

# **CERTIFICATION OF APPROVAL**

## **UNDERWATER DATA COMMUNICATION PACKAGE**

by

Pan Sophoana 12941

A project dissertation submitted to the  
Information and Communication Technology

Universiti Teknologi PETRONAS

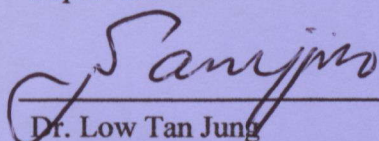
In partial fulfillment of the requirement of the

BACHELOR OF TECHNOLOGY (Hons)

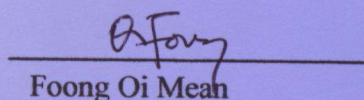
(INFORMATION AND COMMUNICATION TECHNOLOGY)

Approved by,

Supervisor

  
Dr. Low Tan Jung

Co-supervisor

  
Foong Oi Mean

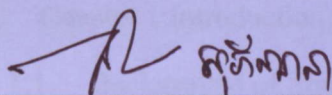
UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

April 2012

## CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



PAN SOPHOANA



## TABLE OF CONTENTS

CERTIFICATION OF APPROVAL .....	I
CERTIFICATION OF ORIGINALITY .....	II
Table of Contents .....	III
List of Figures .....	VI
List of Tables.....	VII
Abstract .....	1
Acknowledgment .....	2
1. Chapter 1 Introduction .....	3
1.1 Background of Study.....	3
1.2 Problem Statement .....	4
1.2.1 Problem Identification.....	4
1.3 Objectives and Scope of Study.....	4
1.3.1 Objectives.....	4
1.3.2 Scope of Study .....	5
1.4 Feasibility of the project.....	5
1.4.1 Scope Feasibility .....	5
1.4.2 Schedule Feasibility .....	5
1.4.3 Technical Feasibility .....	6
1.5 Challenges .....	6
1.5.1 Windows Presentation Foundation (WPF).....	6
1.5.2 Understanding of C Sharp programming language (C#) .....	6
1.5.3 Model View View-Model Architecture (MVVM).....	6
1.6 Expected Outcome .....	7
2. Chapter 2 Literature Review .....	8



2.1	Software Technology .....	8
2.1.1	Window Presentation Foundation (WPF) .....	8
2.1.2	C Sharp (C#) .....	9
2.1.3	Model View View-Model Architecture .....	10
2.2	Underwater Network Communication and Physics .....	10
2.2.1	What is Throughput?.....	10
2.2.2	Data Packet Size and Throughput Efficiency .....	11
2.2.3	Energy Efficiency.....	11
2.2.4	Bit Error Rate (BER).....	12
2.3	The Existing Port Manager .....	12
2.4	The Existing Testbed .....	13
3.	Chapter 3 Methodology of Study .....	14
3.1	Research Methodology .....	14
3.2	Project Activities .....	15
3.3	Key Milestone .....	16
3.4	Programming Architecture .....	16
3.5	Tool .....	17
3.5.1	IDE (Integrated Development Environment).....	17
3.5.2	SQLite Database Tool.....	17
3.5.3	Microsoft Express .....	17
3.5.4	Photoshop CS5 .....	17
4.	Chapter 4 Result and Discussion.....	18
4.1	Research and Understanding of WPF and MVVM Architecture.....	18
4.1.1	WPF Technology .....	18
4.1.2	MVVM Architecture .....	19
4.2	Develop Application's Workspace and Commands required BY MVVM Architecture.....	22



4.2.1	Application's Workspace .....	22
4.2.2	Application's Implementation.....	23
4.3	Development and System Architecture .....	24
4.3.1	Project Description.....	24
4.3.2	Deliverable Underwater Communication Package .....	29
4.4	Protocol Implementation.....	30
4.5	Algorithm Definition .....	30
5.	Chapter 5 Conclusion and Recommendation .....	33
5.1	Conclusion.....	33
5.2	Recommendation.....	34
	Reference.....	35
	Appendix A.....	36
	OSI Model Adaptation .....	36
	OSI Model Implementation.....	36
	Layer Design .....	37
	OSILayer Manager.....	38
	Protocol .....	39
	Protocol Implementation.....	39
	Structure .....	39
	Virtualization.....	39
	Packet.....	40
	DataSend .....	41
	DatatFragemented .....	42
	RawDataPacket .....	42
	Constant and Enumeration .....	43

## LIST OF FIGURES

Figure 1—1: Example Setup of UWA Testbed .....	7
Figure 2—2 Data Packet Format.....	11
Figure 2—3 WinSSD, Port Manager .....	12
Figure 4—1 Defining an Application in XAML.....	18
Figure 4—2 RelayCommand Class.....	20
Figure 4—3 ViewModel Class Hierarchy .....	21
Figure 4—4 Main Workspace of Underwater Communication Package.....	22
Figure 4—5 Wizard Windows for Setting Up Devices .....	23
Figure 4—6 Second Screen of the wizard.....	23
Figure 4—7 Dolphine.OSIModel Class Diagram.....	25
Figure 4—8 Dolphine.Packet Class Diagram .....	26
Figure 4—9 Protocol Setting Dialog .....	27
Figure 4—11 Tested in a pool in Civil Engineering lab .....	27
Figure 4—12 Tested in Small Tank-2.....	27
Figure 4—10 Tested in Small Tank-1 .....	27
Figure 4—13 Underwater Communication Package Main Workspace .....	28
Figure 4—14 Successful Slotted FAMA handshake .....	32



LIST OF TABLES

Table 2-1 WPF Framework..... 8

Table 3-1 Project Activities ..... 16

Table 3-2 Key Milestone..... 16

## ABSTRACT

This project concentrates on simplified and innovated technology to develop an efficient software package for Underwater Acoustic (UWA) communication which helps researcher to have better understanding of the behavior of underwater acoustic network, to cater for the UTP in-house research needs and to set up the relevant basic underwater acoustic communication laboratory based testbed. The existing simulation tool, particularly NS2 can give researchers some basic understanding of underwater network, and this requires certain level of knowledge in C++, TCL and most importantly understating the infrastructure of the simulation. However, researchers will find out that they are not able to simulate the real underwater environment. This project would tackle problems existed in software development by utilizing Windows Foundation Presentation Technology and Model View ViewModel architecture which is an architectural pattern mostly used in software engineering that originated from Microsoft. The author believes that this software package will enable students/ researchers to perform their studies and testing in a real lab based environment with a minimum amount of effort.



## ACKNOWLEDGMENT

Apart from the efforts poured, the success of any project depends largely on the encouragement and guidelines of many others. Author takes this opportunity to express his gratitude to the people who have been instrumental in the successful completion of this project.

Special thanks to Dr. Low Tan Jung, Ms. Foong Oi Mean, Mr. Hai SV Raingsei and Ms. Sophorn Song for the undying commitment and assistance at all times and couldn't say thank you enough for the tremendous support and help.

Also thanks to the family members for strong moral support and advices. The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their constant support and help.

# **CHAPTER 1**

## **INTRODUCTION**

This chapter will entail thoroughly about the overview of this ongoing software development. This chapter will include the following topics:

- Section 1.1 – BACKGROUND OF STUDY
- Section 1.2 – PROBLEM STATEMENT
- Section 1.3 – OBJECTIVES AND SCOPE OF STUDY
- Section 1.4 – FEASIBILITY OF THE PROJECT
- Section 1.5 – CHALLENGES
- Section 1.6 – EXPECTED OUTCOME

### **1.1 BACKGROUND OF STUDY**

The underwater acoustic (UWA) have been used since the early 20<sup>th</sup> century. The first application was to detect icebergs by using sonar waves. Apparently, the underwater acoustic network has been on research for years but in order to attain its significant roles, a lot of tools, simulation software are highly in demand in research study. The author has seen the usefulness of underwater data communication that will facilitate researchers, oceanographic data collection, environment application (pollution monitoring, chemical changes) offshore exploration, disaster prevention, navigation and tactical surveillance applications and in military purpose (unmanned underwater vehicle such as UUV, AUV and submarine vehicle).[9].

Having useful tools for UWA experiments is really advantageous. Yet the existing UWA software prototype that is developed by previous researcher has many features to be improved and implemented. The essential objective of this project is to develop a user-friendly and cost effective software package which is very much easier in term of installation and configuration. It is to improve the usability and understandability of the software package which plays a crucial part to ease many UWA studies of many researchers.

It could be a disclosure to say that after many years of researches and studies, there are not much usable software has been developed to serve this underwater acoustic



communication. However, there are also a number of software package that have been developed and yet, the user-friendliness, simplicity of implementation and usability of system is not met the desire of users and researchers.

## **1.2 PROBLEM STATEMENT**

### **1.2.1 Problem Identification**

The underwater data communication package should be easy to use, provided with simple steps of installation and fast deployment in every underwater lab. If underwater data communication software is going to be used in most of the underwater lab, it must be usable with a minimum amount of explanation and training. The problem with existing software is poor design with complicated user interface; it uses a lot of direct manipulation (a style of user-system interaction) which lead to some major problems such that the derivation of appropriate information space model can be very difficult and the interface become complex to program and make heavy demand on computer system. To provide ease of use and learnability of the software, the implementation of the software package should be straight forward, which enables users to easily alter, modify, install or re-configure some setting of the system; graphic user interface should be simple and easy to understand and use. Scholars and researchers who are to use the underwater data communication software should be able to use it with a minimum amount of effort.

In addition, the currently available software is very much expensive and very much customized for specific purpose of studies and testing. Particularly, there is no specific UWA communication software packages have been developed by UTP students/ researchers to cater for the UTP in-house research needs and to set up the relevant basic UWA communication laboratory based testbed.

## **1.3 OBJECTIVES AND SCOPE OF STUDY**

### **1.3.1 Objectives**

The objectives of this project are:

- To develop a underwater data communication software package
- To make the software scalable which is easy to re-configure and modify.



- To cater for the UTP in-house research needs
- To set up basic UWA communication laboratory based testbed.
- To improvise the learnability of the software.

### **1.3.2 Scope of Study**

This project is focusing on usability of system which is able to communicate under water, using a pair of devices connected to computer via serial communication port. Usability, by definition, is easily accessible by many different types of computer users even without knowledge of programming. This project is also to develop a well-structured algorithm with which data can be communicated stably under the water within 200meters to 2000meters. In conclusion this research project is aiming to build a user-friendly application and the underwater data communication is stabilized.

## **1.4 FEASIBILITY OF THE PROJECT**

### **1.4.1 Scope Feasibility**

This project will primarily focus on improvising and developing an underwater data communication application. There are a lot of research have been done on this field, some of which have suggest network design, some other have proposed the system architecture and algorithm for building the underwater data communication software. According to research author has found a lot of suggested algorithm which focus on data optimization and for first phase of development, author will take a deep focus on windows based application, this is because many research lab and educational organization are using windows based operating system.

### **1.4.2 Schedule Feasibility**

The development of this project will take approximately two semesters in which it will be divided into two parts. They first part of this project or FYP part I will cover the planning, requirement analysis and design phase. Other than that, the first part of the project will also involve with in-depth study of current application and the relevancy of this project.



