: 37
nodeid: 130 pktid: 38
nodeid: 130 pktid: 39
nodeid: 130 pktid: 40
10 packets received, check loss.
```

Figure 5.2.4.8: Segment of images displayed in Cygwin bash shell

Figure 5.2.4.8 above shows some segment of image data being forwarded to PC and displayed on Cygwin bash shell.

5.3 Summary/Concluding Remark

In summary, this section has explained the hardware design and its actual implementation. The TinyOS and MySQL installation guideline also provided in this section. The necessary files such as java file and nesC code needed in this project also included as well. Also, the MICAz compilation steps also provided to user who wish to install the system. Lastly, some of the screenshot of workable program also provided as well.

CHAPTER 6 DISCUSSION**6.1 System Verification****6.1.1 Raw Data Packet Reliability Rate Using Point-to-point Protocol**

This plan is designed to test out the reliability rate using point-to-point DAT-ACK protocol. There are two experiments to be tested, first is single hop which only source node and base station are involved, second is two hops which included intermediate node act as a relay node to forward data. The source node is required to send out total unique 200 packets (raw data) to the base station and based on that to calculate how many packets are lost. Besides that, the experiment test at the same time also accumulates the total number of duplicate packets received at the PC side. Each of the tests is also needed to be repeated 10 times to get the total average of lost packet. These tests are conducted at Block N corridor and the distance between hops is 10 meter away. For single hop experiment test, the result is shown as below:

No.	Total unique packet sent	Total loss packet	Total duplicate packet received
1	200	4	5
2	200	2	4
3	200	0	1
4	200	5	1
5	200	0	0
6	200	3	10
7	200	5	9
8	200	1	2
9	200	2	2
10	200	4	7
Average:	200	2.6	4.1

Table 6.1.1.1: Single hop packet reliability rate for raw data

For two hops testing, the result is shown as follow:

No.	Total unique packet sent	Total loss packet	Total duplicate packet received
1	200	0	3
2	200	0	0
3	200	6	11
4	200	2	6
5	200	0	4
6	200	3	7
7	200	4	3
8	200	7	4
9	200	7	6
10	200	1	3
Average:	200	3	4.7

Table 6.1.1.2: Two hops packet reliability rate for raw data

Based on the two experiment results, the average total loss packet for single hop and two hops is almost the same which is around 2.5 – 3 packets loss. As can be seen in both results, some of the result can be very good, which achieved no loss found and no duplicate packet receive while some of the result is bad, which the loss packet can reach to 7 and duplicate packet can hike to 11. All these result can be affected by the environment issues or some object (such as human) moving around that cause the transmitted signal to be reflected.

The reason of why the receiver received duplicate packets is because the point-to-point DAT-ACK protocol being implemented into system. This is due to the sender will send the packet up to three times to receiver if the sender didn't receive ACK message from receiver. However, the ACK message might have lost during transmission or the radio is busy which can't send out the ACK message, therefore, the sender assume that receiver didn't receive the packet and when time out, the packet is being sent to receiver again, which cause the receiver received duplicate packet.

Based on the result, we can see that both of the tests are achieving less than 5% of packet loss.

6.1.2 Image Data Packet Reliability Rate Using Point-to-point and End-to-end Protocol

The purpose of this verification testing is to test the packet reliability rate for image data with layer 2 and layer 4 protocols implemented. This test is similar as the test done in section 6.1.1. But this test is going to calculate the total number of loss packet and whether the loss packets are able to be resent back to receiver.

Since every image might have different size, the total number of packets sent is also different. Therefore, the range of the packets is within 200 to 250 (each packet contain 70 bytes of image data), which means the size of an image is about 14KB to 17KB.

The test is also conducted at Block N corridor. The experiment testing is also done in two ways: single hop and two hops which is multi-hop. The results of two tests are shown in the following tables.

