UNIVERSITY OF AGDER

Marine Data Collection based on Embedded System with Wired and Wireless Transmission

By

Mingli Yue, Yihai Sun

Supervisors: Lei Jiao, Frank Yong Li

Master Thesis in Information and Communication Technology
IKT590, Spring 2013

Faculty of Engineering and Science
University of Agder

Grimstad, 3 June 2013

Status: Final

Abstract:

A great interest of boat manufacturers is to improve their products by knowing how the boats are used after sale. In order to gather information about the condition of usages, a system needs to be developed in order to collect data from different marine electronics mounted on the boat. Through this thesis work, we developed such data collecting system for leisure boats which support CAN Bus the message-based protocol. The data collection system has been developed and installed on a Linux-based embedded system connected to the CAN Bus network through a gateway in our laboratory. Through the data collection system, all data generated from different marine electronics in the network can be captured, filtered, transmitted, displayed and then stored in the system. For data transmission and access, we have implemented three methods through wired or wireless networks, i.e., the fixed Internet, 3G/LTE cellular networks and Wi-Fi networks.

Furthermore, the prototype implementation has been extensively tested in both lab and real-life environment.

Keywords: CAN Bus, NMEA 2000, embedded system, data filtering, data transmission

Preface

This report is the result of the master thesis IKT 590 (30 ECTS) which is part of our 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 2013 and ended on 3 June 2013. We have completed the main goal of our project “Marine Data Collection based on Embedded System with Wired and Wireless Transmission”.

This project is part of the ECO-Boat MOL project which is funded by the Research council of Norway. We would like to thank our project supervisors Dr. Lei Jiao and Professor Frank Y. Li for the guidance in giving feedback on technical and content of report throughout this project. Through this thesis work, we learnt a lot about project content and technical report writing. We also thank Mr. Ahmad Noor for his constructive suggestions and technical support. At last, we would like to give our thanks to Marex AS for providing their boats for real-life experiments.

Grimstad

3 June 2013

Mingli Yue and Yihai Sun

Contents

Abbreviations 9
1 Introduction 11
1.1 Background and Motivation 11
1.2 Problem Statement 11
1.3 Approaches 11
1.4 Thesis Outline 12
2 Technology Background 13
2.1 CAN Bus and NMEA 2000 13
2.2 Access Modes 13
2.2.1 Wired connection 13
2.2.2 Wi-Fi 14
2.2.3 Cellular networks 14
2.3 Web Service 14
2.4 Secure Shell 15
2.5 Netcat 15
2.6 MySQL 16
2.7 Secure Sockets Layer 16
2.8 Advanced Encryption Standard 17
3 Requirements and System Design 18
3.1 Requirements 18
3.2 System Architecture 18
3.2.1 Local component 19
3.2.2 Remote component 19
3.3 Embedded System 19
3.3.1 Hardware 19
3.3.2 Software 21
3.4 Remote Server 22
3.4.1 Hardware 22
3.4.2 Software 22
4 Implementation of Basic Functionalities 24
4.1 Data Collection 24
4.2 Data Filtering 25
4.2.1 Filtering by device 25
4.2.2 Filtering by type of message 27
4.2.3 Filtering by message update interval 28
4.3 Data Storage 28
4.4 Data Transmission 29
4.4.1 Real-time data transmission 30
4.4.2 Historical data transmission 30
4.5 Automatic Operation of Data Collection and Transmission 31
4.6 Data Browsing 33
4.6.1 Data browsing on the embedded system 33
4.6.2 Data browsing on the remote server 33
4.6.3 Data browsing from external Internet connection 35
5 Implementation of Advanced Features 39
5.1 Data Security 39
5.2 Transmission Security 40
5.2.1 Secure connection for MySQL 40
5.2.2 Secure connection for FTP 40
5.3 Local Access and Control by Using SSH 41
5.4 Reverse Control 42

6 Test and Validation........46
6.1 Test Scenarios........46
6.2 Experiment on Access Modes........47
6.3 Experiment on Real-Time Transmission........48
6.4 Experiment on Local Access and Control using Android Phone........48
6.5 Experiment on Boat........50
7 Discussions........53
7.1 Operating System........53
7.2 Wi-Fi Stability on BeagleBoard-xM........53
7.3 Device ID Assignment........53
7.4 Portability of Software........53
7.5 IP Address Allocation........54
8 Conclusions and Future Work........55
8.1 Contributions........55
8.2 Future Work........55
References........56
Appendix A Installing NGT-1 gateway on embedded system........57
Appendix B Remote control on embedded system through SSH........58
Appendix C Improvement on user experience........59
Appendix D Integration with other systems........61
Appendix E Results of real-life experiment........62

Figures

Figure 1 Application scenario of data collection and transmission system 12
Figure 2 Description of command parameters in Netcat 16
Figure 3 System architecture 19
Figure 4 BeagleBoard-xM and PandaBoard ES...20
Figure 5 Types of RESTful Web service in Netbeans 22
Figure 6 Translate raw data to human-readable message 24
Figure 7 Priority of filtering criteria 25
Figure 8 Messages from specified device 26
Figure 9 Messages from devices not in the blacklist 27
Figure 10 Messages with a specified PGN 28
Figure 11 Table structure in local database 28
Figure 12 Two modes for data transmission 29
Figure 13 Configuration of startup applications 32
Figure 14 Add startup program 32
Figure 15 List of startup programs 32
Figure 16 Table structure on remote server 34
Figure 17 Viewing data on remote server 34
Figure 18 Viewing data in Netbeans 35
Figure 19 Functions of Web service..36
Figure 20 Inquery for single message 37
Figure 21 Inquery for multiple messages 37
Figure 22 Message count 38
Figure 23 AES encryption and decryption in MySQL 39
Figure 24 Encrypted data and transmission..41
Figure 25 Flow chart of reverse control...42
Figure 26 Netcat listening port 42
Figure 27 Reverse control on the embedded system 44
Figure 28 Control terminal on the remote server 44
Figure 29 Testbed 46
Figure 30 Access modes 47
Figure 31 Data received in the remote server 47
Figure 32 Connect the embedded system to mobile phone through Wi-Fi 48
Figure 33 Login screen 49
Figure 34 Log in the embedded system 49
Figure 35 Local access 50
Figure 36 Real-life experiment 51
Figure 37 Real-time data in real-life experiment 51
Figure 38 PuTTY configuration 58
Figure 39 Execute the shell script 59
Figure 40 Set message update interval 59
Figure 41 Select filtering mode 60
Figure 42 Specify device ID 60
Figure 43 coordinator and wireless sensor 61

Tables

Table 1 Technical specifications of PandaBoard ES 20
Table 2 Filtering criteria and application scenarios 25
Table 3 Rule 1: Specified device....26
Table 4 Rule 2: Blacklist 26
Table 5 Filtering by PGN 27
Table 6 Filtering by update interval 28
Table 7 Internet-dependent functionalities under different access modes 47
Table 8 Compatibility of operating system on the embedded system 53

Abbreviations

CAN	Controller Area Network
NMEA	National Marine Electronic Association
GPON	Gigabit-capable Passive Optical Networks
AP	Access Point
LTE	Long Term Evolution
MIMO	Multi-Input Multi-Output
WLAN	Wireless Local Area Networks
SSH	Secure Shell
FTP	File Transfer Protocol
PGN	Parameter Group Number
AES	Advanced Encryption Standard
SSL	Secure Sockets Layer
CA	Certificate Authority
LAN	Local Area Network
SSID	Service Set Identifier
SSH	Secure Shell
GUI	Graphical User Interface
WSN	Wireless Sensor Network

1 Introduction

In Section 1.1, the motivation of this thesis is introduced. Section 1.2 illustrates the problems need to solved while Section 1.3 describes our solution. At last, Section 1.4 gives the outline of this thesis.

1.1 Background and Motivation

In recent years, the market of leisure boats is on steady growth. Especially in North America and Europe, there are a large number of leisure boats in use. Meanwhile, Asian market is becoming more active. Therefore, it has become the main interest for boat manufacturers to improve their products according to various market demands. For that reason, monitoring data on boats is required for analysing operation state of equipment, user preference and security.

Now marine sensors are widely implemented on leisure boats to provide monitoring information. However, as far as we know, there is not any data collection and management system on the market. Thus, we aim at developing such a system for boat manufacturers and owners.

1.2 Problem Statement

The main task of this thesis is to develop an embedded system which can be used to collect, filter, transmit and store the data from different marine electronics. Furthermore, with the Internet accessibility, the system can transmit the data to the appointed server, so that the data can be used for commercial or safety analysis. The problems to be solved can be summarized as follows:

- To create a testbed – a CAN Bus network with sensors connected to it;
- To select a proper embedded platform with Linux installed on it;
- To install CAN Bus gateway on embedded system;
- To develop a program to implement the functionalities of data collection, filtering, storage and publishing.
- To set up a server receives the data from the embedded systems.

These requirements are the functional demands of the system, which will be described in the next chapter.

1.3 Approaches

In this thesis, we design a marine data collection and transmission system. It is based on an ARM-based single-board computer, which is small in size but provides excellent performance.

Figure 1 illustrates the overall structure of the data collection and transmission system. The system can be implemented on boats, being connected to CAN Bus. In the CAN Bus network, there are NMEA 2000-compliant sensors generating monitoring data and broadcasting it through the bus. The embedded system is designed to collect the data, and store it after filtering. Furthermore, the data will be transmitted to the designated remote server through Internet. It is worth mentioning that, the embedded system supports flexible connectivity, so that we can choose wired or wireless connection according to circumstances.

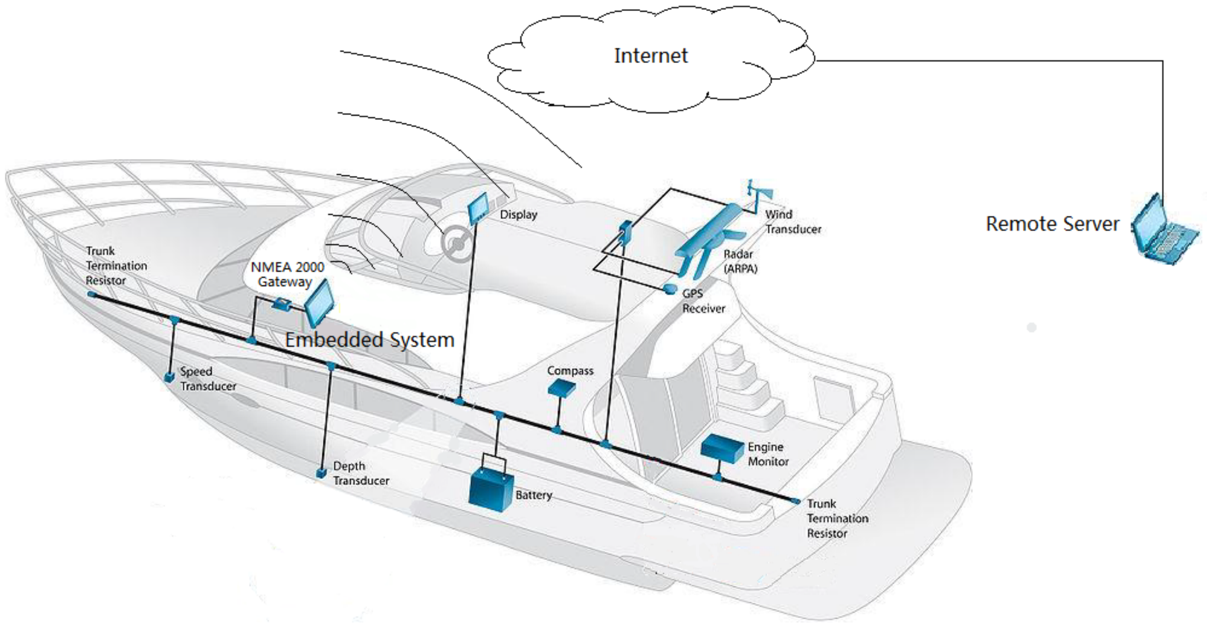


Figure 1 Application scenario of data collection and transmission system

For developing the data collection and transmission system, we have done the following:

- To build a CAN Bus testbed with different sensors and an NMEA 2000 gateway.
- To develop and an embedded system to fulfil the requirements of data collection, filtering, storage and transmission.
- To set up a server which receive the data from embedded system and publish the data on the Internet.
- To improve the system with data/transmission security and remote control.

And the prototype of the system has been tested in experiments.

1.4 Thesis Outline

The remaining thesis is structured as follows.

- In Chapter 2, a brief introduction of background technologies adopted in this project is given.
- Chapter 3 describes the functional requirements of the system as well as the system architecture.
- Chapter 4 and 5 present the basic functionalities and advanced features of the system respectively.
- In Chapter 6, tests on the system and results are presented.
- Discussions is in Chapter 7
- Chapter 8 gives the conclusions and future works in this thesis.

2 Technology Background

As mentioned in the previous chapter, this data collection system is designed for NMEA 2000-compliant devices in a CAN Bus network. In the following paragraphs, we will introduce these two standards. Then we will give a description on the cellular network and Web service technology, which enable the data transmission and data publishing.

2.1 CAN Bus and NMEA 2000

CAN is the abbreviation for Controller Area Network. CAN Bus is originally developed for the automobile industry. However, CAN Bus standard is now also widely implemented in boat manufacturing. There are even some specific messages that are specially designed for the marine use [1].

The main advantage of a CAN Bus is that devices connected to it may exchange data in all directions. And the transmission via CAN Bus is very quick and fulfill the requirements for most devices. It is worth mentioning that the CAN Bus is a worldwide standard, which means that all CAN Bus-compliant devices can exchange data regardless of manufacturer. The CAN Bus system operates in a manner that all devices may listen to the messages transmitted in CAN Bus. Devices only accept the messages which are needed, while discard the others [2].

NMEA 2000, which is defined by National Marine Electronic Association, is a data network for communications between marine electronic devices. It is based on CAN Bus which connects devices together in a common channel. It means that different devices such as temperature sensors, GPS and fuel monitor can exchange the data between each other. The main goal of the standard is to share marine information in an easy way. We can say that the NMEA 2000 is a language defined based on CAN Bus. In an NMEA 2000 network, the data transmitted in CAN Bus should follow the frame structure defined in NMEA 2000 standard.

In the NMEA family, there are NMEA 0183 and NMEA 2000. NMEA 0183 standard is the predecessor of the other and it is not based on CAN Bus. One advantage of the new standard is that it has higher data rate, i.e., 256000 bps compared with 4800 bps in NMEA 0183. Another advantage is that more compact binary messages are used in NMEA 2000, which makes it more efficient than NMEA 0183. In this project, all the devices we use comply with NMEA 2000 standard.

2.2 Access Modes

In order to transmit the sensor data from the embedded system to the dedicated remote server, we need them both connected to the Internet. In this section, we will introduce three access modes for the embedded system which apply to different application scenarios.

2.2.1 Wired connection

When anchoring at the harbour, it is possible to find wired connection to the Internet. It requires that there is an RJ45 port on the embedded system so that wired access mode can be adopted.

Normally, wired connection can support higher data transmission rate than wireless connection. Take optical network for example, the data rate in uplink can reach 2.4 Gbps and 1.2 Gbps in downlink according to the standard of GPON [3].

However, when sailing, it is not possible to obtain wired connection. Thus we need to adopt wireless technology which is suitable for mobile devices.