The second part of this project will commence in the second semester whereby the output from part I will be transformed into workable codes and testing and maintenance procedure will be executed.

### **1.4.3 Technical Feasibility**

The development of the software will mostly depend on C# programming with Windows Presentation Foundation Technology (WPF). C# is a programming language which is very much similar to C/C++ programming language. Windows Presentation Foundation (WPF) is a UI framework that creates rich, interactive client applications. [6]

## **1.5 CHALLENGES**

### **1.5.1 Windows Presentation Foundation (WPF)**

WPF is a window-based graphical subsystem that is developed by Microsoft. WPF is a product under .NET Framework 3.0. WPF is very robust and powerful framework compare to the old graphical system. WPF is rendered under DirectX. The powerfulness of WPF comes from the separation between interface and business logic.

### **1.5.2 Understanding of C Sharp programming language (C#)**

C# has become a sophisticated programming language that can achieve many goals, but many programmers are left wondering what techniques to use to attain that. [C#,2008] To get the full benefit from this programming language, solid understanding of the esoteric C# features is required. As a student who has little exposure and limited understanding of C#, process of learning and developing software using C# could be challenging.

### **1.5.3 Model view view-model architecture (MVVM)**

It is another big challenge to develop software with professional-level user interface. A lot of technique, such as data, interaction design, visual design, connectivity, multithreading, security, internationalization, validation, unit testing, and a touch of voodoo will be blended together to achieve the user interface at such level. [9] It is preferable to use WPF together with MVVM which was designed to support for

commends in WPF. Because it is well suited to WPF platform, learning new software architecture and differentiating between MVVM and WPF could be challenging to author too. [5]

1.6 EXPECTED OUTCOME

At the end of this project development the author is creating software to help researchers to have better understanding of the behavior of underwater acoustic network, to cater for the UTP in-house research needs and to set up the relevant basic underwater acoustic communication (UWA) laboratory based testbed.

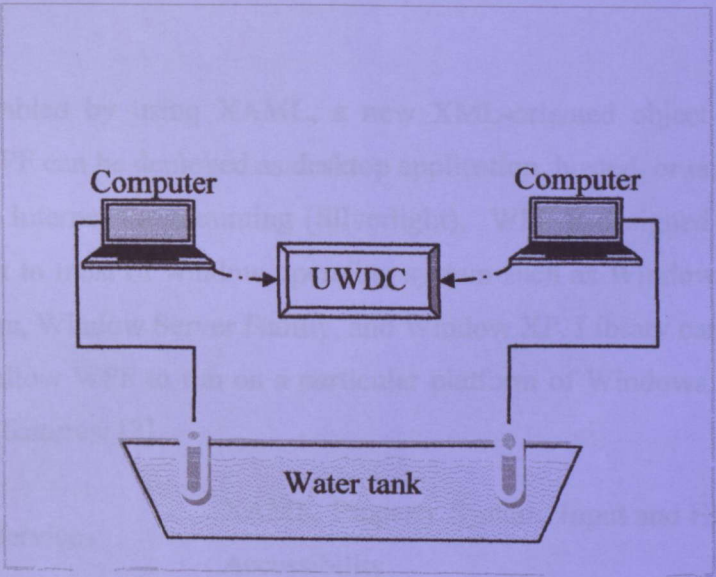


Figure 1—1: Example Setup of UWA Testbed



**CHAPTER 2**  
**LITERATURE REVIEW**

**2.1 SOFTWARE TECHNOLOGY**

**2.1.1 Window Presentation Foundation (WPF)**

WPF is a window-based graphical subsystem that is developed by Microsoft. WPF is a product under .NET Framework 3.0. WPF is very robust and powerful framework compare to the old graphical system. WPF is rendered under DirectX. The powerfulness of WPF comes from the separation between interface and business logic.

WPF is resembled by using XAML, a new XML-oriented object models. The capacity of WPF can be deployed as desktop application, hosted, or embedded object using in Rich Internet Programming (Silverlight). WPF is designed in such a way that it can suit to most of window operating system such as Window 8, Window 7, Windows Vista, Window Server Family, and Window XP. Library can be installed if necessary to allow WPF to run on a particular platform of Windows. WPF contains the following features: [3]

Base Services	XAML, Property System, Input and Eventing, Accessibility
Media services	2D,3D,Autdio, Video , Text, Imaging, Animation, Effects, Composition Engine
Document Services	XPS Documents, Open Packaging Conventions
User Interface Services	Application Services, Deployment, Controls, Layout , Data Binding

Table 2-1 WPF Framework

Unlike other GUI technology, WPF is carefully designed with the following principle:



- **Integration:** Provide a single model that is orthogonal across all services such as GDI, Direct3D or OpenGL. It allows seamless integration of content within a single application.
- **Vector Graphics:** As the name implied, it uses vector as it base design and the heart of the system. This has some advantage of scaling and resolution problem for a specific machine. There are two main graphic rendering such as Hardware and Software. They can be used interchangeably.
- **Declarative Programming:** It introduces a new language called XAML (eXtensible Application Markup Language), an XML-based language for instantiating and populating nested object hierarchies. XAML be access as object in code-behind model embodied in WPF.
- **Easy Deployment:** It offers the best of both deployment model and Web-browser application.
- **Document Lifecycle:** It provides a new set of document and print technologies. Application can use the Open Packaging Conventions.

### 2.1.2 C Sharp (C#)

C Sharp programming is a multi-paradigm programming language that is developed by Microsoft. It comes together with .NET framework. It is developed by Anders Hejlsberg for the purpose of simple, modern, general-purpose and object oriented. C# is based on Common Language Infrastructure (CLI). The virtual machine manages memory, handles object references and perform Just-In-Time (JIT) compiling of Common Intermediate Language Code. C# is derived from C++ Programming Language and it has introduced a lot of distinguish features such as:[1]

- Automatic garbage collection
- Optional Pointer features
- Reflection Capability
- Indexer
- Conditional Compilation
- Simplified multithreading



### 2.1.3 Model View View-Model Architecture

The Model View View-Model (MVVM) is an architecture pattern that is developed by Martin Fowler for using with Microsoft Application Development as a specialization of the Presentation Model Design. MVVM is used most with Microsoft development platforms such as WPF or Silverlight. The architecture designed to separate between business logic, data and application. It promotes the 3 tiers architecture layer. MVVM has the following pattern descriptions: [4]

- Model: represents data, real state content or data access layer that represents that content
- View: refers to all element displayed by GUI such as window, button, graphics, animation, video, and other controls
- View-Model: represents an abstraction of the view that also serves in data binding between the View and the Model. It can also be a converter. It contains all the commands that performed by the UI.

MVVM is very well suited with WPF developers. It can be said MVVM is the lingua franca of WPF developers. Data binding is the great and important aspect for MVVM architecture. Another aspect of MVVM is the usability of data templates and the resource system. Developers who apply this architecture will gain back the smooth of programming workflows. Therefore the development team can focus on creating robust ViewModel classes and the design team can focus on making user-friendly Views.

## 2.2 UNDERWATER NETWORK COMMUNICATION AND PHYSICS

### 2.2.1 What is Throughput?

The throughput efficiency is the ratio of delivered bit rate and the total number of transmitted bit. This is the very basic of throughput [9].

$$\text{Throughput} = \frac{ND}{NT} \quad (2.1)$$

Where: ND is the number of delivered bits;  
NT is the total number of bits sent.



### 2.2.2 Data Packet Size and Throughput Efficiency

Based on data link layer, the format of an packet size is as in Figure 2 below and assume that each data packet consists of a total of  $N$  composes of  $N_l$  data bits plus the  $N_{oh}$  data packet overhead bits ( $\alpha + \tau$ ). So the Total is:

$$N = N_l + N_{oh} \quad (2.2)$$

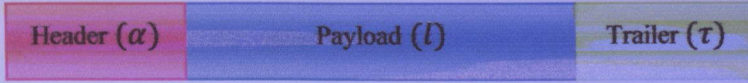


Figure 2—2 Data Packet Format

with a given a set of physical layer parameters ( $P_e, R, d$ ) where  $P_e$  is the probability of packet error,  $R$  is the bit rate, and  $d$  is the distance between transmitter and the receiver; the **throughput efficiency** can be written in the form of:

$$\eta = (1 - P_e)^{N_l + N_{oh}} \times \left( \frac{N_l}{N_l + \mu} \right) \quad (2.3)$$

Where,

$$\mu = N_{oh} + \frac{T_w R}{g} = \mu_o + \frac{2}{gc} l R \quad (2.4)$$

### 2.2.3 Energy Efficiency

In data communications, energy is consumed during transmission of the data energy expended at the transmitter) and when framing and error correction is performed. So in communication energy can be generally taken as the sum of the energy required to transmit the data and the energy required to perform encoding and decoding of the data. Therefore the Energy Efficiency can be expressed as below[9]:

$$\eta = \frac{k_1 N_l}{k_1 (N_l + N_{oh} + \tau) + k_2 + E_{dec}} (1 - \text{PER}) \quad (2.5)$$

Where  $(1 - \text{PER})$  is the packet acceptance rate i.e. the data reliability rate.



2.2.4 Bit Error Rate (BER)

One of the changes that digital communications systems has brought to wireless transmission is the need for good end-to-end performance which is usually quantified by the bit error rate (BER)[9]. It quantifies the reliability of the entire radio system. BER starts off as a simple concept with a definition of,

BER = NE/NT (2.6)

Where NE is the number of error bits and NT is the total number of bits sent. BER is considered insignificant if a strong signal can be relayed through an unperturbed communication link.

2.3 The existing port manager

Since SAM-1 Senor Network (Sensing Acoustic Modem) is using COM Port, the port manager application is becoming in demand. Much progress has been made toward setting up and configures the sensor node (SAM-1). This COM port manager is called WinSSD whose screen shot is going to present in a short following session.

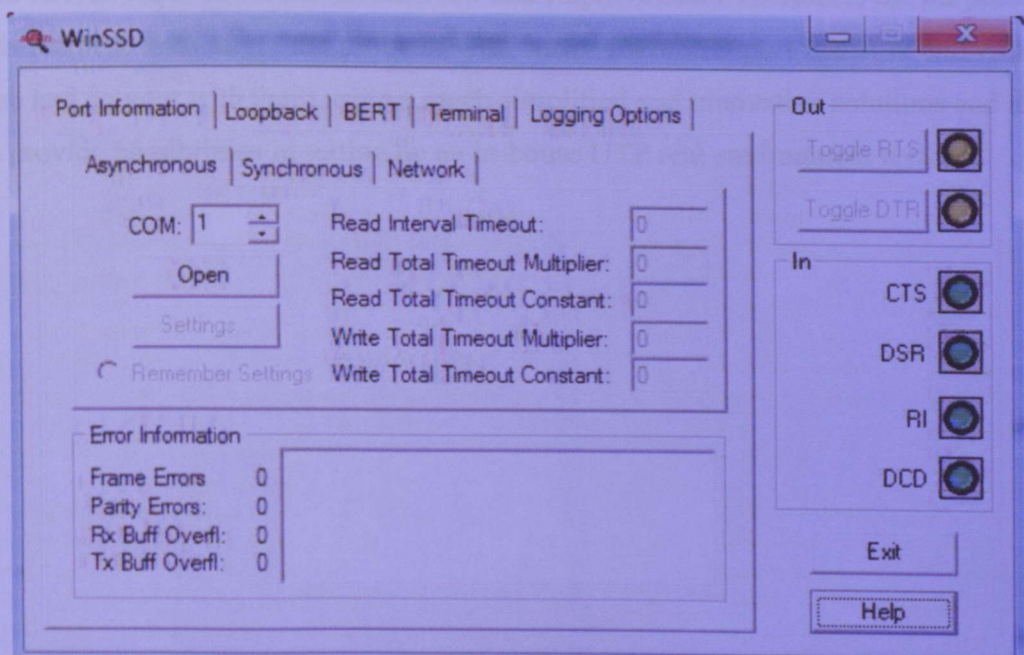


Figure 2—3 WinSSD, Port Manager

The limitation of winSSD is that everything is done from command line and it only is able to sent data as a text string, there is no feature for sending a file to another end of next work. Sending and receiving itself is rather too simple, without protocol. It would push and send all the packet which are in sending queue.

