



**Marine Data Collection and Transmission System for  
ECO-boat**

by

*Gaoqiang Zhuo*

Supervisor: Lei Jiao

**Master Thesis in Information and Communication Technology  
IKT 590, Spring 2014**

Faculty of Engineering and Science  
University of Agder

Grimstad, 02 June 2014

Status: Final

## Abstract

Marine data collection is of great importance because the analytical results based on the collected data can be utilized for many purposes during the design phase before sale as well as the maintenance phase of the boat after sale. In this thesis, I have designed and implemented a marine data collection and transmission system for yachts. The system can collect data from an NMEA network, a wireless sensor network, a custom-made CAN-bus network, and other sensors such as a 9 degree-of-freedom inertial measurement units. The collected data is stored both at the boat and at a remote server. In this design, I use a Pandaboard as the main board and I also adopt a watchdog to control the Pandaboard. The watchdog is formed by an Arduino Uno, which controls a relay associated with voltage regulators. I have also configured the Pandaboard as an access point with a fixed private IP address and have initialed the DHCP functionality. In this way, I can access the Pandaboard as an access point and control it freely in its vicinity via another laptop. For long distance communication, with a 3G modem, I can transmit the data from the boat to a remote server in the lab. The designed system has gone through a rigorous lab test as well as a simulation test via a boat-movement simulator. Afterwards, I install the box on a yacht manufactured by Viknes. Through various testing and boat implementations, I find that the system works smoothly and stable. Two groups of students have done some previous work on this topic in their master theses, but my work are better than the combination of their work. My system is also better than the current commercial product. The commercial product supports only local data storage while my system supports long distance data transmission in addition to local data storage.

**Keywords:** Marine data collection, Pandaboard, Watchdog, Scripts, Data storage

## **Preface**

This report is the result of the master thesis IKT 590 (30 ECTS) which is a part of my fourth semester MSc study at the Faculty of Engineering and Science, University of Agder (UiA) in Grimstad, Norway. The work on this project started from 1 January 2014 and ended on 2 June 2014. I have completed the main goals of my project “Marine Data Collection and Transmission System for ECO-boat”.

This project is part of the ECO-Boat MOL project that is funded by the Research Council of Norway. I would like to thank my supervisor Dr. Lei Jiao for the guidance in both technical questions and content of the report throughout this project. Through this thesis work, I have learnt a lot about project content and technical report writing. I want to thank Prof. Frank Li for his advices and caring about the process. I also want to thank Mr. Torgeir Bråtane and Ms. Vivian Wigvoll Skaim from Inventas for the design of the encapsulation of the box. I am also thankful to Mr. Tornd-Ivar Lynghaug in University of Agder for providing soldering equipment and giving guidance about it. Last but not least, I would like to thank Mr. Kristian Grøndal Sivertsen from Viknes for the support to install the system on boat.

# Contents

<b>1</b>	<b>Introduction.....</b>	<b>10</b>
1.1	Background .....	10
1.2	Problem statement.....	10
1.3	Problem solutions .....	10
1.4	Report outline .....	11
<b>2</b>	<b>State of the art.....</b>	<b>12</b>
2.1	Existing previous work .....	12
2.2	Existing commercial product .....	13
<b>3</b>	<b>System design .....</b>	<b>15</b>
3.1	System structure .....	15
3.2	Hardware overview .....	16
3.2.1	Pandaboard ES .....	16
3.2.2	Arduino Uno/Mega/Lilypad.....	16
3.2.3	NGT-1 .....	18
3.2.4	9DOF Razor IMU .....	18
3.2.5	3G modem .....	18
3.3	Hardware connectivity and interaction .....	19
3.3.1	Watchdog and mainboard.....	19
3.3.2	Gateway between various devices and mainboard.....	20
3.3.3	Power isolation .....	22
3.4	Software overview.....	22
3.4.1	Ubuntu12.04 .....	22
3.4.2	MySQL .....	22
3.4.3	Arduino.....	23
3.4.4	Openssh.....	23
3.5	Overview of software design .....	23
3.5.1	Operating system .....	23
3.5.2	Data transmission .....	23
3.5.3	Data storage.....	23
3.5.4	Access Pandaboard from vicinity.....	24
<b>4</b>	<b>Implementation .....</b>	<b>25</b>
4.1	Ubuntu installation .....	25
4.1.1	Format SD card and create boot and file system.....	25
4.1.2	Create new boot file from source.....	25
4.1.3	Create new kernel from source.....	26
4.1.4	Choose drivers.....	26
4.1.5	Create ulmage and install modules .....	28
4.2	Install important software .....	29
4.2.1	Set up 3G modem .....	29
4.2.2	Install MySQL.....	30
4.2.3	Access point configuration and wireless access .....	32



<b>4.3</b>	<b>Implement C programs and scripts .....</b>	<b>34</b>
4.3.1	C programs for IMU and NMEA.....	34
4.3.2	Mapping ports .....	43
4.3.3	Watchdog.....	45
<b>4.4</b>	<b>Set up hardware .....</b>	<b>48</b>
<b>4.5</b>	<b>Start up scripts .....</b>	<b>48</b>
<b>4.6</b>	<b>Integration in a box.....</b>	<b>51</b>
<b>5</b>	<b>Testing .....</b>	<b>53</b>
5.1	Testing in the lab .....	53
5.2	Testing in a boat-movement simulator.....	55
<b>6</b>	<b>Installation on boat .....</b>	<b>59</b>
6.1	Installation procedure .....	59
6.2	Testing and validation .....	62
<b>7</b>	<b>Conclusion and future work.....</b>	<b>64</b>
	<b>Reference.....</b>	<b>65</b>
	<b>Appendices .....</b>	<b>67</b>

## List of Figures

Figure 2.1 System structure in [1] .....	12
Figure 2.2 System structure in [2] .....	13
Figure 2.3 VDR100 [3].....	14
Figure 3.1 System structure.....	15
Figure 3.2 Top view of Pandaboard [5] .....	16
Figure 3.3 Top view of Arduino Mega [6] .....	17
Figure 3.4 Top view of Arduino Uno [7] .....	17
Figure 3.5 Top view of Arduino Lilypad [8] .....	17
Figure 3.6 NGT-1 gateway [10] .....	18
Figure 3.7 Top view of the IMU module [11] .....	18
Figure 3.8 3G modem [12].....	19
Figure 3.9 Reaction of watchdog in normal case.....	19
Figure 3.10 Reaction of watchdog in abnormal case .....	20
Figure 3.11 WSN gateway design.....	20
Figure 3.12 CAN gateway design.....	21
Figure 3.13 NMEA gateway design.....	21
Figure 3.14 IMU gateway design .....	22
Figure 3.15 Voltage regulator and isolator .....	22
Figure 3.16 GUI of Arduino software.....	23
Figure 3.17 Access Pandaboard from vicinity .....	24
Figure 3.18 Signal flows for accessing Pandaboard .....	24
Figure 4.1 Menuconfig window.....	26
Figure 4.2 How to choose a driver .....	27
Figure 4.3 GUI of the Ubuntu 12,04 LTS.....	28
Figure 4.4 Successful connection of the 3G modem.....	30
Figure 4.5 Access point is found .....	33
Figure 4.6 Successful log in from another host.....	34
Figure 4.7 Initial version of IMU output.....	35
Figure 4.8 New version of IMU output.....	35
Figure 4.9 Flow chart of the IMU data collection program .....	36
Figure 4.10 Part of IMU data in database.....	37
Figure 4.11 Data in IMU.txt.....	38
Figure 4.12 IMU data in json format .....	38
Figure 4.13 Files created by date .....	40
Figure 4.14 Local data from NMEA.....	42
Figure 4.15 NMEA data in the remote server.....	42
Figure 4.16 Illustration of USB stick auto mount program.....	43
Figure 4.17 Nicknames of ports .....	44
Figure 4.18 Device information .....	45
Figure 4.19 Flow chart of the watchdog program .....	46
Figure 4.20 Flow chart of the watchdog script.....	47
Figure 4.21 Flow chart of the polr.sh script .....	47

Figure 4.22 Flow chart of the startup script for NMEA .....	49
Figure 4.23 Flow chart of the startup script for IMU .....	49
Figure 4.24 Flow chart for the database backup script .....	50
Figure 4.25 Flow chart of the circular storage script .....	51
Figure 4.26 Inner layers of the box.....	51
Figure 4.27 Top view of the inner layers in the box.....	52
Figure 4.28 Outlook of the box without cover .....	52
Figure 5.1 Testing phase 1 .....	53
Figure 5.2 Power supply in the lab .....	53
Figure 5.3 Testing phase 2 .....	54
Figure 5.4 Outside appearance of the integration .....	54
Figure 5.5 Fix the box in the simulator .....	55
Figure 5.6 Testing in simulator without power - a .....	56
Figure 5.7 Testing in simulator without power - b.....	56
Figure 5.8 Testing in simulator with power .....	57
Figure 5.9 IMU data during the testing in simulator.....	57
Figure 6.1 Discussion about the installation .....	59
Figure 6.2 The position of the NMEA network on boat .....	59
Figure 6.3 NMEA network on the boat.....	60
Figure 6.4 Measuring the diameter of the button .....	60
Figure 6.5 Choosing the position for the button.....	60
Figure 6.6 Installation of the button .....	61
Figure 6.7 Closer look of the button.....	61
Figure 6.8 Install the system under the seat .....	61
Figure 6.9 Overview of the system after installation.....	62
Figure 6.10 Testing the system on sailing .....	62
Figure 6.11 3G modem is working.....	63
Figure 6.12 Detecting all sensors on boat.....	63

## List of Tables

Table 4.1 Connection betlen watchdog and Pandaboard .....	48
Table 4.2 Connection betlen watchdog and relay.....	48
Table 4.3 Connection betlen watchdog and polr button .....	48

## Abbreviations

CAN	Controller area network
DHCP	Dynamic host configuration protocol
DOF	Degree of freedom
GPIO	General purpose input/output
IMU	Inertial measurement unit
LTS	Long term support
NGT	NMEA gateway
NMEA	National marine electronics association
OTG	On the go
SSH	Secure shell
WSN	Wireless sensor network

# 1 Introduction

In this chapter, I am going to provide a short background of my thesis. I will also give the problem statement and problem solution in briefly. At the end of this chapter, the outline of the whole report will be given.

## 1.1 Background

My master thesis is a part of a project called “ECO-boat mid of life” funded by Research Council of Norway. And the target of my thesis is to build a marine data collection and transmission system for ECO-boats. There are many sources of marine data from various sensors, and how to integrate them in one system is a challenge. I need to find the right hardware and also need to create necessary software to support all the functionalities. Two groups of students did some work on this topic in their master theses last year and I will continue their work and integrate the whole system together.

## 1.2 Problem statement

In this thesis, I am going to develop the following functionalities.

- ✧ To collect data from four different sources and store the collected data in an embedded system. The four sources are NMEA 2000 network, IMU, WSN and custom-made CAN protocol based network.
- ✧ To implement a watchdog for the embedded system.
- ✧ To store data in both MySQL databases and in text files.
- ✧ To access the embedded system in vicinity through wireless communications.
- ✧ To upload the sensed data to a remote server.
- ✧ To integrate the system into a box that is appropriate to install on boats.
- ✧ To ensure the stability and reliability of the system.

## 1.3 Problem solutions

Based on the above requirement of the system, I propose the following solutions for the system.

- ✧ I will use Pandaboard as the main board to collect the data from all the four sources. To achieve this goal, I need to integrate many equipments (hardware) together with the corresponding programs (software).
- ✧ I will use an Arduino Uno as the watchdog.
- ✧ I will configure the Pandaboard as an WiFi access point, so that I can access it from its vicinity with the help of SSH software.
- ✧ I will utilize a 3G interface for long distance communication.
- ✧ I will select a box and arrange all the hardware inside the box.
- ✧ I will test the system in different scenarios to ensure the stability and reliability of the system.

## **1.4 Report outline**

In Chapter 2, I will introduce the state of the art of the thesis. In Chapter 3, I am going to present my system design, which includes hardware design and software design. The implementation details are demonstrated in Chapter 4. In Chapter 5 and 6, I will present the testing and installation of the system. Chapter 7 concludes the report and gives the possible future work.

## 2 State of the art

In this chapter, I am going to introduce the state of the art. The state of the art includes two parts. The first part includes two data collection systems designed by two groups of students last year during their master thesis. In the second part, I will introduce a commercial data recorder.

### 2.1 Existing previous work

In this subsection, I am going to introduce two systems designed by two previous groups of students. The two systems have similarity as well as differences.

The first system was designed by Xiaolong Chen and Baddam in [1], and the system structure is shown in Figure 2.1.

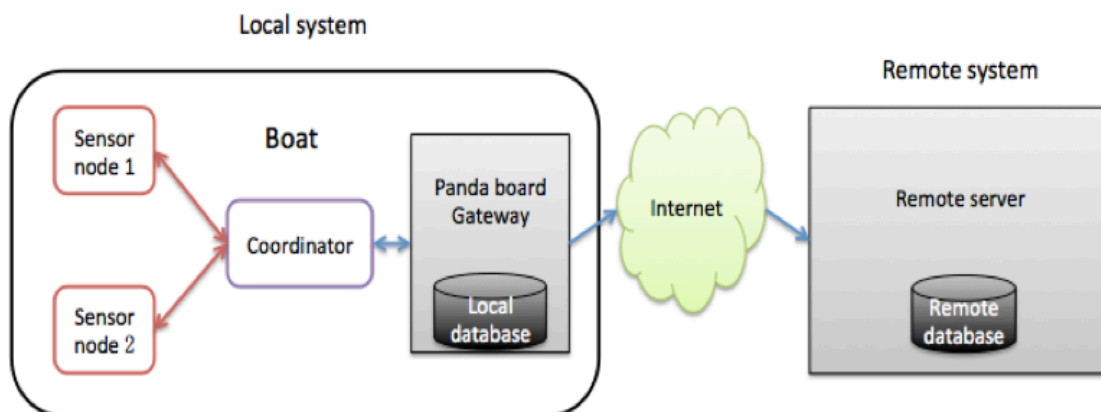


Figure 2.1 System structure in [1]

The data source in this system is a wireless sensor network. The data is generated from the sensor nodes, and then sent to the coordinator. The coordinator is connected to the Pandaboard that will store the data to the local database and at the same time the Pandaboard will send the data to a remote server via Telenor's 3G network.

The second system was designed by Mingli Yue and Yihai Sun in [2], and the system structure is shown in Figure 2.2.



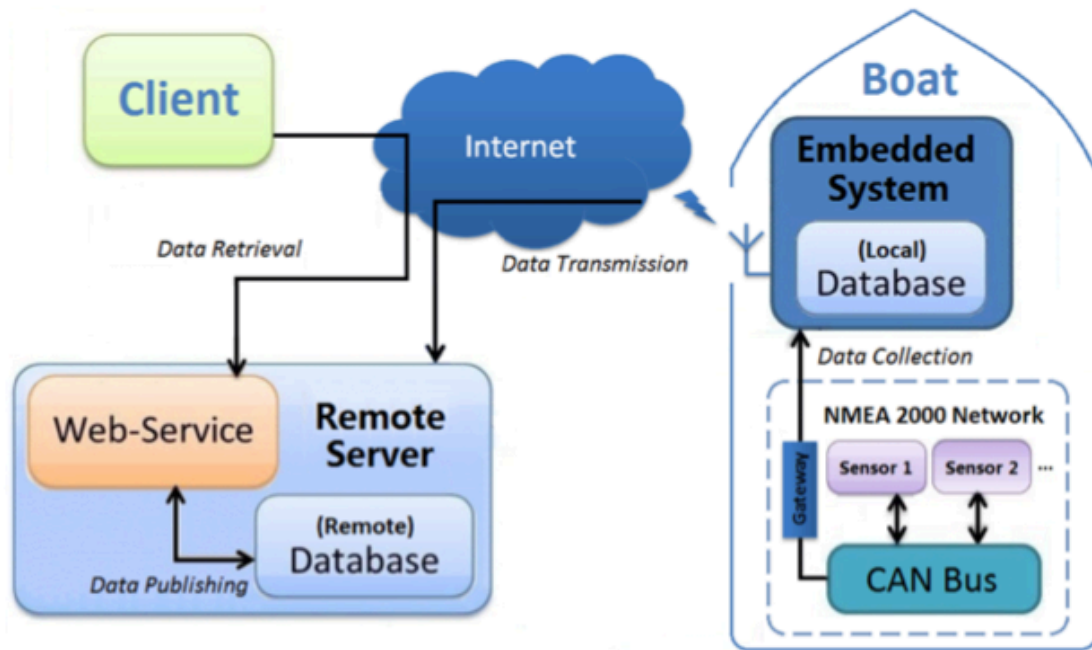


Figure 2.2 System structure in [2]

The data source in this system is an NMEA 2000 network. The data is generated from the NMEA 2000 network. The embedded system receives the data from a gateway and stores the data into a local database. At the same time, the embedded system sends the data to a remote server via Telenor's 3G networks. The remote server will publish the data through a web service. The embedded system here is also a Pandaboard.

The above two systems have almost the same structure. Both of them can store the sensed data into a local database and also a remote server. The differences are 1) the data source in the first system is a WSN and the data source in the second system is a NMEA 2000 network, 2) the second system has an extra function that will publish the data through a web service.

However, these two systems can only store data from one source respectively. In my system, both of them will be included.

## 2.2 Existing commercial product

There already exists a commercial product to record data from NMEA 2000. This product is called Maretron Vessel Data Recorder (VDR100), as shown in Figure 2.3.



Figure 2.3 VDR100 [3]

VDR100 can record data from all sensors that are connected to the NMEA 2000 network, and all data are stored in a removable USB flash drive. The stored data can be easily accessed by using the free software N2K Extractor.

Both the existing previous work and the existing commercial product have their disadvantages. The existing previous work can only store data from one data source, for example either WSN or NMEA 2000. The existing commercial product can only store data from NMEA 2000 network and it does not support long distance transmission. As a result, the commercial product can only store data locally.

Based on the facts that there are no satisfactory products in the market, I am going to design or integrate a more advanced system. My system will be able to store data from various data sources and store data both locally and remotely. My system will have all the functions that current work or products have and will have many additional functions.

### 3 System design

In this chapter, I am going to explain my system design. This chapter will firstly present an overview of system structure, and then introduce the selected hardware and software platform for this system. The overall hardware and software design is detailed thereafter.

#### 3.1 System structure

The overall system structure is shown in Figure 3.1. The WSN is a wireless sensor network that can generate and send sensed information to the gateway. The CAN is a custom-made CANbus based network that can collect data from all sensors that can be used with CANbus. The NMEA is an NMEA 2000 network that is used collect data from the boat, such as the tank level, GPS and so on. The IMU is a 9 degree-of-freedom inertial measurement unit. The data from different network will have a distinct gateway. The Pandaboard collects data through the corresponding gateways. The watchdog can control the on and off of the Pandaboard and if the Pandaboard crashes the watchdog will restart the Pandaboard. All the above hardware will be on the boat. There is also a remote server that is not located on the boat. The remote server can receive the data from the Pandaboard via Telenor's 3G networks.

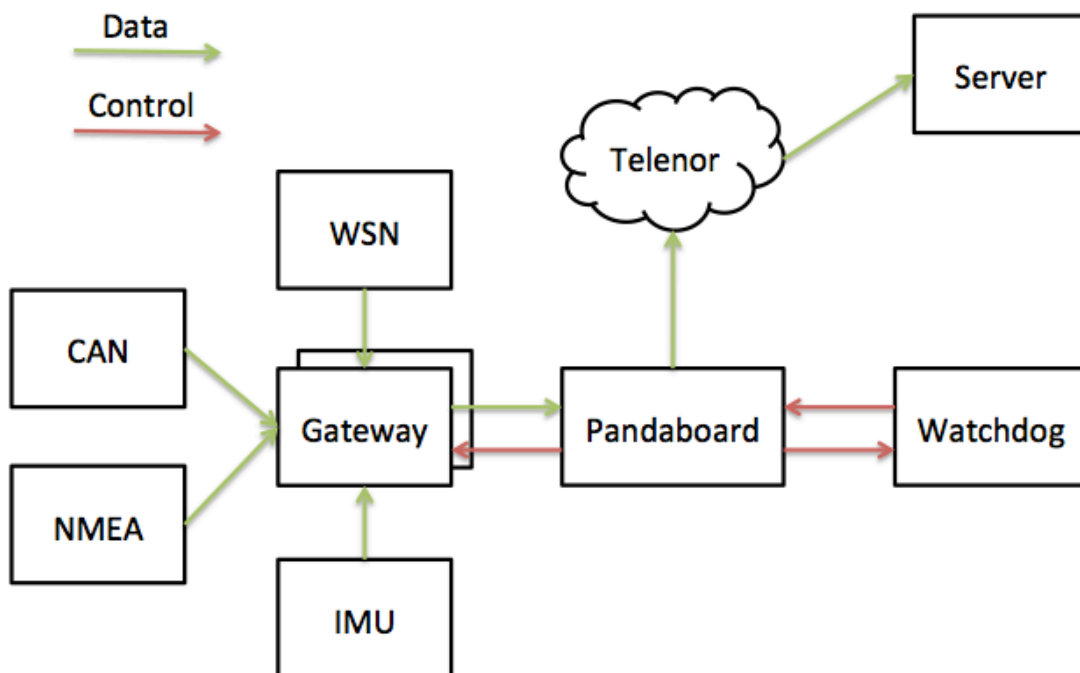


Figure 3.1 System structure

## 3.2 Hardware overview

I will use many kinds of hardware in this project. In what follows, I am going to introduce a few most important ones.