2.2.2 Wi-Fi

Nowadays Wi-Fi is widely used in our daily life. It enables electronic devices to exchange data wirelessly at a high data rate. For example, according to IEEE 802.11n standard, the maximum data rate can reach 600 Mbps in theory by increasing the transmission bandwidth and adopting MIMO.

Although Wi-Fi is a good solution of wireless transmission, the problem lies in the limitation of transmission range and interference due to license free. In the outdoor environment, the transmission range is usually around 160 m. When sailing, it will be natural that boats sail out of the range. And because the frequency band of Wi-Fi is free to use, so it may be shared by many users and interference may occur.

Besides the Internet access, Wi-Fi can be used for local communication, e.g., establishing connection between the embedded system and a mobile router or a laptop.

2.2.3 Cellular networks

Today, mobile phones are so popular that almost every person has one. According to BBC news, there have been about 6 billion mobile phone subscriptions all over the word at the end of 2011 while the world population was nearly 7 billion [4]. By using a data-service-enabled mobile phone as the wireless access point (AP), we can easily access to the Internet via cellular networks.

The transmission range in cellular networks is about 10 to 15 km. By deploying more base stations, the coverage of cellular network is close to 100% coverage along Norwegian coasts, i.e., nearly all the offshore areas where leisure boats usually sail are under the coverage of cellular networks.

The data rate of cellular network is quite high. Take 3G networks for example, the theoretical maximum data rate is 384 kbps while moving. And under practical situation, the data rate is normally higher than 200 kbps. In an LTE network, the data rate is even much higher.

Since it is easy for us to get access to the Internet via cellular network and the data rate and coverage meet the requirements of data transmission, the cellular network can be considered as the most popular Internet access mode.

2.3 Web Service

Web-Service is a standard-based system that makes applications to communicate with an API, which transmits formatted requests from other remote machines through different transport protocols.

Generally, Web service has following characters:

- Communication over network
- Communication among multiple applications
- Interoperability between disparate systems
- Enables loosely coupled design
- Open protocol is used for establishing communication
- Exposed interface is platform independent

In our case, we used Restful Web service [5] as application for collect and check real-time data from embedded system.

Restful Web service: Representational State Transfer (REST) is an architecture style described by a researcher named Roy Fielding. In Restful service, once its functionality is enabled, service expose resources as a URI and clients can access the resources and invoke them by four HTTP verbs, which are *GET*, *PUT*, *POST* and *DELETE*, respectively.

Restful architectures have following basic principles:

- All resources use four HTTP verbs
- The Restful service is stateless
- The protocol is cacheable
- By standard URIs, resources are addressable and can be used as hypermedia links
- It is layered system
- Uniform interface

RESTful Web service allows that resources have different representations, such as JSON, TXT and XML. The RESTful client can send request for specific representations via the HTTP protocols.

2.4 Secure Shell

Nowadays, more and more people have multiple computers, such as working laptop in office and stationary desktop at home. Thus it would be much more convenient if people can make connections between these computers. For instance, you might want to execute commands in your remote computer, or transfer files between machines over network. There is variety of protocols for these functions. For example, Telnet for remote login, RCP and FTP for file transfer.

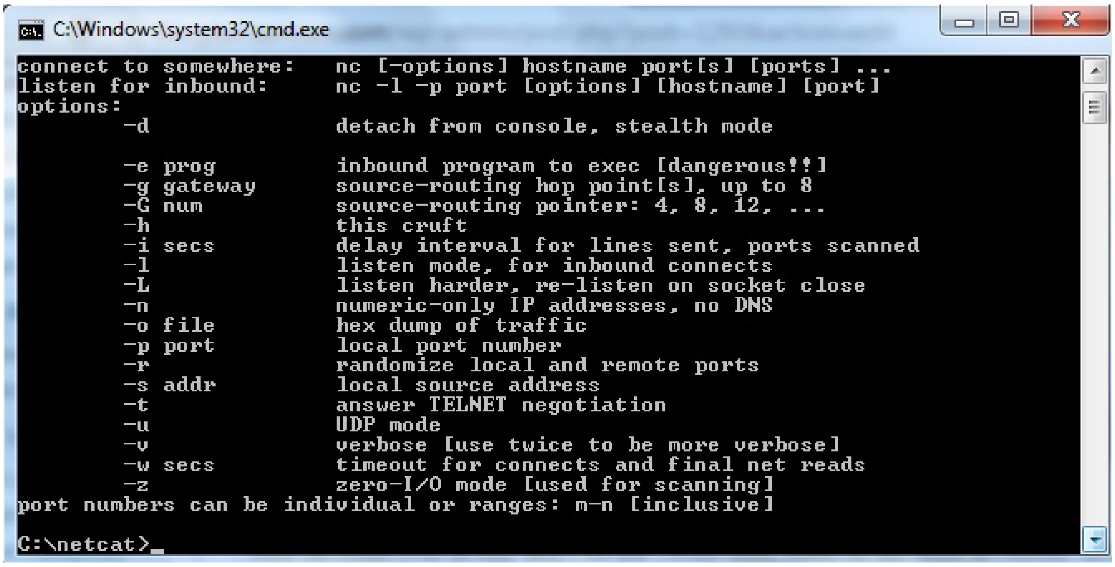
However, these protocols basically meet an inevitable problem, which is the security risk. When you transmit any important files through ftp, a potential intruder can intercept and obtain the data. Moreover, if you use telnet to access another machine and remotely execute an application, your username and password can be intercepted during the transmission.

To improve security, SSH, the secure shell was standarized by IETF, which is a popular, software-based approach [6]. Whenever the data transport through the network, SSH automatically encrypts it. After the data reached its destination, SSH automatically decrypts it. Although it has encryption and decryption during the transmission, the users can work normally and locally regardless of the process of transmission. In addition, SSH uses secure and modern encryption algorithms to provide enough protection during transmission.

2.5 Netcat

Netcat is a network debugging tool, which helps in reading and writing data cross network connections. In Netcat, it uses TCP/UDP for its functionality working across the networks. In addition, it is configurable and can be driven by scripts. There are a lot of inbuilt commands that can add different features to the utility.

The following figure shows the description of various command parameters when Netcat is working:



```
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [options] [hostname] [port]
options:
        -d              detach from console, stealth mode

        -e prog         inbound program to exec [dangerous!!]
        -g gateway      source-routing hop point[s], up to 8
        -G num          source-routing pointer: 4, 8, 12, ...
        -h              this cruft
        -i secs         delay interval for lines sent, ports scanned
        -l              listen mode, for inbound connects
        -L              listen harder, re-listen on socket close
        -n              numeric-only IP addresses, no DNS
        -o file         hex dump of traffic
        -p port         local port number
        -r              randomize local and remote ports
        -s addr         local source address
        -t              answer TELNET negotiation
        -u              UDP mode
        -v              verbose [use twice to be more verbose]
        -w secs         timeout for connects and final net reads
        -z              zero-I/O mode [used for scanning]
port numbers can be individual or ranges: m-n [inclusive]

C:\netcat>_
```

Figure 2 Description of command parameters in Netcat

In our project, it acts as a port listener that keeps listening to a specific port to check whether there is an in-coming connection request. Once the client side creates an active TCP connection to the host on the specified port, we will get a prompt on the server side and can successfully control client afterwards.

2.6 MySQL

MySQL is a relational database management system that can helps us to create a database with tables, columns and indexes. It can be used to store, sort, manage and display data content, which is reliable, fast, easy to use and suitable for application of any size.

The following features show the advantages of MySQL database:

- MySQL has an open source license, therefore it is no cost for the users.
- MySQL uses a uniform standard SQL data language
- MySQL has fast working performance and also works well with large data sets.
- MySQL widely opens interface to PHP, which contributes a lot in Web service development
- MySQL runs on more than 20 operation systems

Because of its reliability and consistent fast performance, it becomes the most popular open source database and a new choice for the LAMP stack applications (Linux, Apache, MySQL and PHP).

2.7 Secure Sockets Layer

The SSL, Secure Sockets Layer protocol is originally developed by Netscape to provide protection for the Web browser. After 20 years development, it has become the most accepted Web security standard. The main role of SSL is manage authentication and encrypted communication for Web traffic. It provides the security in terms of message integrity, authentication and confidentiality [7].

- Confidentiality, by encrypting the data message, only application endpoints understand the data
- Integrity, where the protocol will detects if any data was changed or loss during the transmission
- Authentication, which validate the identity of endpoint users or applications.

SSL achieves above security features through the use of digital signatures, certificates and cryptography.

2.8 Advanced Encryption Standard

Advanced Encryption Standard (AES) is an encryption algorithm to process the data block by using a single as a part of the encryption process. The size of key can be 128 bits, 192 bits or 256 bits. In AES, both encryption and decryption procedure are performed using the same key.

There are 6 different ways to use symmetric key in AES encryption method, which are named *Modes of Operation*. The following list shows different modes of operation that can be used in AES encryption [8].

- Counter (CTR)
- Cipher block chaining (CBC)
- Counter (CTR)
- Cipher feedback (CFB)
- Output feedback (OFB)
- Galois/Counter Mode (GCM)

By these modes, AES provides strong encryption mechanism and thus it is widely used in the field of data encryption.

In the next chapter, we will introduce the requirements to the data collection and transmission system and the structure of the system.

3 Requirements and System Design

In this chapter, we will introduce the requirement and design goal then present the overview of system architecture.

3.1 Requirements

As the main goal of this project is to develop a system used to collect, filter and store the data of marine sensors as well as to transmit data to a remote server from which people can obtain the data for analysis. The system should fulfill the following functional requirements.

Basic functionalities:

1. Data collection: to collect NMEA 2000 messages from CAN Bus network and translated the binary raw message in to the format which is human readable.
2. Data filtering: to filter data by predefined criteria.
3. Data storage: the filtered data should be properly stored and managed.
4. Data transmission: to transmit data from the embedded system to the server.
5. Data publishing: to publish the data on the Internet, users can access the appointed web page for the data.

For data transmission, there are two different working modes: real-time data transmission and historical data transmission. The former occurs immediately after the data collection and filtering while the latter is performed after the startup of Ubuntu.

Advanced features:

1. Data security: to encrypt the data in MySQL.
2. Transmission security: to encrypt the data in transmission.
3. Local control: to access and control the embedded system by using SSH.
4. Reverse control: to enable the remote server to obtain the control on the embedded system.

Basic functionalities are the needs for implementing basic operation of the system, and the advanced features are the improvements on security, local and remote control. The requirements on them are the main goals of our design, which will be introduced in detail in the next two chapters. In the following section, we will present the architecture of the system.

3.2 System Architecture

The system consists of two physical entities, one is embedded data collection system and the other is the remote server. They together form a total solution for collecting data from marine electronics on boats and publishing the data online.

3.2.1 Local component

The local component indicates the embedded system. As shown in Figure 3, the embedded system collects data from CAN Bus and then filter it. The filtered data will be stored in the local database. Meanwhile, if the embedded system and remote server are both online, the data will also be transmitted to the remote server and stored in the server's database.

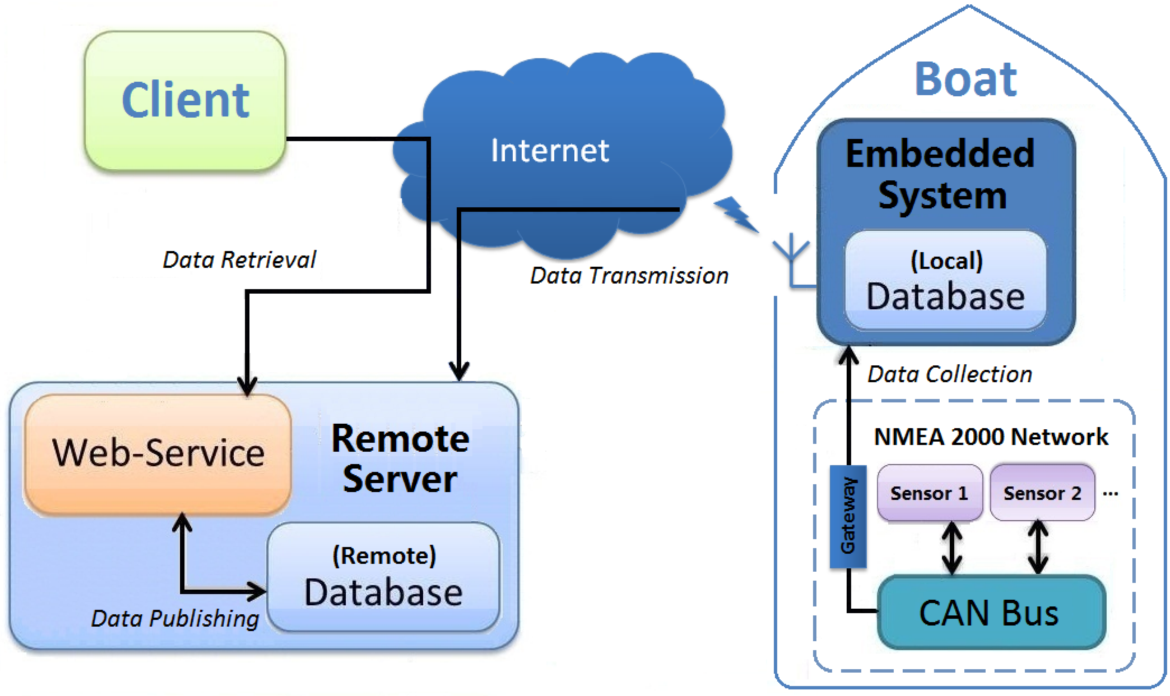


Figure 3 System architecture

3.2.2 Remote component

The remote component refers to the server which stores the data transmitted from the embedded system and publishes it online through Web service.

In the following sections, these two components will be described respectively.

3.3 Embedded System

The embedded system consists of both hardware and the software installed on it. In the following subsections, we will introduce them respectively.

3.3.1 Hardware

In the initial stage, two development boards are tested as the hardware platform of our data collection system as shown in Figure 4. One is BeagleBoard-xM and the other is PandaBoard ES. Both of them have good enough performance for running the Linux-based system.

However, as there is no integrated wireless adaptor on the BeagleBoard-xM, so we have to add a USB adapter to it. We have tried Belkin N150 micro wireless USB adapter and Edimax EW-7811Un Nano USB adapter, both are powered by Realtek's WLAN chip. During the test, it is found that the USB adapters are not well supported by Ubuntu. When transmitting the data, the embedded system disconnects from the network frequently and the connection cannot recover automatically unless rebooting the BeagleBoard-xM or re-plug the USB adapter. We check this problem on Internet and find that it is very common among Ubuntu users. It also happens to other wireless USB adapters.

After reinstalling the drivers and manually configuring the wireless network, the problem remains. Since the stability of wireless network is important for data transmission, we have to abandon it and look for a new platform.