## 2.4 The existing testbed

Aqua-Lab is configurable lat testbed for underwater acoustic sensor networks. Aqua-Lab contains a real physical environment, a set of communication hardware, a programming library and an emulator. In short, Aqua-Lab is a flexible testbed. It contains real acoustic communication channels, it provides an environment closer to reality that that by simulators. However Aqua-lab has yet to completed and had more room for improvement, such as implement and test more protocol such as MAC, reliable transfer protocols.

As seen above, most of the underwater communication and testbed are from United Stated and used for their own beneficial purpose and yet they have yet come to be implemented in Malaysia. The limited underwater communication software and underwater testbed have shown that the current existing underwater communication and testbed required further amendment and improvement. Therefore, the purpose of this research is to target the problem with current WinSSD port manager and Aqua-Lab and counter with them using a much simplified and innovative solutions and also to provide possibilities in setting up an in-house UTP real environment testbed.



## CHAPTER 3

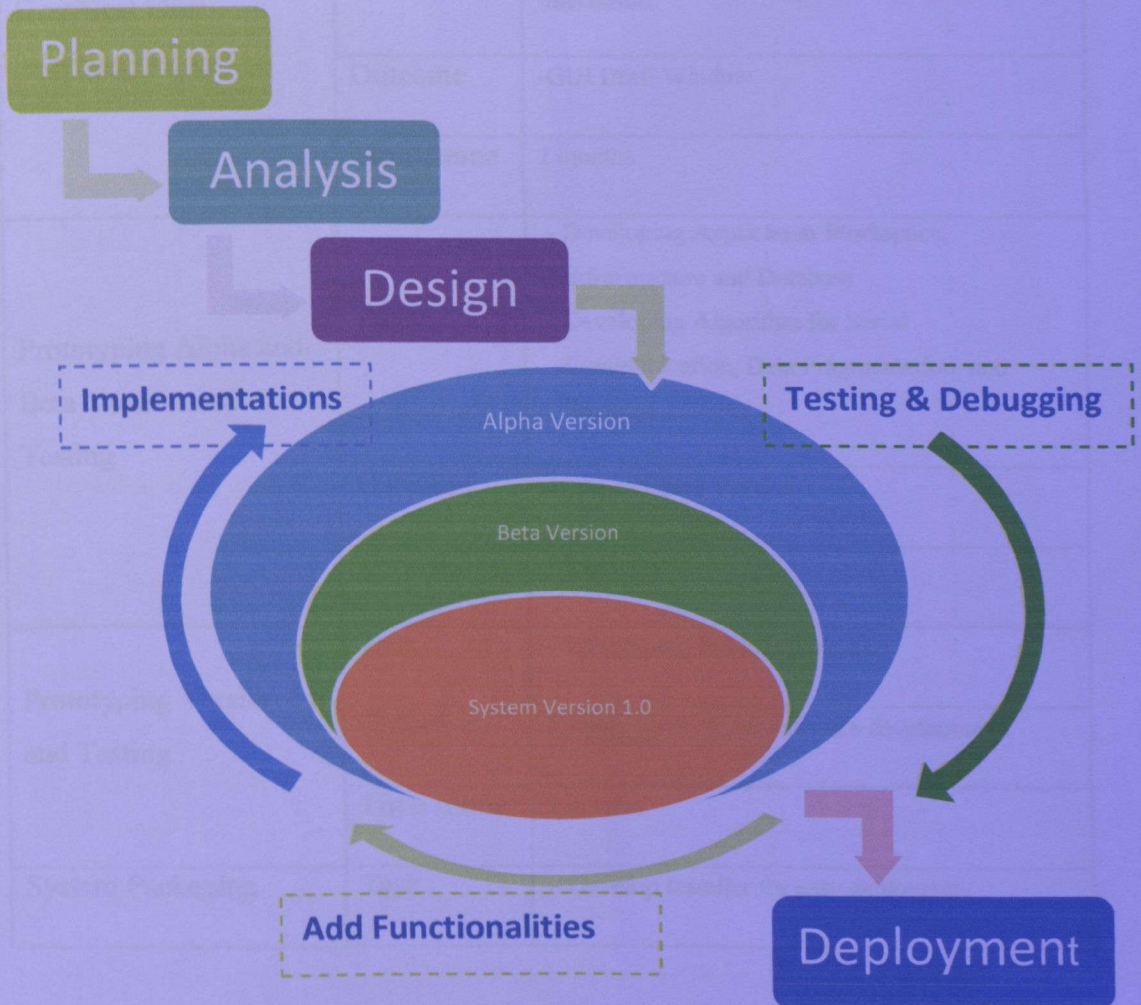
### METHODOLOGY OF STUDY

#### 3.1 RESEARCH METHODOLOGY

Methodology is crucial part of a software development as it will determine the success of a technology solution. [7] With a better methodology, there will be fewer defects or bugs in software development, it results faster delivery time with better price.

In this project, author will use a combination of *Waterfall* and *Prototyping*. The reason is because development it is very flexible as the System keeps iterating as its functionalities keep improving. Each of the iteration, the system is well tested and debugged which help to minimize the errors and save more time.

Below diagram is the overall methodology used in this project.





### 3.2 PROJECT ACTIVITIES

System Planning and Analysis	Task	-Gathering information do develop system model by doing system analysis design
	Outcome	-Developing use-case diagram, sequential diagram, interaction diagram, and other system analysis components
	Time frame	2 months
System Design	Task	-Develop Software components using WPF technology and MVVM architecture.
	Outcome	- UML diagram
	Time frame	4-6 months.
Graphical User Interface Design	Task	-Developing Friendly and interactive graphical user interface using concept of Human Computer Interaction.
	Outcome	-GUI Draft Window
	Time frame	2 months
Prototyping Alpha and Beta Version and Testing	Task	- Developing Application Workspace, Infrastructure and Database - Developing Algorithm for Serial Communication, Data Fragmentation, and Protocol
	Outcome	Alpha and Beta Version
	Time frame	1-2 months
Prototyping Version 1.0 and Testing	Task	- Developing full functionalities
	Outcome	A ready platform for software development.
	Time frame	2 months
System Packaging	Task	- Develop Installer for easy deployment



	Outcome	- Application Installers
	Time frame	1-2 days
Documentation and Dissertation Completion	Task	- Complete final documentation and dissertation
	Outcome	- Documentation and Dissertation
	Time frame	1-2 months
System Deployment (Delivery)	Task	Delivery complete system with documentation
	Outcome	Report and Project submission
	Time frame	1-2 months

Table 3-1 Project activities

### 3.3 KEY MILESTONE

Milestone	Expected Date Delivery
1. Completion of GUI Drafts	Week 11
2. Completion of Project infrastructure and architecture and data flow	Week 9
3. Completion testing and Alpha Version Delivery	Week 11
4. Completion testing and Beta Version Delivery	Week 13
5. Complete Final Testing	Week 14
6. Delivery of Final Version	Week 14
7. Delivery of System Package	Week 14

Table 3-2 Key milestone

### 3.4 PROGRAMMING ARCHITECTURE

Technical detail of programming architecture is attached at the Appendix A.



## 3.5 TOOL

### 3.5.1 IDE (Integrated Development Environment)

To develop WPF application, *Microsoft Visual Studio 2010* is used. This IDE is very powerful provides comprehensive tools that help to develop faster and debug effectively. It also comes with deployment tool that allow developer to pack all the necessary files and libraries and create fast and easy installer. Moreover, the new improvement feature of intelligence help programmer to code very fast and find out all of the error faster. There are many extra tools such as flowchart diagram, use case diagram, database diagram that help developer throughout the entire application lifecycle.

### 3.5.2 SQLite Database Tool

Application use SQLite as it backbone database. It store data for application to use include calculation data, application data (properties) and history. SQLite is suite for desktop application. To view the database, **SQLite Database Browser** is also used to explore data in the database.

### 3.5.3 Microsoft Express

This is another product from Microsoft that help developer to develop complex GUI. It is like WYSIWYG (What You See Is What You Get Tool) that allows developer to develop fast WPF GUI. Developer just copy and paste the XAML into MV2010. Developers can also develop their own template and create theme for the whole application as well.

### 3.5.4 Photoshop CS5

This is part of development design. This tool is used for image editor and color blending in order to have a good GUI design.



## CHAPTER 4

### RESULT AND DISCUSSION

At this point of time, the progress of this project has reached Deployment which repackaging developed software with modem's drivers software and creating one installer file. Previous phase – Design phase – which is developing application workspace and commands required by MVVM architecture, developing application's infrastructure, developing application's database (SQL Lite) and developing algorithm (serial communication, data fragmentation, protocol). Previous phase which are Planning and Analysis have been completed thoroughly. The project is gradually proceeding to next phase which is Prototyping and Testing. This is the phase where the prototype is being developed and testing& debugging process are being carried on until an acceptable prototype is finally achieved. The result of the progress of this project will be discussed in the following sections:

#### 4.1 RESEARCH AND UNDERSTANDING OF WPF AND MVVM ARCHITECTURE

##### 4.1.1 WPF Technology

WPF application is starting up in XAML format and accompanied by its companion implementation file with extension cs. The application definition is all in the XAML file as shown in figure below:

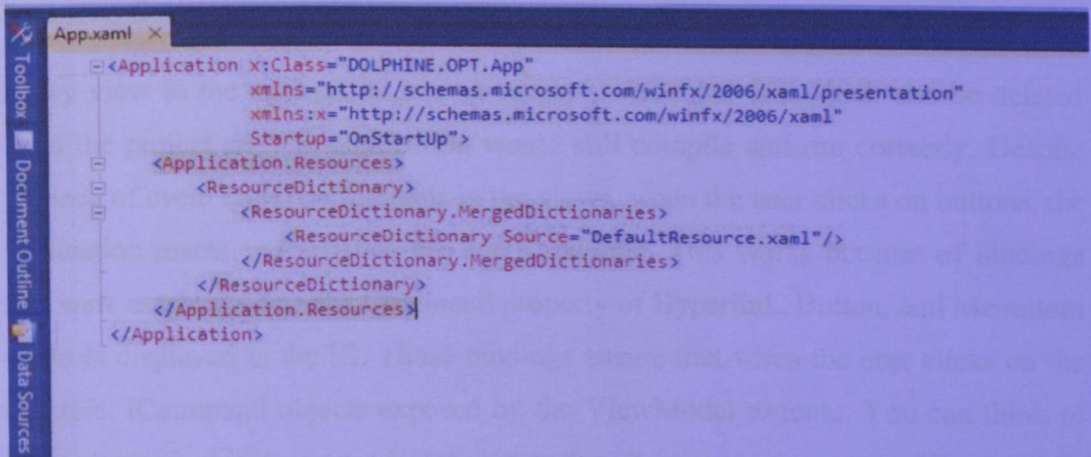


Figure 4—1 Defining an application in XAML



Defining an application, initializing and launching code in Main() could be very much heavier as compared to XAML –based application definition because it is simpler to change a definition instead of changing code. The three main reasons for using XAML –based are declarative programming, Windows Vista, and navigation hosting. [12]

WPF addresses all concerns with respect to user interface, multimedia and document lifecycle. Most developers might think of windows form application to implement of the windows UI, Adobe Flash to tackle multimedia aspect and using PDF to handle the documentation works. Instead WPF could handle them all in a single, unified base technology.