### 3.2.1 Pandaboard ES

The top view of the Pandaboard is shown in Figure 3.2. From Wikipedia, I know that “The PandaBoard is a low-power, low-cost single-board computer development platform based on the Texas Instruments OMAP4430 system on a chip (SoC)” [4]. “The PandaBoard ES is a newer version based on the OMAP4460 SoC, with the CPU and GPU running at higher clock rates” [4]. I choose Pandaboard because it has many integrated functions that other boards do not have, such as the WiFi chip, the GPIO pins and so on. The memory size and CPU speed are also sufficient for the applications.

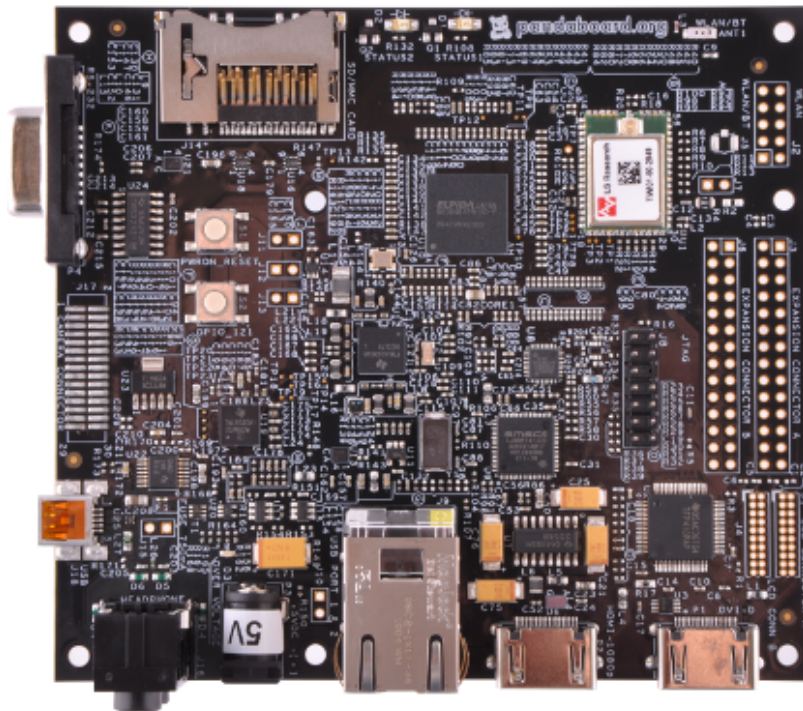


Figure 3.2 Top view of Pandaboard [5]

### 3.2.2 Arduino Uno/Mega/Lilypad

In this system, Arduino Mega, Uno and Lilypad are adopted. I select Arduino Mega or Uno to work as gateway between WSN and PandaBoard. I also use the Uno as the watchdog. Both Mega and Uno are very useful platforms, and they offer a lot of digital and analogue pins. The top views of Arduino Mega and Uno are shown in Figure 3.3 and Figure 3.4 respectively.

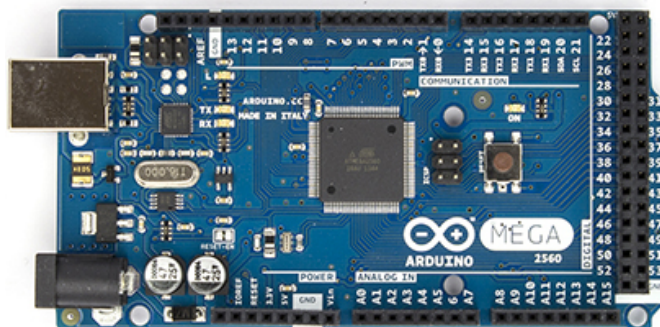


Figure 3.3 Top view of Arduino Mega [6]

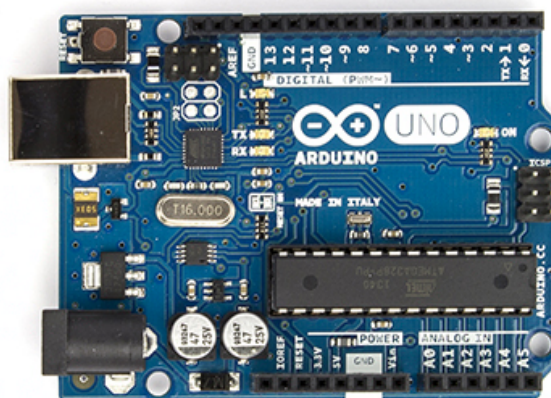


Figure 3.4 Top view of Arduino Uno [7]

The LilyPad is another kind of Arduino board, and it is smaller in size. I use LilyPad for the sensor node in wireless sensor network. The reasons are as follows: 1) I can make the sensor node small and 2) the computational capability of this board is enough for sensing information processing and forwarding. The top view of LilyPad is shown in Figure 3.5.

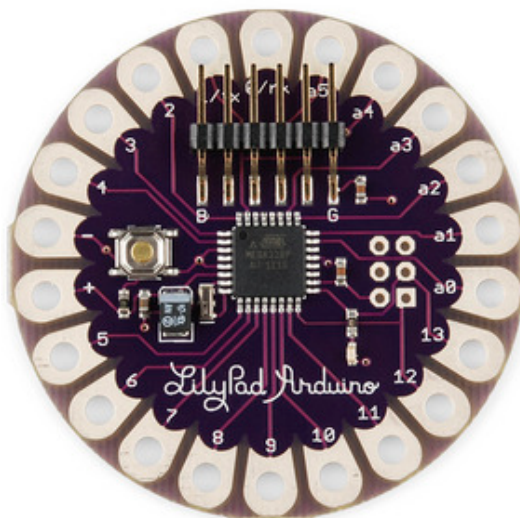


Figure 3.5 Top view of Arduino Lilypad [8]



### 3.2.3 NGT-1

The NGT-1 is a gateway for NMEA 2000 and USB 2.0 protocol, which can transmit messages to/from the NMEA 2000 bus from/to USB [9]. The picture of the NGT-1 gateway is shown in Figure 3.6.



Figure 3.6 NGT-1 gateway [10]

### 3.2.4 9DOF Razor IMU

I select the 9 degree-of freedom (DOF) inertial measurement unit to measure the acceleration, magnet, and gyro values. The IMU module is shown in Figure 3.7.

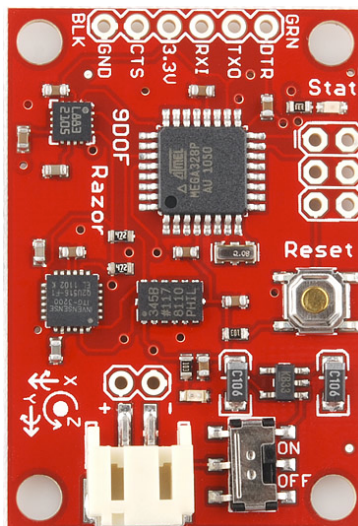


Figure 3.7 Top view of the IMU module [11]

### 3.2.5 3G modem

In this project, I will use a 3G modem to have access to the Telenor's 3G network for long distance communication. The picture of the modem is shown in Figure 3.8.

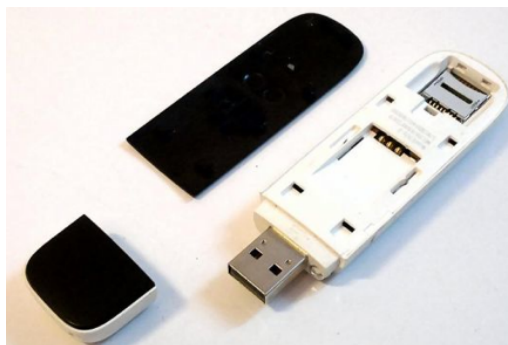


Figure 3.8 3G modem [12]

This 3G modem is an Huawei E353, and it needs a normal SIM card to work. Furthermore, an extension port is available for external antenna. It is very tiny in size and it supports hot plug.

### 3.3 Hardware connectivity and interaction

In this subsection, I will give an overall description about the hardware design.

#### 3.3.1 Watchdog and mainboard

The watchdog plays an important role in my system. It is responsible for controlling the booting and shutting down processes of the operation system in normal situations and it can also check if the Pandaboard is freezed or not. If Pandaboard becomes freezed for some reason, the watchdog can reboot the system. In the following paragraphs, I will use two cases to show the functions of the watchdog. The first case is the normal case, as shown in Figure 3.9.

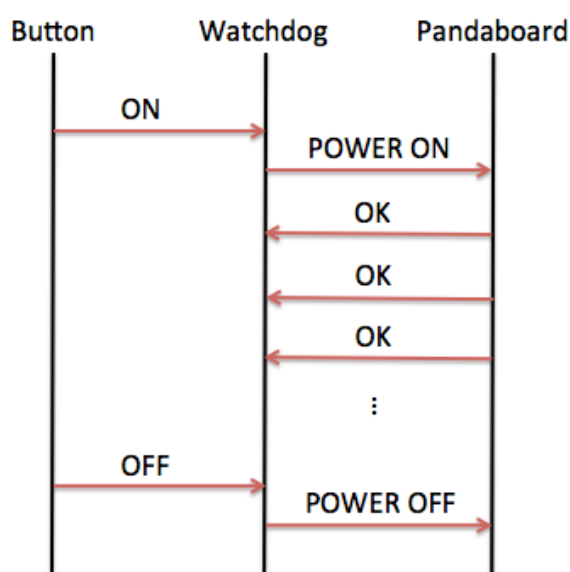


Figure 3.9 Reaction of watchdog in normal case

In the normal case, when the button is pressed, meaning an “ON” signal is given to the watchdog, then the watchdog will enable the power supply to the Pandaboard, so that the Pandaboard can boot. After booting successfully, the Pandaboard will send “OK” back to watchdog periodically via GPIO pins. When the watchdog receives the “OK” message, it believes that the Pandaboard is working, and will not do anything to the Pandaboard. When the button is pressed off, then the watchdog will ask the Pandaboard to power off through GPIO and then disable the power supply to the Pandaboard after power off. The second case is the abnormal case, as shown in Figure 3.10.

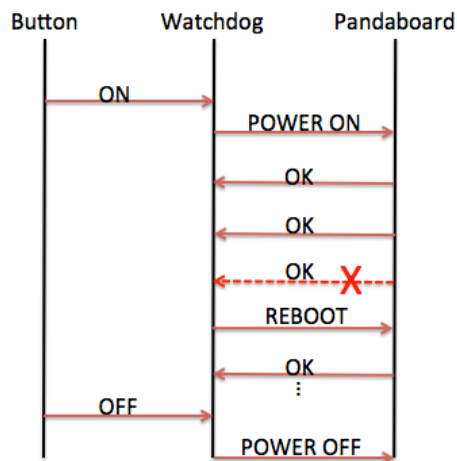


Figure 3.10 Reaction of watchdog in abnormal case

In the abnormal case, the Pandaboard crashes while working. Once this happens, the watchdog will detect this and repower on the Pandaboard again through a relay. The rest procedures are the same as the normal case.

### 3.3.2 Gateway between various devices and mainboard

The gateway for the WSN consists of an Arduino Uno, an Arduino shield and an Xbee module, as shown in the following Figure 3.11.

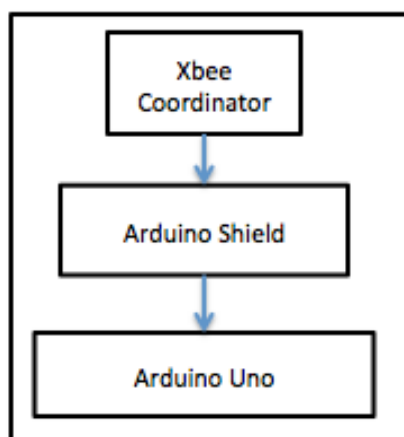


Figure 3.11 WSN gateway design



The Xbee works as a coordinator and when it receives data from a sender, it will pass the data to the Arduino Uno by serial communication through the shield board. The Arduino Uno will process the data and send the processed data to the Pandaboard via USB cable.

The gateway for the CAN network consists of a CANbus connector, a CANbus shield and an Arduino Uno, as shown in Figure 3.12.

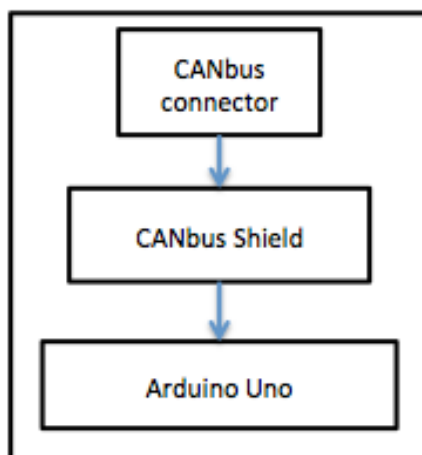


Figure 3.12 CAN gateway design

As can be observed from the figure, the data from the CAN network will be sent to the CANbus connector, and the connector will pass the data to the Arduino Uno. On the Arduino Uno, the data from the CAN bus is processed and sent to the Pandaboard via USB cable.

The gateway for the NMEA is a commercial NGT-1 gateway, as shown in Figure 3.13.

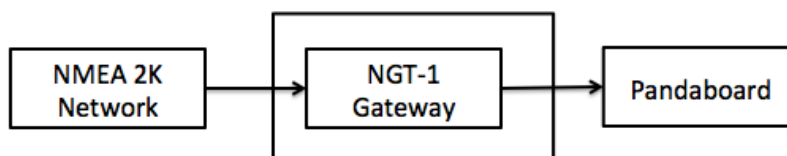


Figure 3.13 NMEA gateway design

The NGT-1 gateway can send all the data from the NMEA 2000 network to the Pandaboard via the USB cable.

The gateway for the 9 DOF IMU is a serial to USB connector as shown in Figure 3.14.

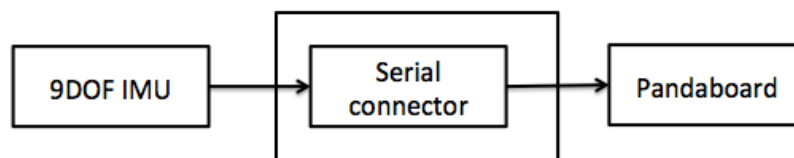


Figure 3.14 IMU gateway design

### 3.3.3 Power isolation

To improve the stability of my system, I take the power isolation into consideration for. In my system, I have isolation between the power supply and the power input of my system. I achieve the isolation by means of a constant output voltage regulator and isolator, as shown in Figure 3.15.

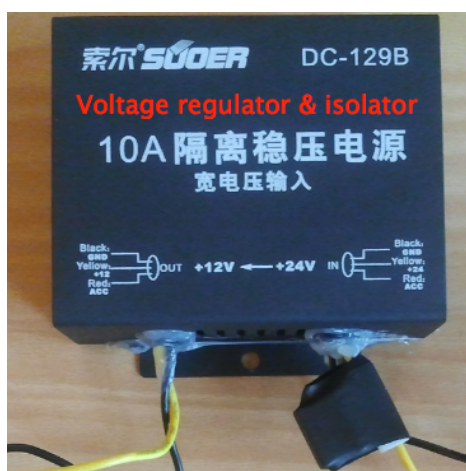


Figure 3.15 Voltage regulator and isolator

This regulator has an input range from 12V to 24V and a constant output value 12V. With this regulator and isolator, I do not need to worry about the voltage shift of the power supply, which can increase the stability of the system.

## 3.4 Software overview

In this subsection, I am going to introduce a few of the most important programs that I will use in this project.

### 3.4.1 Ubuntu12.04

Ubuntu is a free Linux operation system. It is very famous and very easy to use. In this project, I use Ubuntu 12.04 LTS as the operation system.

### 3.4.2 MySQL

MySQL is the world's second widely used open source database [13]. In this project, I need MySQL in both the embedded system and the remote server. MySQL can be used on Windows, Mac OS X and Linux.

### 3.4.3 Arduino

I have already introduced the Arduino Mega, Uno and Lilypad, To upload programs to these boards, I need a software called “Arduino”. The Arduino software supports almost all the popular platforms. The language for the Arduino software is simplified version of C language and it is very easy to use. The GUI of the Arduino software is shown in Figure 3.16.

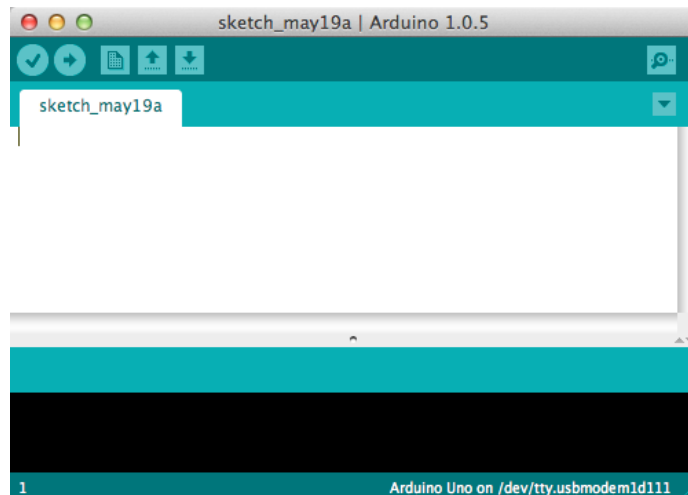


Figure 3.16 GUI of Arduino software

### 3.4.4 Openssh

Openssh is a set of computer programs that offer encrypted communication sessions over a computer network via the SSH protocol [14]. In this project, I need to install openssh in Ubuntu, so that I can access the system from another host with the help of SSH protocol.

## 3.5 Overview of software design

### 3.5.1 Operating system

I am going to use Ubuntu 12.04 as the operating system. However, to make the operating system more efficient, I recompile the kernel so that I can keep the drivers that I really need. The procedures will be given in Chapter 4 in more details.

### 3.5.2 Data transmission

I have a lot of data transmission in my system. All the programs for data transmission purpose are written with C language. The details of the C programs are explained in Chapter 4 and the source code can be found in Appendices.

### 3.5.3 Data storage

In my system, I will design two methods to store data. The first method is to store all

the data into the MySQL database. The other method is to store all the data into a text file. Each of them has advantages and disadvantages. I have implemented both of the storage method and the details of the methods will be given in Chapter 4.

### 3.5.4 Access Pandaboard from vicinity

Once all the design and encapsulation of hardware are finished, the Pandaboard will be inside a box where a monitor is not applicable. In order to access the Pandaboard, I will configure the Pandaboard as an access point that can assign IP address to its clients. The Pandaboard itself has a fixed local IP, so that I can access the Pandaboard through SSH method. The Figure 3.17 and Figure 3.18 show how this works.

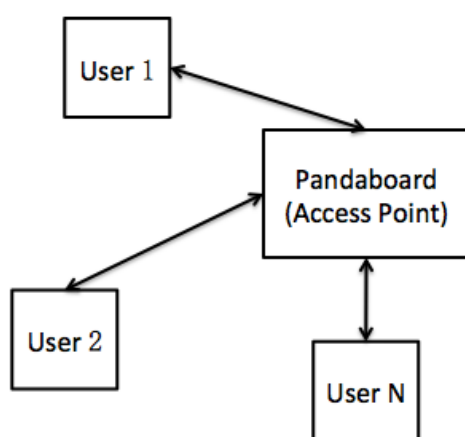


Figure 3.17 Access Pandaboard from vicinity

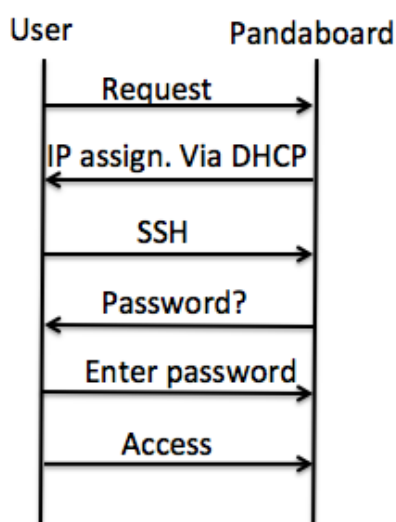


Figure 3.18 Signal flows for accessing Pandaboard

In this chapter, I have illustrated my system design in terms of hardware and software at a high level. In next chapter, I am going to implement my system design.

## 4 Implementation

In this chapter, I am going to explain in details how I implement the design and how I build up the system step by step. There will be six steps.

- Step 1: Install operating system on Pandaboard;
- Step 2: Install important software;
- Step 3: Implement important C programs and scripts;
- Step 4: Set up all hardware;
- Step 5: Design startup scripts;
- Step 6: Integration in a box.

### 4.1 Ubuntu installation

I choose Ubuntu 12.04 for the Pandaboard B3. However, to keep the system lighter, I am going to recompile the kernel. I will only keep the necessary drivers.

#### 4.1.1 Format SD card and create boot and file system

I need to format the SD card first, and then create two partitions called “boot” and “rootfs”. The boot partition will contain all the boot files. The rootfs partition will contain the file system. The command lines for achieving this are given below [15].

```
$cd /Ubuntu-1204-b3/ #this is offered by the seller#  
$chmod +x mkcard.sh  
$sudo ./mkcard.sh --device /media/sdb
```

After following the above command lines, I now have a fresh Ubuntu 12.04 operating system installed on the SD card. In the next steps, I am going to recompile the kernel.