CHAPTER 6

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

No.	Total unique packet sent	Total loss packet (able to receive again)	Total duplicate packet received
1	207	4	9
2	220	1	0
3	214	0	0
4	220	1	0
5	235	2	4
6	235	4	2
7	235	5	3
8	235	1	1
9	235	6	3
10	235	9	4
Average:	227.1	3.3	2.6

Table 6.1.2.1: One hop packet reliability rate for image data

No.	Total unique packet sent	Total loss packet (able to receive again)	Total duplicate packet received
1	235	0	1
2	235	1	2
3	235	4	6
4	235	0	2
5	235	1	0
6	235	4	3
7	242	4	2
8	242	2	0
9	242	3	1
10	242	5	6
Average:	237.8	2.4	2.3

Table 6.1.2.2: Two hops packet reliability rate for image data

Table 6.1.2.1 shows the result of one hop testing while table 6.1.2.2 shows the result of two hops testing of packet reliability rate for image data. The average loss packets which were able to be retransmitted from source node is 3.3 (single hop) and

2.4 (two hops). The average duplicate packets received is 2.6 (single hop) and 2.3 (two hops).

Although the results above show that all missing segment packets were able to receive from the source node again. However, for each test, there is 1 to 2 occurrences out of 10 times that the missing packets failed to be received again. This is because the source node was unable to receive layer 4 ACK or NACK message, these could cause the image transmission being terminated and can't be able to fully collect all the image data to do the experiment testing.

The root cause of this problem is that whenever PC trying to send either layer 4 NACK or ACK control message to source node, the control messages was unable to reach the intermediate node or even the source node. Figure 6.1.2.1 shows that the PC tried three times sending layer 4 ACK message to source node and yet no layer 2 ACK control message is received. If the source node didn't receive layer 4 ACK message from PCs. The timer will expired and the source node will terminate current image transmission and start sending raw data.

```
Retry sending1 0
nodeid: 130 pktid: 71
nodeid: 130 pktid: 72
nodeid: 130 pktid: 73
nodeid: 130 pktid: 74
nodeid: 130 pktid: 75
nodeid: 130 pktid: 76
nodeid: 130 pktid: 77
nodeid: 130 pktid: 78
nodeid: 130 pktid: 79
nodeid: 130 pktid: 80
10 packets received, check loss.
Retry sending1 0
Retry sending1 0
Retry sending1 0
```

Figure 6.1.2.1: Source node failed to receive layer 4 ACK message

6.1.3 Total Time Delay to Receive all Image Data

This test is planned to get the total time needed for whole image data to be received by PC. This plan also design to do two testing which is single hop and multi-hop (2 hops). The results are shown on tables below:

No.	Total unique packet sent	Total time delay
1	207	1:0:422
2	220	0:49:968
3	214	0:42:156
4	220	0:44:0
5	235	0:50:313
6	235	0:48:297
7	235	0:48:282
8	235	0:48:234
9	235	0:48:281
10	235	0:56:15
Average:	227.1	0:49:596

Table 6.1.2.1: Total time delay to receive whole image (single hop)

No.	Total unique packet sent	Total time delay
1	235	1:32:250
2	235	1:38:329
3	235	1:56:297
4	235	1:2:14
5	235	1:38:250
6	235	1:48:469
7	242	1:46:249
8	242	1:46:352
9	242	1:50:250
10	242	1:48:479
Average:	237.8	1:41:006

Table 6.1.2.2: Total time delay to receive whole image (two hops)

Based on the results from tables above, we can see that the average time required to receive all segments of image is around 49 sec (single hop) while it takes around 1 min 41sec to receive all image data from two hops away, The reason why it would take that long to receive all image data is because the implementation of active ACK control message and also the reliable data collection protocols that implemented into the whole system. For instance, for single hop, the source node is required to send a beacon every 2 sec to parent' node before the actual data sending can happen. If an image data is required to have total 200 packets to be transferred, and each round only can send 10 packets, the minimum time requires sending all 200 packets would be 20×2 sec which is around 40sec (assuming the parent node is in coverage area and the parent node is always reply active ACK back to source node). Besides, some of the delay could due to the layer 2 DAT- ACK message that sending back and forth. This is because when the source didn't receive layer 2 ACK message, the timer will time out (100 milli sec) and resend the segment image again. Therefore, the accumulation of time to resend packet will cause to increase the total time delay as well.

As compared to both tests, the total time delay is doubled up when transferring image data through multi-hop (2 hops away). This is because the adding of intermediate node with protocols implemented would cause the total time to be increased as well.

6.2 Problems Encountered and Solutions

6.2.1 MICAz can't Synchronize with C329 Camera

This problem is encountered when trying to synchronize C329 camera using the commands set provided by C329 camera datasheet. At first, when MICAz sending SYNC command trying to synchronize with C329 camera, the C329 didn't respond any command such as ACK back to MICAz. After searching some website, the author found out that the default baud rate of C329 camera is 14400 whereas the default baud rate of MICAz is 57600.