Developing software with a professional graphic user interface is always a challenge to many developers because it could be a very heavy mixture of data, interaction design, visual design, connectivity, multithreading, security, internationalization, validation, unit testing, and a touch of voodoo. With this volatile situation, WPF achieves its potential features in software development.

#### **4.1.2 MVVM Architecture**

It is difficult to distinguish between WPF and MVVM once developers become very much familiar with them. MVVM is well suited to the WPF platform which is designed to make it easy to develop applications using MVVM pattern.

##### **Relaying Command Logic**

Every view in the application has an empty codebehind file which can be deleted from the project and the application would still compile and run correctly. Despite the lack of event handling methods in the views, when the user clicks on buttons, the application reacts and satisfies the user's requests. This works because of bindings that were established on the Command property of Hyperlink, Button, and MenuItem controls displayed in the UI. Those bindings ensure that when the user clicks on the controls, ICommand objects exposed by the ViewModel execute. You can think of the command object as an adapter that makes it easy to consume a ViewModel's functionality from a view declared in XAML. [11]



When a ViewModel exposes an instance property of type I-Command, the command object typically uses that ViewModel object to get its job done. One possible implementation pattern is to create a private nested class within the ViewModel class, so that the command has access to private members of its containing ViewModel and does not pollute the namespace. That nested class implements the ICommand interface, and a reference to the containing ViewModel object is injected into its constructor. However, creating a nested class that implements ICommand for each command exposed by a ViewModel can bloat the size of the ViewModel class. More code means a greater potential for bugs. [11]

```
1  public class RelayCommand : ICommand
2  {
3      #region Fields
4
5      readonly Action<object> _execute;
6      readonly Predicate<object> _canExecute;
7
8      #endregion // Fields
9
10     #region Constructors
11
12     public RelayCommand(Action<object> execute)
13     : this(execute, null)
14     {
15     }
16
17     public RelayCommand(Action<object> execute, Predicate<object> canExecute)
18     {
19         if (execute == null)
20             throw new ArgumentNullException("execute");
21
22         _execute = execute;
23         _canExecute = canExecute;
24     }
25     #endregion // Constructors
26
27     #region ICommand Members
28
29     [DebuggerStepThrough]
30     public bool CanExecute(object parameter)
31     {
32         return _canExecute == null ? true : _canExecute(parameter);
33     }
34
35     public event EventHandler CanExecuteChanged
36     {
```

Figure 4—2 RelayCommand Class

ViewModel Class Hierarchy

ViewModel base class, at least, is created once or twice in a whole circle of an application development because most the ViewModel classes need the same features and also often need to implement the *INotifyPropertyChanged* interface. Other ViewModel classes can just inherit all of the common functionality from this base class. [11]

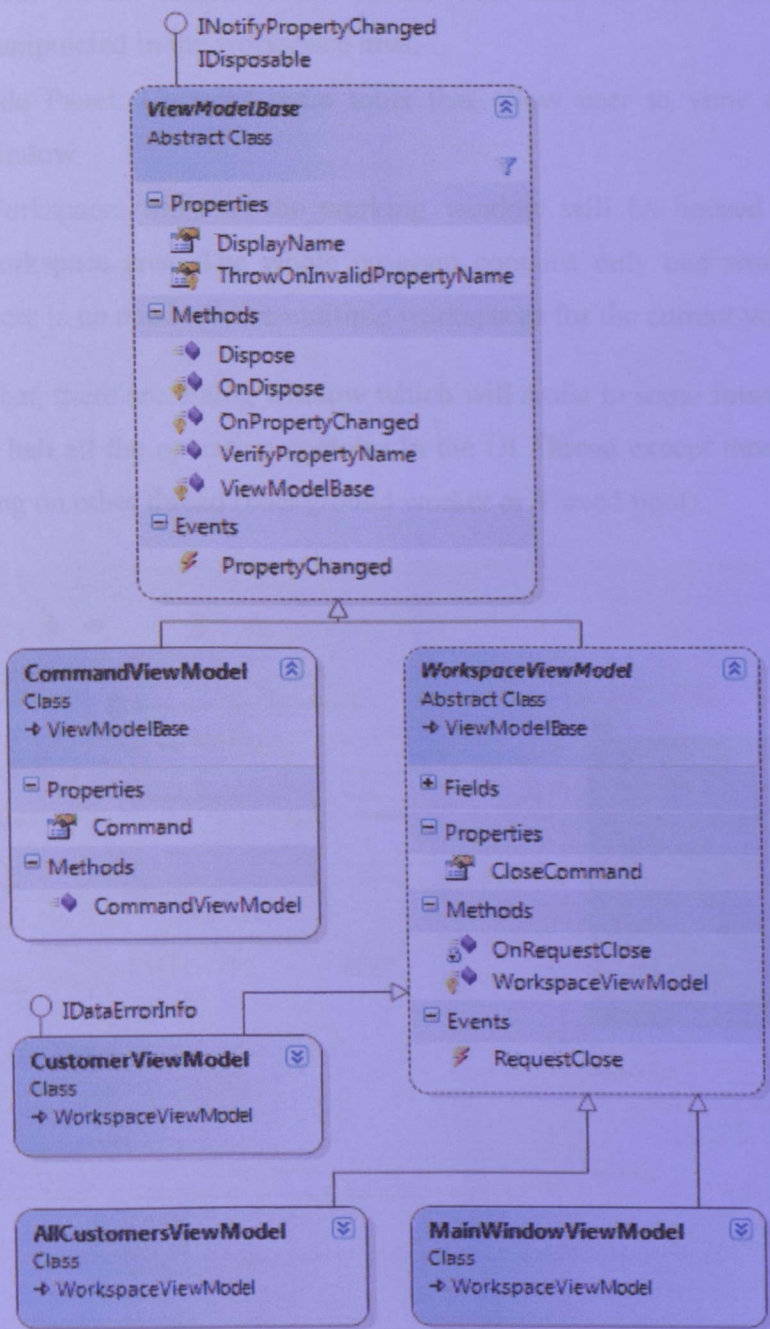


Figure 4—3 ViewModel Class hierarchy



## 4.2 DEVELOP APPLICATION’S WORKSPACE AND COMMANDS REQUIRED BY MVVM ARCHITECTURE

### 4.2.1 Application’s Workspace

#### Application’s Structure

Application that is being developed is divided into three areas which are:

- Main Menu: Contains all menus and functions that needed to be manipulated in the workspace area.
- Side Panel: Contains extra tools that allow user to view a particular window
- Workspace: Most of the working window will be housed inside this workspace area. The whole program contains only one workspace and there is no need to have multiple workspaces for the current version.

Aside from that, there are dialog window which will assist in some function. Dialog window will halt all the operation contains in the UI Thread except those processes who is running on other thread (background worker or Thread pool).

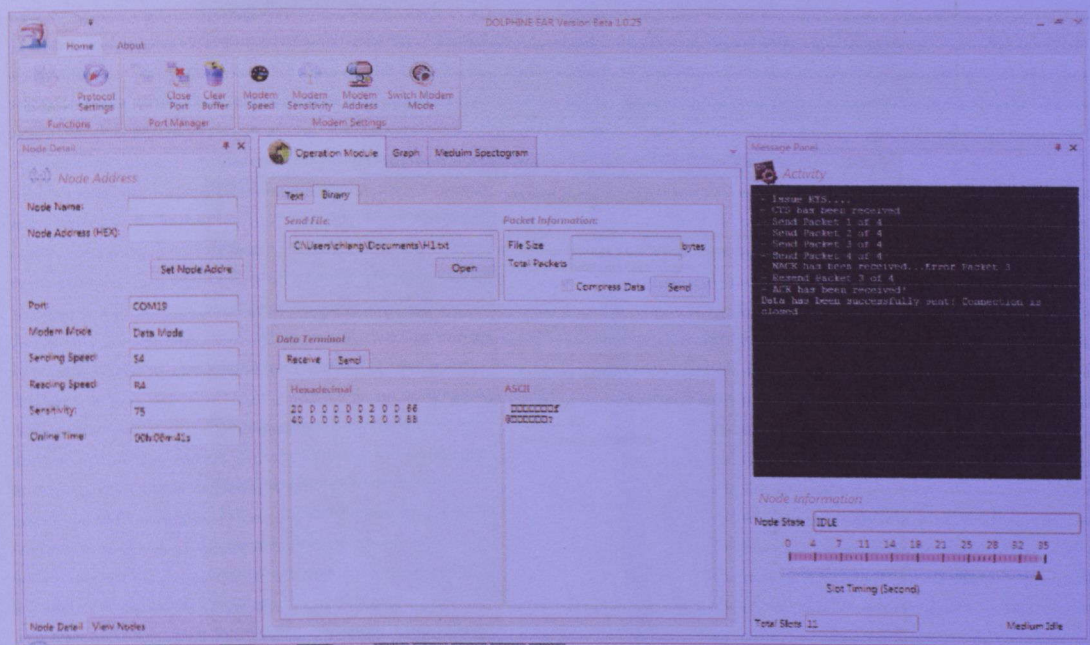


Figure 4—4 Main workspace of Underwater Communication Package



## 4.2.2 Application's Implementation

### Hardware Configuration Wizard

The purpose of having this wizard is to allow the user to perform on shot configuration. The configuration will be saved once the user has performed the first configuration. The configuration also can be redo when user wishes to perform or when there is some changes in the hardware connectivity.