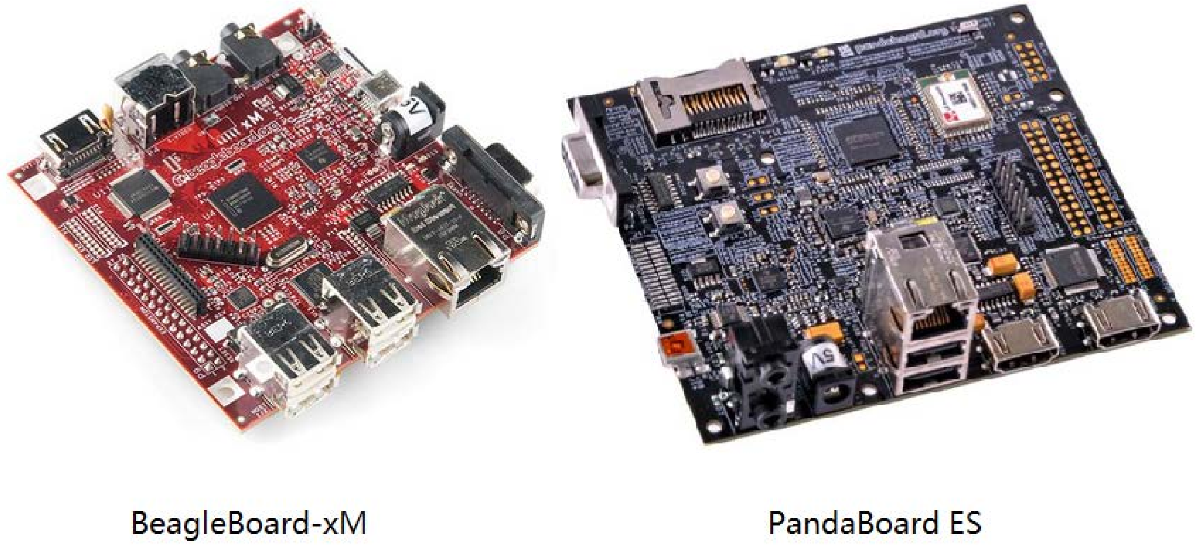


Figure 4 BeagleBoard-xM and PandaBoard ES

PandaBoard ES is a high-performance single-board computer with an on-board wireless adapter which supports IEEE 802.11 b/g/n standards. The processor is the dual-core ARM-based OMAP4460, operating frequency of which is up to 1.2 GHz. For more technical specifications, please refer to Table 1.

Table 1 Technical specifications of PandaBoard ES

Processor	OMAP4460 1.2 GHz dual-core ARM Cortex-A9 MPCore with SGX540 384 MHz Graphics Core
Memory	1 GB low power DDR2 RAM
Extension Memory	Full size SD/MMC card
Display	HDMI v1.3 connector DVI-D connector
USB Ports	2x USB 2.0 High-Speed host ports 1x USB 2.0 High-Speed On-the-go port
Connectivity	On-board 10/100 Ethernet (with RJ45 interface)
Wireless Connectivity	On-board wireless adapter (compatible with IEEE 802.11 b/g/n)
Length	114.3 mm
Width	101.6 mm
Weight	81.5 g

With the help of on-board wireless adapter, the wireless connection between PandaBoard ES and the wireless access point is stable. Even when disconnecting from the network, the on-board wireless adapter can automatically re-establish the wireless connection if the access point remains.

Due to the high performance, small size and low power consumption, the PandaBoard ES is adopted as the hardware platform for the embedded system.

3.3.2 Software

The software to be installed on PandaBoard ES includes the operating system, database, FTP client and the data collection software developed by us.

Operating system

We choose Ubuntu as our operating system as it is a popular Linux-based system. After installing different version on PandaBoard ES and BeagleBoard-xM, we find that only Ubuntu 12.04 LTS (desktop version for OMAP board) is available on them. The image of install CD can be found on the official website of Ubuntu [9]. There are two versions for OMAP3 and OMAP4 respectively. According to the model of processor, we choose OMAP4 version for PandaBoard ES.

Database

We adopt MySQL (version 5.5.22 for Ubuntu) to store and manage the data on the embedded system. Hereinafter, the database installed on the PandaBoard ES is called local database while the one on the remote server is called remote database.

In Ubuntu system, MySQL can be downloaded in the Software Center. After installing it on PandaBoard ES, we need to log in the database and create a device list table by typing the following command before perform the data storage.

```
CREATE TABLE device(id INT NOT NULL AUTO_INCREMENT UNIQUE KEY, deviceid INT UNSIGNED);
```

The data collection system will record the IDs assigned to the sensors in the NMEA 2000 network in this table. According to this table, the data collection system can detect new devices in the network and create new tables for storing the data generated by them.

FTP client

We use FTP tool to transfer the data stored on the PandaBoard ES to the remote server. Since the data contains no real-time information, it is hereinafter called historical data.

In this project, we use LFTP for historical data transmission. Because it is a command-line FTP client, by shell scripting, we can have automatic data uploading which will be introduced in Section 4.5.

Data collection software

We develop this data collection software in C by referring to part of an open source program called CANboat [10]. The files we refer to are *actisense-serial* and *analyzer*. The *actisense-serial* is used for retrieving the raw NMEA 2000 messages from the Actisense NGT-1 gateway while the *analyzer* is used for translating the raw data to human-readable formats.

All the message formats are defined in NMEA 2000 Appendix B – PGN (NMEA Network Messages) Database [11]. We have purchased NMEA 2000 Appendix B in order to use it for data translation.

We also add new features to *analyzer*. One is data filtering, which helps users to concentrate on the data of interest and reduce the traffic of unnecessary data transmission. Furthermore, by invoking the MySQL API for C, the filtered data will be store in the local database. Meanwhile, if the embedded system and remote server are both online, the data will also be transmitted to the remote server and stored in server's database. Since the data is transmitted right after being received, it is hereinafter called real-time data.

All functionalities of the software will be described in details in Chapter 4 as well as the usages.

PHP

Since we need to run a PHP script to establish the connection with the remote server for reverse control, we have to install the PHP developing environment beforehand by using the command

```
sudo apt-get install php5 libapache2-mod-php5 php5-cli php-pear php5-mysql php5-pgsql
```

3.4 Remote Server

In order to collect real-time data and monitor it in any place with Internet access, we must configure a full-function remote server that can deal with these tasks. We will elaborate our configuration of server in terms of hardware and software.

3.4.1 Hardware

In thesis, we use our own laptop as a server. The laptop is powered by an Intel Core i5 processor, and with4GB 1333 MHz RAM and 500GB storage, which support all the software we need and work smoothly.

3.4.2 Software

On the remote server, we need to install the following software and applications.

Operating system

The default operating system on the laptop is Windows 7 Professional (64-bit).

Web service developing environment

In order to provide Web service, we install the latest Netbeans version 7.3 on the laptop as development environment. In our thesis, Java is used as our main developing language for Web service establishment.

From Java EE 6 and Java EE 5, Web service standards are supported by Netbeans. We use GlassFish open source edition as our server to make service access available from external Internet. Since RESTful service is stateless for server and have a simple architecture between client and server, we decide to use it as our Web service style. Moreover, Netbeans IDE assists us to create RESTful Web services directly from MySQL, thus this functionality helps us to wrap entity beans and provide easy CRUD (create, read, update and delete) operation. Figure 5 shows the different architecture types for RESTful Web service in Netbeans.

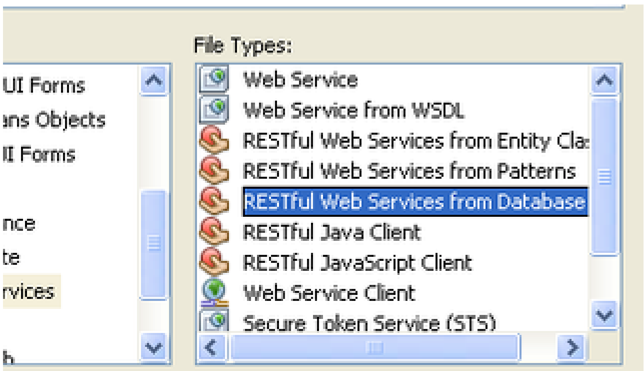

UI Forms
ans Objects
II Forms
nce
te
rvices
h
File Types:
Web Service
Web Service from WSDL
RESTful Web Services from Entity Cla:
RESTful Web Services from Patterns
RESTful Web Services from Database
RESTful Java Client
RESTful JavaScript Client
Web Service Client
Secure Token Service (STS)

Figure 5 Types of RESTful Web service in Netbeans

Database

Similar to the embedded system, MySQL (version 5.5.31 for Windows 7) is installed on the remote server for receiving and storing the data transmitted from the embedded system.

Since the default setting of access control in MySQL only allow the access from local IP address 127.0.0.1, we need to make the database can be accessed from any IP address. By using the following commands after log in MySQL:

```
using mysql;

update user set host = '%' where user = 'root' and host = 'localhost'

flush privileges;
```

Netcat installation for reverse control

Besides the function of Web service, the reverse control on the embedded system needs to be performed on the remote server. For reverse control, we installed Netcat as a role of scanner and listener on the server side.

FTP server

For historical data transmission, we need to install a FTP server on the remote server. In our case, Serv-U (version 14.0.0.6) is adopted for providing FTP service.

In this chapter, we have presented the requirements and architecture of the system. And the requirements are divided into two parts: basic functionalities and advanced features, which will be introduced in Chapter 4 and 5 respectively.

4 Implementation of Basic Functionalities

The basic functionalities meet the design requirements mentioned in Section 3.1, which includes:

1. Data collection: to collect NMEA 2000 messages from CAN Bus network and translated the binary raw message in to the format which is human readable.

2. Data filtering: to filter data by predefined criteria.

3. Data storage: the filtered data should be properly stored and managed.

4. Data transmission: to transmit data from the embedded system to the server. It consists of two parts: real-time data transmission and historical data transmission.

5. Data publishing: to publish the data on the Internet, users can access the appointed web page for the data.

Function 1, 2, 3 and real-time data transmission are implemented by programming in C using MySQL API for C. For the implementation of historical data transmission, Linux shell scripting is used with LFTP. And the data publishing on the remote server is implemented by using RESTful Web service architecture with MySQL.

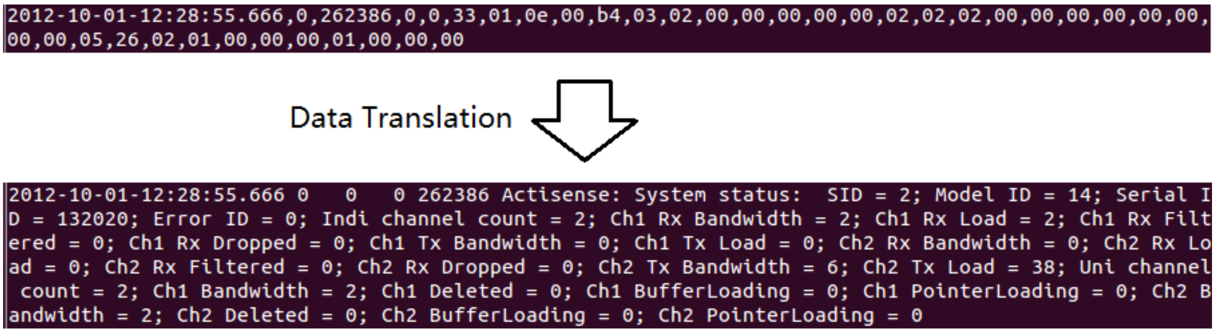
With these functionalities, the embedded system and the server constitute an integral system which provides data collection and transmission service.

4.1 Data Collection

As mentioned above, using the program CANboat [10], we are able to collect data from CAN Bus. Before performing data collection, we need to install the NGT-1 gateway on PandaBoard ES. Refer to Appendix B for the installation guide. Then input the following command in the terminal to display the collected messages.

actisense-serial /dev/ttyUSB0 | *analyzer*

The actisense-serial reads the raw data from the *ttyUSB0* (NGT-1 gateway) and output it to the standard output stream. If we use *actisense-serial* alone, the raw data will be displayed in the terminal. When used with analyzer, the output of *actisense-serial* will become the input of *analyzer* and the raw data will be translated into human-readable message.



```
2012-10-01-12:28:55.666,0,262386,0,0,33,01,0e,00,b4,03,02,00,00,00,00,00,02,02,02,00,00,00,00,00,00,
00,00,05,26,02,01,00,00,00,01,00,00,00
```

```
2012-10-01-12:28:55.666 0   0   0 262386 Actisense: System status:  SID = 2; Model ID = 14; Serial I
D = 132020; Error ID = 0; Indi channel count = 2; Ch1 Rx Bandwidth = 2; Ch1 Rx Load = 2; Ch1 Rx Filt
ered = 0; Ch1 Rx Dropped = 0; Ch1 Tx Bandwidth = 0; Ch1 Tx Load = 0; Ch2 Rx Bandwidth = 0; Ch2 Rx Lo
ad = 0; Ch2 Rx Filtered = 0; Ch2 Rx Dropped = 0; Ch2 Tx Bandwidth = 6; Ch2 Tx Load = 38; Uni channel
 count = 2; Ch1 Bandwidth = 2; Ch1 Deleted = 0; Ch1 BufferLoading = 0; Ch1 PointerLoading = 0; Ch2 B
andwidth = 2; Ch2 Deleted = 0; Ch2 BufferLoading = 0; Ch2 PointerLoading = 0
```

Figure 6 Translate raw data to human-readable message

As shown in Figure 6, the raw data mainly consist of a group of 8-bit binary data which can be translated according to the definitions in NMEA 2000 Appendix B [11].

4.2 Data Filtering

The first functionality we add to the system is data filtering. Because there could be different types of sensors in an NMEA 2000 network while every sensor sends multiple types of messages, there will be abundant of messages generated and transmitted in a short period of time. Data bursts onto the screen and updates at a high frequency, which leads to high data traffic and it is impossible for human eye to catch the information. Furthermore, we sometimes want only to store the data in which we are interested. Therefore, data filtering is a necessary feature for our system.

We develop the system in the way that data can be filtered according to different criteria set by users. In Table 2, we can see that there are criteria apply to different scenarios.

Table 2 Filtering criteria and application scenarios

Criteria	Application Scenario
Filtering by device	ID of desired or undesired device is known
Filtering by type of message	PGN of designed message type is known
Filtering by message update interval	Message update frequency needs to be set

The filtering criteria can be adopted individually or concurrently. If multiple criteria are designated simultaneously, they will be performed according to the priority illustrated in Figure 7.

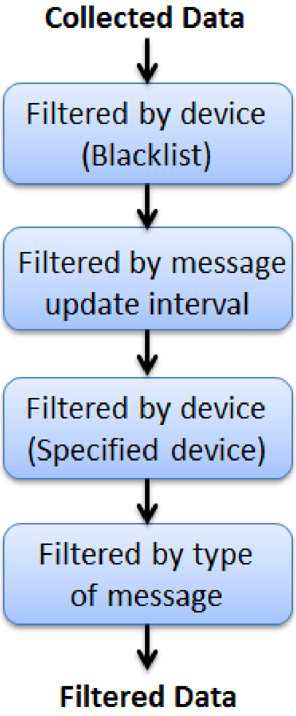

Collected Data
Filtered by device (Blacklist)
Filtered by message update interval
Filtered by device (Specified device)
Filtered by type of message
Filtered Data

Figure 7 Priority of filtering criteria

For the instructions on using the criteria, Please refer to the following subsections.

4.2.1 Filtering by device

Every device in an NMEA 2000 network will be assigned with a unique 8-bit device ID [12], which is an integer ranges from 0 to 254 (255 is broadcast ID). And when a message is transmitted, the device ID of the transmitter is included in the message. Thus the messages from different devices can be distinguished according to the transmitters' IDs. We define two different filtering rules for filtering by device.