#### 4.1.2 Create new boot file from source

Before I compile the kernel, I need to install a compiler called “gcc-arm-linux-gnueabi” and an image maker called “uboot-mkimage”. Since I compile the kernel from another desktop, I will need to cross-compile the kernel for ARM core. The command lines for achieving these are given below [15].

```
$sudo apt-get install build-essential gcc-arm-linux-gnueabi uboot-mkimage  
$cd /u-boot-pandaboard-ES-RevB3/  
$make CROSS_COMPILE=arm-linux-gnueabi- omap4_panda  
$cp MLO u-boot.bin /media/boot
```

The first line installs the necessary tools. The third line cross-compile related files and produce two important files called “MLO” and “u-boot.bin”. The last line copy the newly produced files into the boot partition on the SD card.

### 4.1.3 Create new kernel from source

Here I am going to make a new kernel that is tailored for the application. First, I need to install a configuration tool that provides an API for choosing the drivers. Then I use that configuration tool to choose the drivers that I need. The command lines are given below [15].

```
$sudo apt-get install menuconfig #install this if you don't have it
$cd /kernel-pandaboard-ES-RevB3/
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-omap4plus_defconfig
#create default kernel config
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-menuconfig
```

After doing these command lines, a window will pop up, as show in Figure 4.1.

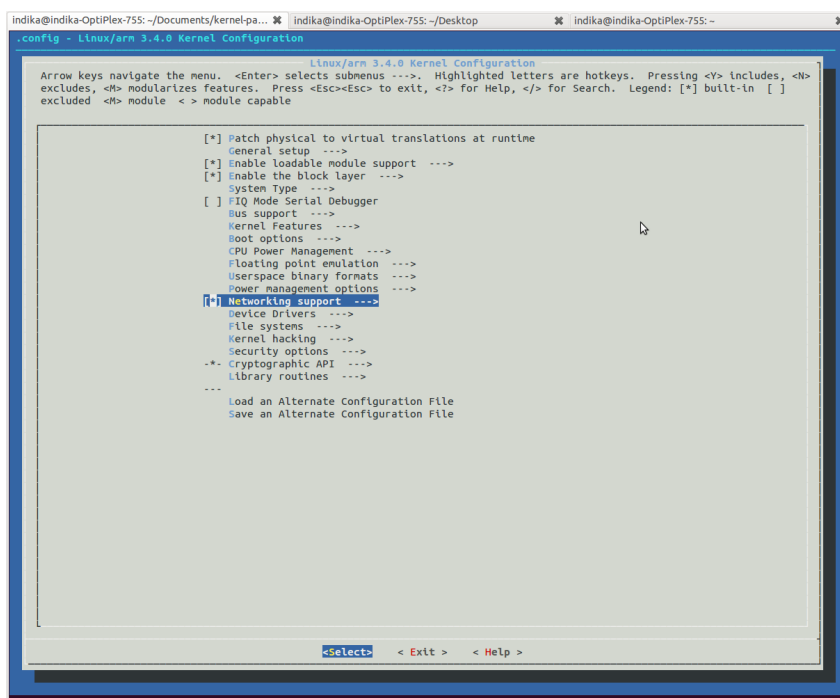


Figure 4.1 Menuconfig window

### 4.1.4 Choose drivers

After the configuration window pops up, choose the drivers according to the following list.

#### Device Drivers

- >Network device support
  - <\*> PPP (point-to-point) support
  - <\*> PPP BSD-compress
  - <\*> PPP Deflate compression
  - <\*> PPP for async serial ports

```

<*> PPP for sync tty ports
[*] Wireless LAN --->
    <*> IEEE 802.11 for Host AP (Prism2/...)
        [*] Support downloading
        [*] Support for non-volatile...
    [*] TI wireless LAN Support --->
        <M> TI wl1251 driver support --->
            <M> TI wl1251 SDIO

->GPIO Support --->
    [*] /sys/class/gpio/...(sysfs interface)
->Watchdog Timer Support --->
    [*] Watchdog Timer Driver Core
    <*> Software watchdog
->USB Support --->
    <*> USB Modem (CDC ACM) Support
    [*] OTG Support
    <*> USB Serial Converter Support --->
        [*] USB Generic Serial Driver
        [*] USB FTDI...
        <M> USB driver for GSM and CDMA modems
->Real Time Clock --->
    [*] Dallas/Maxim DS1307/37/...

```

Figure 4.2 shows how to choose the driver.

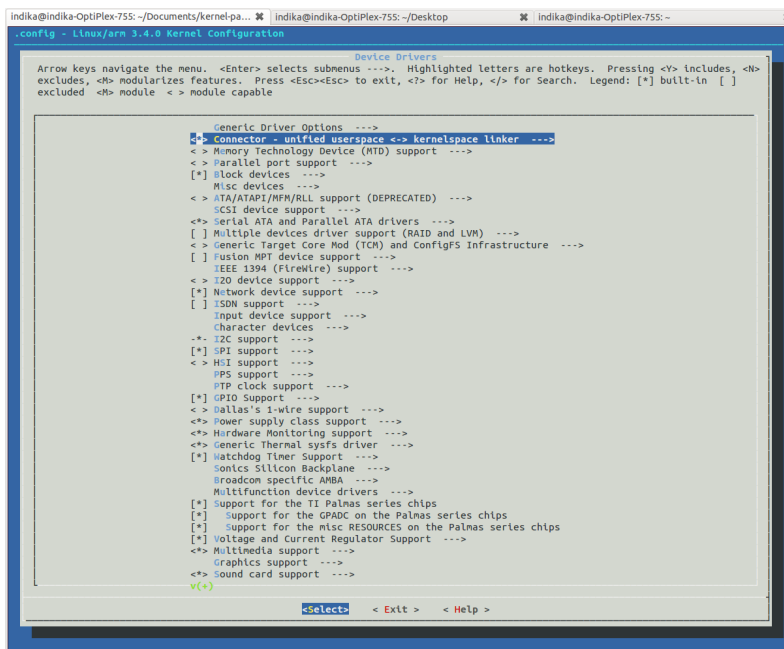


Figure 4.2 How to choose a driver

I select the PPP drivers because these drivers will be required by the 3G modem to connect the Internet. I select the wireless LAN drivers because I need to use the





## 4.2 Install important software

### 4.2.1 Set up 3G modem

In my design, I need a 3G modem to connect to the Internet so that the local data on boat can be transmitted to the remote server. In order to achieve that, I need to install software called “wvdial” and this is a popular method to connect the 3G modem to the Internet [16]. After installing the software, I need to edit the configuration file. The command lines are given in below. I need to install the wvdial first and then edit the configuration file for configuration purpose.

```
$sudo apt-get install wvdial  
$sudo gedit /etc/wvdial.conf
```

The content of the configuration file is shown below.

```
[Dialer Defaults]  
APN = telenor  
Init1 = AT+CPIN=4384  
Init2 = ATZ  
Init3 = ATQ0 V1 E1 S0=0 &C1 &D2  
Init4 = AT+CGDCONT=1,"IP","telenor"  
Modem Type = Analog Modem  
Phone = *99#  
ISDN = 0  
Password = <>  
New PPPD = yes  
Username = <>  
Modem = /dev/gsmmodem  
Baud = 9600  
Auto DNS = on  
Auto route = on
```

The line “Modem = /dev/gsmmodem” specifies the serial port for the 3G modem. This port is actually the nickname created by following [17].

To connect the 3G modem, just type “wvdial” in the terminal and the Internet will be connected, as shown in Figure 4.4. The constant light means the connection is successful.



Figure 4.4 Successful connection of the 3G modem

### 4.2.2 Install MySQL

I will store all the data into MySQL databases. On the Pandaboard, I install both MySQL server and client. I also need to install a MySQL library that will be used by C programs. On the server side, I need to install the MySQL server and client.

#### On Pandaboard

The installation of MySQL is done by the following command lines.

```
$sudo apt-get install mysql-server  
$sudo apt-get install mysql-client  
$sudo apt-get install libmysqlclient-dev
```

The first line installs the MySQL server, the second line installs the MySQL client and the last line installs the library that will be used by C programs. If a C program needs to insert data into the MySQL database, then this library is a necessity.

When all those programs are installed, I can log into the MySQL server and create databases and tables. The following command lines show that I create two databases, named by NMEA and IMU respectively.

```
$mysql -uroot -p  
$create database NMEA;  
$create database IMU;
```

The first line enables us to log into the MySQL server, the second line creates a database “NMEA” that will keep all the data from the NMEA network. The last line creates a database “IMU” that keeps all the data from the IMU sensor. After creating these databases, I also need to create tables for each database. The command lines are given below.

First, create a table “device” in the database “NMEA”.

```
$use NMEA;
```

```
$create table device(  
id int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,  
src int(10) UNSIGNED NULL  
);
```

Then create a table "imu" in the database "IMU".

```
$use IMU;  
$CREATE TABLE imu(  
id int(50) NOT NULL PRIMARY KEY auto_increment,  
Ax varchar(100) NOT NULL,  
Ay varchar(100) NOT NULL,  
Az varchar(100) NOT NULL,  
Mx varchar(100) NOT NULL,  
My varchar(100) NOT NULL,  
Mz varchar(100) NOT NULL,  
Gx varchar(100) NOT NULL,  
Gy varchar(100) NOT NULL,  
Gz varchar(100) NOT NULL,  
No varchar(100) NOT NULL,  
date datetime NOT NULL  
);
```

After finishing the above command lines, I can move to the server side.

### **MySQL server configurations at the remote server**

First, I need to install the MySQL server and client. This is the same as that in 3.4.1. After installing the server and the client, I can create databases and tables. The command lines are given below.

```
$mysql -uroot -p  
$create database NMEA;  
$use NMEA;  
$CREATE TABLE canboat(  
id int(11) NOT NULL PRIMARY KEY auto_increment,  
boatname varchar(8) NULL,  
date date NULL,  
time time NULL,  
src int(10) UNSIGNED NULL,  
dst int(10) UNSIGNED NULL,  
prio int(10) UNSIGNED NULL,  
pgn int(10) UNSIGNED NULL,  
data varchar(8200) NULL  
);
```

After creating the database and the table, there is one important step that I need to do. Because I want the data to be transmitted from the Pandaboard to the remote server side, the Pandaboard needs a privilege to access the remote server. The command lines to achieve this goal are given below.

```
$mysql -uroot -p
$use mysql;
$update user set host='%' where user='root' and host='localhost';
$flush privileges;
$exit
$sudo gedit /etc/mysql/my.cnf
change bind-address from 127.0.0.1 to server's current IP address.
```

The last line above binds the IP address of the server to the MySQL, so that from now on I should log into the MySQL server with the following command.

```
$mysql -uroot -h SERVER_IP -p
```

### 4.2.3 Access point configuration and wireless access

In order to access the Pandaboard from a local host wirelessly, I will configure the Pandaboard as an access point with DHCP enabled. As a result, when I want to control the Pandaboard locally, I just need to connect to the access point and then use the SSH to log into the Pandaboard. The access point will have a fixed private IP address.

#### Access point configuration

The command lines to configure the access point are given below.

```
$sudo apt-get update
$sudo apt-get install hostapd
$sudo apt-get install udhcpd
$sudo gedit /etc/hostapd.conf
$sudo gedit /etc/udhcpd.conf
```

The first line is necessary and the remaining installation will fail otherwise. The second and the third lines install two programs. The “hostapd” can configure the Pandaboard as an access point and the “udhcpd” can provide the DHCP function so that the access point can assign each user a different IP address. The last two lines edit two configuration files. The content of the “hostapd.conf” is given below [18].

```
interface=wlan0
driver=nl80211
ssid=Pandaboard1
```

```
hw_mode=g
channel=11
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

In my design, I name the access point “PandaboardN” where the “N” is sequential number to distinguish different boards and the password is defined via “wpa\_passphrase=12345678”. The IP address of the access point is the fixed “192.168.0.1” and the user’s IP address ranges from “192.168.0.2” to “192.168.0.20”, which means that at most nineteen users can access the Pandaboard at the same time. The password for the access point is set up via “wpa\_passphrase=12345678”. It is obvious that the range can be changed easily. The content of the “udhcpd.conf” is given in the Appendix B.

### Wireless access

After configuring the access point, I need to install the “openssh-server” on the Pandaboard so that the user of the access point can log into the Pandaboard from the ssh command line. The command line to install the openssh is given below.

```
$sudo apt-get install openssh-server
```

After that, I can find the access point call “Pandaboard X” from another laptop, as shown in Figure 4.5.

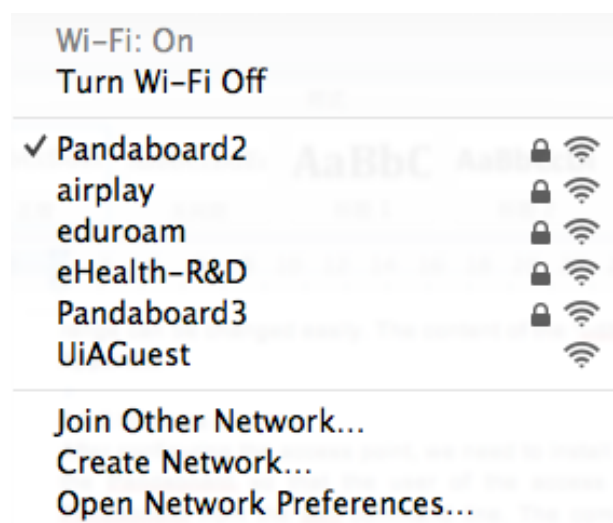


Figure 4.5 Access point is found

I can connect to the access point with the password defined above, which in my case is “12345678”. After entering the password, the access point will assign an IP address to the laptop automatically. Then I can access the Pandaboard wirelessly with the following command line [19].

```
$ssh "username"@192.168.0.1
```

The “username” in the above command is the username of the Ubuntu in the Pandaboard. You will also be asked for the password for the “username”. The password is the same as the one used to log into the Pandaboard. If everything goes Ill, now I have accessed the Pandaboard wirelessly successfully. By doing so, I have a complete control of the Pandaboard from another computer, as shown in Figure 4.6.

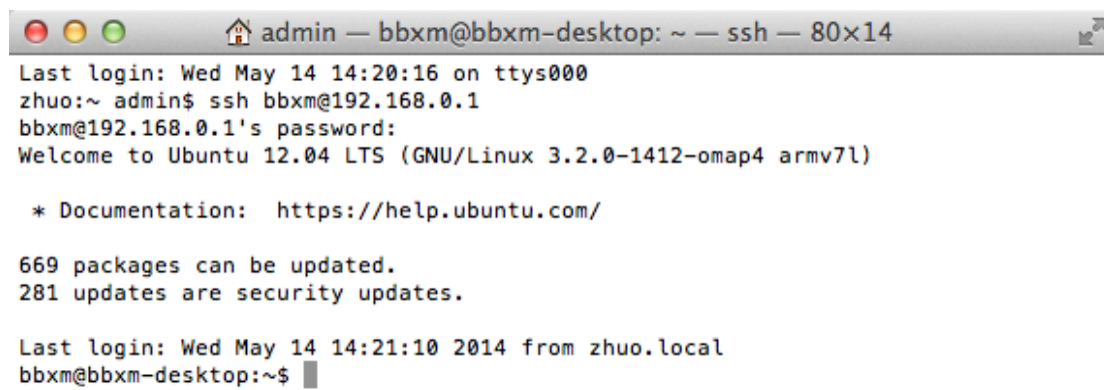
A terminal window titled 'admin — bbxm@bbxm-desktop: ~ — ssh — 80x14'. The terminal output shows the user 'zhuo' at the desktop running 'ssh bbxm@192.168.0.1'. The password is accepted, and the user is logged into the Pandaboard as 'bbxm'. The terminal displays the Ubuntu 12.04 LTS login banner, including the version '3.2.0-1412-omap4 armv7l', documentation link 'https://help.ubuntu.com/', and update information: '669 packages can be updated. 281 updates are security updates.' The session ends with the prompt 'bbxm@bbxm-desktop:~\$'.

Figure 4.6 Successful log in from another host

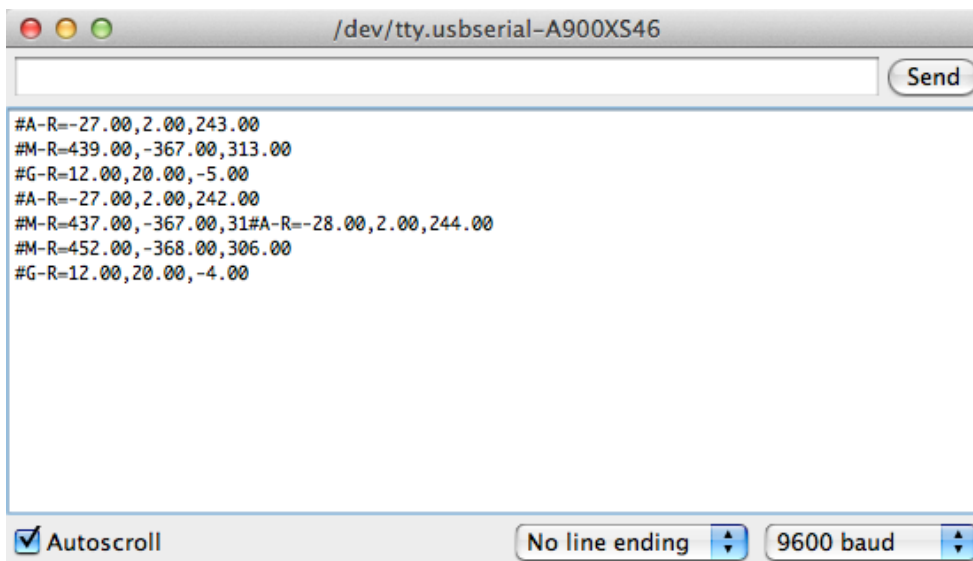
## 4.3 Implement C programs and scripts

### 4.3.1 C programs for IMU and NMEA

The programs for WSN and CAN have already been designed by another two group of students [1] [2]. As a result, I will only introduce the programs for IMU and NMEA in this report.

#### C program for IMU

In order to collect data from the IMU sensor, I write a C program. This program will read data from the serial port and insert the data into the MySQL database “IMU”. Initially, the IMU module will output data shown in Figure 4.7.



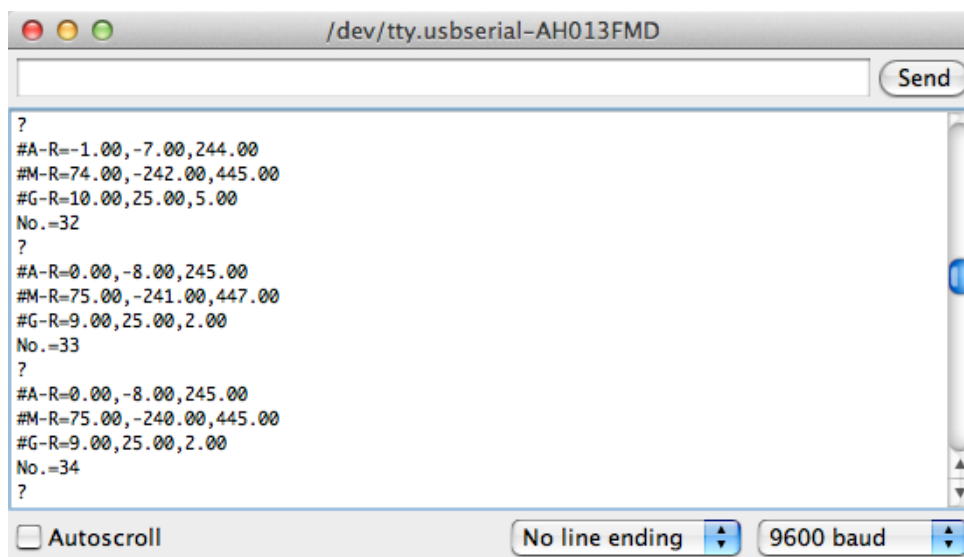
```

/dev/tty.usbserial-A900XS46
#A-R=-27.00,2.00,243.00
#M-R=439.00,-367.00,313.00
#G-R=12.00,20.00,-5.00
#A-R=-27.00,2.00,242.00
#M-R=437.00,-367.00,31#A-R=-28.00,2.00,244.00
#M-R=452.00,-368.00,306.00
#G-R=12.00,20.00,-4.00

```

Figure 4.7 Initial version of IMU output

In order to mark the end of a data set, I attach a special char symbol “?” at the end of every set of output. This is done by the codes uploaded to the IMU processor. And the new version of IMU output is shown in Figure 4.8.