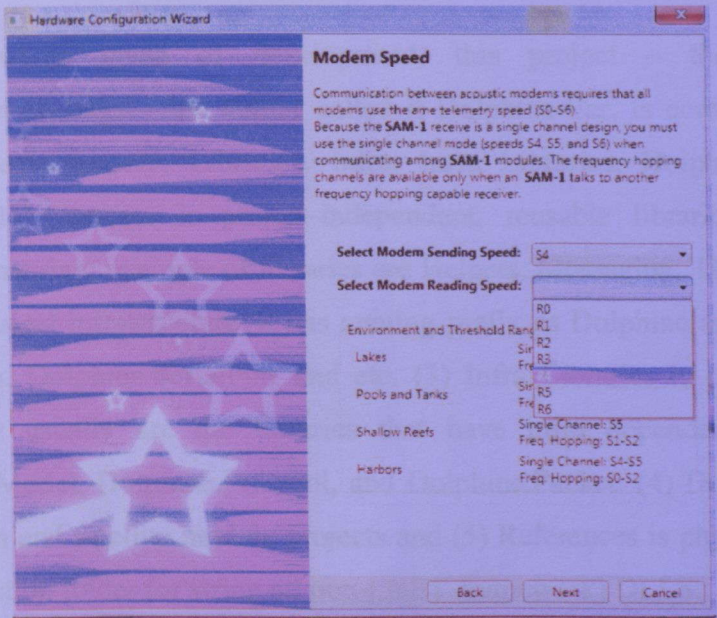


Figure 4—5 Wizard Windows for Setting Up the device

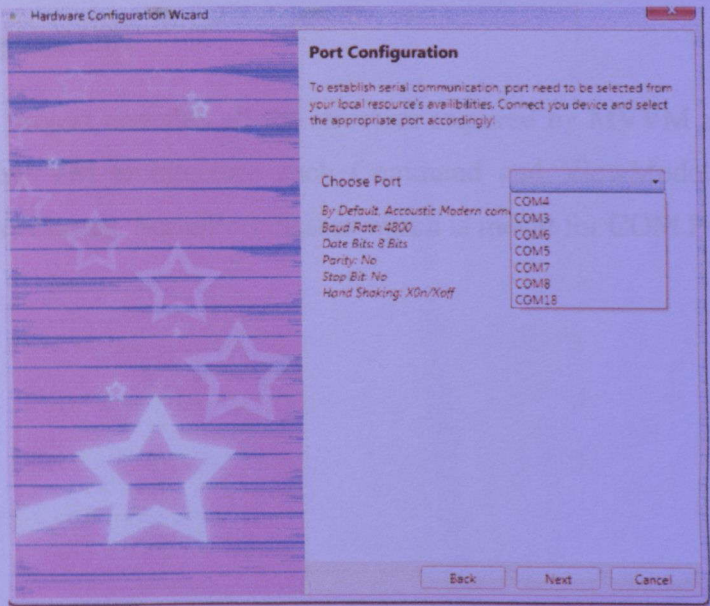


Figure 4—6 Second Screen of the wizard



### 4.3 DEVELOPMENT AND SYSTEM ARCHITECTURE

System architecture gives a large picture of the overall system; it is really helpful for future development and for giving developers who are going to maintain the project or to enhance project, a better understanding of how the system was being developed. In this chapter, detail of system architecture will be discussed.

#### 4.3.1 Project description

During the whole cycle of development, this project – the underwater communication package – the author called it Dolphine-Ear, is comprised of five main components namely, Libraries, Infra, Infra Libraries, Dolphine.OPT, and References. (1) Libraries is project-independent, reusable libraries, it has no restriction on naming; example of libraries are Helpers, Messaging, Threading class. (2) Infra is project infrastructure, it has naming prefix as Dolphine as for example: Dolphine.Infra, Dolphine.SerialPort and etc. (3) Infra Libraries is project-specific libraries, they usually are the libraries that have infra dependencies such as Dolphine.OSIModel, Dolphine.Protocol, and Dolphine.Packet. (4) Dolphine.OPT is the application and satellite module projects and (5) References is physical folder in Dolphine Solution folder for string all non-(.NET-framework) DLLS.

The project contains five sub-projects whose functionalities are given as below:

- Dolphine.Infra

Dolphine.Infra project contains all the structure required by MVVM as discussed in section 4.1.2 MVVM architecture such Command and ViewModel base. It also contains Helpers classes, SerialPort classes which is meant for COM Port manager.

- Dolphine.OSIModel

Dolphine.OSIModel contains interface class for implementing the seven layer classes of OSI Layers which are Physical Layer, Data link layer, Network layer, Transport Layer, Session layer, Presentation Layer and Application layer [14]. In this project only five layers are implemented namely Data link layer, Network layer, Physical layer, Presentation later and Transport layer. Dolphine.OSIModel also handles messages passing from one layer to another.

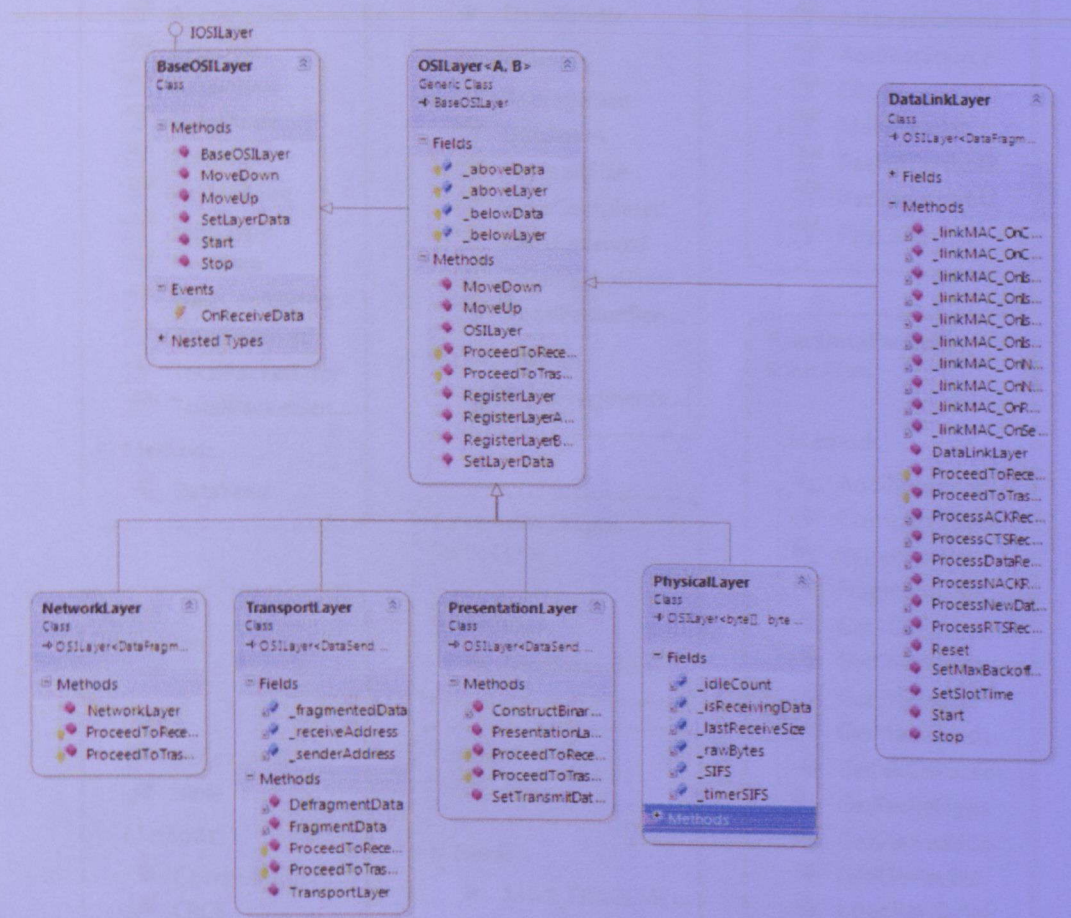


Figure 4—7 Dolphine.OSIModel class diagram



- Dolphine.Packet

Dolphine.Packet is responsible for packet sending and receiving, data fragmentation and as well as for performing Cyclic Redundancy Checksum.

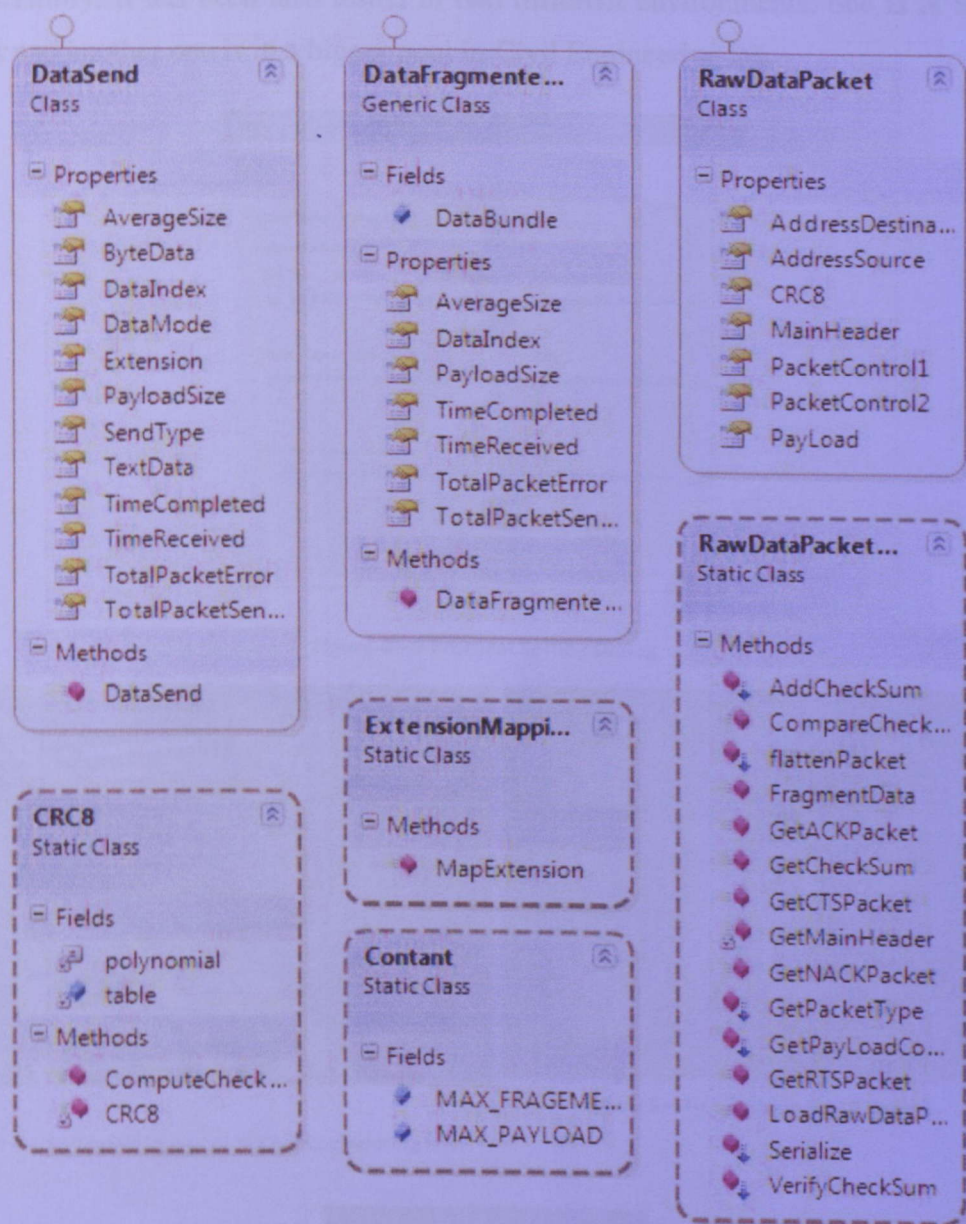


Figure 4—8 Dolphine.Packet Class Diagram



- Dolphine.Protocol

Dolphine.Protocol is where network protocols are supposed to be implemented at. Particularly in this project, Slotted FAMA has been selected and implemented successfully. It has been also tested in two different environments, one is in Small Tank and another one is in a bigger pool in Civil Engineering lab.