Specified device

Rule 1 is used to retrieve data only from the specified device. Refer to Table 3 for the usage.

Table 3 Rule 1: Specified device

Purpose	To extract the messages from a specified device
Usage	actisense-serial /dev/ttyUSB0 | analyzer –src <Device ID>

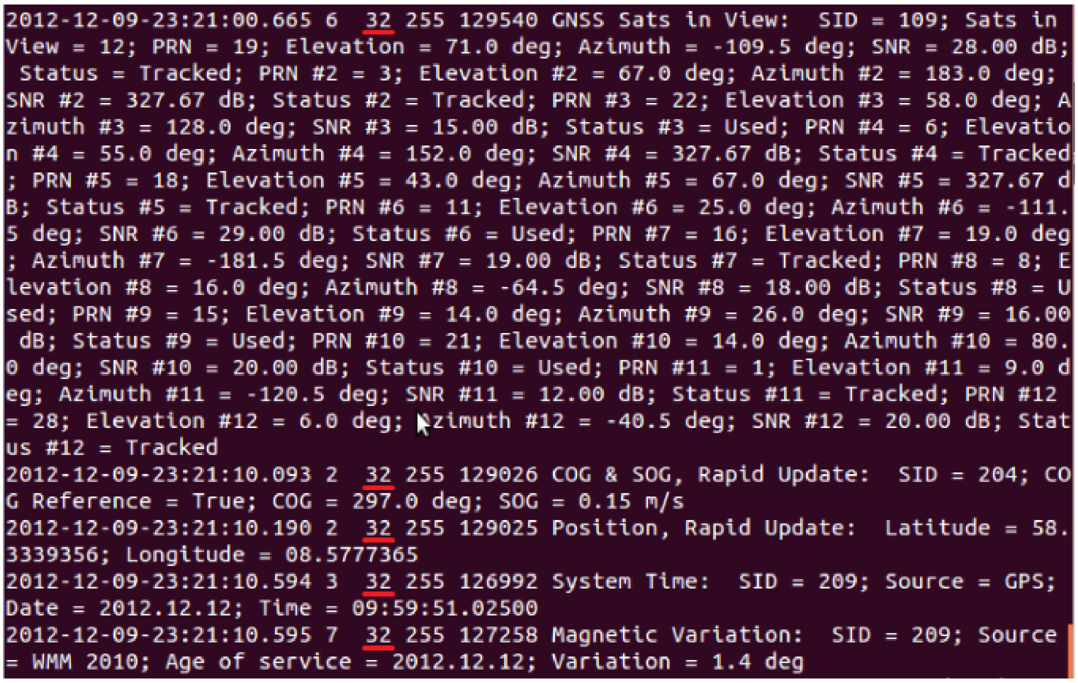
For instance, this rule applies to the scenario that only GPS information are needed, and the device ID of GPS receiver is known (ID is 32), the following steps should be performed:

Step 1: Open the terminal and go to the directory of actisense-serial and analyzer.

Step 2: Type the following command:

./actisense-serial -r /dev/ttyUSB0 | *./analyzer -src 32*

By specifying the device ID *32*, all the messages from the GPS receiver will be captured and displayed on the screen as shown in Figure 8.



```
2012-12-09-23:21:00.665 6  32 255 129540 GNSS Sats in View:  SID = 109; Sats in
View = 12; PRN = 19; Elevation = 71.0 deg; Azimuth = -109.5 deg; SNR = 28.00 dB;
 Status = Tracked; PRN #2 = 3; Elevation #2 = 67.0 deg; Azimuth #2 = 183.0 deg;
SNR #2 = 327.67 dB; Status #2 = Tracked; PRN #3 = 22; Elevation #3 = 58.0 deg; A
zimuth #3 = 128.0 deg; SNR #3 = 15.00 dB; Status #3 = Used; PRN #4 = 6; Elevatio
n #4 = 55.0 deg; Azimuth #4 = 152.0 deg; SNR #4 = 327.67 dB; Status #4 = Tracked
; PRN #5 = 18; Elevation #5 = 43.0 deg; Azimuth #5 = 67.0 deg; SNR #5 = 327.67 d
B; Status #5 = Tracked; PRN #6 = 11; Elevation #6 = 25.0 deg; Azimuth #6 = -111.
5 deg; SNR #6 = 29.00 dB; Status #6 = Used; PRN #7 = 16; Elevation #7 = 19.0 deg
; Azimuth #7 = -181.5 deg; SNR #7 = 19.00 dB; Status #7 = Tracked; PRN #8 = 8; E
levation #8 = 16.0 deg; Azimuth #8 = -64.5 deg; SNR #8 = 18.00 dB; Status #8 = U
sed; PRN #9 = 15; Elevation #9 = 14.0 deg; Azimuth #9 = 26.0 deg; SNR #9 = 16.00
 dB; Status #9 = Used; PRN #10 = 21; Elevation #10 = 14.0 deg; Azimuth #10 = 80.
0 deg; SNR #10 = 20.00 dB; Status #10 = Used; PRN #11 = 1; Elevation #11 = 9.0 d
eg; Azimuth #11 = -120.5 deg; SNR #11 = 12.00 dB; Status #11 = Tracked; PRN #12
= 28; Elevation #12 = 6.0 deg; Azimuth #12 = -40.5 deg; SNR #12 = 20.00 dB; Stat
us #12 = Tracked
2012-12-09-23:21:10.093 2  32 255 129026 COG & SOG, Rapid Update:  SID = 204; CO
G Reference = True; COG = 297.0 deg; SOG = 0.15 m/s
2012-12-09-23:21:10.190 2  32 255 129025 Position, Rapid Update:  Latitude = 58.
3339356; Longitude = 08.5777365
2012-12-09-23:21:10.594 3  32 255 126992 System Time:  SID = 209; Source = GPS;
Date = 2012.12.12; Time = 09:59:51.02500
2012-12-09-23:21:10.595 7  32 255 127258 Magnetic Variation:  SID = 209; Source
= WMM 2010; Age of service = 2012.12.12; Variation = 1.4 deg
```

Figure 8 Messages from specified device

Blacklist

Rule 2 is used to discard the data from the device(s) listed in the blacklist, the remaining of the data will be captured and displayed.

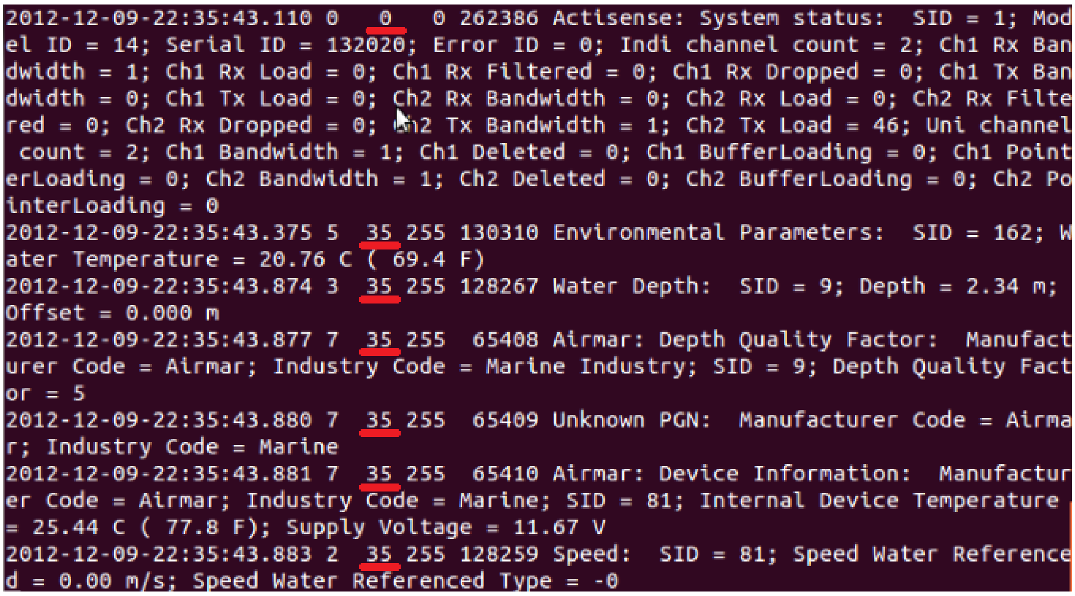
Table 4 Rule 2: Blacklist

Purpose	To extract the messages from devices which are not in the blacklist
Usage	actisense-serial /dev/ttyUSB0 | analyzer –nsrc <Device ID1>< Device ID2 (optional)>…

Take our testbed for example, there are four devices connected by a CAN bus: a DST110 Triducer (ID is 112), a GPS receiver (ID is 32), a TLM100 tank level monitor (ID is 35) and an Actisense NGT-1 USB Gateway (ID is 0). In this NMEA 2000 network, if users are only interested in the messages from the tank level monitor and Actisense NGT-1 USB Gateway. Then the following command should be used for collecting data only from these two devices.

./actisense-serial -r /dev/ttyUSB0 | *./analyzer -nsrc 32 112*

By listing all the unwanted devices in the blacklist, the messages sent by the GPS receiver and DST110 Triducer will be discarded. On contrary, the messages from the tank level monitor and NGT-1 Gateway, which are not in the list, are retrieved and displayed on the screen as shown in Figure 9.



```
2012-12-09-22:35:43.110 0   0   0 262386 Actisense: System status:  SID = 1; Mod
el ID = 14; Serial ID = 132020; Error ID = 0; Indi channel count = 2; Ch1 Rx Ban
dwidth = 1; Ch1 Rx Load = 0; Ch1 Rx Filtered = 0; Ch1 Rx Dropped = 0; Ch1 Tx Ban
dwidth = 0; Ch1 Tx Load = 0; Ch2 Rx Bandwidth = 0; Ch2 Rx Load = 0; Ch2 Rx Filte
red = 0; Ch2 Rx Dropped = 0; Ch2 Tx Bandwidth = 1; Ch2 Tx Load = 46; Uni channel
 count = 2; Ch1 Bandwidth = 1; Ch1 Deleted = 0; Ch1 BufferLoading = 0; Ch1 Point
erLoading = 0; Ch2 Bandwidth = 1; Ch2 Deleted = 0; Ch2 BufferLoading = 0; Ch2 Po
interLoading = 0
2012-12-09-22:35:43.375 5  35 255 130310 Environmental Parameters:  SID = 162; W
ater Temperature = 20.76 C ( 69.4 F)
2012-12-09-22:35:43.874 3  35 255 128267 Water Depth:  SID = 9; Depth = 2.34 m;
Offset = 0.000 m
2012-12-09-22:35:43.877 7  35 255  65408 Airmar: Depth Quality Factor:  Manufact
urer Code = Airmar; Industry Code = Marine Industry; SID = 9; Depth Quality Fact
or = 5
2012-12-09-22:35:43.880 7  35 255  65409 Unknown PGN:  Manufacturer Code = Airma
r; Industry Code = Marine
2012-12-09-22:35:43.881 7  35 255  65410 Airmar: Device Information:  Manufactur
er Code = Airmar; Industry Code = Marine; SID = 81; Internal Device Temperature
= 25.44 C ( 77.8 F); Supply Voltage = 11.67 V
2012-12-09-22:35:43.883 2  35 255 128259 Speed:  SID = 81; Speed Water Reference
d = 0.00 m/s; Speed Water Referenced Type = -0
```

Figure 9 Messages from devices not in the blacklist

Please note that multiple devices can be put into the blacklist while only one can be selected as the specified device. And if a device is designated as the specified device while it is in the blacklist, then the messages from this device will be filtered out since Rule 2 has higher priority than Rule 1 as shown in Figure 7.

4.2.2 Filtering by type of message

The messages transmitted in the NMEA 2000 network are organized into parameter groups that are identified by Parameter Group Number (PGN), i.e., the PGN identifies the types of messages. Thus we can differentiate messages according to their PGNs.

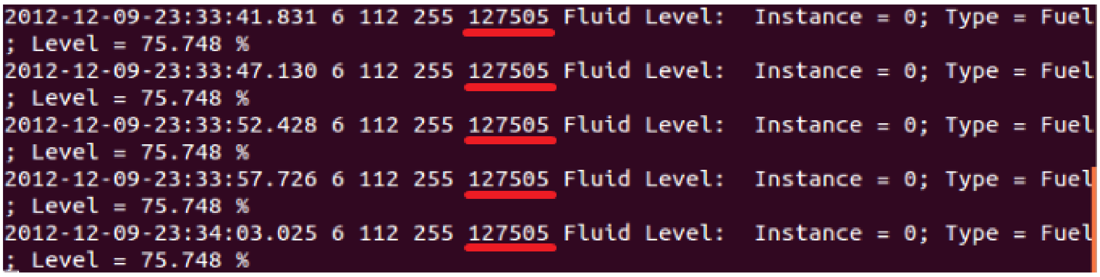
Table 5 Filtering by PGN

Purpose	To extract a specified type of messages
Usage	actisense-serial /dev/ttyUSB0 | analyzer <PGN>

Take the TLM100 tank level monitor as example. It generates six types of messages and each one has a PGN. One of them contains the information of fuel level. When users are only interested in this type of information, the following command can be used for obtaining the specified type of messages (PGN is 127505). The parameter and argument *–itv 5* will be explained in the next subsection.

./actisense-serial -r /dev/ttyUSB0 | *./analyzer 127505 –itv 5*

The output is shown in Figure 10.



```
2012-12-09-23:33:41.831 6 112 255 127505 Fluid Level:  Instance = 0; Type = Fuel
; Level = 75.748 %
2012-12-09-23:33:47.130 6 112 255 127505 Fluid Level:  Instance = 0; Type = Fuel
; Level = 75.748 %
2012-12-09-23:33:52.428 6 112 255 127505 Fluid Level:  Instance = 0; Type = Fuel
; Level = 75.748 %
2012-12-09-23:33:57.726 6 112 255 127505 Fluid Level:  Instance = 0; Type = Fuel
; Level = 75.748 %
2012-12-09-23:34:03.025 6 112 255 127505 Fluid Level:  Instance = 0; Type = Fuel
; Level = 75.748 %
```

Figure 10 Messages with a specified PGN

For another example, a TLM100 (device ID is 112) is installed in the main tank and another one (device id is 113) in the auxiliary tank respectively, if users only care about the fuel level in the main tank, the following command can be used:

./actisense-serial -r /dev/ttyUSB0 | *./analyser –src 112 127505*

Only the fluid level messages from the main tank will be displayed. It is also an example of data filtering under multiple criteria. The software can automatically distinguish PGNs from Device IDs, because the Device ID ranges from 0 to 254 while PGNs are all greater than 256.

4.2.3 Filtering by message update interval

Message update interval is the time interval between two messages which are of the same type and from the same sender. As mentioned in Section 4.2, it is conventional that marine electronics generate messages at a high frequency. Thus a large number of messages need to be stored or transmitted. If users want to reduce the memory consumption or the transmission data traffic, the command in Table 6 will be useful.