```

/dev/tty.usbserial-AH013FMD
?
#A-R=-1.00,-7.00,244.00
#M-R=74.00,-242.00,445.00
#G-R=10.00,25.00,5.00
No.=32
?
#A-R=0.00,-8.00,245.00
#M-R=75.00,-241.00,447.00
#G-R=9.00,25.00,2.00
No.=33
?
#A-R=0.00,-8.00,245.00
#M-R=75.00,-240.00,445.00
#G-R=9.00,25.00,2.00
No.=34
?

```

Figure 4.8 New version of IMU output

The most important part of the codes to produce the above output format is given in the Appendix C.

On the Pandaboard, a C program is used to collect the data from the IMU sensor. A flow chart in Figure 4.9 shows how this program works. A part of this program is given in the Appendix D.

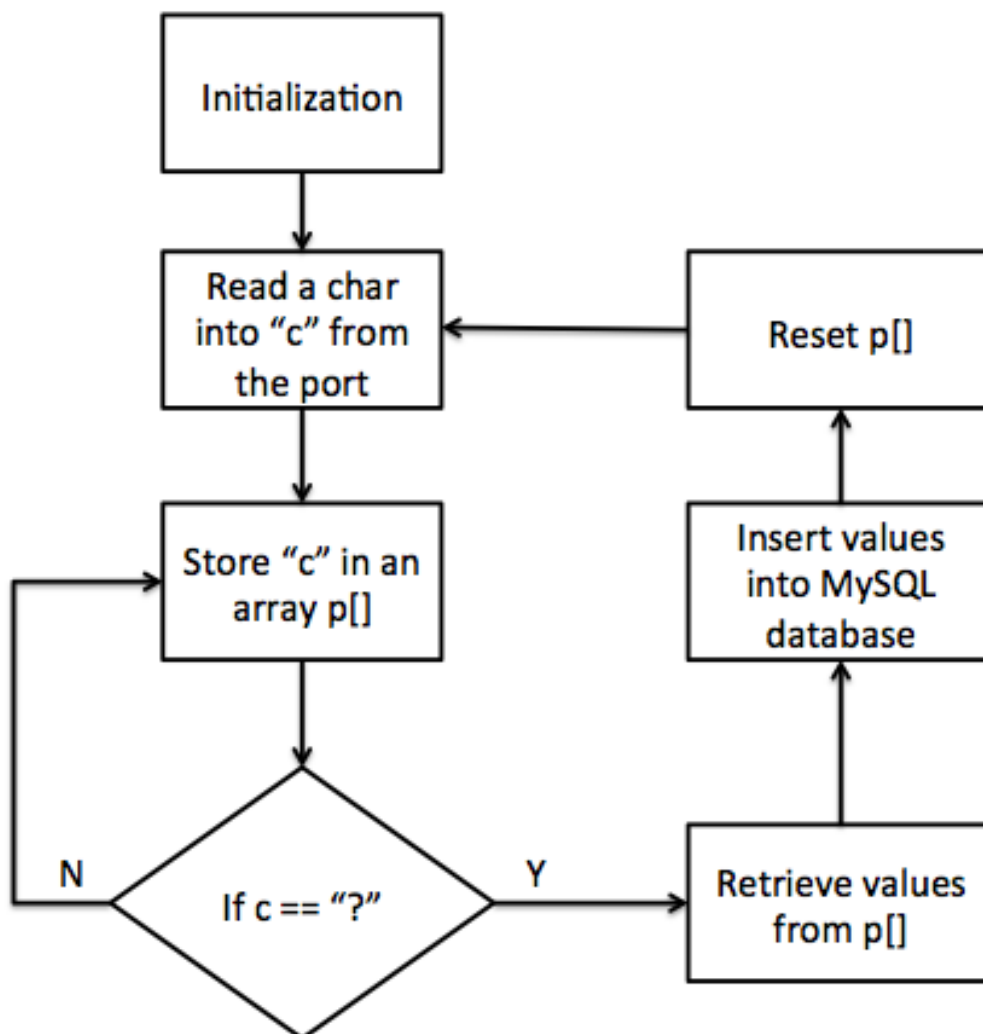


Figure 4.9 Flow chart of the IMU data collection program

This is a while-loop that keeps listening to the specific serial port. If there is data coming, the data will be saved in an array. If the data is the char "?", the information will be derived from the array and saved into the database with the integer data type. Because the data rate from the IMU sensor is very high, I only save the data into the database on the Pandaboard instead of sending it to the remote server.

The above C program can store the data into MySQL database. And the data in the MySQL database is shown in Figure 4.10.

After the booting of the IMU sensor, it will start producing data, and along with data I add a sequence number with each group of data. This sequence number can be used as the time stamp for the data. At the receiver side, I can tell whether any data is lost by checking the sequence numbers.



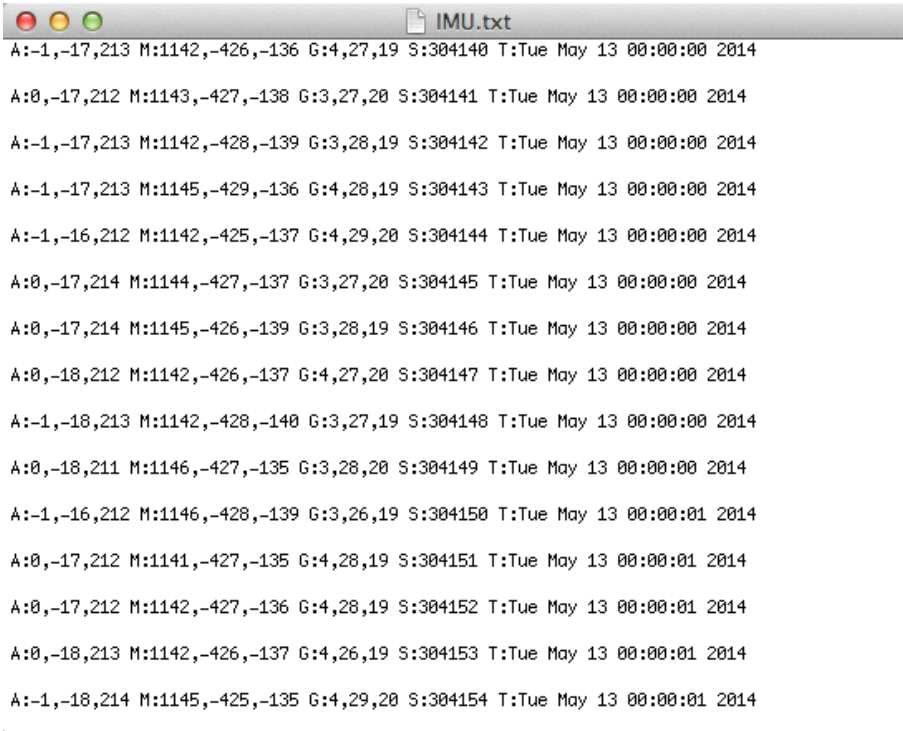
13237	-2	-1	206	780	-18	-423	19	67	4	15509	2014-04-04 12:28:07
13238	-3	-1	206	780	-20	-424	19	67	5	15510	2014-04-04 12:28:07
13239	-3	-1	206	780	-17	-426	18	67	6	15511	2014-04-04 12:28:07
13240	-3	1	206	781	-20	-422	19	68	6	15512	2014-04-04 12:28:07
13241	-2	0	205	777	-18	-422	21	67	5	15513	2014-04-04 12:28:07
13242	-2	2	208	780	-18	-423	19	67	5	15514	2014-04-04 12:28:07
13243	-5	0	205	782	-20	-422	18	67	5	15515	2014-04-04 12:28:07
13244	-2	0	206	777	-23	-423	18	66	4	15516	2014-04-04 12:28:07
13245	-3	0	204	778	-19	-423	19	66	4	15517	2014-04-04 12:28:08
13246	-5	0	207	778	-22	-425	18	66	5	15518	2014-04-04 12:28:08
13247	-5	-1	205	779	-20	-424	20	67	5	15519	2014-04-04 12:28:08
13248	-3	-1	204	777	-23	-424	19	68	4	15520	2014-04-04 12:28:08
13249	-3	-1	207	777	-22	-423	19	68	5	15521	2014-04-04 12:28:08
13250	-2	0	205	777	-20	-425	19	67	6	15522	2014-04-04 12:28:08
13251	-2	0	206	780	-18	-422	18	67	4	15523	2014-04-04 12:28:08
13252	-5	1	207	778	-20	-423	19	67	3	15524	2014-04-04 12:28:08
13253	-1	-1	207	779	-21	-423	19	67	4	15525	2014-04-04 12:28:08
13254	-5	-2	206	777	-18	-423	19	68	5	15526	2014-04-04 12:28:08
13255	-3	0	207	778	-22	-422	19	67	6	15527	2014-04-04 12:28:08
13256	-4	-1	204	779	-19	-422	19	66	5	15528	2014-04-04 12:28:09
13257	-3	0	207	777	-20	-425	19	66	5	15529	2014-04-04 12:28:09
13258	-4	0	205	779	-19	-422	18	66	5	15530	2014-04-04 12:28:09
13259	-3	0	206	779	-18	-421	18	66	4	15531	2014-04-04 12:28:09
13260	-4	-1	204	779	-20	-418	19	68	5	15532	2014-04-04 12:28:09
13261	-2	-1	205	778	-22	-426	18	68	5	15533	2014-04-04 12:28:09
13262	-3	1	207	780	-19	-424	19	69	5	15534	2014-04-04 12:28:09
13263	-5	1	205	778	-21	-423	19	67	5	15535	2014-04-04 12:28:09
13264	-4	0	206	776	-24	-422	18	67	5	15536	2014-04-04 12:28:09
13265	-5	0	205	779	-20	-425	20	66	5	15537	2014-04-04 12:28:09
13266	-3	0	207	782	-18	-425	19	67	5	15538	2014-04-04 12:28:10
13267	-1	1	207	782	-22	-422	18	67	5	15539	2014-04-04 12:28:10
13268	-3	0	207	780	-17	-426	19	66	4	15540	2014-04-04 12:28:10
13269	-6	0	207	778	-19	-423	20	67	5	15541	2014-04-04 12:28:10
13270	-2	0	207	778	-18	-424	18	66	4	15542	2014-04-04 12:28:10
13271	-5	0	204	776	-20	-423	18	67	5	15543	2014-04-04 12:28:10
13272	-2	0	205	781	-20	-422	19	68	4	15544	2014-04-04 12:28:10
13273	-4	1	203	779	-23	-424	19	66	6	15545	2014-04-04 12:28:10
13274	-4	0	204	776	-19	-425	18	66	6	15546	2014-04-04 12:28:10
13275	-1	1	207	779	-21	-426	19	67	4	15547	2014-04-04 12:28:10
13276	-4	1	206	780	-20	-426	18	67	5	15548	2014-04-04 12:28:10
13277	-3	1	207	777	-18	-424	19	67	5	15549	2014-04-04 12:28:11
13278	-3	0	205	782	-21	-421	19	66	4	15550	2014-04-04 12:28:11
13279	-4	-1	204	780	-20	-422	19	67	5	15551	2014-04-04 12:28:11
13280	-5	0	203	782	-19	-422	18	67	4	15552	2014-04-04 12:28:11
13281	-3	-1	204	780	-19	-423	19	67	5	15553	2014-04-04 12:28:11
13282	-4	0	204	779	-18	-422	18	67	4	15554	2014-04-04 12:28:11
13283	-3	0	206	780	-19	-423	18	66	5	15555	2014-04-04 12:28:11
13284	-3	0	206	780	-22	-427	19	67	5	15556	2014-04-04 12:28:11
13285	-4	0	204	778	-20	-422	18	66	5	15557	2014-04-04 12:28:11
13286	-4	0	206	779	-21	-422	19	67	4	15558	2014-04-04 12:28:11
13287	-2	1	207	778	-22	-425	19	67	4	15559	2014-04-04 12:28:11
13288	-5	-1	203	780	-21	-422	19	66	4	15560	2014-04-04 12:28:12
13289	-2	0	205	778	-24	-420	19	67	5	15561	2014-04-04 12:28:12

Figure 4.10 Part of IMU data in database

Another method to store the data is to log the data into a text. The reason why I want to store the data into a text file is that the storing speed for the MySQL is not fast enough, I might have data loss. The most important change of the code is given below.

```
FILE *f = fopen("/media/DATA/IMU.txt", "a+");
if (f == NULL)
{
    printf("Error opening file!\n");
    exit(1);
}
current_time = time(NULL);
c_time_string = ctime(&current_time);
//fprintf(f, "A:%d,%d,%d M:%d,%d,%d G:%d,%d,%d S:%lu T:%s\n", ax, ay, az, mx,
my, mz, gx, gy, gz, no,c_time_string);
fclose(f);
```

The first line of the above program will open a file called "IMU.txt" and if the file does not exist, a new file with the same name will be created. And if the file is already there, then a new line will be attached into the content of the file [20]. The data in the IMU.txt is shown in Figure 4.11.



```

A:-1,-17,213 M:1142,-426,-136 G:4,27,19 S:304140 T:Tue May 13 00:00:00 2014
A:0,-17,212 M:1143,-427,-138 G:3,27,20 S:304141 T:Tue May 13 00:00:00 2014
A:-1,-17,213 M:1142,-428,-139 G:3,28,19 S:304142 T:Tue May 13 00:00:00 2014
A:-1,-17,213 M:1145,-429,-136 G:4,28,19 S:304143 T:Tue May 13 00:00:00 2014
A:-1,-16,212 M:1142,-425,-137 G:4,29,20 S:304144 T:Tue May 13 00:00:00 2014
A:0,-17,214 M:1144,-427,-137 G:3,27,20 S:304145 T:Tue May 13 00:00:00 2014
A:0,-17,214 M:1145,-426,-139 G:3,28,19 S:304146 T:Tue May 13 00:00:00 2014
A:0,-18,212 M:1142,-426,-137 G:4,27,20 S:304147 T:Tue May 13 00:00:00 2014
A:-1,-18,213 M:1142,-428,-140 G:3,27,19 S:304148 T:Tue May 13 00:00:00 2014
A:0,-18,211 M:1146,-427,-135 G:3,28,20 S:304149 T:Tue May 13 00:00:00 2014
A:-1,-16,212 M:1146,-428,-139 G:3,26,19 S:304150 T:Tue May 13 00:00:01 2014
A:0,-17,212 M:1141,-427,-135 G:4,28,19 S:304151 T:Tue May 13 00:00:01 2014
A:0,-17,212 M:1142,-427,-136 G:4,28,19 S:304152 T:Tue May 13 00:00:01 2014
A:0,-18,213 M:1142,-426,-137 G:4,26,19 S:304153 T:Tue May 13 00:00:01 2014
A:-1,-18,214 M:1145,-425,-135 G:4,29,20 S:304154 T:Tue May 13 00:00:01 2014

```

Figure 4.11 Data in IMU.txt

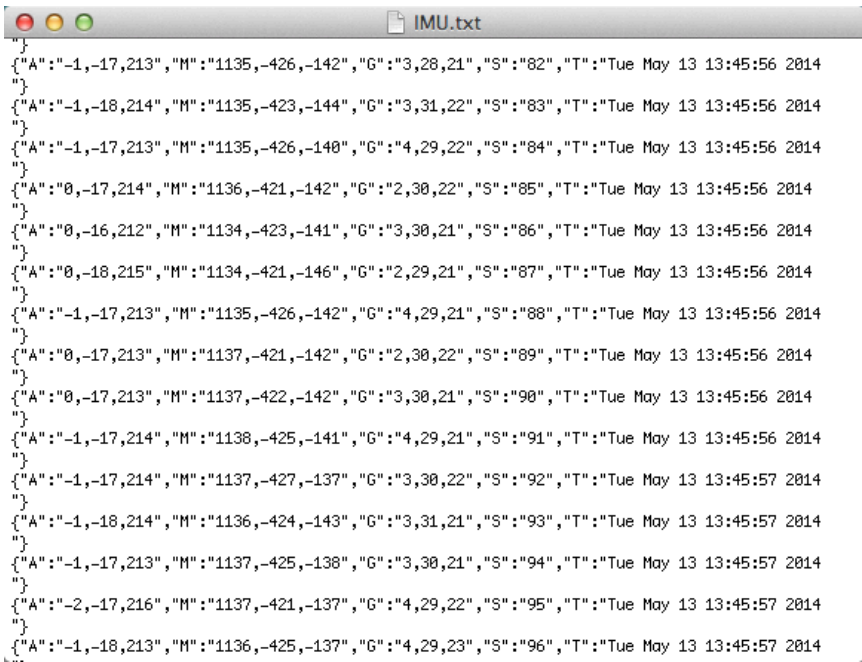
I can also change the data into json format by using the following codes [21].

```

fprintf(f,
"{\"A\": \"%d,%d,%d\", \"M\": \"%d,%d,%d\", \"G\": \"%d,%d,%d\", \"S\": \"%lu\", \"T\": \"%s\"}
\n", ax, ay, az, mx, my, mz, gx, gy, gz, no, c_time_string);

```

Figure 4.12 shows the data in json format.



```

{"A": "-1,-17,213", "M": "1135,-426,-142", "G": "3,28,21", "S": "82", "T": "Tue May 13 13:45:56 2014"}
{"A": "-1,-18,214", "M": "1135,-423,-144", "G": "3,31,22", "S": "83", "T": "Tue May 13 13:45:56 2014"}
{"A": "-1,-17,213", "M": "1135,-426,-140", "G": "4,29,22", "S": "84", "T": "Tue May 13 13:45:56 2014"}
{"A": "0,-17,214", "M": "1136,-421,-142", "G": "2,30,22", "S": "85", "T": "Tue May 13 13:45:56 2014"}
{"A": "0,-16,212", "M": "1134,-423,-141", "G": "3,30,21", "S": "86", "T": "Tue May 13 13:45:56 2014"}
{"A": "0,-18,215", "M": "1134,-421,-146", "G": "2,29,21", "S": "87", "T": "Tue May 13 13:45:56 2014"}
{"A": "-1,-17,213", "M": "1135,-426,-142", "G": "4,29,21", "S": "88", "T": "Tue May 13 13:45:56 2014"}
{"A": "0,-17,213", "M": "1137,-421,-142", "G": "2,30,22", "S": "89", "T": "Tue May 13 13:45:56 2014"}
{"A": "0,-17,213", "M": "1137,-422,-142", "G": "3,30,21", "S": "90", "T": "Tue May 13 13:45:56 2014"}
{"A": "-1,-17,214", "M": "1138,-425,-141", "G": "4,29,21", "S": "91", "T": "Tue May 13 13:45:56 2014"}
{"A": "-1,-17,214", "M": "1137,-427,-137", "G": "3,30,22", "S": "92", "T": "Tue May 13 13:45:57 2014"}
{"A": "-1,-18,214", "M": "1136,-424,-143", "G": "3,31,21", "S": "93", "T": "Tue May 13 13:45:57 2014"}
{"A": "-1,-17,213", "M": "1137,-425,-138", "G": "3,30,21", "S": "94", "T": "Tue May 13 13:45:57 2014"}
{"A": "-2,-17,216", "M": "1137,-421,-137", "G": "4,29,22", "S": "95", "T": "Tue May 13 13:45:57 2014"}
{"A": "-1,-18,213", "M": "1136,-425,-137", "G": "4,29,23", "S": "96", "T": "Tue May 13 13:45:57 2014"}

```

Figure 4.12 IMU data in json format

To further improve the storage method, I design a program to store the data by date. I add the following codes into the C program [22].

```
static char* now(void){
static char str[64];
struct timeval tv;
struct tm *nowtm;
time_t nowtime;
gettimeofday(&tv, NULL);
nowtime = tv.tv_sec;
nowtm = localtime(&nowtime);
strftime(str, sizeof(str), "/media/DATA/IMU%Y%m%d.txt", nowtm);
return str;
}
```

This function will return a pointer that points to a string with the format of “/media/DATA/IMU20140512.txt” In this string, the time is 2014.05.12. It is the time when this function is used. With the help of this function, and combined with the following codes, I can successfully store data into texts by sorted date [20].

```
char *t = now();
FILE *f = fopen(t, "a+");
if (f == NULL)
{
    printf("Error opening file!\n");
    exit(1);
}
current_time = time(NULL);
c_time_string = ctime(&current_time);
fprintf(f,
"{\"A\": \"%d,%d,%d\", \"M\": \"%d,%d,%d\", \"G\": \"%d,%d,%d\", \"S\": \"%lu\", \"T\": \"%s\"}
\n", ax, ay, az, mx, my, mz, gx, gy, gz, no, c_time_string);
fclose(f);
```

The first line define a pointer “t” and the value of “t” is the address of the string “/media/DATA/IMU%Y%m%d.txt”. As a result, the data will always be stored into the text with the newest date. Figure 4.13 shows the files named by date.

```

admin — bbxm@bbxm-desktop: ~ — ssh — 80x34

669 packages can be updated.
281 updates are security updates.