To solve the problem, the author changed the default baud rate of MICAz to be 14400 as well. Therefore, the MICAz can finally first synchronize with C329 camera and also be able to send other commands set as well.

6.2.2 Can't Form the Whole Image after Receiving Image Data

```

pktid: 32 total image size: 11856 total block size: 11265
incoming data count: 8897 image written in flash memory count: 8883
Gap difference: 448
pktid: 33 total image size: 11856 total block size: 11265
incoming data count: 9167 image written in flash memory count: 9153
Gap difference: 462
pktid: 34 total image size: 11856 total block size: 11265
incoming data count: 9437 image written in flash memory count: 9423
Gap difference: 476
pktid: 35 total image size: 11856 total block size: 11265
incoming data count: 9707 image written in flash memory count: 9693
Gap difference: 490
pktid: 36 total image size: 11856 total block size: 11265
incoming data count: 9977 image written in flash memory count: 9963
Gap difference: 504
pktid: 37 total image size: 11856 total block size: 11265
incoming data count: 10247 image written in flash memory count: 10233
Gap difference: 518
pktid: 38 total image size: 11856 total block size: 11265
incoming data count: 10517 image written in flash memory count: 10503
Gap difference: 532
pktid: 39 total image size: 11856 total block size: 11265
incoming data count: 10787 image written in flash memory count: 10773
Gap difference: 546
pktid: 40 total image size: 11856 total block size: 11265
incoming data count: 11057 image written in flash memory count: 11043
Gap difference: 560
pktid: 41 total image size: 11856 total block size: 11265
incoming data count: 11327 image written in flash memory count: 11313
Gap difference: 574
pktid: 42 total image size: 11856 total block size: 11265
incoming data count: 11597 image written in flash memory count: 11583
Gap difference: 588
pktid: 43 total image size: 11856 total block size: 11265
incoming data count: 11856 image written in flash memory count: 11853
Gap difference: 591

```

Figure 6.2.2.1: Image data failed to write into flash memory

This problem occurred when the author tried to store all the image data that received from C329 UART serial communication straight into flash memory. The author found that some of the image data may loss if straight writing into flash memory. This is because each byte of image data is coming too fast while the I/O overhead of writing byte by byte data to flash memory causing it can't handle the fast come in of image data. Therefore, some if the image data are failed to write into flash memory.

As can be seen in figure 6.2.2.1 above the total image size is the actual size of image while the total block size is the actual image size that contained in flash memory. There is clearly a total loss of 591 bytes for the whole image. The incoming data count is the counter of counting the current number of data received from C329 camera whereas the total block size is the actual number of byte images that are successfully stored into flash memory. From the figure, we can see that the loss happened very frequently. Therefore, some crucial byte of image data may loss and hence causing the PC can't reform the picture.

To solve the problem, the author has designed to storing the incoming byte of image data into temporary buffer, after received total 500 bytes of image data then only store them into flash memory at one shot. This could reduce the I/O overhead of writing data into flash memory. For more details of this design, please go back to figure 4.2.5.5.

6.2.3 Flash memory can't be Accessed Simultaneously

This happen when the author tried to storing the raw data into flash memory, at the same time the image data is also stored into flash memory. This causes the loss of the data either raw data or image data that trying to access the flash memory due to the flash memory is currently granted by others.

To solve the problem, whenever user request image data, the current activity of reading raw data are stopped. Until the entire image data are transmitted to PC successfully, then the raw data activity will start again to avoid both activities trying to compete granting the flash memory resources.

6.2.4 Source Node Failed to Receive Layer 4 Control Message

This happened when first implementing the end-to-end protocol into MICAz. During the testing, the author found out that the source node was not able to receive layer 4 ACK/NACK message very frequently which could cause the whole image transmission being terminated. This is due to no layer 2 DAT-ACK protocol being implemented in sending message from destination which is PC to source. After implemented, the loss of layer 4 ACK/NACK control message doesn't occur so frequent.

6.3 Achievement

In this project, the first objective is to enable and activate the functionalities of temperature and humidity as well as VGA camera sensors on MICAz platform. The author is able to start-up and triggers both the SHT11 sensor and C329 camera. Therefore, the first objective is achieved.