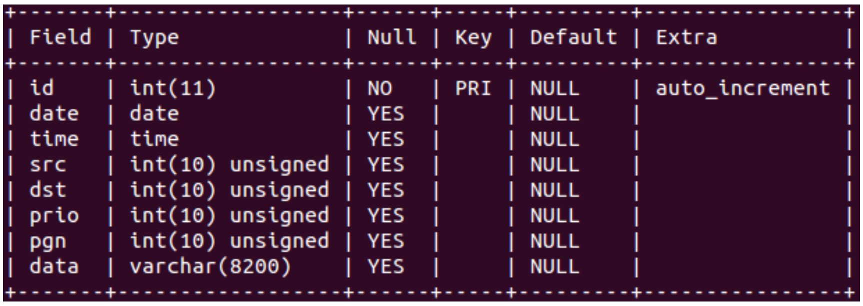
Table 6 Filtering by update interval

Purpose	To change the update frequency of messages
Usage	actisense-serial /dev/ttyUSB0 | analyzer –itv <Update Interval (0-59)>

The message update interval can be set in the range from 0 to 59 (in seconds). Set 0 for storing and displaying all the messages. If another value *n* is set, message will be updated only once during each update interval (which is equal or larger than *n* seconds). For example, as shown in Figure 10, the messages updates every 5 seconds. If we set the interval to 10 second, within the same duration, the output data will be half-sized.

4.3 Data Storage

After being filtered, the data will be sorted by device and stored in the local MySQL database. If the data is from a new device, the system will create a new table automatically and then insert the data into it.



```
+-------+-------------------+------+-----+---------+----------------+
| Field | Type              | Null | Key | Default | Extra          |
+-------+-------------------+------+-----+---------+----------------+
| id    | int(11)           | NO   | PRI | NULL    | auto_increment |
| date  | date              | YES  |     | NULL    |                |
| time  | time              | YES  |     | NULL    |                |
| src   | int(10) unsigned  | YES  |     | NULL    |                |
| dst   | int(10) unsigned  | YES  |     | NULL    |                |
| prio  | int(10) unsigned  | YES  |     | NULL    |                |
| pgn   | int(10) unsigned  | YES  |     | NULL    |                |
| data  | varchar(8200)     | YES  |     | NULL    |                |
+-------+-------------------+------+-----+---------+----------------+
```

Figure 11 Table structure in local database

All the tables for storing data are named by the ID assigned to devices in NMEA 2000 network, e.g., the data generated by GPS receiver (ID is 32) will be put into the table *src32* while the information of engine (ID is 17) will be stored in table *src17*. These tables have an identical structure which is shown in Figure 11.

The definitions of the parameters in table are as follows:

- id: a serial number for each message in table;
- date: the date on which the message is received;
- time: the time when the message is received;
- src: ID of the device by which the message is generated;
- dst: ID of the device to which the message is sent. 255 is the broadcast ID;
- prio: priority of the message;
- pgn: PGN of the message which defines the message type;
- data: content of the message.

Except the message serial number, all the other parameters are intercepted from messages. It is useful for the user who searches for specified messages.

4.4 Data Transmission

In order to publish the data online, the data collected by the embedded system needs to be transmitted to the remote server. According to the timeliness of data, we have two data transmission modes.

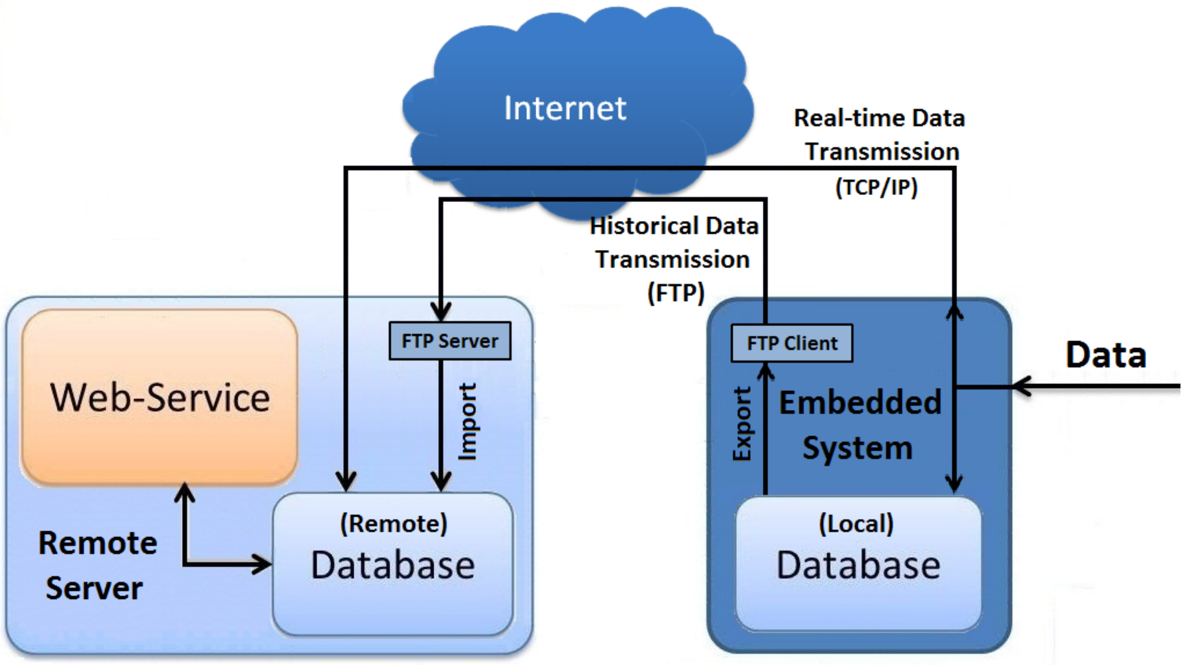


Figure 12 Two modes for data transmission

As shown in Figure 12, the collected data will be immediately transmitted to the remote server and simultaneously stored in the local database on the embedded system. The data in local database is called historical data and the one transmitted in real time is called real-time data.

4.4.1 Real-time data transmission

The real-time data transmission starts together with data collection. And it is implemented by establishing TCP/IP connectivity between the embedded system and remote MySQL database.

The source code for real-time transmission in *analyser.c* is as follows.

```
 1 MYSQL re_connection;
 2 int res;
 3 mysql_init(&re_connection);
 4 if (mysql_real_connect(&re_connection, <IP of remote database>, <username>, <password>,
   <name of database>, 0, NULL, CLIENT_FOUND_ROWS))
 5 {
 6     char sql_insert_re[8400];
 7     sprintf(sql_insert_re, "INSERT INTO b_001 values(NULL, '%s', '%s', '%s', %u, %u, %u, %u,
       '%s');",<boatname>, strdate, strtime, prio_sql, src_sql, dst_sql, pgn_sql, mbuf);
 8     res = mysql_query(&re_connection, sql_insert_re);
 9     if (res)
10     {
11         printf("Fail to write to REMOTE database\n");
12     }
13 }
14 else
15 {
16     printf("Fail to connect to REMOTE database\n");
17 }
18 mysql_close(&re_connection);
```

Each filtered message will be passed to the function with above codes for real-time transmission. The first line of the codes defines a MySQL connection, and then it is initialized in Line 3. In line 4, function *mysql_real_connect()* is used to establish the connect to the remote database with correct IP of remote database, username, password and database name. The 7th line is for insert the message to the remote database according the data structure. After transmitting the message, use *mysql_close()* to close the connection. The process will repeat when another message is passed to the function.

If no available connection between the embedded system and remote server, after a few seconds, the operation will be time-out and return an error message -- *Fail to connect to REMOTE database*. However, the message will still be stored into the local database and then the system will begin to process the next message.

4.4.2 Historical data transmission

The messages in the local database contain the past information of boats, and are uploaded to the server through FTP transmission. Although the historical data contains no real-time information, it includes complete information on the running state of boats which holds value to boat manufacturers and owners for data analysis.

In order to simplify the operation of FTP uploading, we develop the following shell script *upload.sh*.

```
 1 #!bin/bash
 2 mysqldump –u<MySQL username> -p<MySQL password> b_001 > <directory of exported
   data>/b_001-$(date +%y%m%d).sql
 3 lftp -e "set net:timeout 10;set net:max-retries 3" <<EOF
 4 open <IP of FTP server>:<port no>
 5 user <FTP username> <FTP password>
 6 put <directory of exported data>/b_001-$(date +%y%m%d).sql
 7 bye
 8 EOF
 9 rm <directory of exported data>/b_001-$(date +%y%m%d).sql
10 rm <directory of exported data>/b_001-$(date +%y%m%d -d "1 day ago").sql
```

The first line defines the script compiler. The second one is used to export the data from MySQL and save it as an *sql* file under the appointed directory. Line 3 is used to set the timeout and retry limit for connecting to FTP server in avoiding infinity reconnection. Line 4, 5 and 6 are for logging in FTP server login and transferring the file of exported data. Finally, the last two lines are used for deleting the files created on the day and the day before.

With the shell script, we still need to type the following command manually in the terminal to start the operation.

```
cd <directory of upload.sh>
sh upload.sh
```

In order to make the embedded system self-running, we find a solution which will be described in the next section.

4.5 Automatic Operation of Data Collection and Transmission

For setting up automatic operation, refer to the instructions bellow.

Step 1: Create a shell script *data.sh* for running the data collection system on PandaBoard ES.

```
#!bin/bash
cd <directory of actisense-serial and analyzer>
./actisense-serial /dev/ttyUSB0 | ./analyzer -itv 30
```

Step 2: Click the system button on the top right corner and select *Startup Applications* as shown in Figure 13.

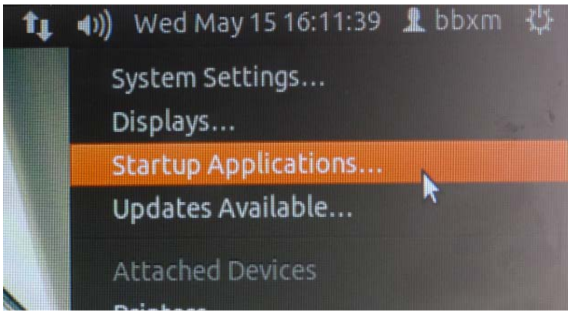
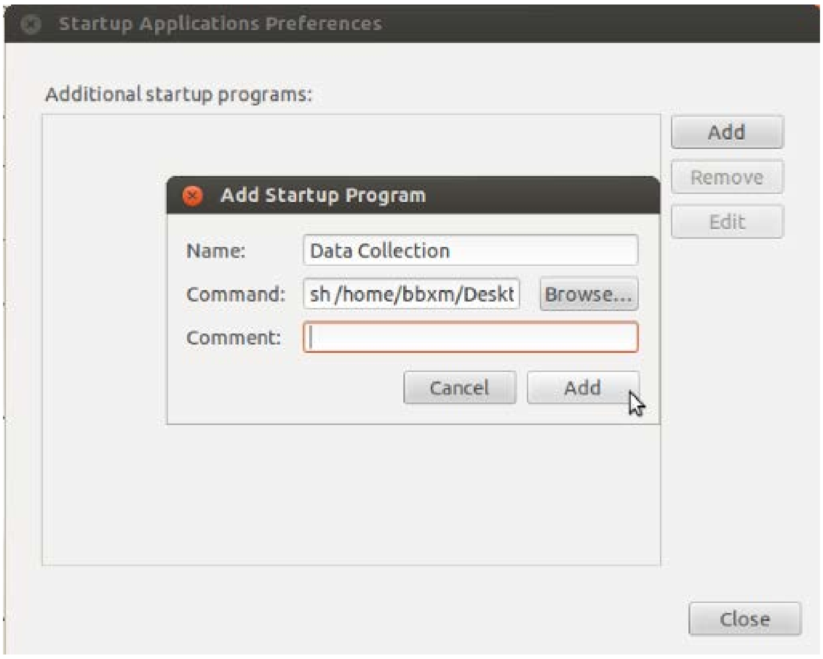


Figure 13 Configuration of startup applications

Step 3: Click *Add* button, then input the name and command for adding *data.sh.*


Startup Applications Preferences
Additional startup programs:
Add
Remove
Edit
Add Startup Program
Browse...
Cancel
Add
Close

Name: Data Collection

Command: sh /home/bbxm/Deskt

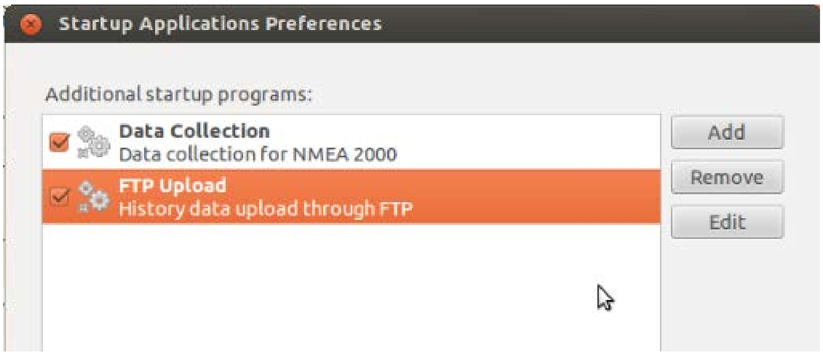
Comment:

Figure 14 Add startup program

The command should be in the following format:

sh <path name to the script>

Step 4: Repeat Step 3 to add *upload.sh* to startup programs.


Startup Applications Preferences
Additional startup programs:
Add
Remove
Edit

☑ **Data Collection**
Data collection for NMEA 2000

☑ **FTP Upload**
History data upload through FTP

Figure 15 List of startup programs

Now, after system startup, the data collection and real-time transmission will be started and keep running in background. Meanwhile, historical data will be exported from database and uploaded to the FTP server. The process will be terminated when either the exported data is successfully transmitted or connecting requests exceed the limit.

4.6 Data Browsing

Since data is already filtered and transmitted by embedded system, users need to check it both from the server and external Internet. For boat administrative staff, browsing the data on the server can help them know whether data is successfully transmitted to the server and check the boat's current state. For the people who only have the Internet connection but want to browse the data, they can directly access the Web-service. It displays all the sensors' parameters which are exactly same as it in the server side.

4.6.1 Data browsing on the embedded system

For the boat owners, it is easy to log in the local database and view the data by using following MySQL commands on the embedded system.

Step 1: Open the Ubuntu terminal and type:

```
mysql –u<username> -p<password>
```

Step 2: Choose the database in which the data is stored.

```
use <database name>;
```

Step 3: There could be multiple tables in the selected database. Show all tables by typing:

```
show tables;
```

Step 4: View the data in a dedicated table.

```
Select * from <table name>;
```

4.6.2 Data browsing on the remote server

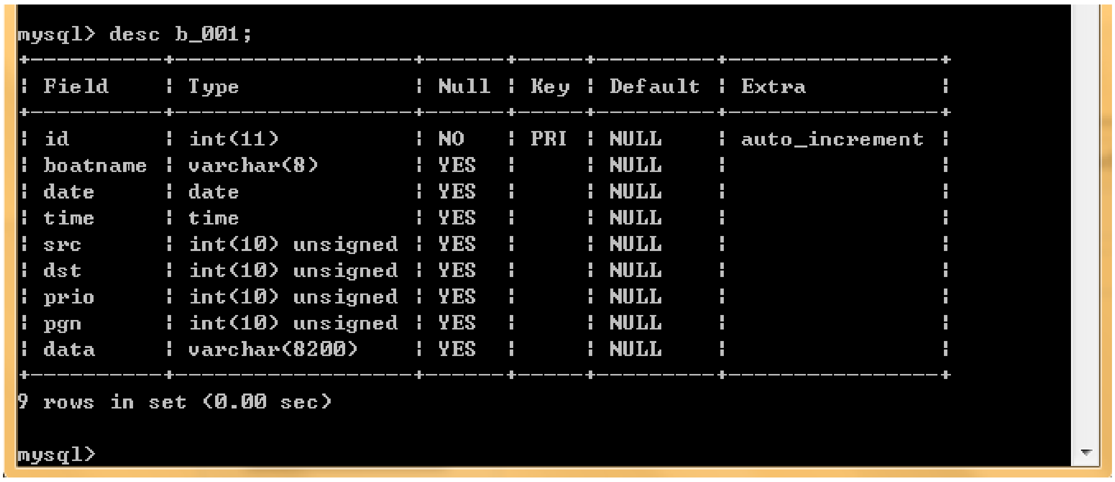
As we mentioned in data storage part, the real-time data is stored both in local embedded system and remote server. In our thesis, we have designed two approaches to view data in server side: by using MySQL command application or viewing it from Netbeans IDE.