Last login: Wed May 14 14:21:10 2014 from zhuo.local
bbxm@bbxm-desktop:~$ ls -lh /media/DATA/
total 1.5G
drwx----- 2 bbxm bbxm 32K May  5 14:29 IMU
-rw-r--r-- 1 bbxm bbxm 65M May  1 23:59 IMU20140501.txt
-rw-r--r-- 1 bbxm bbxm 42M May  2 15:11 IMU20140502.txt
-rw-r--r-- 1 bbxm bbxm 217K May  3 16:06 IMU20140503.txt
-rw-r--r-- 1 bbxm bbxm 22M May  4 23:59 IMU20140504.txt
-rw-r--r-- 1 bbxm bbxm 38M May  5 14:28 IMU20140505.txt
-rw-r--r-- 1 bbxm bbxm 23M May  8 23:59 IMU20140508.txt
-rw-r--r-- 1 bbxm bbxm 65M May  9 23:59 IMU20140509.txt
-rw-r--r-- 1 bbxm bbxm 66M May 10 23:59 IMU20140510.txt
-rw-r--r-- 1 bbxm bbxm 66M May 11 23:59 IMU20140511.txt
-rw-r--r-- 1 bbxm bbxm 64M May 12 23:59 IMU20140512.txt
-rw-r--r-- 1 bbxm bbxm 67M May 13 23:59 IMU20140513.txt
-rw-r--r-- 1 bbxm bbxm 53M May 14 15:18 IMU20140514.txt
drwx----- 3 bbxm bbxm 32K May  5 14:42 Mingli
-rw-r--r-- 1 bbxm bbxm 116M May  2 15:11 NMEA20140502.txt
-rw-r--r-- 1 bbxm bbxm 90K May  3 16:06 NMEA20140503.txt
-rw-r--r-- 1 bbxm bbxm 44M May  4 23:59 NMEA20140504.txt
-rw-r--r-- 1 bbxm bbxm 79M May  5 14:27 NMEA20140505.txt
-rw-r--r-- 1 bbxm bbxm 9.8M May  8 23:59 NMEA20140508.txt
-rw-r--r-- 1 bbxm bbxm 28M May  9 23:59 NMEA20140509.txt
-rw-r--r-- 1 bbxm bbxm 28M May 10 23:59 NMEA20140510.txt
-rw-r--r-- 1 bbxm bbxm 28M May 11 23:59 NMEA20140511.txt
-rw-r--r-- 1 bbxm bbxm 97M May 12 23:59 NMEA20140512.txt
-rw-r--r-- 1 bbxm bbxm 264M May 13 23:59 NMEA20140513.txt
-rw-r--r-- 1 bbxm bbxm 178M May 14 15:18 NMEA20140514.txt
drwx----- 2 bbxm bbxm 32K May  5 14:29 Scripts
bbxm@bbxm-desktop:~$ █

```

Figure 4.13 Files created by date

## C program for NMEA

I have already had the C program for the NMEA data collection [2]. However, I have two improvements over the previous program. First improvement is the update “now()” function. The “now()” function in the “actisense-serial.c” is not accurate, as it cannot return the right time. The code for the old function is given below.

```

static char * now(void)
{
    static char str[64];
    struct timeval tv;
    struct tm tm;
    str[0] = 0;
    if (gettimeofday(&tv, 0) != 0)
    {
        return "?";
    }
    gmtime_r(&tv.tv_sec, &tm);
    snprintf(str, sizeof(str), "%u", (unsigned int) tv.tv_sec);
    strftime(str, sizeof(str) - 5, "%F-%T", &tm);
}

```

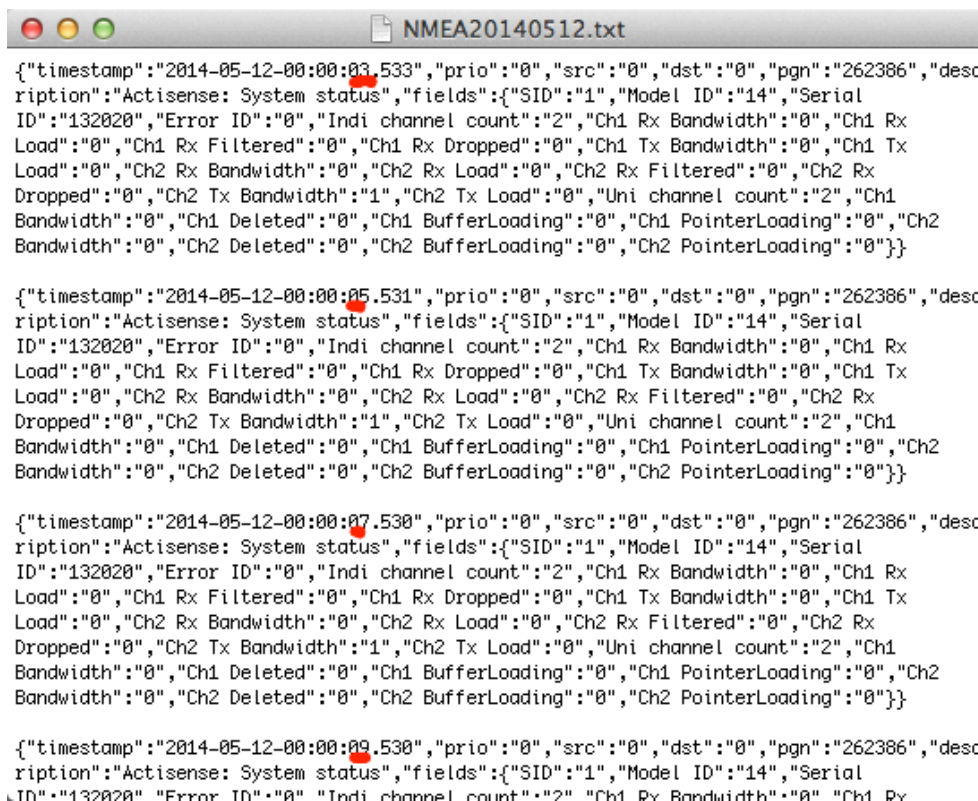
```
snprintf(str + strlen(str), 5, ".%03d", (int) (tv.tv_usec / 1000L));  
return str;  
}
```

The new “now()” function is given below [22].

```
static char * now(void)  
{  
    static char str[64];  
    struct timeval tv;  
    struct tm *nowtm;  
    time_t nowtime;  
    gettimeofday(&tv, NULL);  
    nowtime = tv.tv_sec;  
    nowtm = localtime(&nowtime);  
    strftime(str, sizeof(str), "%Y-%m-%d-%H:%M:%S", nowtm);  
    snprintf(str + strlen(str), 5, ".%03d", (int) (tv.tv_usec / 1000L));  
    return str;  
}
```

The difference between the two “now()” functions is that the latter one uses a “localtime()” function which can output a correct local time.

My second improvement is that I split the “analyzer.c” into two C programs “analyzer-local.c” and “analyzer-remote.c”. The reason is that I want to have a higher data rate for the local storage while have a lower data rate for the remote storage. If I do not split the program into two, the local data storage will be dependent on the remote data storage. As a result, when the interval is very short, for example every 2 seconds, then the remote data storage will have a big delay impact on the local data storage due to the limited data rate of wireless communications for remote storage. To use the output of one process as the input of two other processes, I use the “tee” command. The grammar for the tee command is “process 1 | tee >(process 2) | process 3” [23]. Figure 4.14 shows the NMEA data that is stored locally. As can be seen from this figure, the data is stored every two seconds.



```

{"timestamp": "2014-05-12-00:00:00.533", "prio": "0", "src": "0", "dst": "0", "pgn": "262386", "description": "Actisense: System status", "fields": {"SID": "1", "Model ID": "14", "Serial ID": "132020", "Error ID": "0", "Indi channel count": "2", "Ch1 Rx Bandwidth": "0", "Ch1 Rx Load": "0", "Ch1 Rx Filtered": "0", "Ch1 Rx Dropped": "0", "Ch1 Tx Bandwidth": "0", "Ch1 Tx Load": "0", "Ch2 Rx Bandwidth": "0", "Ch2 Rx Load": "0", "Ch2 Rx Filtered": "0", "Ch2 Rx Dropped": "0", "Ch2 Tx Bandwidth": "1", "Ch2 Tx Load": "0", "Uni channel count": "2", "Ch1 Bandwidth": "0", "Ch1 Deleted": "0", "Ch1 BufferLoading": "0", "Ch1 PointerLoading": "0", "Ch2 Bandwidth": "0", "Ch2 Deleted": "0", "Ch2 BufferLoading": "0", "Ch2 PointerLoading": "0"}}

{"timestamp": "2014-05-12-00:00:00.531", "prio": "0", "src": "0", "dst": "0", "pgn": "262386", "description": "Actisense: System status", "fields": {"SID": "1", "Model ID": "14", "Serial ID": "132020", "Error ID": "0", "Indi channel count": "2", "Ch1 Rx Bandwidth": "0", "Ch1 Rx Load": "0", "Ch1 Rx Filtered": "0", "Ch1 Rx Dropped": "0", "Ch1 Tx Bandwidth": "0", "Ch1 Tx Load": "0", "Ch2 Rx Bandwidth": "0", "Ch2 Rx Load": "0", "Ch2 Rx Filtered": "0", "Ch2 Rx Dropped": "0", "Ch2 Tx Bandwidth": "1", "Ch2 Tx Load": "0", "Uni channel count": "2", "Ch1 Bandwidth": "0", "Ch1 Deleted": "0", "Ch1 BufferLoading": "0", "Ch1 PointerLoading": "0", "Ch2 Bandwidth": "0", "Ch2 Deleted": "0", "Ch2 BufferLoading": "0", "Ch2 PointerLoading": "0"}}

{"timestamp": "2014-05-12-00:00:00.530", "prio": "0", "src": "0", "dst": "0", "pgn": "262386", "description": "Actisense: System status", "fields": {"SID": "1", "Model ID": "14", "Serial ID": "132020", "Error ID": "0", "Indi channel count": "2", "Ch1 Rx Bandwidth": "0", "Ch1 Rx Load": "0", "Ch1 Rx Filtered": "0", "Ch1 Rx Dropped": "0", "Ch1 Tx Bandwidth": "0", "Ch1 Tx Load": "0", "Ch2 Rx Bandwidth": "0", "Ch2 Rx Load": "0", "Ch2 Rx Filtered": "0", "Ch2 Rx Dropped": "0", "Ch2 Tx Bandwidth": "1", "Ch2 Tx Load": "0", "Uni channel count": "2", "Ch1 Bandwidth": "0", "Ch1 Deleted": "0", "Ch1 BufferLoading": "0", "Ch1 PointerLoading": "0", "Ch2 Bandwidth": "0", "Ch2 Deleted": "0", "Ch2 BufferLoading": "0", "Ch2 PointerLoading": "0"}}

{"timestamp": "2014-05-12-00:00:00.530", "prio": "0", "src": "0", "dst": "0", "pgn": "262386", "description": "Actisense: System status", "fields": {"SID": "1", "Model ID": "14", "Serial ID": "132020", "Error ID": "0", "Indi channel count": "2", "Ch1 Rx Bandwidth": "0", "Ch1 Rx Load": "0", "Ch1 Rx Filtered": "0", "Ch1 Rx Dropped": "0", "Ch1 Tx Bandwidth": "0", "Ch1 Tx Load": "0", "Ch2 Rx Bandwidth": "0", "Ch2 Rx Load": "0", "Ch2 Rx Filtered": "0", "Ch2 Rx Dropped": "0", "Ch2 Tx Bandwidth": "1", "Ch2 Tx Load": "0", "Uni channel count": "2", "Ch1 Bandwidth": "0", "Ch1 Deleted": "0", "Ch1 BufferLoading": "0", "Ch1 PointerLoading": "0", "Ch2 Bandwidth": "0", "Ch2 Deleted": "0", "Ch2 BufferLoading": "0", "Ch2 PointerLoading": "0"}}

```

Figure 4.14 Local data from NMEA

Figure 4.15 shows the NMEA data in the remote server.



```

indika@indika-OptiPlex-755: ~/Documents/kernel-pa... indika@indika-OptiPlex-755: ~/Desktop indika@indika-OptiPlex-755: ~
| 244216 | A | 2014-05-11 | 14:54:01 | 6 | 3 | 255 | 129540 | 2014-05-11-14:54:01.285 6 | 3 255 129540 GNSS Sats in V
| SID = 87; Mode = 1; Sats in View = 10; PRN = 1; Elevation = 13.0 deg; Azimuth = 24.0 deg; SNR = 44.48 dB; Range residuals = 0;
| Status = Used; PRN #2 = 11; Elevation #2 = 5.0 deg; Azimuth #2 = 5.0 deg; SNR #2 = 43.21 dB; Range residuals #2 = 0; Status #2 = Used
| PRN #3 = 12; Elevation #3 = 29.0 deg; Azimuth #3 = -156.5 deg; SNR #3 = 44.71 dB; Range residuals #3 = 0; Status #3 = Used; PRN #4
| = 14; Elevation #4 = 15.0 deg; Azimuth #4 = -61.5 deg; SNR #4 = 42.58 dB; Range residuals #4 = 0; Status #4 = Used; PRN #5 = 15; Elev
| ation #5 = 39.0 deg; Azimuth #5 = 179.0 deg; SNR #5 = 43.56 dB; Range residuals #5 = 0; Status #5 = Used; PRN #6 = 17; Elevation #6 =
| 40.0 deg; Azimuth #6 = 85.0 deg; SNR #6 = 49.62 dB; Range residuals #6 = 0; Status #6 = Used; PRN #7 = 18; Elevation #7 = 21.0 deg;
| Azimuth #7 = -110.5 deg; SNR #7 = 44.03 dB; Range residuals #7 = 0; Status #7 = Used; PRN #8 = 22; Elevation #8 = 25.0 deg; Azimuth #
| 8 = -79.5 deg; SNR #8 = 42.62 dB; Range residuals #8 = 0; Status #8 = Used; PRN #9 = 24; Elevation #9 = 75.0 deg; Azimuth #9 = -126.5
| deg; SNR #9 = 47.67 dB; Range residuals #9 = 0; Status #9 = Used; PRN #10 = 28; Elevation #10 = 16.0 deg; Azimuth #10 = 55.0 deg; SN
| R #10 = 40.54 dB; Range residuals #10 = 0; Status #10 = Used
|
| 244217 | A | 2014-05-11 | 14:54:01 | 7 | 2 | 255 | 126720 | 2014-05-11-14:54:01.411 7 | 2 255 126720 Manufacturer P
| roprietary: Addressable Multi-Frame: Manufacturer Code = Garmin; Industry Code = Marine
|
| 244218 | A | 2014-05-11 | 14:54:03 | 7 | 3 | 255 | 126720 | 2014-05-11-14:54:03.326 7 | 3 255 126720 Manufacturer P
| roprietary: Addressable Multi-Frame: Manufacturer Code = Garmin; Industry Code = Marine
|
| 244219 | A | 2014-05-11 | 14:54:03 | 0 | 0 | 0 | 262386 | 2014-05-11-14:54:03.647 0 | 0 0 262386 Actisense: Sys
| tem status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi channel count = 2; Ch1 Rx Bandwidth = 6; Ch1 Rx Load = 2;
| Ch1 Rx Filtered = 0; Ch1 Rx Dropped = 0; Ch1 Tx Bandwidth = 0; Ch1 Tx Load = 0; Ch2 Rx Bandwidth = 0; Ch2 Rx Load = 0; Ch2 Rx Filter
| ed = 0; Ch2 Rx Dropped = 0; Ch2 Tx Bandwidth = 13; Ch2 Tx Load = 90; Uni channel count = 2; Ch1 Bandwidth = 3; Ch1 Deleted = 0; Ch1 B
| ufferLoading = 0; Ch1 PointerLoading = 0; Ch2 Bandwidth = 4; Ch2 Deleted = 0; Ch2 BufferLoading = 0; Ch2 PointerLoading = 0
|
| 244220 | A | 2014-05-11 | 14:54:31 | 2 | 2 | 255 | 127250 | 2014-05-11-14:54:31.011 2 | 2 255 127250 Vessel Heading
| : Heading = 16.8 deg; Reference = Magnetic

```

Figure 4.15 NMEA data in the remote server

### 4.3.2 Mapping ports

In this part, I am going to introduce how to make an USB stick auto mount and how to map the port of each device with its nickname.

#### Auto mount USB flash drive

After compiling the kernel there is no program to auto mount the USB flash drive, I make use of a rule that can mount the USB drive automatically. First, I need to create a rule in a proper place, so that every time the system boots, the rule will be executed. I create the file with the following command line.

```
$sudo nano /etc/udev/rules.d/automount.rules
```

Then, I write a script in this file which has the functions shown in Figure 4.16. The whole script can be found in the Appendix A.

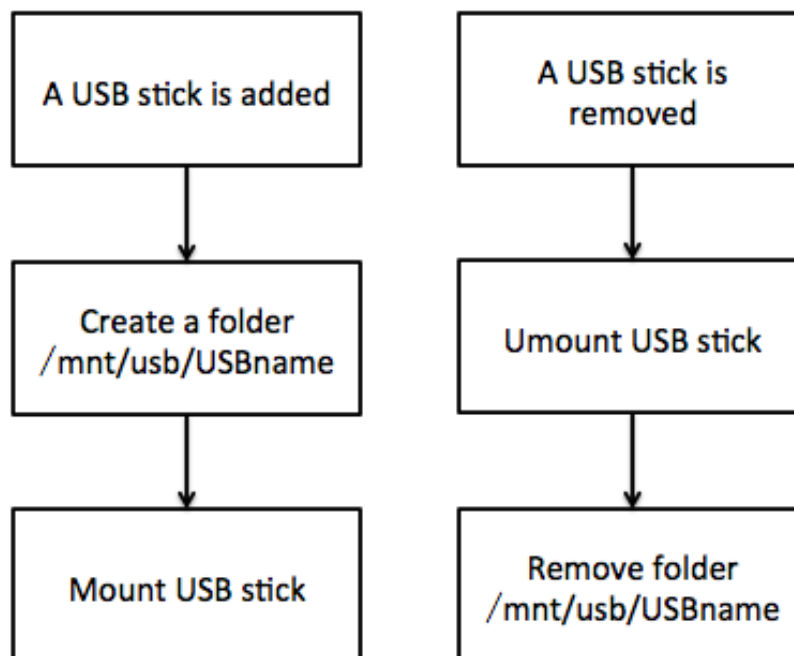


Figure 4.16 Illustration of USB stick auto mount program

After saving the above file, the USB flash drive will be mounted automatically during the next booting.

#### Create ports for NGT and IMU

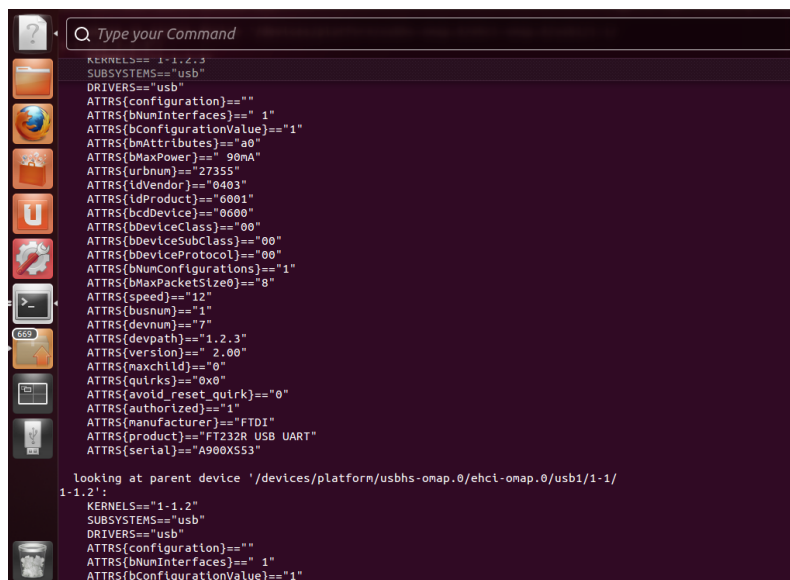
By default, the system will assign a port name for the NGT gateway and the IMU sensor. And the port name looks like “ttyUSB\*” under the “/udev/” directory. This will cause a problem that I cannot tell which port belongs to NGT gateway and which belongs to IMU sensor. To solve this problem, I will give a nickname for each serial port.





For different devices the above information is different, all the information can be found from the following command line. Figure 4.18 illustrates a portion of the results of this command line [24].

`$sudevadm info -a -n /dev/ttyUSB*` #the "\*" is the number assigned by the system.