For second objective which is to design and implement reliable data collection protocol suitable for self-powered WSNs in agriculture monitoring. The author has designed and implemented point-to-point DAT-ACK protocol for transferring SHT11 readings and also another end-to-end protocol combined with point-to-point protocol to transferring the image data. Therefore, the second objective is achieved as well.

For third objective is to collect the sensed data and image from MICAz using the protocol installed. The author is able to collect both SHT11 readings and image data queried by user through multihopping using the protocol installed. So, the third objective is also achieved.

For fourth objective which is to set up a database to store the both SHT11 values as well as image data. The author has successfully set up a MySQL database with two tables to store the SHT11 readings and image data into respective table. Therefore, fourth objective is achieved as well.

For fifth objective which is to create user interface to display the environmental information and crops image. The author has create a simple GUI that allows user to view the temperature and humidity values, the user can also request to capture an image and the whole image are able to transfer and display on GUI as well. So, fifth objective is achieved.

For last objective which is to carry out performance study in terms of data packet reliability rate according to the number of hops. The author has also carried out

the test and some discussions also provided in chapter 5 as well, therefore, the last objective is achieved.

In a nut shell, all the objectives in this project are achieved.

6.4 SWOT

Strength	Weakness
<ul style="list-style-type: none"> • Transmitting raw and image data in a reliable way • Achieving less than 5% of packet loss • Data are relayed through multi-hop communication • User can view the environmental readings and request image when needed 	<ul style="list-style-type: none"> • Transmission of image data is slow and the time is doubled up when hops increase • Too many control messages going back and forth which could drain out battery very quickly
Opportunity	Threat
<ul style="list-style-type: none"> • WSN in agriculture monitoring still new in Malaysia market • Can combine with some routing protocol to find its own path way • can combine with some other end system such as crops image analysis or some plantation system to be a comprehensive agriculture monitoring system 	<ul style="list-style-type: none"> • Farmer or plantation owner are reluctant to try out new technology for monitor their farm land

6.5 Summary/Concluding Remarks

As a summary, two verification plans are done which are the data packet reliability rate using point-to-point as well as end-to-end protocol. Besides, another test is carried out to determine total time delay required to receive whole segment of image. Some of the problem encountered during implementation and its solutions are discussed as well. Lastly, the achievements and SWOT analysis of this project is defined as well.

Chapter 7 CONCLUSION

7.1 Overall Conclusion

As previously mentioned under the Problem Statement and Motivation, the scalar data and crops image are needed to be collected in a proper and reliable way back to farmers so that further actions such as crops health analysis can be taken. However, due to the unique characteristic of image data, it poses a challenge to transmit the data back to farmer. In chapter 1, in order to solve the problem, project scope and objectives have been stated down clearly. The final aims of this project is to design and implement a reliable data collection protocol on the MICAz platform which have solar panels supported, and temperature and humidity sensors as well as VGA camera sensors to gather the critical environmental data.

In chapter 2, four related papers were reviewed and their advantages and disadvantages have been listed, the comments about these articles were written out as well. In chapter 3, agile model is chosen to be used throughout the whole FYP. The system requirement specifications are stated so as to allow the author to prepare for the FYP2. Project timeline and budgets are also listed to ensure the project can be delivered in time while controlled the budget plan listed in this chapter.

In chapter 4, the overall design of the system has been elaborated. Besides, two protocols which is layer 2 point-to-point DAT-ACK protocol as well as layer 4 end-to-end reliable data collection protocol are designed also. The flow chart of enabling SHT11 sensor and C329 camera are drawn.

In chapter 5, the hardware implementation such as circuit design of SHT11 sensor and C329 camera are explained. The TinyOS and MySQL installation guideline also provided as well. Besides, all of the files and folders required in this project is stated clearly and some example code is explained as well. The steps to

compile and run the MICAz and its applications are also provided. Besides, some of the screenshot of the workable system are also included in this chapter.

In chapter 6, two verification plans are defined and carried out and the discussion of the result is also included under this chapter. Besides, the author also stated out some problem encountered during the implementation of this project and providing the solutions so that user can be aware of it when implement this system again. Lastly, the achievements and SWOT analysis have been defined and verified as well.

7.2 Future Enhancement

This project only involved three nodes which is source, intermediate and base station. For future enhancement, the intermediate node can be added more to test the reliability as well. Besides, it can be enhanced by implementing routing protocol that could allow the sensor node to self-discovered its path way and form its parent-child relationship. Whenever intermediate node is down, there will still have an alternative path to go. Therefore, the data can still be forwarded to another end even though its parent's node is down.