Viewing data though MySQL command line client

Firstly, we launch the MySQL command line client and type the enter password to access. Using the *ecoboat* database, then type the command *show tables*. As shown in Figure 16, it displays all the current tables in our *ecoboat* database.

By using the following command, we can see table structure in Figure 16.

```
describe b_001;
```



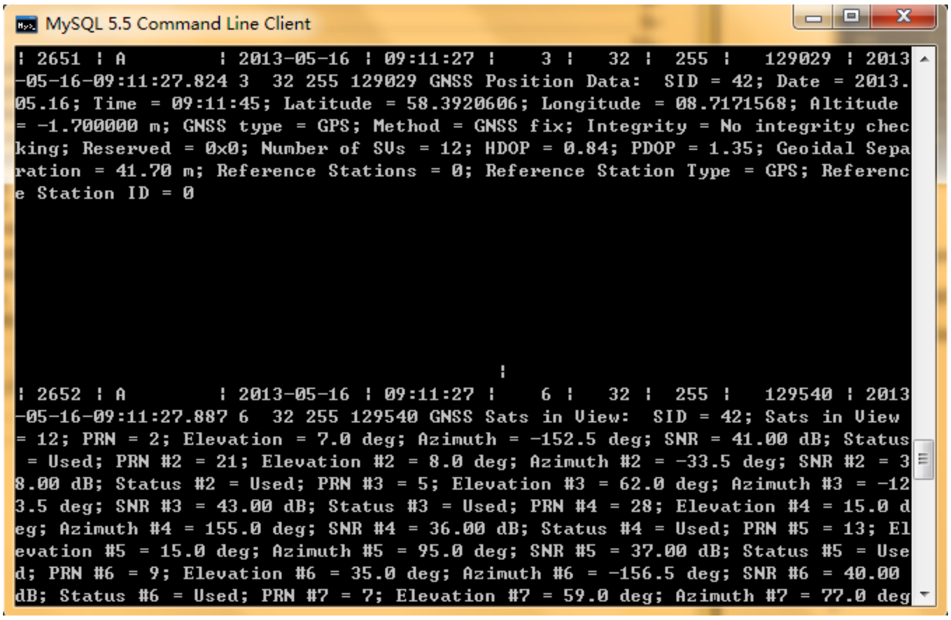
```
mysql> desc b_001;
+----------+-------------------+------+-----+---------+----------------+
¦ Field    ¦ Type              ¦ Null ¦ Key ¦ Default ¦ Extra          ¦
+----------+-------------------+------+-----+---------+----------------+
¦ id       ¦ int(11)           ¦ NO   ¦ PRI ¦ NULL    ¦ auto_increment ¦
¦ boatname ¦ varchar(8)        ¦ YES  ¦     ¦ NULL    ¦                ¦
¦ date     ¦ date              ¦ YES  ¦     ¦ NULL    ¦                ¦
¦ time     ¦ time              ¦ YES  ¦     ¦ NULL    ¦                ¦
¦ src      ¦ int(10) unsigned  ¦ YES  ¦     ¦ NULL    ¦                ¦
¦ dst      ¦ int(10) unsigned  ¦ YES  ¦     ¦ NULL    ¦                ¦
¦ prio     ¦ int(10) unsigned  ¦ YES  ¦     ¦ NULL    ¦                ¦
¦ pgn      ¦ int(10) unsigned  ¦ YES  ¦     ¦ NULL    ¦                ¦
¦ data     ¦ varchar(8200)     ¦ YES  ¦     ¦ NULL    ¦                ¦
+----------+-------------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)

mysql>
```

Figure 16 Table structure on remote server

In order to view the real-time data in this table, we need to enter the command:

*select * from b_001;*

Then all the collected data in client will be displayed as illustrated in Figure 17.



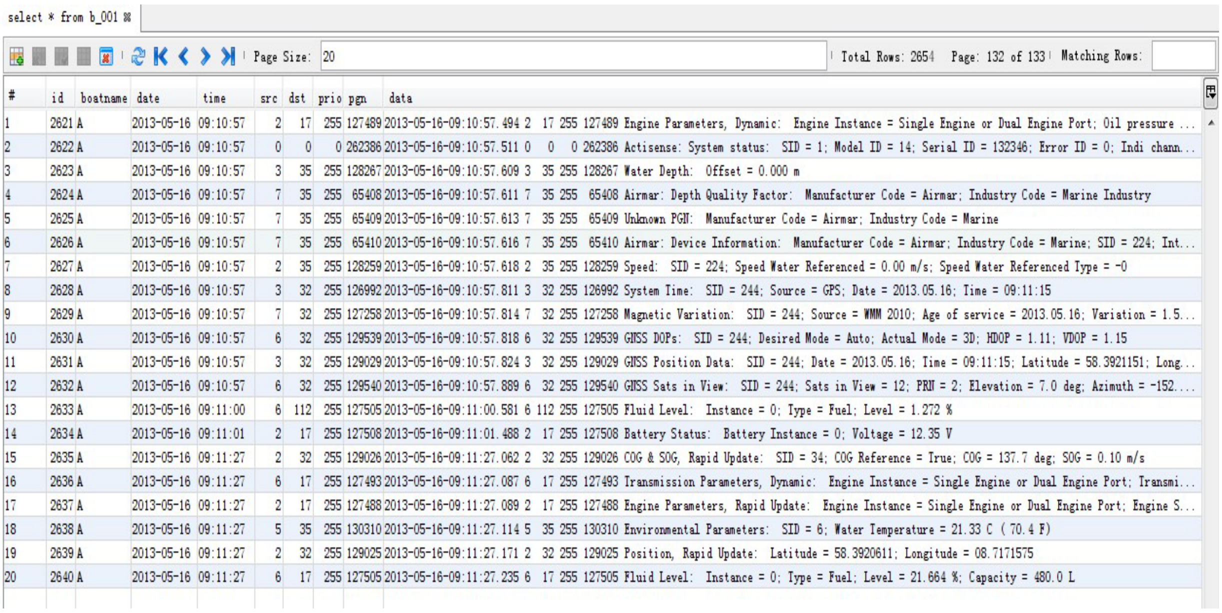
```
¦ 2651 ¦ A          ¦ 2013-05-16 ¦ 09:11:27 ¦    3 ¦   32 ¦  255 ¦   129029 ¦ 2013
-05-16-09:11:27.824 3  32 255 129029 GNSS Position Data:  SID = 42; Date = 2013.
05.16; Time = 09:11:45; Latitude = 58.3920606; Longitude = 08.7171568; Altitude
= -1.700000 m; GNSS type = GPS; Method = GNSS fix; Integrity = No integrity chec
king; Reserved = 0x0; Number of SVs = 12; HDOP = 0.84; PDOP = 1.35; Geoidal Sepa
ration = 41.70 m; Reference Stations = 0; Reference Station Type = GPS; Referenc
e Station ID = 0





                                           ¦
¦ 2652 ¦ A          ¦ 2013-05-16 ¦ 09:11:27 ¦    6 ¦   32 ¦  255 ¦   129540 ¦ 2013
-05-16-09:11:27.887 6  32 255 129540 GNSS Sats in View:  SID = 42; Sats in View
= 12; PRN = 2; Elevation = 7.0 deg; Azimuth = -152.5 deg; SNR = 41.00 dB; Status
 = Used; PRN #2 = 21; Elevation #2 = 8.0 deg; Azimuth #2 = -33.5 deg; SNR #2 = 3
8.00 dB; Status #2 = Used; PRN #3 = 5; Elevation #3 = 62.0 deg; Azimuth #3 = -12
3.5 deg; SNR #3 = 43.00 dB; Status #3 = Used; PRN #4 = 28; Elevation #4 = 15.0 d
eg; Azimuth #4 = 155.0 deg; SNR #4 = 36.00 dB; Status #4 = Used; PRN #5 = 13; El
evation #5 = 15.0 deg; Azimuth #5 = 95.0 deg; SNR #5 = 37.00 dB; Status #5 = Use
d; PRN #6 = 9; Elevation #6 = 35.0 deg; Azimuth #6 = -156.5 deg; SNR #6 = 40.00
dB; Status #6 = Used; PRN #7 = 7; Elevation #7 = 59.0 deg; Azimuth #7 = 77.0 deg
```

Figure 17 Viewing data on remote server

Viewing data though Netbeans IDE

Another approach to see data on the server side is to use our Netbeans IDE since the IDE has integrated and synchronized the MySQL database. It is a user-friendly graphic interface so that the data structure and contents will be showed more clearly in it.

As shown in Figure 18, by opening the Netbeans and choosing *Database* in the left bar, to launch java database driver *ecoboat* and choose the *Tables* under the ecoboat directory. There are 2 tables in the list: *b_001* and *src32*, right click on b_001 and view data, we can observe that all the data that will be display in the IDE.



select * from b_001

Page Size: 20

Total Rows: 2654 Page: 132 of 133 Matching Rows:

#	id	boatname	date	time	src	dst	prio	pgn	data
1	2621	A	2013-05-16	09:10:57	2	17	255	127489	2013-05-16-09:10:57.494 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pressure ...
2	2622	A	2013-05-16	09:10:57	0	0	0	262386	2013-05-16-09:10:57.511 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi chann...
3	2623	A	2013-05-16	09:10:57	3	35	255	128267	2013-05-16-09:10:57.609 3 35 255 128267 Water Depth: Offset = 0.000 m
4	2624	A	2013-05-16	09:10:57	7	35	255	65408	2013-05-16-09:10:57.611 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
5	2625	A	2013-05-16	09:10:57	7	35	255	65409	2013-05-16-09:10:57.613 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
6	2626	A	2013-05-16	09:10:57	7	35	255	65410	2013-05-16-09:10:57.616 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 224; Int...
7	2627	A	2013-05-16	09:10:57	2	35	255	128259	2013-05-16-09:10:57.618 2 35 255 128259 Speed: SID = 224; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
8	2628	A	2013-05-16	09:10:57	3	32	255	126992	2013-05-16-09:10:57.811 3 32 255 126992 System Time: SID = 244; Source = GPS; Date = 2013.05.16; Time = 09:11:15
9	2629	A	2013-05-16	09:10:57	7	32	255	127258	2013-05-16-09:10:57.814 7 32 255 127258 Magnetic Variation: SID = 244; Source = WMM 2010; Age of service = 2013.05.16; Variation = 1.5...
10	2630	A	2013-05-16	09:10:57	6	32	255	129539	2013-05-16-09:10:57.818 6 32 255 129539 GNSS DOPs: SID = 244; Desired Mode = Auto; Actual Mode = 3D; HDOP = 1.11; VDOP = 1.15
11	2631	A	2013-05-16	09:10:57	3	32	255	129029	2013-05-16-09:10:57.824 3 32 255 129029 GNSS Position Data: SID = 244; Date = 2013.05.16; Time = 09:11:15; Latitude = 58.3921151; Long...
12	2632	A	2013-05-16	09:10:57	6	32	255	129540	2013-05-16-09:10:57.889 6 32 255 129540 GNSS Sats in View: SID = 244; Sats in View = 12; PRN = 2; Elevation = 7.0 deg; Azimuth = -152....
13	2633	A	2013-05-16	09:11:00	6	112	255	127505	2013-05-16-09:11:00.581 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 1.272 %
14	2634	A	2013-05-16	09:11:01	2	17	255	127508	2013-05-16-09:11:01.488 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.35 V
15	2635	A	2013-05-16	09:11:27	2	32	255	129026	2013-05-16-09:11:27.062 2 32 255 129026 COG & SOG, Rapid Update: SID = 34; COG Reference = True; COG = 137.7 deg; SOG = 0.10 m/s
16	2636	A	2013-05-16	09:11:27	6	17	255	127493	2013-05-16-09:11:27.087 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Transmi...
17	2637	A	2013-05-16	09:11:27	2	17	255	127488	2013-05-16-09:11:27.089 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engine S...
18	2638	A	2013-05-16	09:11:27	5	35	255	130310	2013-05-16-09:11:27.114 5 35 255 130310 Environmental Parameters: SID = 6; Water Temperature = 21.33 C (70.4 F)
19	2639	A	2013-05-16	09:11:27	2	32	255	129025	2013-05-16-09:11:27.171 2 32 255 129025 Position, Rapid Update: Latitude = 58.3920611; Longitude = 08.7171575
20	2640	A	2013-05-16	09:11:27	6	17	255	127505	2013-05-16-09:11:27.235 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 21.664 %; Capacity = 480.0 L

Figure 18 Viewing data in Netbeans

4.6.3 Data browsing from external Internet connection

For the users who are away from administration office or the server, but still want to view the data through the Internet, they can access our Web service, which is designed for external users to browse the data at any time and location.

In order to publish the Web service, we install the Netbeans 7.3 IDE and make several trials under Java developing environment. Here are our steps to publish the RESTful Web service:

Firstly, we create a MySQL table class, which defines the different sensors' parameters such as *id*, *boatname*, *date*, *pgn* and *src*.

Then we can invoke these parameters to see the data by the command *NamedQueries*. For example, @NamedQuery(name = "B001.findById", query = "SELECT b FROM B001 b WHERE b.id = :id").

Next, we compile table b_001 into RESTful Web service class. By using 4 operations (GET, POST, PUT, DELETE), we can easily invoke the specific parameter and view its content. Here we choose three main episodes in my codes to explain how the RESTful service works with MySQL database.

```
@Stateless
   @Path("boattest.b001")
   public class B001FacadeREST extends AbstractFacade<B001> {
      @PersistenceContext(unitName = "boattestPU")
      private EntityManager em;
         public B001FacadeREST() {
            super(B001.class);
         }
```

This segment of codes illustrates that class B001FacadeREST is defined to have the same method as table class B001 and makes the class B001FacadeREST to be stateless.