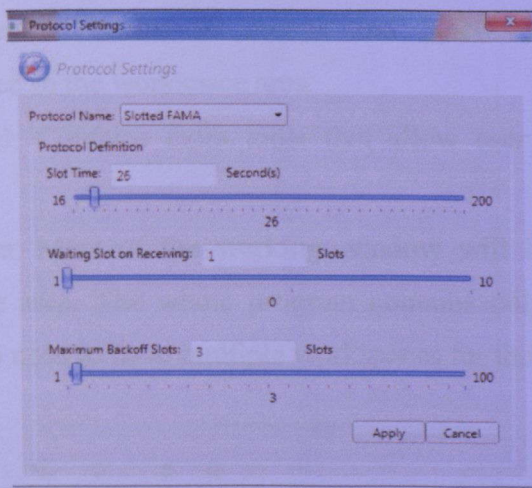


Figure 4—9 Protocol Setting Dialog

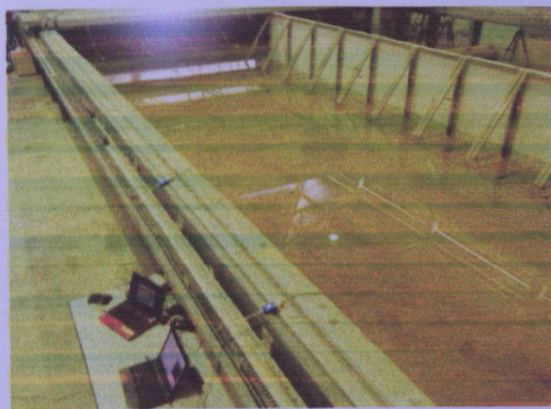


Figure 4—11 Tested in a pool in Civil Engineering lab

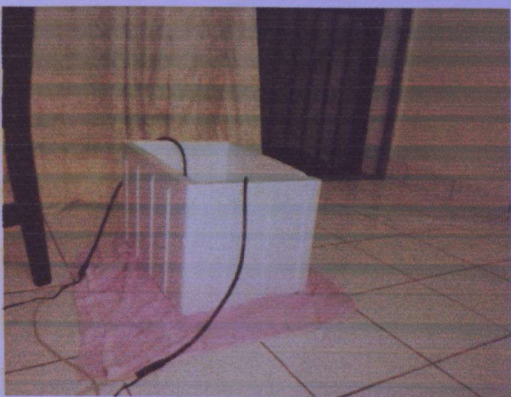


Figure 4—10 Tested in Small Tank-1

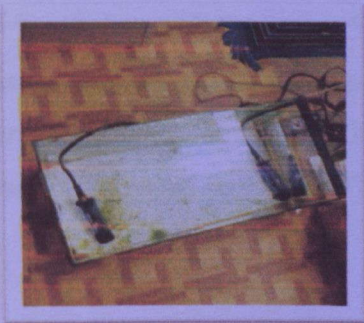


Figure 4—12 Tested in Small Tank-2



- Dolphine.OPT

Dolphine.OPT is main application and core module for User Interface. User input and interaction are all handled in this sub project. Most of User Interface are seen on the underwater communication package are found in this. The main workspace for instant is divided into three areas which are:

- Main Menu: Contains all menus and functions that needed to be manipulated in the workspace area.
- Side Panel: Contains extra tools that allow user to view a particular window
- Workspace: Most of the working window will be housed inside this workspace area. The whole program contains only one workspace and there is no need to have multiple workspaces for the current version.

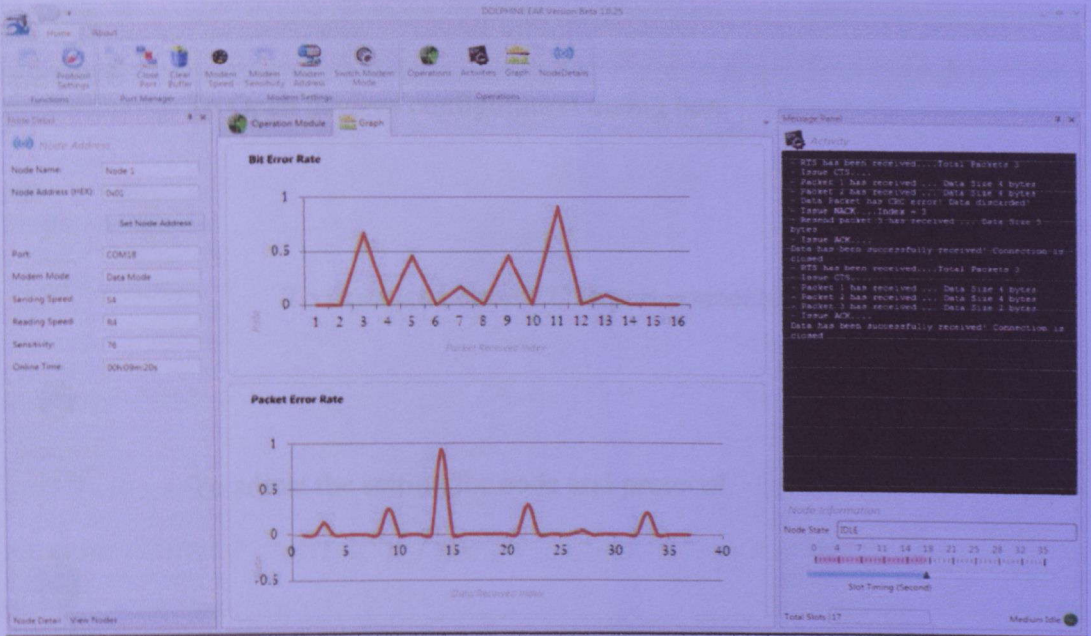


Figure 4—13 Underwater Communication Package Main Workspace

Aside from that, there are dialog window which will assist in some function. Dialog window will halt all the operation contains in the UI Thread except those processes who is running on other thread (background worker or Thread pool).

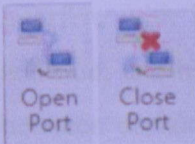


### 4.3.2 Deliverable underwater communication package

The underwater communication package has improved on the existing port manager application by giving user abilities to send files from one end to another based on the chosen protocol, slotted FAMA (Floor Acquisition Multiple Access). Below are some explanations of tool added in underwater communication package and the demonstrated steps needed for sending a file. Modem Sensitivity is sensitivity of the receive modem is adjusted by setting the detection threshold. A low detection threshold makes the modem very useful. This may increase the maximum data exchanged range. Modem speed required, for communication between acoustic modems, that all modems use the same telemetry speed (S0-S6), the most stable protocol tested is S4 (Sending Speed), T4 (Received fore 2pm) and below here are the example use of the underwater communication system.



To Open devices configuration dialog box



To Open and close COM port connected to SAM-1 node



To adjust the setting for node and protocol.



To empty sending queue in SAM-1 node



To Setup name and address for node



To open the Modem Sensitivity setting dialog



#### 4.4 Protocol implementation

Underwater acoustic (UWA) Network consists of a number of sensor nodes that communicate with each other using acoustic signal. These networks have many characteristics that make them different from packet radio networks. Low speed of sound causes long propagation delay and extremely limited low bandwidth in acoustic signal apparently distinguish from packet radio networks. In UWA networks the power management is very crucial because the battery is limited and they could be easily replaced, recharged or changes when depleted. Since the first ALOHA protocol has been proposed, many MAC protocol have been proposed as well. The main purpose of these protocols is to prevent simultaneous transmission that leads to collision between two or more stations transmitting at the same time. Stations are required to "listen" to the channel before starting to transmit to avoid collisions with other ongoing transmission. When the packet duration is bigger as compared to propagation delays in a fully connected network, CSMA comes to its most efficient. In contract to that, when the delay increases, the efficiency is exponentially lost. There is situation where one station cannot sense one or more nodes that can interfere with its transmission. This situation is called "hidden terminal". [13] In addition, another problem arises in ad-hoc networks due to the lack of connectivity between certain nodes is called "exposed terminal". These two problems are very likely to occur in UWA network. To overcome these problems FAMA protocol [13] has been introduced. However, as discussed in [13], using the original FAMA protocol in the UWA networks would not be efficient due to the required length of RTS and CTS packet. In order to tackle this problem, slotted FAMA protocol should be used. In this slotted FAMA, the time is slotted to eliminate the asynchronous nature of the protocols and RTS (Request-to-send), CTS (Clear-to-send), DATA or ACK has to be transmitted at the beginning of one slot. The slot length itself has to be determined in such a way that there will be less or zero collision.

#### 4.5 Algorithm definition

When a node wants to send a packet it waits until the next slot and transmits an RTS packet. This packet is received by the destination node and all the terminals in the neighborhood of the source node within the slot time.



Every node sends the packet at the beginning of each time slot. RTS packet is generated and sent at the beginning of the time slot when a node wants to establish a connection to its neighborhood and sends packets. The neighbors nodes receive RTS packet within the slot time and send a CTS packet at the beginning of the next slot. The source node and all the terminal also receive this CTS packet with slot time. After receiving the CTS, all the terminal understands there will be a connection established and it will set itself to backoff mode to avoid collision. At the beginning of next slot, the source node starts sending the data packet.

When the receiver has the entire data packet it sends an ACK packet to indicate that the transmission has been successful. As the FAMA protocol, slotted FAMA is based on carrier sensing. This means that terminals are constantly listening to the channel. Terminals stay in Idle state until they sense the carrier in the channel or until they have a packet ready to transmit. If a packet is ready to be transmitted at the beginning of a slot and no carrier has been detected, terminal sends an RTS and waits two slots (current slot and the next one) to receive a CTS packet. If no CTS is received during this time, a collision is assumed and the terminal goes to Backoff state for a random number of slots. After that, the RTS packet is re-sent if no carrier has been sensed during the backoff time. When CTS is successfully received, the terminal will start sending the data packet in the next slot. [13]

When a terminal detects carrier on the channel it goes to a Receiving state while it is receiving the packet. The type of packet received will determine the receiver actions as listed below.