Besides, one of the enhancement can be done is to research on finding a way to reduce down the control message. One way is to determine the probability of loss for sending image data. If the probability of loss is low in the current environment condition, the total number of packet to be checked can be enlarged. Therefore, reduce down the layer 4 ACK message to be sent.

Furthermore, the environmental data stored in the database can be retrieved and furthered process to some useful information. Besides, some image processing algorithm can be applied onto MICAz with C329 camera attached. The C329 camera

CHAPTER 7

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

will self-capture the image with some time interval. After that, the MICAz will do some image processing to detect whether the crops is in good condition or not. If there is not much changes of the crops condition, the image will not be sent to user to save up the energy of MICAz.

REFERENCES

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

REFERENCES

BayAmp, 2011, *Agile Software Development Process*. Available from: < <http://www.bayamp.com/methodology.html>>. [14 August 2012].

ComputerSci.ca, 2007, *Educational flaws: Programming with the waterfall model*. Available from: < <http://compsci.ca/blog/educational-flaws-programming-with-the-waterfall-model/>>. [14 August 2012].

Crossbow n.d., *MICAZ: Wireless Measurement System*. Available from: < http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf>. [19 July 2012].

Crossbow Technology n.d. *MPR/MIB User's Manual*. Available from: http://bullseye.xbow.com:81/Support/Support_pdf_files/MPR-MIB_Series_Users_Manual.pdf. [16 August 2013].

Electronics123.com, Inc. n.d. *C328 and C329 Cameras*. Available from: < <http://www.electronics123.com/s.nl/sc.8/category.207684/.f>>. [18 July 2012].

Electronics123.com, Inc. 2010, *C329-UART User Manual*. Available from: <www.electronics123.net/amazon/datasheet/C329_UART_UM.pdf>. [16 August 2013].

Goh, H. G., Sim, M. L. & Hwe, H. T. 2007, 'Agriculture Monitoring' in *Sensor Networks and Configuration*. ed. by Nitaigour P. Mahalik. New York: Springer, pp. 439-462.

Hu, F. & Cao, X., 2010, *Wireless Sensor Networks: Principles and Practice*. New York: CRC Press, Taylor & Francis Group, LLC.

REFERENCES

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

Jacques P., Seshagiri R., Prabhakar, T.V., Jean-Pierre H. & Jamadagni H. S. 2007, 'COOMONSense Net: A Wireless Sensor Network for Resource-Poor Agriculture in the Semiarid Areas of Developing Countries' *Journal of Information Technologies and International Development*, vol. 4, no. 1, pp. 51-67.

Karim, L., Nasser, N. & Salti, T. E. 2011, 'Efficient Zone-based Routing Protocol of Sensor Network in agriculture monitoring systems.' *Conference on Communications and Information Technology (ICCIT)*. Held 29-31 March 2011 at Aqaba. IEEE Computer Society, pp. 167 – 170.

Kwong, K.H., Wu T., Goh, H.G., Sasloglou, K., Stephen, B., Glover, I., Shen, C., Du, W., Craig, M. & Andonovic, I. 2012, 'Practical Considerations for Wireless Sensor Networks in Cattle Monitoring Applications' *Computers and Electronics in Agriculture*, vol. 81, pp. 33-44.

Marin-Perianu, M. & Havinga, p., 2005. 'Experiments with Reliable Data Delivery in Wireless Sensor Network' *Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference*. pp. 109 – 114.

Oracle Corporation, 2012, *MySQL: The world's most popular open source database*. Available from: < <http://www.mysql.com/>>. [19 July 2012].

Philip, L, Sam, M., Joseph, P., Robert, S., Alec, W., David, G., Jason, H., Matt, W., Eric, B. & David, C. 2005, 'TinyOS: An operating system for sensor networks' in *Ambient Intelligence* eds. by Weber, W., Rabaey, J. & Aarts, Emile H.L. Springer Verlag, pp. 115-148.

REFERENCES

Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring

SENSIRION n.d., *SHT1 – Digital Humidity Sensor (RH&T)*. Available from: <<http://www.sensirion.com/en/products/humidity-temperature/humidity-sensor-sht11/>>. [18 July 2012].