```

Type your Command
KERNELS=="1-1.2.3"
SUBSYSTEMS=="usb"
DRIVERS=="usb"
ATTRS(configuration)==" "
ATTRS(bNumInterfaces)=="1"
ATTRS(bConfigurationValue)=="1"
ATTRS(bmAttributes)=="a0"
ATTRS(bMaxPower)=="90mA"
ATTRS(urbnum)=="27355"
ATTRS(idVendor)=="0403"
ATTRS(idProduct)=="6001"
ATTRS(bcdDevice)=="0600"
ATTRS(bDeviceClass)=="00"
ATTRS(bDeviceSubclass)=="00"
ATTRS(bDeviceProtocol)=="00"
ATTRS(bNumConfigurations)=="1"
ATTRS(bMaxPacketSize0)=="8"
ATTRS(speed)=="12"
ATTRS(busnum)=="1"
ATTRS(devnum)=="7"
ATTRS(devpath)=="1.2.3"
ATTRS(version)=="2.00"
ATTRS(maxch1ld)=="0"
ATTRS(quirks)=="0x0"
ATTRS(avoid_reset_quirk)=="0"
ATTRS(authorized)=="1"
ATTRS(manufacturer)=="FTDI"
ATTRS(product)=="FT232R USB UART"
ATTRS(serial)=="A900X553"

looking at parent device '/devices/platform/usbhs-omap.0/ehci-omap.0/usb1/1-1/1-1.2':
KERNELS=="1-1.2"
SUBSYSTEMS=="usb"
DRIVERS=="usb"
ATTRS(configuration)==" "
ATTRS(bNumInterfaces)=="1"
ATTRS(bConfigurationValue)=="1"

```

Figure 4.18 Device information

### 4.3.3 Watchdog

I use the Arduino Uno as a watchdog. The watchdog is connected to the output pin of a voltage regulator. And the output voltage is 7 Volts according to the requirement of Arduino Uno. The watchdog will control a relay, if the watchdog gives the relay a HIGH level, then the relay will be enabled, otherwise the relay is disabled. Another voltage regulator will input 5 Volts into the input pin of the relay, so that if the relay is enabled the 5 Volts will be passed to the Pandaboard. The watchdog also has direct interaction with the Pandaboard too. When the Pandaboard is working, it will send a periodical signal to the watchdog to indicate that it is alive. When the watchdog receives the signal from the Pandaboard, it will reset a timer that is counting inside the watchdog. The direct interaction between the Pandaboard and the watchdog is through three GPIO Pins. The flow chart of the watchdog program is shown below in Figure 4.19

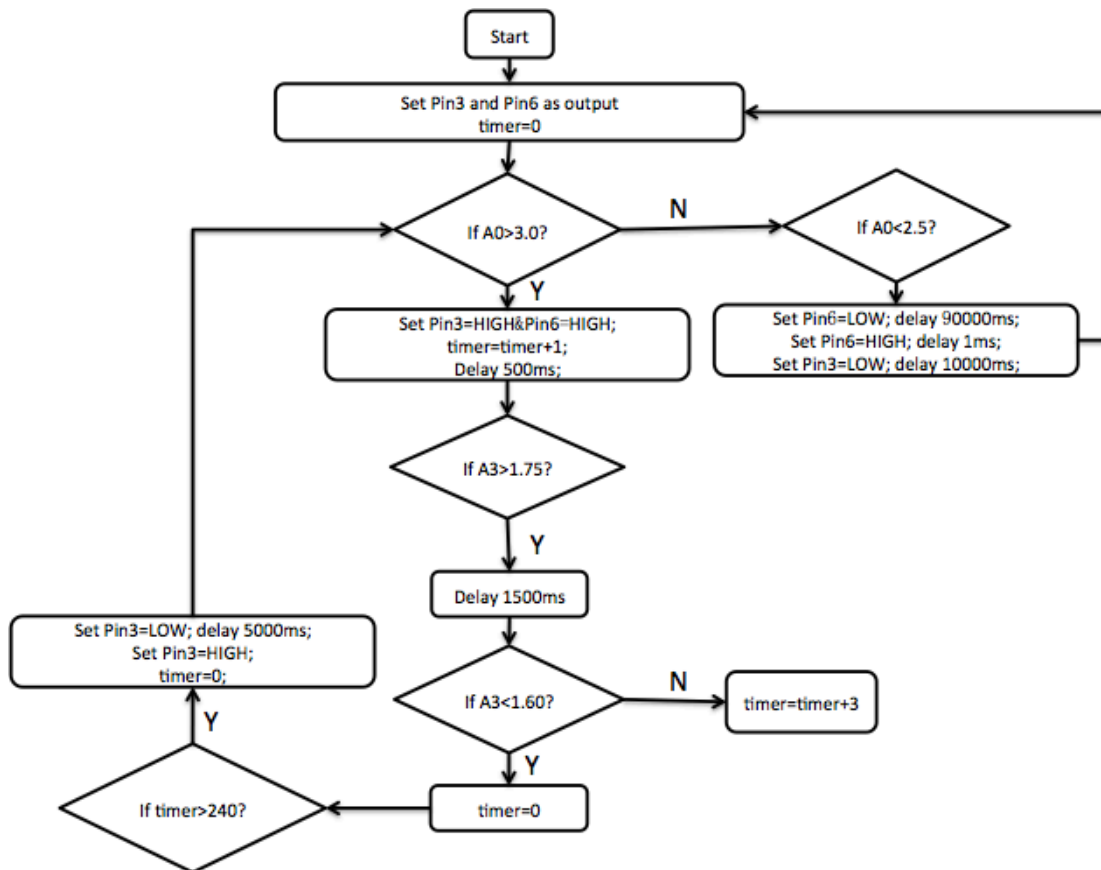


Figure 4.19 Flow chart of the watchdog program

The full program is given in the Appendix E.

Here is the explanation for this program. When the watchdog is powered on, it will check if the ON/OFF button is pressed on. This is achieved by checking the voltage value of pin A0. If the button is pressed on, then the watchdog will set the pin3 as HIGH. Then the relay will be enabled, so that the Pandaboard is powered on. At the same time a HIGH signal will be sent to the pin6 of the watchdog that is connected to the pin20 of J6 on the Pandaboard. The function of pin20 of J6 will be explained later. The timer on the watchdog starts to count now, and it will add one to itself every 500ms. If a high level from A3 is detected, then after 1500ms the watchdog will check A3 again, and if the level is low now, then the watchdog is confident that the Pandaboard is working fine now and it will reset the timer to zero. If the timer is now reset to zero within two minutes then the watchdog believes that the Pandaboard has crashed, and it will disable the relay in order to cut the power to the Pandaboard and enable the relay again after 5 seconds to give the power to Pandaboard again. If the value of the A0 is less than 2.5 Volts, then the Pandaboard will not be powered. I use 2.5 Volts instead of 3 Volts to avoid the impact of the floating of the voltage level. For example, if the level drops below 3 Volts during working condition but nobody presses the OFF button, then the system will not be shut down. If the level is below 2.5 Volts, then it is more likely that the OFF button is pressed, so that the Pandaboard will be shut down.

The watchdog program cooperates with two shell scripts in the Pandaboard. One script is called watchdog.sh, and the function of this script is shown in Figure 4.20. The content of this script is given in the Appendix F.

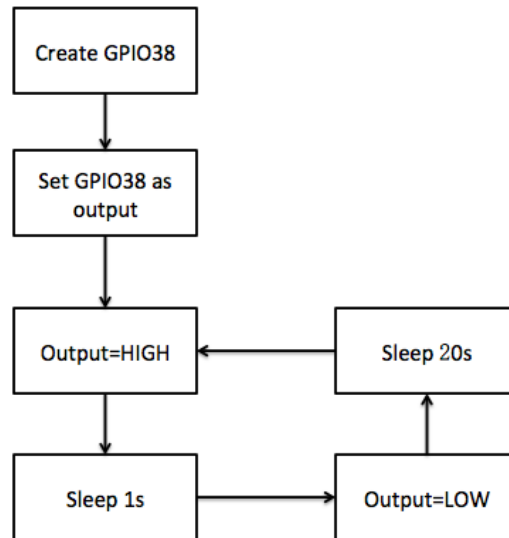


Figure 4.20 Flow chart of the watchdog script

This shell script makes use of the GPIO38. It will send a high level to the watchdog every 21 seconds and the high level lasts 1 second every time. This shell script is used to reset the timer in the watchdog.

Another shell script is called polr.sh, and the flow chart of this script is shown in Figure 4.21. The content of this script can be found in the Appendix G.

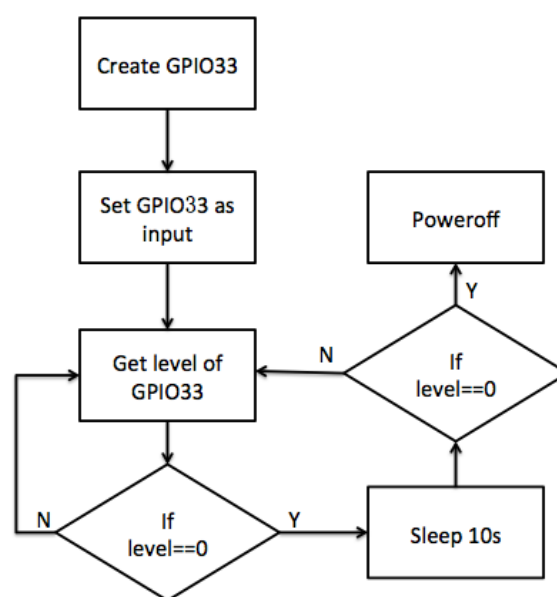


Figure 4.21 Flow chart of the polr.sh script

This shell script utilizes GPIO33. If the watchdog wants to ask Pandaboard to shut down, it will send a low level to the GPIO33 and the low level will last more than 10 seconds. When the Pandaboard detects the low level of the GPIO33, it will wait for 10 seconds and detects again. If the level of GPIO33 is still low then the Pandaboard will shut down itself.

#### 4.4 Set up hardware

Setting up hardware means to connect all hardware together. It is easy to connect the WSN gateway, CAN gateway, NMEA 2000 gateway to the USB hub via just USB cables. Here, I am going to explain how to connect the watchdog to the Pandaboard. The connection method is given below.

Table 4.1 Connection betlen watchdog and Pandaboard

Arduino Uno	Pandaboard
GND	Pin7 of J6
A3	Pin9 of J6
Pin6	Pin20 of J6

Table 4.2 Connection betlen watchdog and relay

Arduino Uno	Relay
GND	DC-
5V	DC+
Pin3	In

Table 4.3 Connection betlen watchdog and polr button

Arduino Uno	Polr Button
3.3V	DC+
A0	DC- & LED+
GND	LED-

#### 4.5 Start up scripts

In order to run all programs automatically, I have designed a few shell scripts to do that. These scripts will be executed during the system booting, and these scripts will start the programs.

The script to start the NMEA program is given in the Appendix H, and the flow chart of this script is given in Figure 4.22.

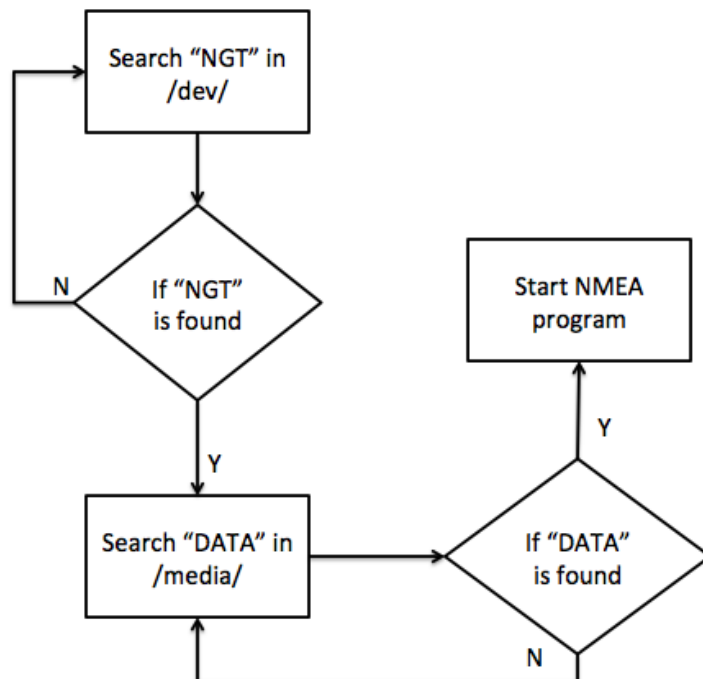


Figure 4.22 Flow chart of the startup script for NMEA

This script will check if the NGT gateway has been mounted. If the NGT gateway is mounted, it will check if the USB stick is mounted or not. The USB stick is named "DATA" in this project. If the USB stick is also mounted, then the script will start the program for collecting data from the NMEA network.

The script to start the IMU program is also given in the Appendix I. The flow chart of this script is given in Figure 4.23.

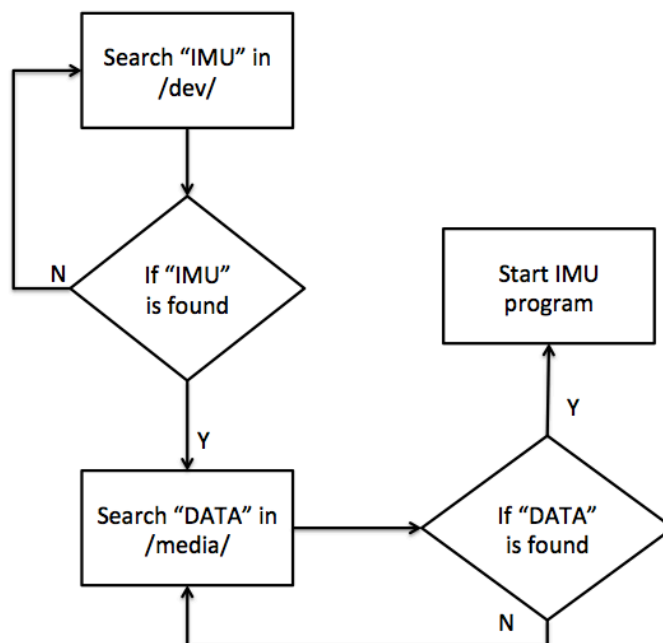


Figure 4.23 Flow chart of the startup script for IMU

This script is similar to the script for the NMEA. The scripts to start the WSN and CAN programs are similar to the script for the NMEA, so that I skip them.

I have also designed a few scripts that have many features. In Figure 4.24, the flow chart represents a script that can backup the MySQL database to an external USB stick. The full script can be found in the Appendix J.

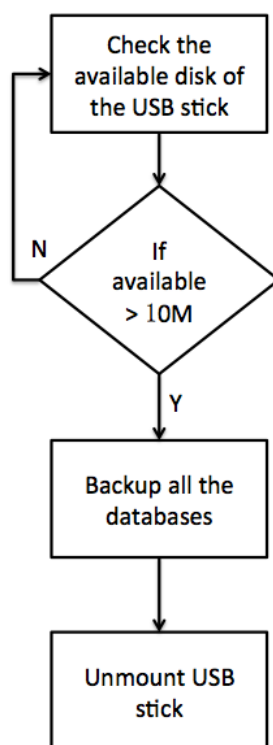


Figure 4.24 Flow chart for the database backup script

The main idea of the database backup script is that when an external USB stick is detected and the available disk of the USB stick is larger than 10M bytes, then all the databases will be copied to the USB stick as a backup. After the backup is finished, the USB stick will be unmounted.

And in Figure 4.25, the flow chart represents a script that can delete the oldest file from the USB stick if the disk usage is less than 500M bytes. So that I can guarantee the newest data can be stored into the USB stick. I call this method the circular storage method. The full script can be found in the Appendix K.

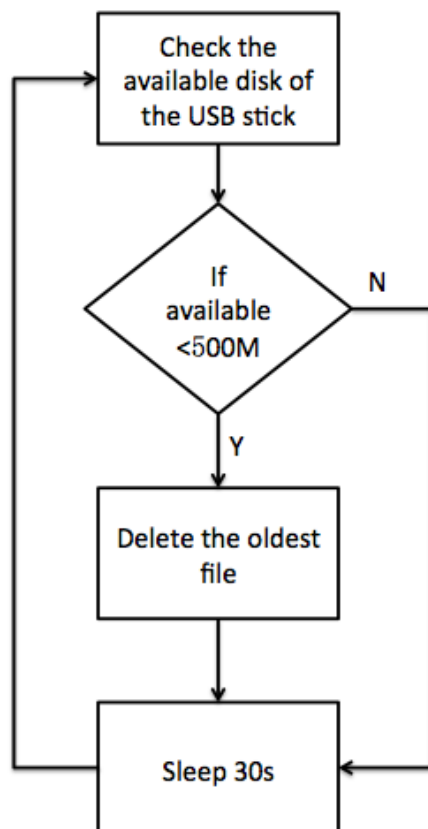


Figure 4.25 Flow chart of the circular storage script

## 4.6 Integration in a box

I am going to use a box bought from Canada with its inner layers design by Ms. Vivian from Inventas. Figure 4.26 shows the inner layers.

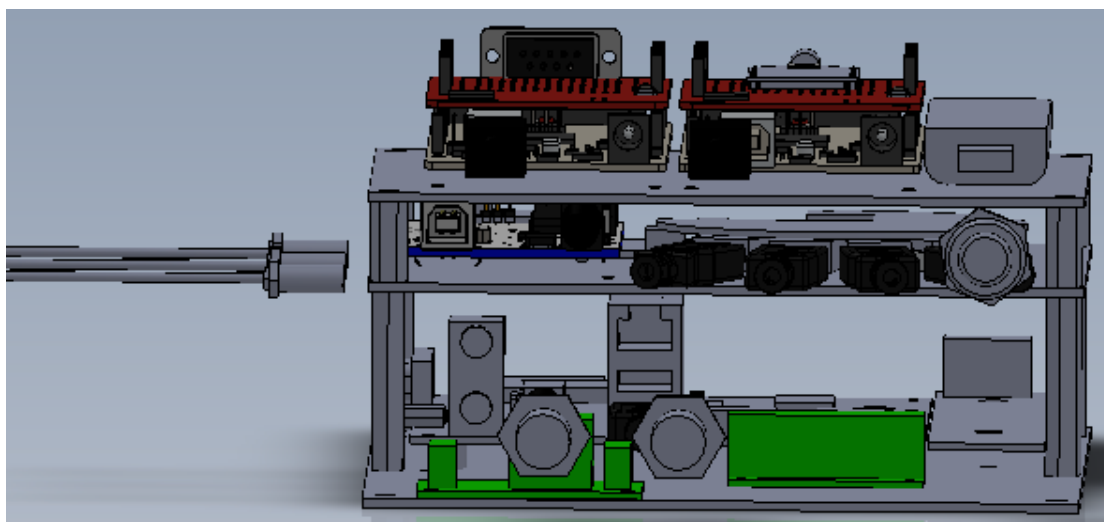


Figure 4.26 Inner layers of the box

There are three layers in total. The top layer includes CAN gateway, WSN gateway and 3G modem. The middle layer includes watchdog and USB hub. The

bottom layer includes Pandaboard, NMEA 2000 gateway, relay and voltage regulator. Figure 4.27 shows the top view of the inner layers.

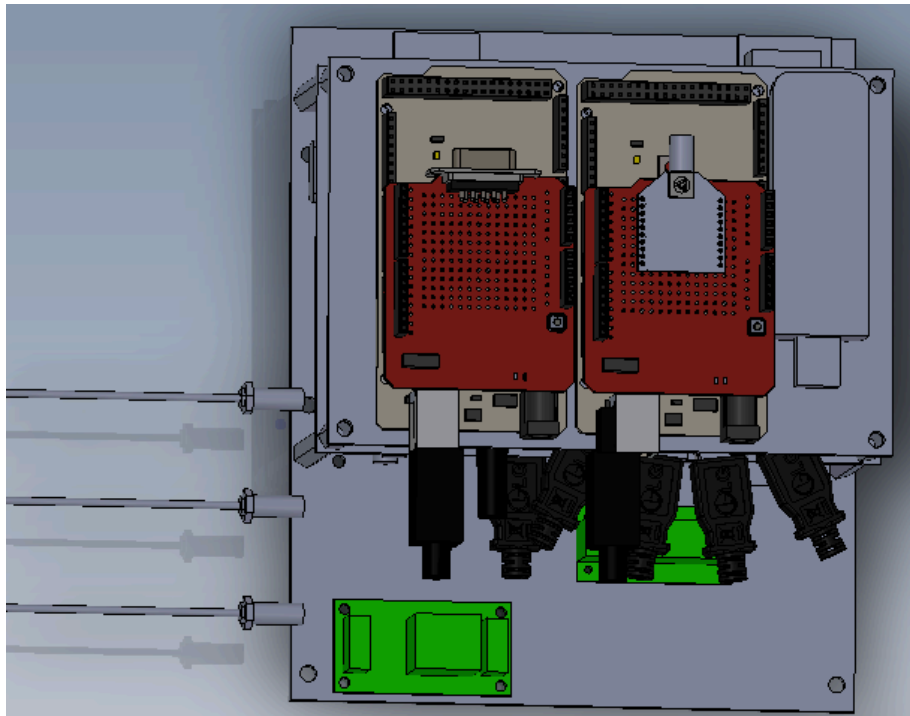


Figure 4.27 Top view of the inner layers in the box

Figure 4.28 shows the outlook of the box.

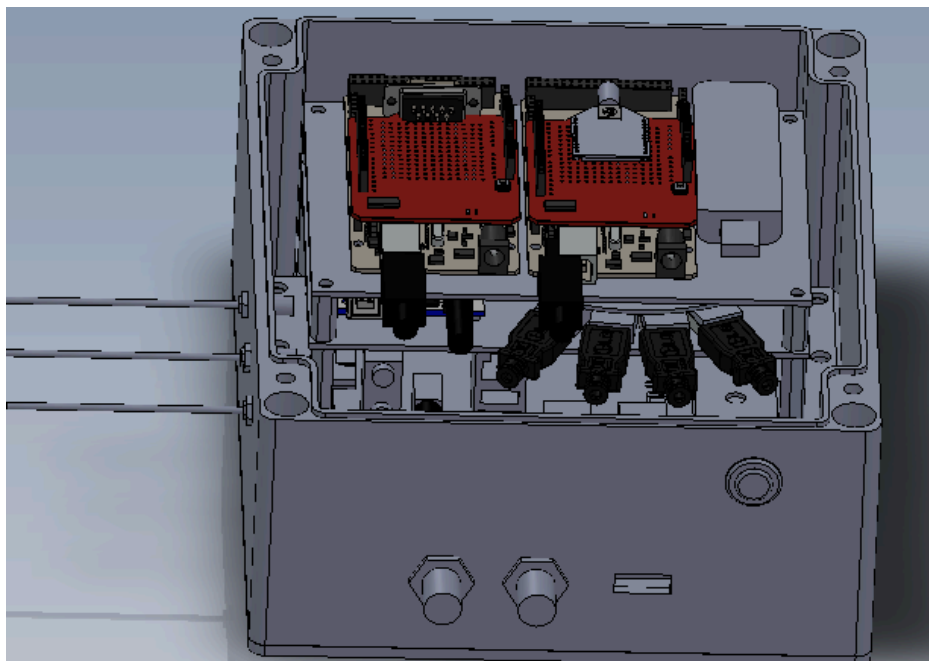


Figure 4.28 Outlook of the box without cover

In this chapter, I have explained the implementation of the system design in details. In next chapter, I am going to demonstrate the testing scenarios of the system.



## 5 Testing

In this chapter, I will test my system in various scenarios. I need to use testing to check the functionality and stability under different situations.

### 5.1 Testing in the lab

I have two phases regarding the testing in the lab. In phase 1, I set up the whole system as a prototype, as shown in Figure 5.1. The CAN and WSN gateways are not included in this figure.

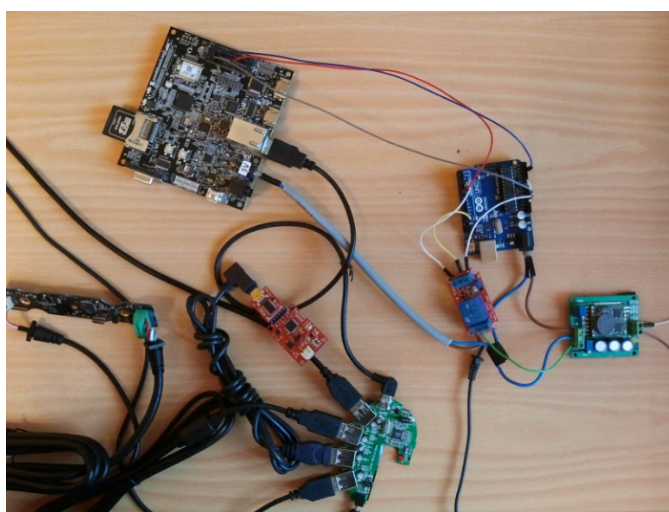


Figure 5.1 Testing phase 1

The power supply in our lab is shown in Figure 5.2.

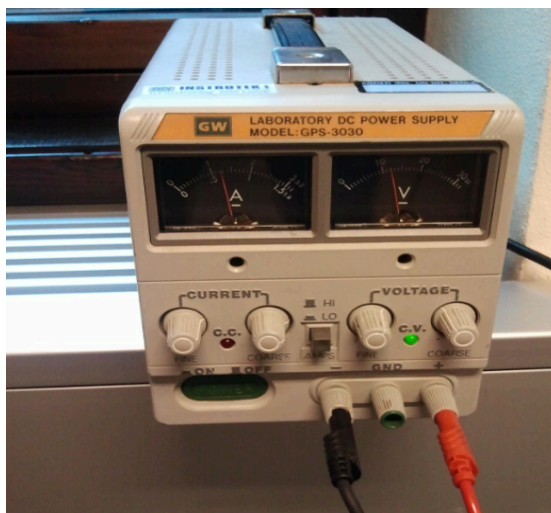


Figure 5.2 Power supply in the lab

When the system is powered on, the Pandaboard can receive data from the WSN, CAN, NMEA and IMU. And the Pandaboard can store the data into MySQL database or into texts successfully. During the testing, the 3G modem and the USB hub are not stable. The 3G modem cannot connect to the Internet sometimes and the USB hub cannot support many devices. The reason is that the 3G modem takes a lot of power and the USB hub cannot offer enough current. I solve this problem by adding a power supply to the USB hub and connect the 3G modem to the Pandaboard instead of to the USB hub.

The phase 2, the system is integrated inside a box, as shown in Figure 5.3.

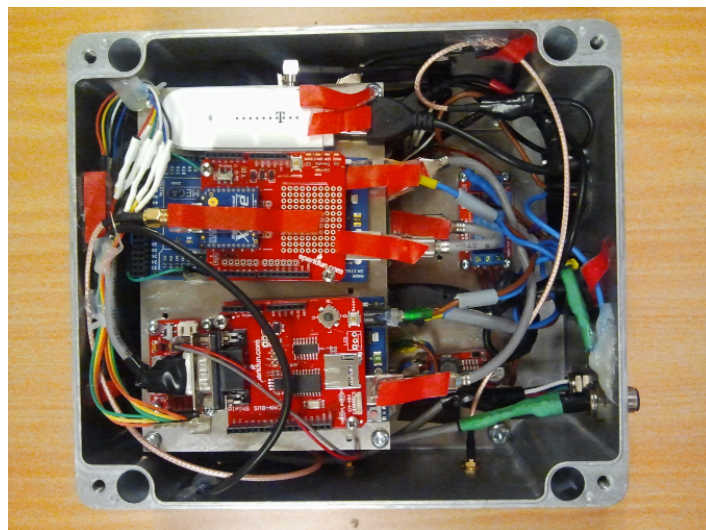


Figure 5.3 Testing phase 2

And the outside appearance is shown in Figure 5.4.

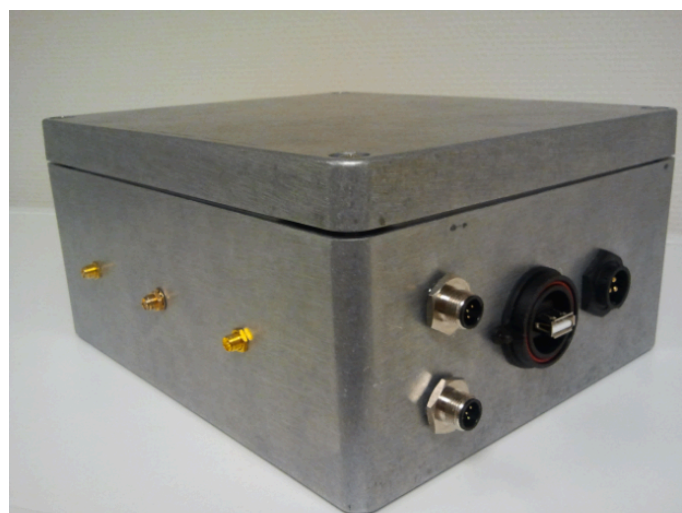


Figure 5.4 Outside appearance of the integration

Due to page limit, I will not write the details of the integration process. During this integration process, I make a lot of cables including the USB cables and the power wires. Everything works smoothly after the integration and adjustment. I have run the system for several days in order to test the functionality and the stability of the system. I can check the local data by accessing the Pandaboard via the access point. All the data are stored successfully. I have also complete data at the remote server.

## 5.2 Testing in a boat-movement simulator

After finishing the test in the lab, I have a further test in a company where there is a boat-movement simulator available. we will fix the system in the boat-movement simulator and test the stability of both the hardware and the functionality on a close-to-reality scenario.

First, we fix the box in the simulator with nails, as shown in Figure 5.5.



Figure 5.5 Fix the box in the simulator

Then we run a simulation program which imitates the movement of a real boat. In this step, the system is not powered. Figure 5.6 and 5.7 show this step.

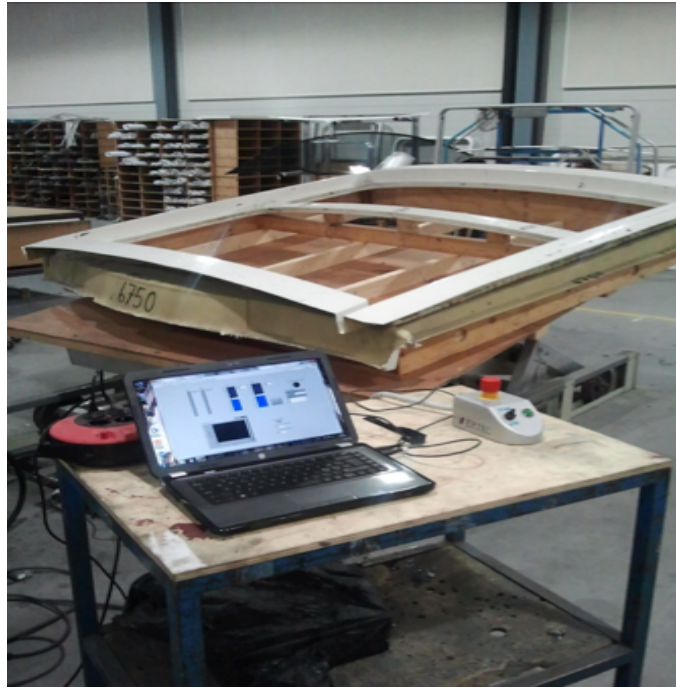


Figure 5.6 Testing in simulator without power - a

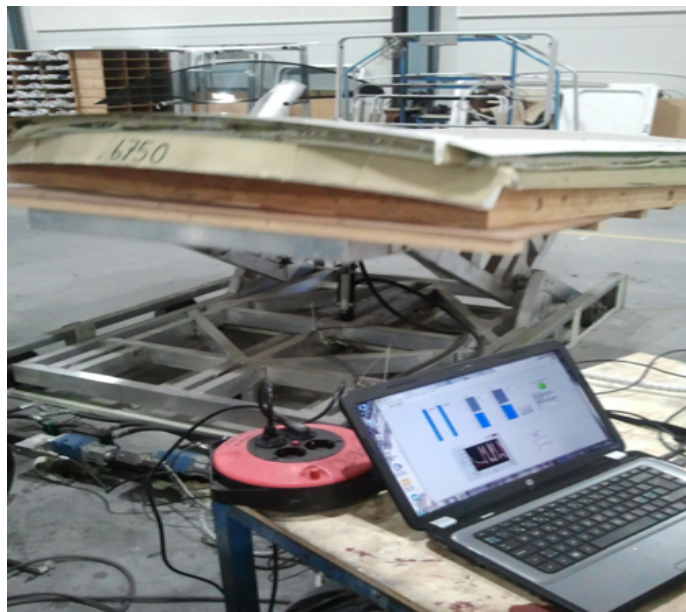


Figure 5.7 Testing in simulator without power - b

After about 30 minutes of simulation, I open the box to check if there are obvious problems like if some cables are broken or if some connections are loose. After careful check, I do not find any obvious problems.

Then repeat this test with power, as shown in Figure 5.8.





Figure 5.8 Testing in simulator with power

After 30 minutes of testing, I stop the simulator. I check the IMU data to see if the system has worked smoothly. Because the IMU data can reflect the movement of the simulator obviously. Figure 5.9 shows a part of the data recorded during the testing.

2255281	-112	-61	20	315	186	583	71	474	-28	271	2014-02-05 11:16:18
2255282	121	1	170	306	184	582	14	40	4	272	2014-02-05 11:16:19
2255283	80	-42	103	296	189	580	37	407	-13	273	2014-02-05 11:16:19
2255284	-132	-23	151	282	183	578	-16	44	4	274	2014-02-05 11:16:19
2255285	63	-58	102	272	185	579	75	417	-18	275	2014-02-05 11:16:19
2255286	-141	25	297	261	184	577	14	73	1	276	2014-02-05 11:16:19
2255287	88	-45	107	267	185	576	35	28	-3	277	2014-02-05 11:16:19
2255288	-43	-4	459	273	188	584	30	-47	-4	278	2014-02-05 11:16:19
2255289	99	5	116	294	186	584	39	-180	-7	279	2014-02-05 11:16:19
2255290	21	-40	329	299	188	581	29	94	-5	280	2014-02-05 11:16:19
2255291	2	59	215	308	185	580	-15	-289	16	281	2014-02-05 11:16:19
2255292	180	-32	-56	323	183	582	62	93	-4	282	2014-02-05 11:16:19
2255293	-98	-66	357	320	183	579	38	-4	-10	283	2014-02-05 11:16:19
2255294	227	10	173	338	186	578	20	-204	-4	284	2014-02-05 11:16:19
2255295	42	40	139	338	187	580	32	287	-22	285	2014-02-05 11:16:19
2255296	18	50	424	330	187	581	-17	-28	3	286	2014-02-05 11:16:19
2255297	69	-6	-38	333	186	579	22	105	-11	287	2014-02-05 11:16:19
2255298	-10	-23	285	325	188	583	33	211	-12	288	2014-02-05 11:16:19
2255299	103	7	214	325	190	583	18	54	-5	289	2014-02-05 11:16:19
2255300	28	1	170	321	184	582	23	133	-5	290	2014-02-05 11:16:19
2255301	44	-9	232	323	187	580	24	111	-3	291	2014-02-05 11:16:19
2255302	122	-4	184	322	188	585	21	71	-2	292	2014-02-05 11:16:19
2255303	91	-1	199	313	187	580	18	219	-6	293	2014-02-05 11:16:19
2255304	88	16	192	303	184	581	25	232	-7	294	2014-02-05 11:16:19
2255305	9	-33	96	283	187	582	23	487	-14	295	2014-02-05 11:16:19
2255306	-53	67	324	266	181	582	7	164	-4	296	2014-02-05 11:16:19
2255307	11	-31	121	260	181	579	2	148	-5	297	2014-02-05 11:16:19
2255308	-80	10	348	261	181	582	21	-65	1	298	2014-02-05 11:16:19
2255309	26	-35	160	275	185	582	20	-112	0	299	2014-02-05 11:16:19
2255310	-1	-18	223	288	186	582	20	-172	0	300	2014-02-05 11:16:19
2255311	82	-2	72	304	185	583	8	-75	0	301	2014-02-05 11:16:19
2255312	-58	38	243	309	186	581	28	-45	-7	302	2014-02-05 11:16:19
2255313	137	-14	234	324	187	583	50	-126	1	303	2014-02-05 11:16:19
2255314	60	-1	143	334	188	583	27	85	-1	304	2014-02-05 11:16:19
2255315	-36	-1	270	333	189	580	19	-71	3	305	2014-02-05 11:16:19
2255316	115	-14	99	345	186	577	10	7	-2	306	2014-02-05 11:16:19

Figure 5.9 IMU data during the testing in simulator

As can be seen from Figure 5.9 that the values of the acceleration (column 2, 3 and 4) change fast due to the movement of the simulator.

Both the results from the tests in the lab and in the boat-movement simulator show that my system is fairly stable and reliable.

In this chapter, I have demonstrated different testing scenarios for the system and the corresponding testing results. The results show that the system is working smoothly with my testing scenarios. In next chapter, I am going to provide the installation steps of the system in a real boat, and show the working status of the system.

## 6 Installation on boat

In this chapter, I am going to introduce how I install the complete system in a real boat. The chapter includes the installation procedure and the results of this installation.

### 6.1 Installation procedure

We have installed the system in a boat from Viknes in Bergen. And the installation procedure is given below.

Step (1): we discussed our installation scenario right after we arrived at Viknes AS, Bergen.

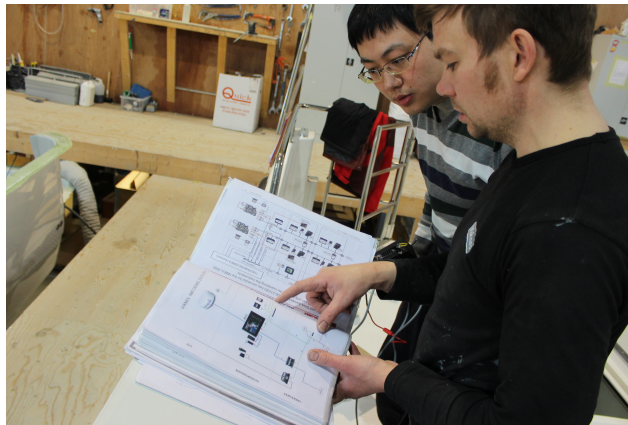


Figure 6.1 Discussion about the installation

Step (2): The NMEA 2000 network backbone is under the steering cabin. we connected the NMEA network to our data collection system via a NGT-1 gateway.



Figure 6.2 The position of the NMEA network on boat

Step (3): The NMEA network on the boat has many sensors, and they are connected

together via CAN bus.

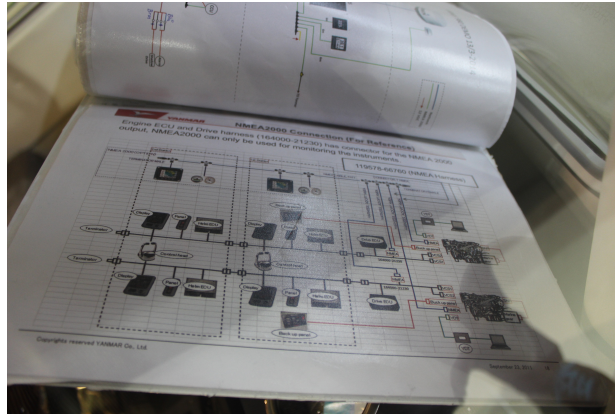


Figure 6.3 NMEA network on the boat

Step (4): we measured the diameter of the Power Button, because we need to install it somewhere.

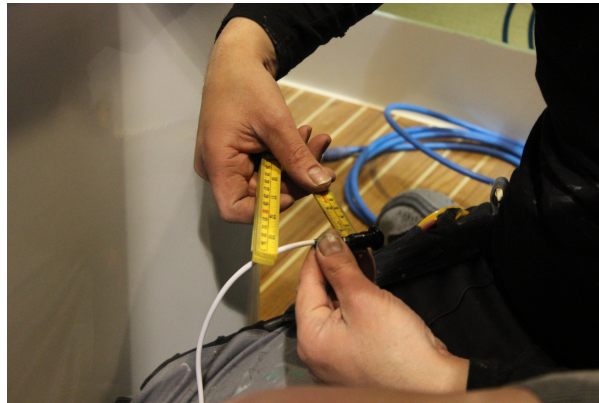


Figure 6.4 Measuring the diameter of the button

Step (5): After discussion, we decided to install the Power Button under the seat.



Figure 6.5 Choosing the position for the button

Step (6): The Power Button was installed through the wall, at the corner.





Figure 6.6 Installation of the button



Figure 6.7 Closer look of the button

Step (7): we installed our system under the seat with nails.

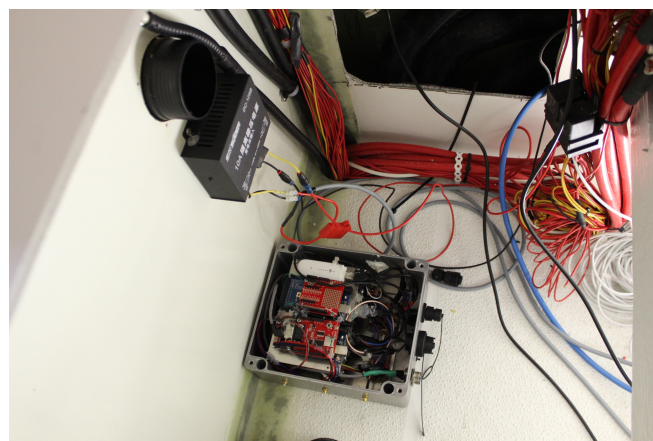


Figure 6.8 Install the system under the seat

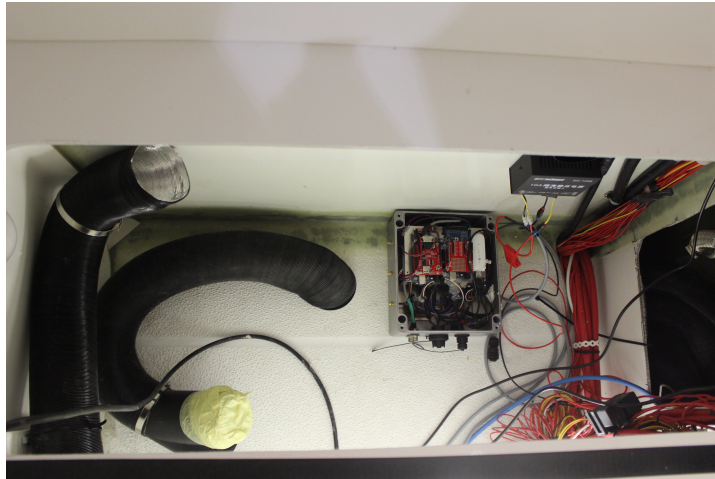


Figure 6.9 Overview of the system after installation

## 6.2 Testing and validation

After finishing the installation, I tested the system when the boat was sailing in the sea. Figure 6.10 showed that the boat was moved from land to water.



Figure 6.10 Testing the system on sailing

After sailing for about 30 minutes, everything went well. Figure 6.11 showed that the 3G modem was working well.