- After receiving an RTS packet intended for another station (xRTS packet) the terminal must wait two slots (long enough for the receiver to send a CTS and the sender to start transmitting data). If after this time no carrier is sensed, the terminal returns to the Idle state. This two slot wait is necessary because of the ARQ protocol. [13]
- After receiving a CTS packet intended for another station (xCTS packet) a terminal must wait long enough to allow the other station to transmit the entire data packet and receive the corresponding ACK. Since the terminal has received the CTS packet, it will also receive the ACK packet and will thus know that data transmission has ended successfully. [13]



- After receiving a Data packet intended for another station (xDATA packet) a terminal must wait long enough to allow the reception of the subsequent ACK or NACK packet. Since it is possible that the terminal cannot hear the ACK or NACK packet, it must wait an additional slot to detect whether the data packet has been re-sent (meaning that a NACK was sent) or not. [13]
- After hearing an ACK packet intended for another station (xACK packet) a terminal only has to wait until the end of the slot since the data transmission has successfully ended. [13]
- After hearing a NACK packet intended for another station (xNACK packet) a terminal must wait long enough to allow for a complete data packet to be transmitted and a new ACK or NACK to be sent (the same time as if it had received an xCTS packet). [13]
- If a terminal senses interference in the channel, a collision is assumed. Since it doesn't know which packets have collided, the worst assumption is made, and it acts as if it had received an xCTS packet which is the situation that requires a longer wait. [13]

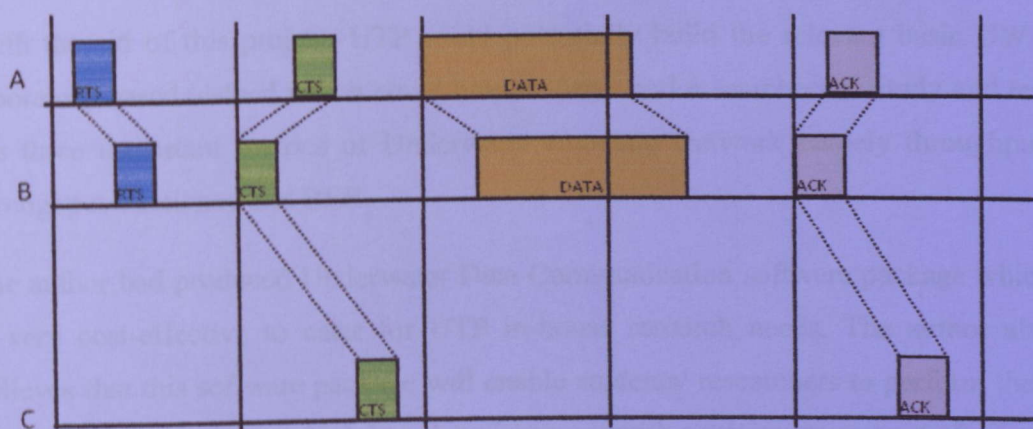


Figure 4—14 Successful Slotted FAMA handshake

In this project, the author will implement this slotted FAMA protocol in the software package by creating an independent class which are meant to be easily integrated with the software package with little amount of afford. By praising this OOP (Object-Oriented Programming), future works which are aiming to attach more proposed protocol to this software package would be pretty much easier and not tedious to understand and implement it.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATION**

#### **5.1 CONCLUSION**

In conclusion, this project is able to serve its objective presented in the earlier section of this paper. To cater for the UTP in-house research needs and to set up the relevant basic underwater acoustic communication laboratory based testbed. The application is successfully developed using Microsoft Visual Studio as Integrated Development Environment (IDE). At the end of the project, basic and simple underwater testbed is set up; with aid of software UWDCP researchers and students could have more opportunity to study their underwater network and test, implement their network routing protocol in actual network environment.

This project's results and outcomes have proven to be both relevant and beneficial to the society. This application will act as an essential tool in learning process especially in learning and researching underwater network.

With the aid of this project, UTP could potentially build the relevant basic UWA laboratory based testbed which could help students and researchers to study and test the three important metrics of Underwater Acoustic Network namely throughput, throughput efficiency and BER.

The author had produced Underwater Data Communication software package which is very cost-effective to cater for UTP in-house research needs. The author also believes that this software package will enable students/ researchers to perform their studies and testing in a real lab based environment with a minimum amount of effort.



## 5.2 RECOMMENDATION

Underwater Acoustic (UWA) Network has become a high interest research area for many researchers, hence many protocols and improvised protocols have been proposed in order to optimized the packet sending in UWA network. Particularly, in this project work, author has implemented slotted FAMA –improvised version of FAMA protocol – for in-house UTP researchers to test the efficiency of the protocol itself in a real lab environment. However, contribution to integrate more protocols to this project is mostly encouraged; for in-house UTP researchers can test more protocols and compare with one another. With that, this project will become more useful and more helpful to in-house UTP underwater researchers and Students. To make this application little more useful, sending and receiving files features have been embedded in this software too, moreover author believes that more contribution would generate an excellent result, therefore when more protocols have been implemented and well tested, it would be easier to choose the suitable and most stable protocol for underwater communication. The ultimate purpose of this project is to help UTP research center to set up underwater testbed. Author also has ambitious to bring this software to become a real medium of communication for underwater after having all the protocol implemented and tested. With this, author strongly hopes that this project would give potential developers a good kickstart to bring about the testbed and more advance underwater communication.

10. Kail, C.H. *Cost Effective Digital Speech Systems*. Pearson, Upper, Elementary Technology PICTORAS, 2011.

11. NVVM Architecture. Retrieved November 22, 2013, from

<http://nvidia.com/page/nvvm-architecture.html>

12. Arlos F., Marc D. 2009. *BPFF in Action*. New York, Wiley, 2009, Manning Publications, Greenwich, CT.

13. Maysil Muthu and Milica Stojanovic. *Slotted FAMA: A Self-Adapted Protocol for Underwater Acoustic Network*.

14. UML Model. Retrieved January 21, 2014 from

[http://en.wikipedia.org/wiki/UML\\_model](http://en.wikipedia.org/wiki/UML_model)

15. Zheng Peng, Jun-Peng Guo and Bing Wang. *UTN: Underwater Network Compiler Design, Implementation and Advancement*.

## REFERENCE

1. C# Programming, Retrieved October 30, 2011, from [http://en.wikipedia.org/wiki/C\\_Sharp\\_\(programming\\_language\)](http://en.wikipedia.org/wiki/C_Sharp_(programming_language))
2. C# Programming, Retrieve October 30, 2011, from [http://en.wikibooks.org/wiki/C\\_Sharp\\_Programming](http://en.wikibooks.org/wiki/C_Sharp_Programming)
3. WPF Apps With The Model-View-ViewModel Design Pattern, Retrieved October 30, 2011, from <http://msdn.microsoft.com/ens/magazine/dd419663.aspx>
4. MVVM Foundation, , Retrieved October 30, 2011, from <http://mvvmfoundation.codeplex.com/>
5. WPF Architecture, Retrieved October 30, 2011, from <http://msdn.microsoft.com/en-us/library/ms750441.aspx>
6. Visual C#, Retrieved October 30, 2011, from <http://msdn.microsoft.com/en-us/library/kx37x362%28v=VS.80%29.aspx>
7. Development Methodology, Retrieved October 23, 2011, from <http://exelanz.com/why-cloud/development-methodology>
8. Development Methodology, Retrieved October 23, 2011, from <http://exelanz.com/why-cloud/development-methodology/>
9. Raingsei, H. Underwater Packet Size Optimization Based on Throughput. *Final Year Dissertation*, University Technology PETRONAS, 2011
10. Koh, C.H. Cost Effective Digital Receipt System. *Interim Report*, University Technology PETRONAS, 2011
11. MVVM Architecture, Retrieved November 29, 2011, from <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>
12. Arlen F., Maxx D. 2009, *WPF In Action With Visual Studio 2008*, Manning Publications, Greenwich, CT
13. Marçal Molins and Milica Stojanovic *Slotted FAMA: a MAC Protocol for underwater acoustic network*
14. OSI Model, Retrieved January 21, 2012, from [http://en.wikipedia.org/wiki/OSI\\_model](http://en.wikipedia.org/wiki/OSI_model)
15. Zheng Peng, Jun-Hong Cui and Bing Wang *An Underwater Network Testbed: Design, Implementation and Measurement*.

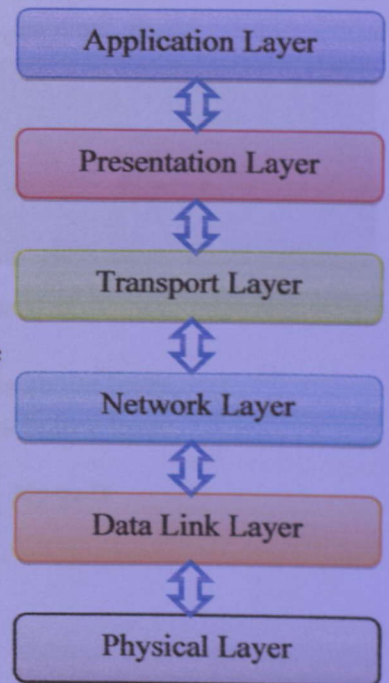


## APPENDIX A

### OSI MODEL ADAPTATION

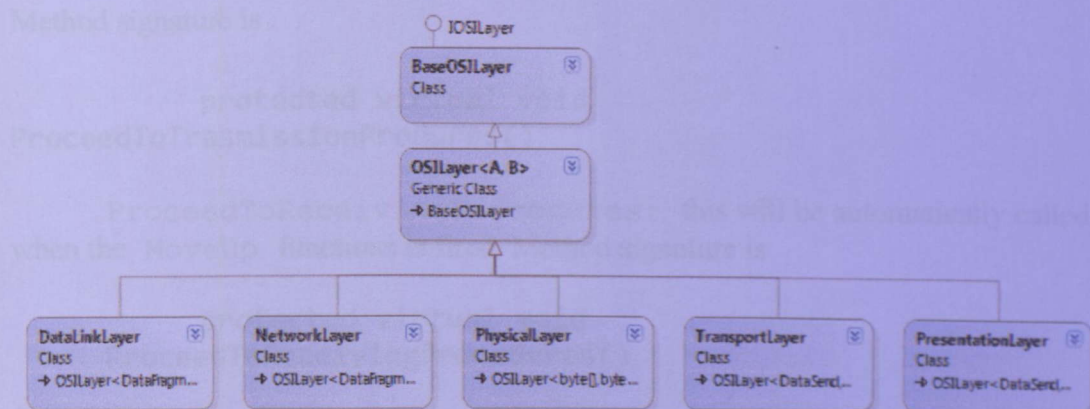
**DOLPHINE OPT** adapts only 6 layers of standard OSI model. The six layers are Application Layer, Presentation Layer, Transport Layer, Network Layer, Data Link Layer and Physical Layer. Among those layers, only Network Layer remains inactive. This means that it only allows data to flow in and flow out without altering any information. Diagram bellows describe the relationship of the six layers. Each of layers contains different data type and different functionalities:

- *Application Layer*: is the top most level layer in the communication design. It contains various operations and act as the main interface between user and application.
- *Presentation Layer*: contain information about data that is being sent out or receive in. The main functionalities of this layer are to manage the resource in and out in the system.
- *Transport Layer*: perform various data fragmentation and defragmentation process.
- *Network Layer*: It does not contain any important functionality and serve as future extension of the model
- *Data Link Layer*: contains MAC protocol. The main functionality of this layer is to the control and to sense the link.
- *Physical Layer*: contains Serial Port Adaptor named as *SerialPortAdaptor* that manage the *SAMHelper* which is used to control the model (hardware).