Shelby, Z. & Bormann, C. 2009, IEEE 802.15.4 Reference. In: *6LoWPAN: The Wireless Embedded Internet*. s.l.:John Wiley & Sons Ltd., pp. 191-194. Available from: WILEY Online Library. [11 August 2012].

Verma, S., Chug, N. & Gadre, D. 2010, 'Wireless Sensor Network for Crop Field Monitoring.' *Conference on Recent Trends in Information, Telecommunication and Computing*. Held 12-13 March at Kochi, Kerala. IEEE Computer Society, pp. 207-211.

Williams, L., 2007, *A Survey of Agile Development Methodologies*. Available from: <<http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf>>. [14 August 2012].

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 4	Study week no.: 2
Student Name & ID: Ng Chor Sheng – 1002711	
Supervisor: Mr Goh Hock Guan	
Project Title: Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring	

1. WORK DONE

- Able to enable and read SHT11 sensor values
- Able to enable C329 camera

2. WORK TO BE DONE

- Finding the root cause of why can't form the image and find the solution to tackle the problem

3. PROBLEMS ENCOUNTERED

- Can't form image after receiving image data

4. SELF EVALUATION OF THE PROGRESS

Feel bad because can't form the image and it's hard to trace the root cause of the problem

Supervisor's signature

Student's Signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 4	Study week no.: 4
Student Name & ID: Ng Chor Sheng – 1002711	
Supervisor: Mr Goh Hock Guan	
Project Title: Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring	

1. WORK DONE

- Able to find the solution to receive all image data and form the image

2. WORK TO BE DONE

- Design and implement the protocols for transferring raw and image data

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

Feel happy and satisfied as the image can be captured and formed successfully.

Supervisor's signature

Student's Signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 4	Study week no.: 6
Student Name & ID: Ng Chor Sheng – 1002711	
Supervisor: Mr Goh Hock Guan	
Project Title: Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring	

1. WORK DONE

- Have designed point-to-point protocol for raw data transferring
- The data are able to relay through multihop

2. WORK TO BE DONE

- Design end-to-end protocol for transferring image data

3. PROBLEMS ENCOUNTERED

-

4. SELF EVALUATION OF THE PROGRESS

Overall progress is running well.

Supervisor's signature

Student's Signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 4	Study week no.: 8
Student Name & ID: Ng Chor Sheng – 1002711	
Supervisor: Mr Goh Hock Guan	
Project Title: Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring	

1. WORK DONE

- The end-to-end protocol is designed and implemented into MICAz
- In the progress of combining all code together

2. WORK TO BE DONE

- Finish combining code
- Design GUI and set up database
- Do some testing

3. PROBLEMS ENCOUNTERED

- source node was unable to receive layer 4 NACK/ACK message

4. SELF EVALUATION OF THE PROGRESS

Overall progress is running well.

Supervisor's signature

Student's Signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 4	Study week no.: 10
Student Name & ID: Ng Chor Sheng – 1002711	
Supervisor: Mr Goh Hock Guan	
Project Title: Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring	

<p>1. WORK DONE</p> <ul style="list-style-type: none">- Combined all code together- one GUI design- Database has set up as well
<p>2. WORK TO BE DONE</p> <ul style="list-style-type: none">- Do final report- Finish experiment test
<p>3. PROBLEMS ENCOUNTERED</p> <p>Take time to combine all the code together as a lot of error need to be debugged.</p>
<p>4. SELF EVALUATION OF THE PROGRESS</p> <p>Overall progress is running well. Feel great that the FYP is almost finished</p>

Supervisor's signature

Student's Signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Year 3 Trimester 4	Study week no.: 12
Student Name & ID: Ng Chor Sheng – 1002711	
Supervisor: Mr Goh Hock Guan	
Project Title: Reliable Data Collection Protocol for Self-powered Wireless Sensor Network in Agriculture Monitoring	

<p>1. WORK DONE</p> <p>- Done experiment testing</p>
<p>2. WORK TO BE DONE</p> <p>- Doing report</p>
<p>3. PROBLEMS ENCOUNTERED</p> <p>-</p>
<p>4. SELF EVALUATION OF THE PROGRESS</p> <p>Rushing FYP report. Hope manage to submit report on time</p>

Supervisor's signature

Student's Signature