```

9 rows in set (0.00 sec)

mysql> quit
Bye
chipsee@chipsee-desktop:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_req=1 ttl=44 time=52.2 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=44 time=51.3 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=44 time=53.5 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=44 time=49.1 ms
64 bytes from 8.8.8.8: icmp_req=5 ttl=44 time=48.4 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 48.492/50.970/53.589/1.917 ms
chipsee@chipsee-desktop:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data:
64 bytes from 8.8.8.8: icmp_req=1 ttl=44 time=51.9 ms
64 bytes from 8.8.8.8: icmp_req=2 ttl=44 time=51.4 ms
64 bytes from 8.8.8.8: icmp_req=3 ttl=44 time=50.2 ms
64 bytes from 8.8.8.8: icmp_req=4 ttl=44 time=46.1 ms
64 bytes from 8.8.8.8: icmp_req=5 ttl=44 time=49.7 ms
64 bytes from 8.8.8.8: icmp_req=6 ttl=44 time=43.7 ms
64 bytes from 8.8.8.8: icmp_req=7 ttl=44 time=51.9 ms
64 bytes from 8.8.8.8: icmp_req=8 ttl=44 time=54.6 ms
64 bytes from 8.8.8.8: icmp_req=9 ttl=44 time=50.6 ms
64 bytes from 8.8.8.8: icmp_req=10 ttl=44 time=49.2 ms
^C
--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9132ms
rtt min/avg/max/mdev = 43.732/49.975/54.688/2.957 ms
chipsee@chipsee-desktop:~$

```

Figure 6.11 3G modem is working

Beside, I can detect all the sensors that are in use on the boat, as shown in Figure 6.12.