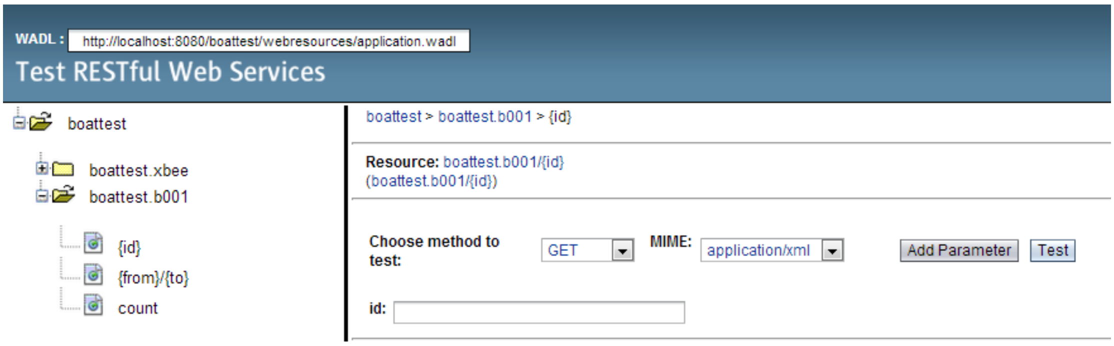
```
@POST
    @Override
    @Consumes({"application/xml", "application/json"})
        public void create(B001 entity) {
            super.create(entity);
        }
```

The segment of codes above shows that a selection function is added into class, which we can decide the type of data content in RESTful Web service. In our case, there are two types of format: *xml* and *json*.

```
@GET
    @Path("count")
    @Produces("text/plain")
        public String countREST() {
            return String.valueOf(super.count());
        }
```

The last segment of codes is invoking the *GET* operation to count the total number of data in our database.

After the RESTful Web service class setup, we need to design our Web service interface. It is a web-page that the clients can access from external Internet and choose which data they want to view. Because of the time limitation, we used the Netbeans default RESTful web-page in our thesis. As shown in Figure 19, it has 3 functions: select a single id to view, select a range of id to view and count the total number of data.



WADL : http://localhost:8080/boattest/webresources/application.wadl

Test RESTful Web Services

boattest > boattest.b001 > {id}

Resource: boattest.b001/{id}
(boattest.b001/{id})

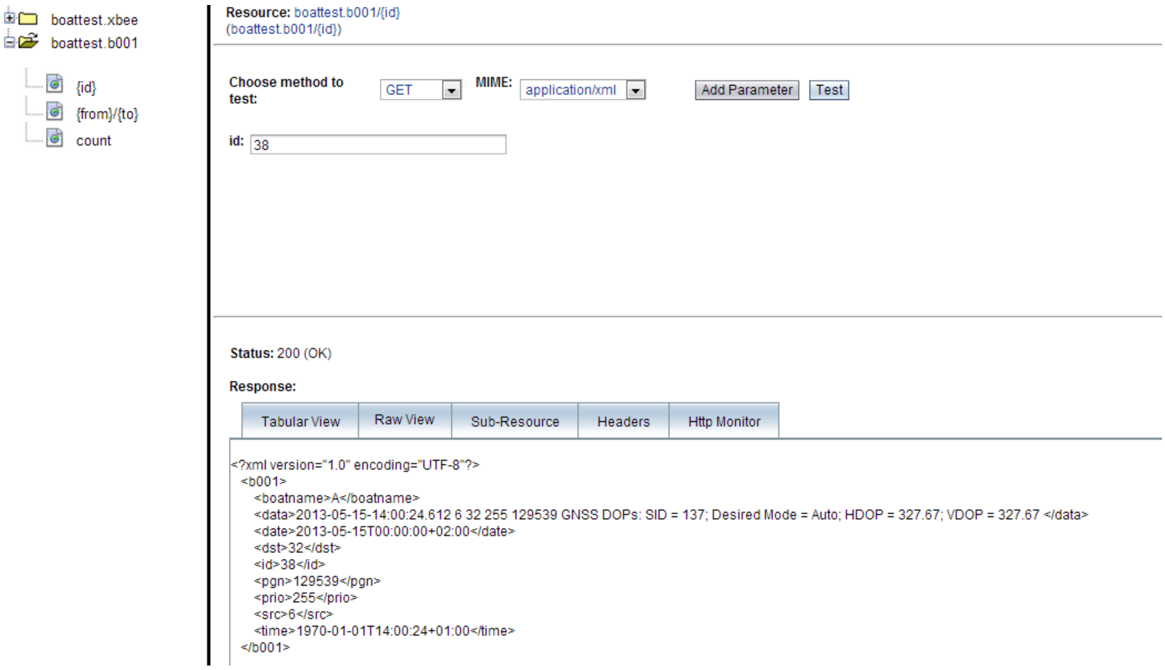
Choose method to test: GET

MIME: application/xml

id:

Figure 19 Functions of Web service

Once we want to check a specific data, for example the data content of id 38. We can use the *{id}* function: to type 38 in id blank, then click Test. It shows entire information of id 38, as shown in Figure 20.



Resource: boattest.b001/{id}
(boattest.b001/{id})

Choose method to test: GET **MIME:** application/xml

id: 38

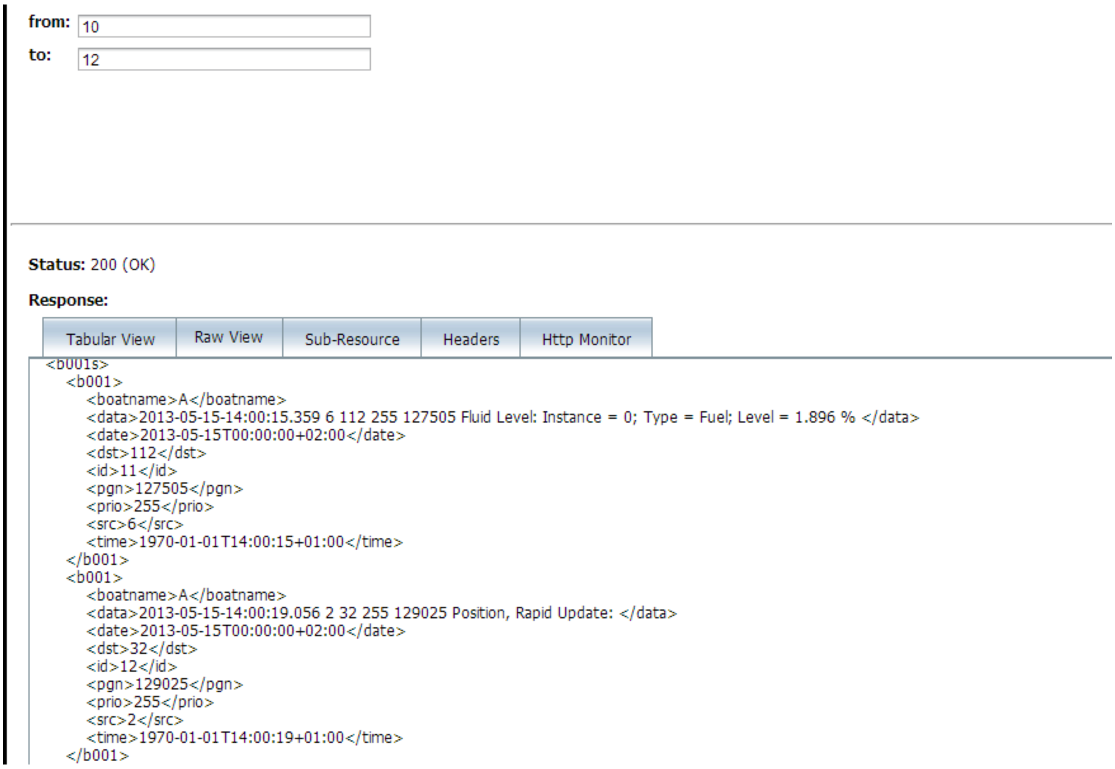
Status: 200 (OK)

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
  <b001>
    <boatname>A</boatname>
    <data>2013-05-15-14:00:24.612 6 32 255 129539 GNSS DOPs: SID = 137; Desired Mode = Auto; HDOP = 327.67; VDOP = 327.67 </data>
    <date>2013-05-15T00:00:00+02:00</date>
    <dst>32</dst>
    <id>38</id>
    <pgn>129539</pgn>
    <prio>255</prio>
    <src>6</src>
    <time>1970-01-01T14:00:24+01:00</time>
  </b001>
```

Figure 20 Inquery for single message

The *{from}/{to}* function in middle of left bar has the similar viewing ability as the previous *{id}* function, but it can display all the ids' content in a range. Figure 21 shows that we choose the id range from 10 to 12 and all the data in this range will be displayed.



from: 10

to: 12

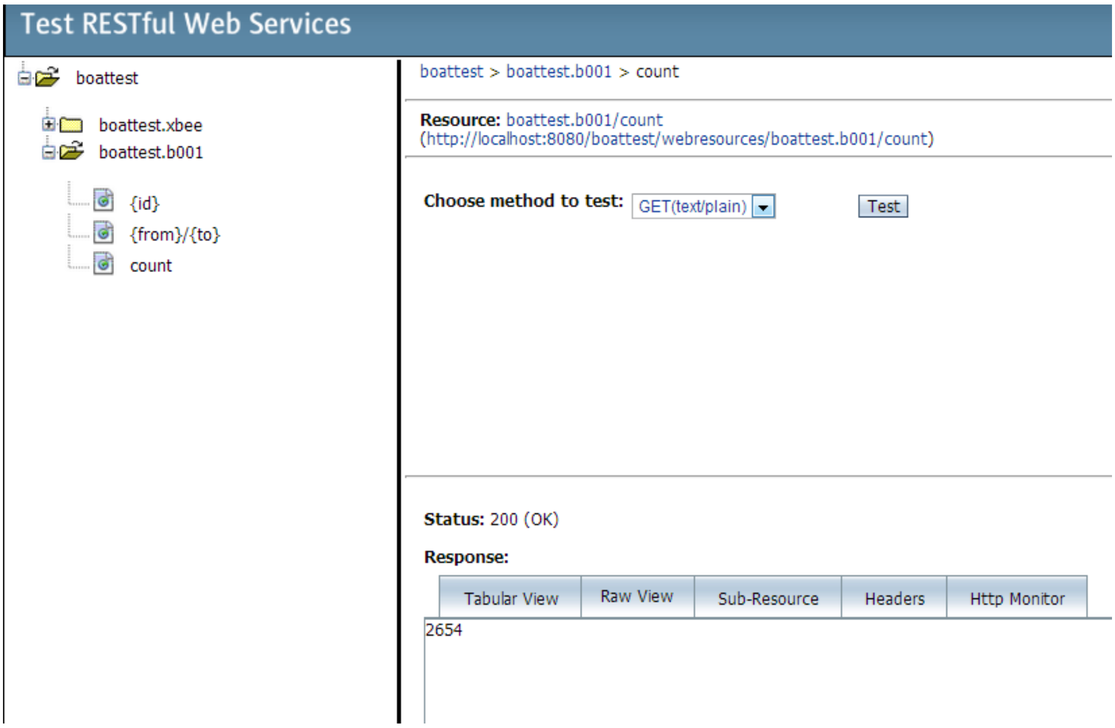
Status: 200 (OK)

Response:

```
<b001s>
  <b001>
    <boatname>A</boatname>
    <data>2013-05-15-14:00:15.359 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 1.896 % </data>
    <date>2013-05-15T00:00:00+02:00</date>
    <dst>112</dst>
    <id>11</id>
    <pgn>127505</pgn>
    <prio>255</prio>
    <src>6</src>
    <time>1970-01-01T14:00:15+01:00</time>
  </b001>
  <b001>
    <boatname>A</boatname>
    <data>2013-05-15-14:00:19.056 2 32 255 129025 Position, Rapid Update: </data>
    <date>2013-05-15T00:00:00+02:00</date>
    <dst>32</dst>
    <id>12</id>
    <pgn>129025</pgn>
    <prio>255</prio>
    <src>2</src>
    <time>1970-01-01T14:00:19+01:00</time>
  </b001>
```

Figure 21 Inquery for multiple messages

The last function is to count the total number of data. We can observe that we have 2654 lines of data in current database as shown in Figure 22.



Resource: boattest.b001/count (http://localhost:8080/boattest/webresources/boattest.b001/count)

Choose method to test: GET(text/plain)

Status: 200 (OK)

Response:

```
2654
```

Figure 22 Message count

5 Implementation of Advanced Features

Comparing with the basic functionalities introduced in the previous chapter, advanced features are not obligatory for the system but the improvement on information security and remote control. As mentioned in Section 3.1, the advanced features of the data collection and transmission system include:

1. Data security: to encrypt the data in MySQL.

2. Transmission security: to encrypt the data in transmission.

3. Local control: to access and control the embedded system by using SSH.

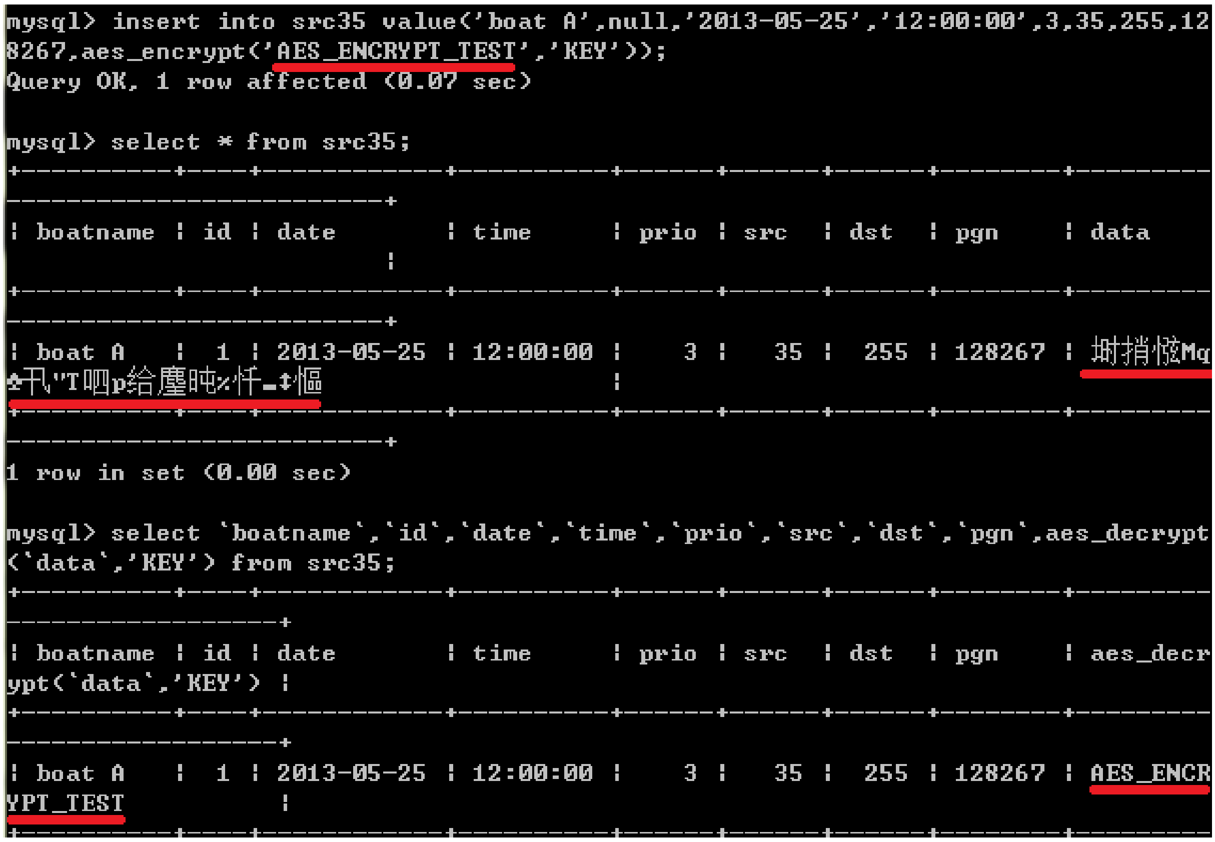
4. Reverse control: to enable the remote server to control the embedded system.

Function 1 is implemented by using the AES encryption and decryption provided by MySQL. Using the SSL encryption on FTP and MySQL, we can implement transmission security. Local control is based on SSH while reverse control is implemented by using port listening.

5.1 Data Security

As the database may be hacked, some privacy information may be exposed, e.g., the current location of the boat. For this reason, data encryption needs to be introduced to the system.

MySQL supports data encryption and decrytion based on AES, which is a standard for the encryption of electronic data. AES is a reliable solotion for data security and has been widely adopted.