### OSI Model Implementation

Each of layers inherits a based class named **OSILayer**. This is a generic class which is also inherits from **BaseOSILayer**. **BaseOSILayer** is the implementation of **IOSILayer**. Diagram bellows describe the relationship of the designed classes.



## Layer Design

Each layer contains two types of data as shown in figure???. They are

- Above Data: contain of the same data type of above layer.
- Below Data: contains the same data type of below layer.

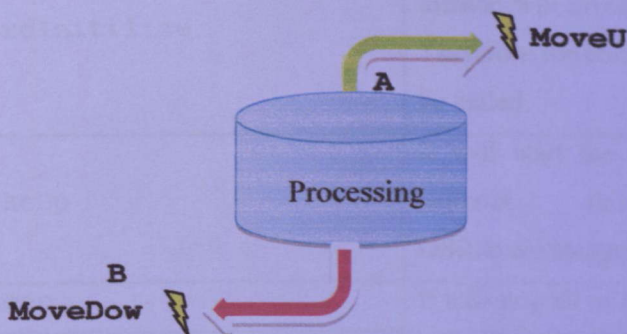
```

public class OSILayer<A, B> : BaseOSILayer
{
    protected BaseOSILayer _aboveLayer;
    protected BaseOSILayer _belowLayer;

    protected A _aboveData;
    protected B _belowData;
}
  
```

It also contains two links of another layer. The links are:

- \_aboveLayer: contains reference object to its above layer.
- \_belowLayer: contains reference object to its below layer.



The MoveUp and MoveDown use to move data from one layer to another layer. Each method contains a *Processing* function. This is used mainly to manipulate and allow programmer to manipulate data pass in or pass out in the layer.

*ProceedToTransmissionProcedure* : this will be automatically call when the MoveDown function is fired.



Method signature is

```
protected virtual void  
ProceedToTrasmissionProdures()
```

ProceedToReceivingProcedures: this will be automatically called when the MoveUp functions is fired. Method signature is

```
protected virtual void  
ProceedToReceivingProcedures()
```

**OSILayer Manager**

OSILayerManager is used to manage all of the six layers. It forms the OSI Model. It also contains the interactions for all layers. It also contains some configurations which will be used in MAC Protocol as well. The following table contains descriptions of methods.

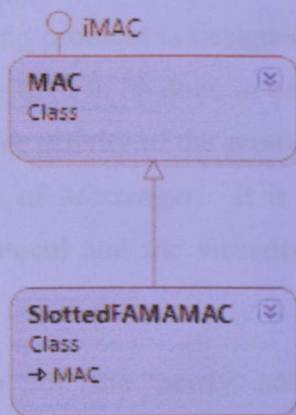
Method	Descriptions
OSILayerManager	Constructor of OSILayerManager class. It will intitalize its property by calling Initialize method
ZeroInitilize	OSILayerManager is a singleton. This method will invoke the only static object. Therefore the constructor of the class will be called.
WakeUp	It will start the function of all layers. Without this method, the OSILayerManager will not functioning.
Sleep	It will stop all of layers' functioning.
OSILayerManagerOnSleep	Event fired on Sleep invocation
OSILayerManagerOnWakeUp	Event fired on WakeUp invocation
Initialise	Initialize objects
TransmitData	Set data to transmit. There are two type <ul style="list-style-type: none"><li>- String data</li><li>- Byte data</li></ul>

## PROTOCOL

In communication, Protocol is very important in allowing two or more nodes exchanging information in a very efficient manner. Protocol is being implemented separately and plug into *Data Link* layer of *OSILayerManager*.

### Protocol Implementation

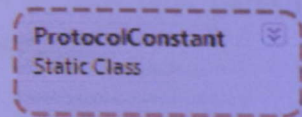
By default, **DOLPHINE OPT** provides a fully implemented protocol of *Slotted FAMA*. It has been proven to be an efficient protocol. Diagram below show the class diagram of **DOLPHINE OPT** protocol structure.



### Structure

All of the protocol must inherit from *MAC* class. Protocol must contain only the following operations:

- Status of the link ( if protocol is meant to be Carrier Sense)
- Status of Communication Node: depends on the implementing protocol. Status is not protocol specific but by default we also provide some basic status which has been enumerated in *ProtocolConstant*.
- *DataLink* Layer invocation method: provides a mean to provoke various data link's functionalities including *MoveDown*, *MoveUp* and so on.



### Virtualization

The virtualization of protocol in **DOLPHINE OPT** has the following purposes:



- Configurations of the protocol
- Status and Activities of the protocol

### Configuration Virtualization

The following steps contain instructions of how to add a new virtualization of a protocol:

- Create a working *View* and *ViewModel* of the newly add protocol
- In the *ProtocolSettingsViewModel* create a new protocol object and add to the *ProtocolItem* list.

For example:

```
this.AvailableProtocol.Add(new ProtocolItem ("Slotted FAMA", new
SlottedFAMASettingsViewModel()));
```

### Status and Activity Virtualization

The virtualization of the existing protocol is designed specific. If a new protocol is added, extra design and coding need to be done in order to cope with the restriction of the existing virtualization. For activity of the protocol, the communication is done by utilizing the functionalities of *Messenger*. It is advisable to use *Messenger* to communicate between the protocol and the virtualization. To achieve that follow below instructions:

- Create a message for the newly added protocol by inherit the *MessageBase* class.
- Identify which message should be broadcast and Send them according the designed message.
- Register your virtualization to the message and process the message accordingly.

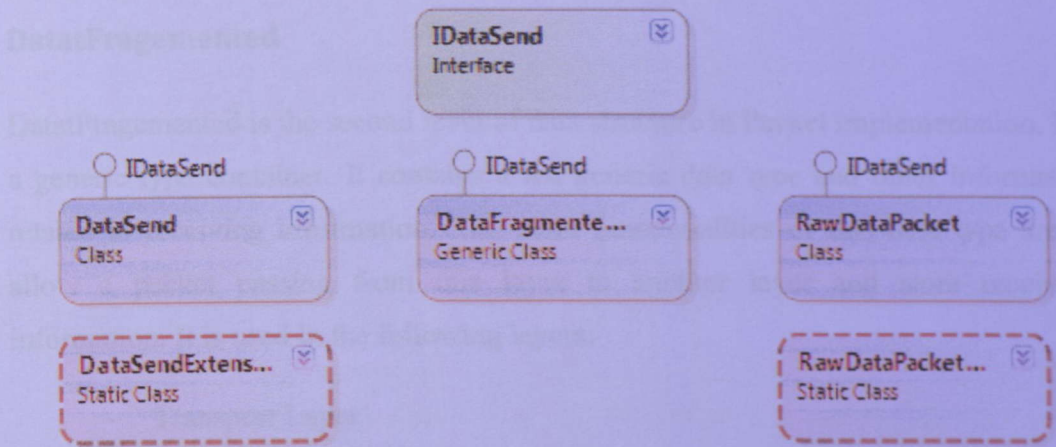
Developer need to make sure that the switching between each protocol is done properly. The procedure including unregister the previous message from the status and activity virtualization, enforce the changing protocol in Data Link Layer plus starting the whole OSI layer structure thought *OSILayerManager*.

## PACKET

Packet is a data structure that will be sending over the network. **DOLPHINE OPT** contains 3 types. They are:

- DataSend
- DataFragmentedBundle
- DataPacket

Each of data type inherits from *IDatasend*. Below is class diagram of *Packet*.



## DataSend

Datasend is the first level of data structure in the Packet implementation. It contains detail of the data that is being sent and received. Below here are the properties of *DataSend*:

- *Data Send Type*: There are two type
  - o Text
  - o Byte
- *Data Mode*: indicate whether data that is being sent need to compress or non-compress
- *Extension* (belong to byte data type): if data is being sent by using file, this will store the extension of the file.
- *TextData / ByteData* ( payload ): store text data or byte data. It is depends sending mode.

It also contains some information about the receive data such as:

- *Time of receiving*: refers to receiving time stamp
- *Time of completion*: refers to the completion time stamp
- *Total Packet send or receive*: indicate the total packet send and receive
- *Total packet error* : refers to the total number of packet error
- *Payload size*: refers to the payload size received
- *Average size*: contains the average size of all sent data packet
- *Data Index*: contains the receiving index number

There is also another extension that attach with *DataSend* contains:

- *Serialize*: is using to convert *DataSend* object to byte data.



- *Deserialize* : is using to convert to byte data type of *DataSend* to *DataSend* object

## DatatFragemented

DatatFragemented is the second level of data structure in Packet implementation. It is a generic type container. It contains a list generic data type and other information related to receiving information. The main functionalities of this data type are to allow a packet passing from one layer to another layer and store receiving information. It is used in the following layers:

- Transport Layer
- Network Layer
- Data Link Layer

## RawDataPacket

RawDataPacket is the third level of data structure in Packet implementation. It contains the following information:

- Main Header: is 1 byte size of data store two type of information:
  - o Packet Type
  - o Pay Load Control
- Address Source: is 1 byte size of data stores address of sender
- Destination Address: is 1 byte size of data stores address of destination
- Packet Control 1: is 1 byte size of data stores different type of information depends on Main Header such as :
  - o Total number of packet send: if Packet Type is **RTS**
  - o Packet Sequence Number: if Packet Type is **Data**
  - o Error Packet Sequence Number: if Packet Type is **NACK**
- Packet Control 2: is 1 byte size of data stores the size of payload.
- CRC8: is 1 byte of data stores the byte code of Cycle Redundancy Check of 8 bits(CRC8).
- Payload: is an array of byte data used to store data to be sent. Since Packet Control 1 and Packet Control 2 are defined in byte data so the total size of each packet is 256. The packet control and CRC8 has taken up 6 bytes. Therefore the maximum size of payload is **250** bytes.

There is also another extension that attaches with *RawDataPacket* and contains the following functions:

- *Serialize*: is used to convert *RawDataPacket* object to byte data.
- *Deserialize* : is used to convert to byte data type of *RawDataPacket* to *RawDataPacket* object

## Constant and Enumeration

There are 5 types of enumeration and 2 static classes such as

- *PacketType*: stores enumerated data of packet type
- *PayloadControl*: stores enumerated data which is used for data control.  
There are two types of this enumeration such as:
  - o *ODD*: is used when payload is odd in number of byte.
  - o *Even*: is used when payload is even in number of byte.
- *FileExtension*: store enumerated data of extension.
- *DataSendType*: there are two types such as
  - o *Byte*
  - o *Text*
- *DataMode*: there are two modes such as
  - o *Compress*
  - o *NonCompress*

The compression algorithm is used G-Compressor of C#.

- *ExtensionMapping*: maps extension in string to *FileExtension* type.
- *Constant*: contains two constants such as:
  - o Max Payload
  - o Max Fragment