```

78 | 2014-04-11 | 12:44:53 | 3 | 5 | 255 | 129283 | 2014-
79 | 2014-04-11 | 12:44:53 | 3 | 5 | 255 | 126720 | 2014-
80 | 2014-04-11 | 12:45:05 | 7 | 5 | 255 | 126720 | 2014-
Industry Code = Marine
|
80 rows in set (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_NMEA |
+-----+
| device         |
| src0           |
| src17          |
| src2           |
| src3           |
| src4           |
| src5           |
+-----+
7 rows in set (0.00 sec)

mysql>

```

Figure 6.12 Detecting all sensors on boat

There is also data at the remote server. This means that the 3G modem is working well.

In this chapter, I have demonstrated the installation steps on a real boat and show that the system is working smoothly. In next chapter, I am going to conclude the report and present possible future work.

## 7 Conclusion and future work

In this thesis, I have developed a marine data collection and transmission system for ECO-boat. My system can collect data from NMEA, WSN, IMU and CAN networks. I contribute to the integration of this system, namely, I have not only integrated all the data collection and transmission modules in one system, but also arranged the whole system into a box. In addition, I have designed a watchdog for my system and the watchdog can greatly enhance the availability of the system. Furthermore, I have designed two methods to store the data. Besides, a lot of shell scripts are designed that can largely improve the efficiency and reliability of the whole system. The whole system is not only tested in lab, but also been tested in a real boat. In a word, I have achieved my main goals for the project. Compared with the state of the art, my system can store data from different sources such as WSN, NMEA 2000 and so on and can store data both locally and remotely via long distance transmission. As far as I know, no other products or systems can achieve this for the moment.

In the future I will extend the sources of the data so that I could collect more types of data from different sensors. I will also try to reduce the energy cost of my system in the future, since energy cost is a very important aspect for a good system. In addition, data analysis is also a possible direction.



## Reference

- [1] S. Baddam and X. Chen, "Data Collection and transmission for Leisure Time Boats Based on Arduino WSNs and LTE," M.S thesis, Dept. ICT, Univ. of Agder, 2013.
- [2] M. Yue and Y. Sun, "Marine Data Collection based on Embedded System with Wired and Wireless Transmission," M.S thesis, Dept. ICT, Univ. of Agder, 2013.
- [3] Maretron.com. (Accessed: 2014, May 30). "VDR100 vessel data recorder"[Online]. Available: <http://www.maretron.com/products/vdr100.php>.
- [4] Wikipedia.org. (Accessed: 2014, May 30). "PandaBoard"[Online]. Available: <http://en.wikipedia.org/wiki/PandaBoard>.
- [5] Omappedia.com. (Accessed: 2014, May 30). "PandaBoard top view"[Online]. Available: [http://www.omappedia.com/wiki/File:PandaBoard\\_top\\_view.png](http://www.omappedia.com/wiki/File:PandaBoard_top_view.png).
- [6] Arduino.cc. (Accessed: 2014, May 30). "Arduino MEGA top view"[Online]. Available: <http://arduino.cc/en/Main/ArduinoBoardMega2560>.
- [7] Arduino.cc. (Accessed: 2014, May 30). "Arduino Uno top view"[Online]. Available: <http://arduino.cc/en/Main/ArduinoBoardUno>.
- [8] Arduino.cc. (Accessed: 2014, May 30). "Arduino Lilypad top view"[Online]. Available: <http://arduino.cc/en/Main/ArduinoBoardLilyPad>.
- [9] Actisense.com. (Accessed: 2014, May 25). "NGT-1 NMEA 2000 to PC Interface"[Online]. Available: <http://www.actisense.com/products/nmea-2000/ngt1.html>.
- [10] Actisense.com. (Accessed: 2014, May 25). "NGT-1 images"[Online]. Available: <http://www.actisense.com/products/nmea-2000/ngt1/ngt1-images.html>.
- [11] Sparkfun.com. (Accessed: 2014, Mar. 20). "IMU images"[Online]. Available: <https://www.sparkfun.com/products/10736>.
- [12] Limundo.com. (Accessed: 2014, May 20). "Huawei E353 modem"[Online]. Available: <http://www.limundo.com/kupovina/Racunari-i-oprema/Mrezni-uredjaji/Modemi/Huawei-E353-HSPA-Telenor-modem-USB-Stick/22228613>.
- [13] MySQL.com. (Accessed: 2014, May 20). "Market Share"[Online]. Available: <http://www.mysql.com/why-mysql/marketshare/>
- [14] Wikipedia.com. (Accessed: 2014, May 20). "OpenSSH"[Online]. Available: <http://en.wikipedia.org/wiki/OpenSSH>.
- [15] Svtronics.com. (Accessed: 2014, Feb. 15). "Pandaboard ES Rev B3 Developer's Guide"[Online]. Available: <http://www.svtronics.com/support/pandaboard-es-b3-developers-guide/>.
- [16] Wikipedia.org. (Accessed: 2014, Mar. 18). "WvDial"[Online]. Available: <http://en.wikipedia.org/wiki/WvDial>.
- [17] Draisberghof.de. (Accessed: 2014, Mar. 18). "USB mode switch"[Online]. Available: [http://www.draisberghof.de/usb\\_modeswitch/](http://www.draisberghof.de/usb_modeswitch/).
- [18] Adafruit.com. (Accessed: 2014, Apr. 10). "Set up access point"[Online]. Available: <https://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/install-software>.

- [19] PSU.edu. (Accessed: 2014, Apr. 10). "Command line SSH user guide"[Online]. Available: [http://rcc.its.psu.edu/user\\_guides/remote\\_connectivity/ssh/](http://rcc.its.psu.edu/user_guides/remote_connectivity/ssh/).
- [20] cplusplus.com. (Accessed: 2014, May 13). "How to use fopen in C/C++"[Online]. Available: <http://www.cplusplus.com/reference/cstdio/fopen/>.
- [21] Wikipedia.org. (Accessed: 2014, May 13). "Json wiki"[Online]. Available: <http://en.wikipedia.org/wiki/JSON>.
- [22] stackflow.com. (Accessed: 2014, May 13). "Time function in C"[Online]. Available: <http://stackoverflow.com/questions/2408976/struct-timeval-to-printable-format>.
- [23] stackflow.com. (Accessed: 2014, May 13). "How to use tee command"[Online]. Available: <http://stackoverflow.com/questions/60942/how-can-i-send-the-stdout-of-one-process-to-multiple-processes-using-preferably>.
- [24] Archlinux.org. (Accessed: 2014, Mar. 18). "Check device information via serial port"[Online]. Available: <https://bbs.archlinux.org/viewtopic.php?id=134705>.
- [25] Blitzbasic.com. (Accessed: 2014, Apr. 15). "Auto mount USB stick"[Online]. Available: <http://www.blitzbasic.com/Community/posts.php?topic=100593>.
- [26] Sites.google.com. (Accessed: 2014, Apr. 15). "How to use GPIO"[Online]. Available: <https://sites.google.com/site/semilleroadt/raspberry-pi-tutorials/gpio>.
- [27] Stackflow.com. (accessed: 2014, Apr. 15). "String compare in script"[Online]. Available: <http://stackoverflow.com/questions/2237080/how-to-compare-strings-in-bash-script>.
- [28] Linuxcommand.org. (Accessed: 2014, Apr. 15). "mysqldump"[Online]. Available: [http://www.linuxcommand.org/man\\_pages/mysqldump1.html](http://www.linuxcommand.org/man_pages/mysqldump1.html)

## Appendices

### Appendix A Program for USB stick auto mount [25]

```

>>>>start<<<<
KERNEL!="sd[a-z]*", GOTO="auto_mount_end"
ACTION=="add", PROGRAM!="/sbin/blkid %N", GOTO="auto_mount_end"
# Set environment
ACTION=="add", IMPORT{program}="/sbin/blkid -o udev -p -s TYPE -s LABEL %N"
# Global mount options
ACTION=="add", ENV{mount_options}="relatime,users,umask=0"
# Filesystem specific options
ACTION=="add", ENV{ID_FS_TYPE}=="vfat",
ENV{mount_options}="%E{mount_options},sholxec"
ACTION=="add", ENV{ID_FS_TYPE}=="ntfs",
ENV{mount_options}="%E{mount_options},utf8"
# Get mount point
# use basename to correctly handle labels such as ../mnt/foo
ACTION=="add", ENV{ID_FS_LABEL}=="?*", PROGRAM="/usr/bin/basename
'%E{ID_FS_LABEL}%', ENV{dir_name}="%c"
ACTION=="add", ENV{dir_name}!="?*", ENV{dir_name}="usbhd-%k"
# Main action
ACTION=="add", ENV{dir_name}=="?*", RUN+="/bin/mkdir -p
'/mnt/usb/%E{dir_name}%', RUN+="/bin/mount -o %E{mount_options} /dev/%k
'/mnt/usb/%E{dir_name}%"
ACTION=="remove", ENV{dir_name}=="?*", RUN+="/bin/umount -l
'/mnt/usb/%E{dir_name}%', RUN+="/bin/rmdir '/mnt/usb/%E{dir_name}%"
LABEL="auto_mount_end"
# label must be cleared
ENV{ID_FS_LABEL}=""
>>>>end<<<<

```

**Appendix B                    udhcpd configuration file [18]**

```
# Sample udhcpd configuration file (/etc/udhcpd.conf)

# The start and end of the IP lease block

start      192.168.0.2 #default: 192.168.0.20
end        192.168.0.20 #default: 192.168.0.254

# The interface that udhcpd will use

interface  wlan0      #default: eth0

# The maximim number of leases (includes addresssed reserved
# by OFFER's, DECLINE's, and ARP conficts

#max_leases  254      #default: 254

# If remaining is true (default), udhcpd will store the time
# remaining for each lease in the udhcpd leases file. This is
# for embedded systems that cannot keep time betlen reboots.
# If you set remaining to no, the absolute time that the lease
# expires at will be stored in the dhcpd.leases file.

remaining  yes      #default: yes

# The time period at which udhcpd will write out a dhcpd.leases
# file. If this is 0, udhcpd will never automatically write a
# lease file. (specified in seconds)

#auto_time  7200      #default: 7200 (2 hours)

# The amount of time that an IP will be reserved (leased) for if a
# DHCP decline message is received (seconds).

#decline_time  3600      #default: 3600 (1 hour)

# The amount of time that an IP will be reserved (leased) for if an
```



```
# ARP conflict occurs. (seconds

#conflict_time 3600      #default: 3600 (1 hour)

# How long an offered address is reserved (leased) in seconds

#offer_time 60      #default: 60 (1 minute)

# If a lease to be given is below this value, the full lease time is
# instead used (seconds).

#min_lease 60      #default: 60

# The location of the leases file

#lease_file /var/lib/misc/udhcpd.leases #default: /var/lib/misc/udhcpd.leases

# The location of the pid file
#pidfile /var/run/udhcpd.pid #default: /var/run/udhcpd.pid

# Everytime udhcpd writes a leases file, the below script will be called.
# Useful for writing the lease file to flash every few hours.

#notify_file      #default: (no script)

#notify_file dumpleases # <--- useful for debugging

# The following are bootp specific options, setable by udhcpd.

#siaddr 192.168.0.22      #default: 0.0.0.0

#sname zorak      #default: (none)

#boot_file /var/nfs_root #default: (none)

# The remainder of options are DHCP options and can be specified with the
# keyword 'opt' or 'option'. If an option can take multiple items, such
# as the dns option, they can be listed on the same line, or multiple
# lines. The only option with a default is 'lease'.

#Examples
opt dns 8.8.8.8 4.2.2.2
```

```
option subnet 255.255.255.0
opt router 192.168.0.1
#opt wins 192.168.0.1
#option dns 129.219.13.81 # appened to above DNS servers for a total of 3
option domain local
option lease 864000 # 10 days of seconds

# Currently supported options, for more info, see options.c
#opt subnet
#opt timezone
#opt router
#opt timesrv
#opt namesrv
#opt dns
#opt logsrv
#opt cookiesrv
#opt lprsrv
#opt bootsize
#opt domain
#opt swapsrv
#opt rootpath
#opt ipttl
#opt mtu
#opt broadcast
#opt wins
#opt lease
#opt ntpsrv
#opt tftp
#opt bootfile
#opt wpad

# Static leases map
#static_lease 00:60:08:11:CE:4E 192.168.0.54
#static_lease 00:60:08:11:CE:3E 192.168.0.44
```

**Appendix C****New program for the IMU output**

```
void output_sensors_text(char raw_or_calibrated)
{
  Serial.print("#A-"); Serial.print(raw_or_calibrated); Serial.print('=');
  Serial.print(accel[0]); Serial.print(",");
  Serial.print(accel[1]); Serial.print(",");
  Serial.print(accel[2]); Serial.println();

  Serial.print("#M-"); Serial.print(raw_or_calibrated); Serial.print('=');
  Serial.print(magnetom[0]); Serial.print(",");
  Serial.print(magnetom[1]); Serial.print(",");
  Serial.print(magnetom[2]); Serial.println();

  Serial.print("#G-"); Serial.print(raw_or_calibrated); Serial.print('=');
  Serial.print(gyro[0]); Serial.print(",");
  Serial.print(gyro[1]); Serial.print(",");
  Serial.print(gyro[2]); Serial.println();

  Serial.print("No.=");Serial.println(time_count++);Serial.println("?");
}
```

**Appendix D****A part of the program for IMU data collection**

```

while (1) {
    read(sd, &c, 1);
    p[j] = c;
    j++;
    if (c == '?') {
        sscanf(p,
"%*[^]=%[^,],%[^,],%[\n]*[^]=%[^,],%[^,],%[\n]*[^]=%[^,],%[^,],%[\n]*[^]=
%[\n]", &Ax[0], &Ay[0], &Az[0], &Mx[0], &My[0], &Mz[0], &Gx[0], &Gy[0], &Gz[0],
&No[0]);
        int ax, ay, az, mx, my,mz, gx, gy, gz;
        unsigned long no;
        ax=(int)(atof(Ax));
        ay=(int)(atof(Ay));
        az=(int)(atof(Az));
        mx=(int)(atof(Mx));
        my=(int)(atof(My));
        mz=(int)(atof(Mz));
        gx=(int)(atof(Gx));
        gy=(int)(atof(Gy));
        gz=(int)(atof(Gz));
        no=(unsigned long)(atof(No));
        fprintf(stderr,
"\n#A=%d,%d,%d\n#M=%d,%d,%d\n#G=%d,%d,%d\nNo.=%lu\n", ax, ay, az, mx,
my, mz, gx, gy, gz, no);

        sprintf(sql_insert, "insert INTO imu
values(NULL, %d, %d, %d, %d, %d, %d, %d, %d, %lu, now());\0", ax, ay, az, mx,
my, mz, gx, gy, gz, no);
        if (mysql_query(&mysql, sql_insert) != 0) {
            fprintf(stderr, "Failed:%s\n", sql_insert);
        } else {
            fprintf(stdout, "Succeeded:%s\n", sql_insert);
        }
        j = 0;
        p[j] = '\0';
    }
}
} // end of "while (1)"

```

## Appendix E                      Arduino program for watchdog

```
int pushButton = A0;//from polr button
int resetPin = A3;//pin 9 of J6
int relayPin = 3;//to relay
//int fromPB = 2;//from pandaboard pin 9 of J6
int toPB = 6;//to pandaboard pin 20 of J6
int timer = 0;

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(relayPin, OUTPUT);
  //pinMode(fromPB, INPUT);
  pinMode(toPB, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input pin:
  int v0 = analogRead(pushButton);
  float v1 = v0 * (5.0 / 1023.0);
  //Serial.println(v1);
  if (v1 > 3.00){
    digitalWrite(relayPin, HIGH);//5V,tell relay to close
    analogWrite(toPB, 255);//5V,to ensure the pin is HIGH level
    //delay (10000);
    //start counting
    timer = timer + 1;
    delay (500);
    int v3 = analogRead(resetPin);
    float v4 = v3 * (5.0 / 1023.0);
    if (v4 > 1.75) {
      delay(1500);
      int v5 = analogRead(resetPin);
      float v6 = v5 * (5.0 / 1023.0);
      if (v6 < 1.60) { //pandaboard is working
        timer = 0;
      }else {
        timer = timer + 3;
      }
    }
  }
  if (timer > 240) {
```

```
//panda board is dead now
digitalWrite(relayPin, LOW);//open relay
delay(5000);
digitalWrite(relayPin, HIGH);//close relay and restart pandaboard
delay(10);
timer = 0;
}
}else if(v1 < 2.50) {
  analogWrite(toPB, 0);//tell panda board to polr off
  delay(90000);//wait for panda board to polr off
  analogWrite(toPB, 255);
  delay(1);
  digitalWrite(relayPin, LOW);//tell relay to open
  delay (10000);
}
}
```

**Appendix F****Shell script for watchdog [26]**

```
#GPMC_AD14#  
#PIN 9 of J6#  
cd /sys/class/gpio  
echo 38 > export  
cd gpio38  
echo "out" > direction  
while true;  
do  
echo 1 > value  
sleep 1s;  
echo 0 > value  
sleep 20s;  
done
```

**Appendix G****Shell script for poweroff Pandaboard [26] [27]**

```
### GPMC_AD9 ###
#PIN 20 of J6#
v=0
cd /sys/class/gpio
echo 33 > export
cd gpio33

while true;
do
flag1=$(cat value | awk '{print $1}');
echo "GPIO 33 level: "$flag1
if [ $flag1 -eq $v ];then

echo "shutdown signal detected..."
sleep 10s;
flag2=$(cat value | awk '{print $1}');

if [ $flag2 -eq $v ];then
echo "shutdown signal confirmed..."
echo "shutting down now..."
sleep 1s;
sudo shutdown -h now
fi
fi
sleep 5s;
done
```



## Appendix H Start up script for NMEA

```
#!/bin/bash
cd /home/bbxm/Desktop/AllSteps/Step5/Mingli/canboat/rel/linux-armv7l
while true;
do
    echo "start searching NGT..."
    flag=$( ls /dev | grep NGT | awk '{print $1 }' );
    if [ "$flag" == "NGT" ]
    then
        echo "target found"
        break;
    fi
done
while true;
do
    echo "start searching /media/DATA..."
    flag2=$( ls /media | grep DATA | awk '{print $1 }' );
    if [ "$flag2" == "DATA" ]
    then
        echo "target found"
        break;
    fi
done
while true;
do
    echo "start searching /media/DATA..."
    flag2=$( ls /media | grep DATA | awk '{print $1 }' );
    if [ "$flag2" == "DATA" ]
    then
        echo "target found"
        break;
    fi
done
echo "start program now"
./actisense-serial /dev/NGT | tee >(/.analyzer_remote -itv 30) >(/.analyzer_local -itv 2
-json)
```

## Appendix I                      Start up script for IMU

```
#!/bin/bash
cd /home/bbxm/Desktop/AllSteps/Step5/IMU

while true;
do
    echo "start searching IMU..."
    flag=$( ls /dev | grep IMU | awk '{print $1 }' );
    if [ "$flag" == "IMU" ];
    then
        echo "target found"
        break;
    fi
done

while true;
do
    echo "start searching /media/DATA..."
    flag2=$( ls /media | grep DATA | awk '{print $1 }' );
    if [ "$flag2" == "DATA" ]
    then
        echo "target found"
        break;
    fi
done

echo "start program"

./IMU-txt IMU
```

**Appendix J****Shell script for database backup**

```
#!/bin/bash
dbuser=root
dbpwd=123456
while true;
do
available=$( df | grep /dev/ | grep -v "/dev/mmcbk0p2" | awk '{print $4 }');
path=$( df | grep /dev/ | grep -v "/dev/mmcbk0p2" | awk '{print $6 }');
if [ $available -ge 10000 ];
then
echo "USB device detected: "$path
echo "copy databases..."
cd $path
mysqldump --quick --single-transaction -u$dbuser -p$dbpwd --all-databases>
backup.sql
sleep 10
echo "copy finished !"
cd ..
#sleep 1
sudo umount $path
else
echo "detecting USB device..."
fi
sleep 5s #set the period to check the disk, suffix:(s,second) (m,minute) (h,hour)
(d,day)
done
```

## Appendix K Shell script for circular storage

```
#!/bin/bash
threshold=500000
cd /media/DATA
while(true)
do
    available=$(df -l /dev/sda1 | grep -v "Filesystem" | awk '{print $4}');
    if [ $available -le $threshold ]; then
        echo "available is "$available
        echo "begin to delete oldest file"
        filename=$(ls -1tr | head -1);
        sudo rm $filename
        echo "oldest file is deleted now"
    else
        echo "there is enough space"
    fi
    sleep 30s
done
```