```
mysql> insert into src35 value('boat A',null,'2013-05-25','12:00:00',3,35,255,12
8267,aes_encrypt('AES_ENCRYPT_TEST','KEY'));
Query OK, 1 row affected (0.07 sec)

mysql> select * from src35;
+----------+----+------------+----------+------+------+------+--------+---------
-------------------------+
| boatname | id | date       | time     | prio | src  | dst  | pgn    | data
                         |
+----------+----+------------+----------+------+------+------+--------+---------
-------------------------+
| boat A   |  1 | 2013-05-25 | 12:00:00 |    3 |   35 |  255 | 128267 | 埘捎惙Mq
≏卂"T呬p给塵吨⅔仟▬‡慪                        |
+----------+----+------------+----------+------+------+------+--------+---------
-------------------------+
1 row in set (0.00 sec)

mysql> select `boatname`,`id`,`date`,`time`,`prio`,`src`,`dst`,`pgn`,aes_decrypt
(`data`,'KEY') from src35;
+----------+----+------------+----------+------+------+------+--------+---------
---------------+
| boatname | id | date       | time     | prio | src  | dst  | pgn    | aes_decr
ypt(`data`,'KEY') |
+----------+----+------------+----------+------+------+------+--------+---------
---------------+
| boat A   |  1 | 2013-05-25 | 12:00:00 |    3 |   35 |  255 | 128267 | AES_ENCR
YPT_TEST          |
+----------+----+------------+----------+------+------+------+--------+---------
```

Figure 23 AES encryption and decryption in MySQL

As shown in Figure 23, when storing the data into MySQL, we can use *aes_encrypt()* to encrypt the message string (*AES_ENCRYPT_TEST*) with a key string (*KEY*). Then the unreadable encrypted data will be stored in the table. If anyone wants to decrypt the data, he must have the key.

By implementing AES encryption, the data stored in MySQL is secured as only the key holders can decrypt it.

5.2 Transmission Security

During the transmission on the Internet, data may be intercepted by hackers. Thus establishing a secure connection between the embedded system and remote server becomes a requirement.

As we have two different connection modes, the introductions will be given respectively.

5.2.1 Secure connection for MySQL

MySQL supports SSL connections between server and client. The following steps should be performed.

Step 1: Create SSL files, including Certificate Authority (CA) certificate, certificates and key files for MySQL server and client.

Step 2: Move CA certificate, server certificate and server key files to the server side.

Step 3: Move CA certificate, client certificate and client key files to the client side.

Step 4: Enable SSL connection on both server and client.

Step 5: Start server and client with their SSL files respectively.

For more details, please refer to the reference manual on the official website for MySQL [13].

After finishing the above operations, we also need to modify the source code in *analyzer.c*. Before calling *mysql_real_connect()*, use the *mysql_ssl_set()* function to specify the path to the certificate and key files for the client side.

```
MYSQL re_connection;
int res;
mysql_init(&re_connection);
mysql_ssl_set(&re_connection, <path name to the key file of client>, <path name to the
  certificate file of client>, <path name to the CA file>, NULL, NULL);
if (mysql_real_connect(&re_connection, <IP of remote database>, <username>, <password>,
  <name of database>, 0, NULL, CLIENT_FOUND_ROWS))
…
```

5.2.2 Secure connection for FTP

Serv-U and LFTP support SSL connections as well. Similar to the MySQL SSL connection, FTP SSL connection requires the following operation.

Step 1: Create SSL files, including CA certificate, certificates and key files for FTP server and client.

Step 2: Move CA certificate, server certificate and server key files to the server side.

Step 3: Move CA certificate, client certificate and client key files to the client side.

Step 4: Configure Serv-U to enable SSL connection. Refer to the website of Serv-U for details [14].

Step 5: Configure LFTP to enable SSL connection by typing the following commands in terminal.

```
lftp
set ftp:ssl-force true
set ftp:ssl-protect-data true
set ssl:verify-certificate no
set ssl:ca-file < path name to the CA file>
set ssl:key-file <path name to the client key file>
set ssl:cert-file <path name to the client certificate file>
```

By now, all the transmission between the embedded system and remote server has been secured by SSL encryption.

As illustrated in Figure 24, with the encryption on both data and transmission, the data is always under the protection. Only the key holders, which could be either human or applications) can decrypt the data.

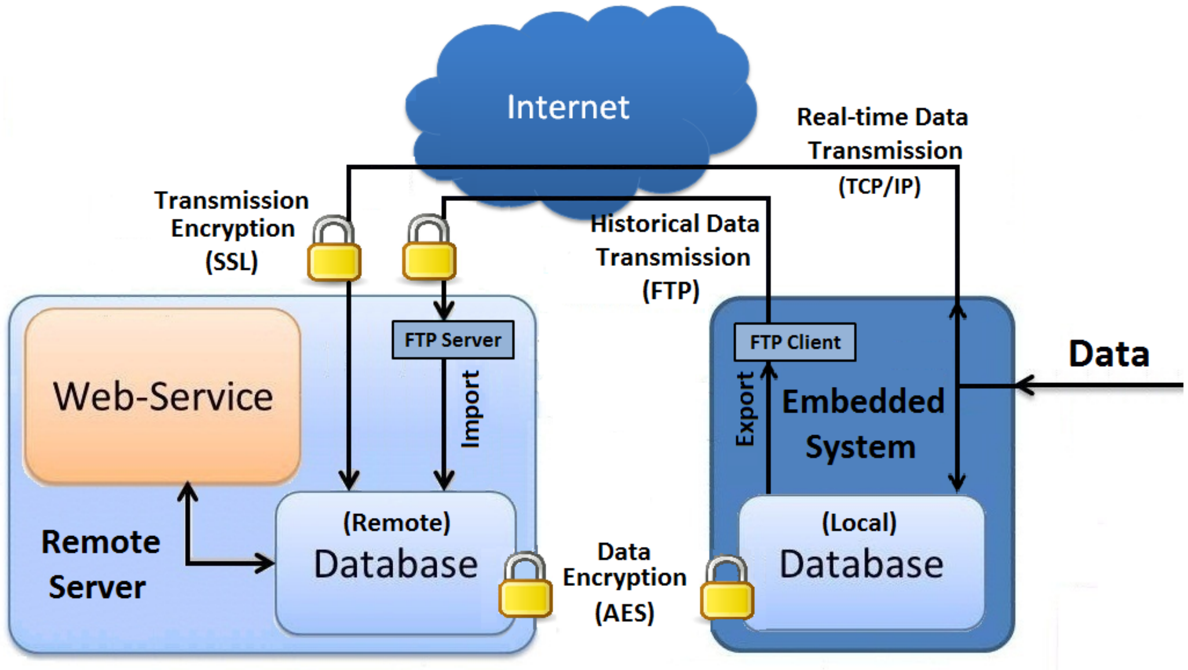


Figure 24 Encrypted data and transmission

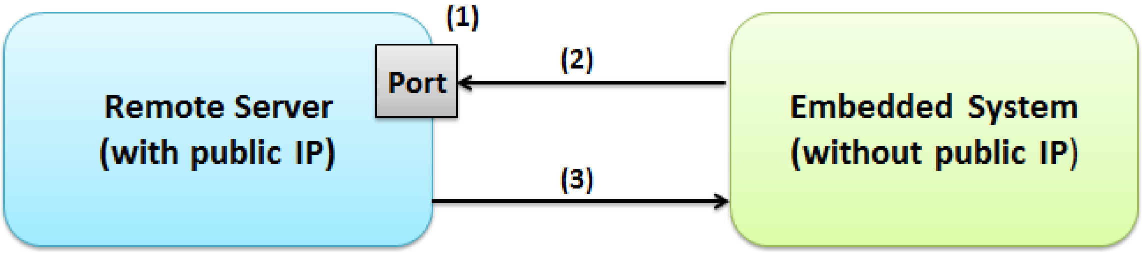
5.3 Local Access and Control by Using SSH

By establishing SSH connection, an authorized user can logging into the embedded system remotely by using a laptop or even a smart phone. If the device you use is in the same local area network (LAN) as the embedded system, you can take the control over the system with knowing its IP address. An example on using an Android phone to control the system will be given in Section 6.4.

5.4 Reverse Control

Reverse Control is a function that can remotely control a non-public IP machine from a local machine with public IP address.

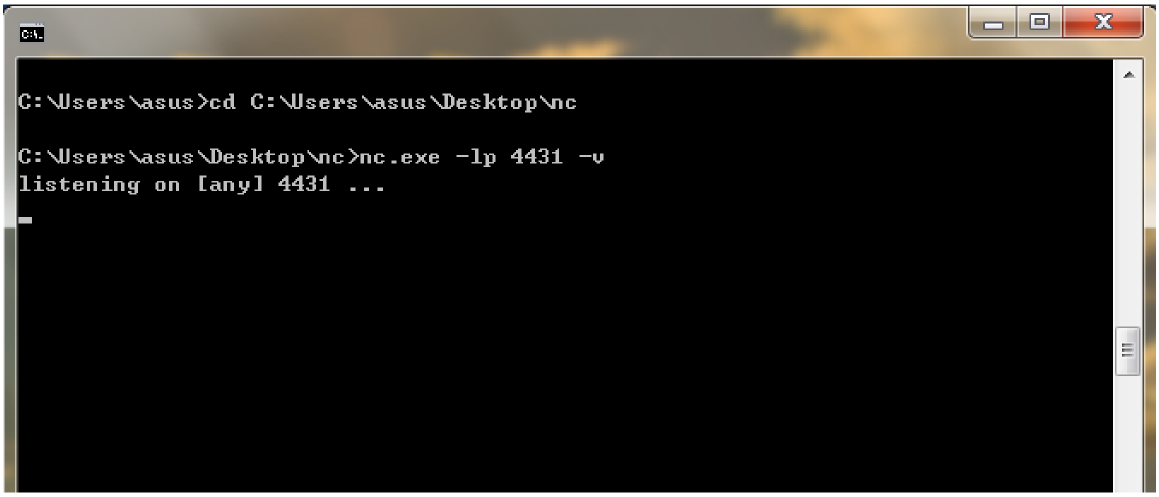
As mentioned in Chapter 3, the reverse control enables the administrative users who are close to the server to remotely control the embedded system even it does not have the public IP address. Figure 25 illustrates the basic procedures of reverse control: firstly, the remote server opens a specific port as scanner and listener. Once the embedded system actively launches a connection to the server, the in-coming connection will be detected by the server. A notice of successful connection will pop out in the command line in which we can send commands to control the embedded system.



(1) Create a listening port on the Remote Server
(2) Send connection request to the remote server through the listening port
(3) Reversely control the embedded system

Figure 25 Flow chart of reverse control

In this scenario, we use Netcat and PHP script to fulfil the reverse control between our server and system. Netcat is used to be a tool that can scan and listen to a specific port on the server side, which has been introduced in Section 2.5.



```
C:\Users\asus>cd C:\Users\asus\Desktop\nc

C:\Users\asus\Desktop\nc>nc.exe -lp 4431 -v
listening on [any] 4431 ...
_
```

Figure 26 Netcat listening port

Figure 26 reflects the working status of Netcat when it was scanning and listening to the port 4431 on the remote server.

In PHP scripts, we firstly defined the server's public IP address, port number, chunk size and daemon process as following code episode:

```
set_time_limit (0);
$VERSION = "1.0";
$ip = '128.39.202.145';
$port = 4431;
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

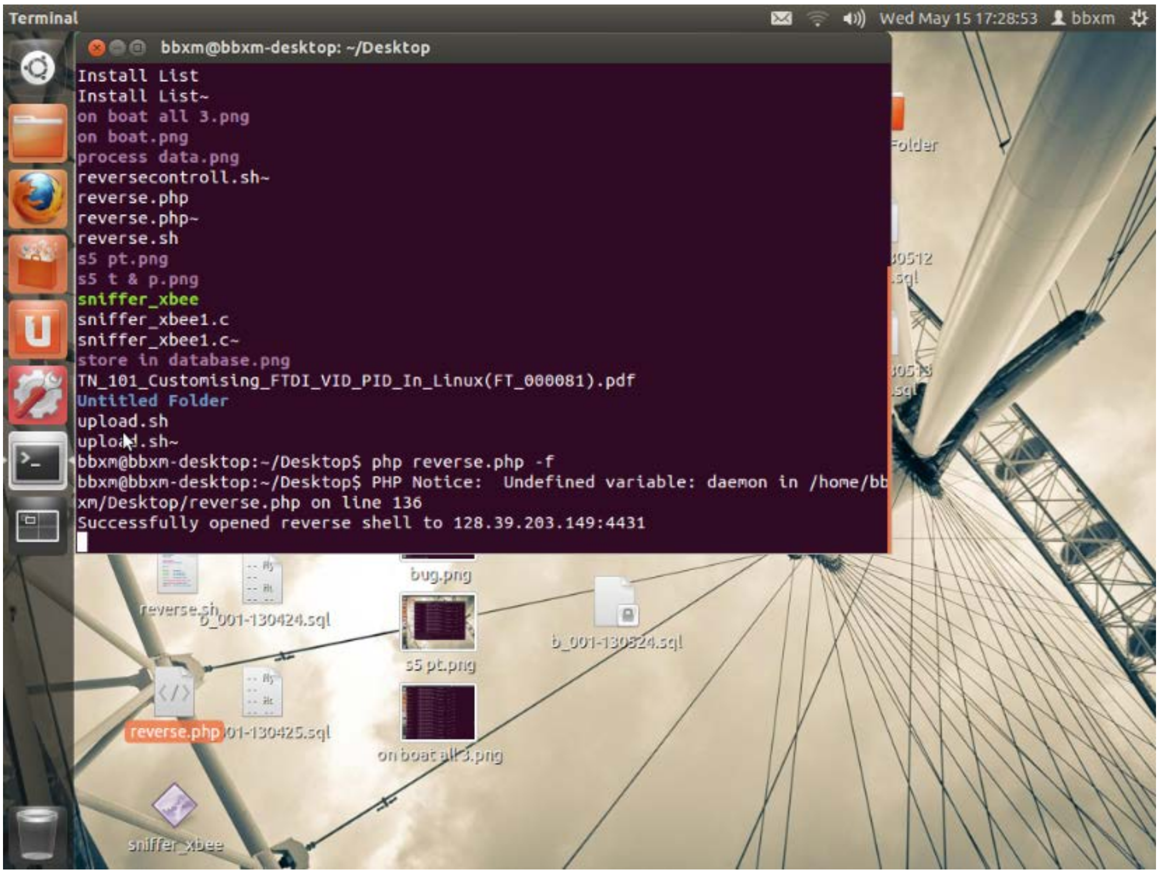
Then we open the reverse control and spawn the shell process, which is achieved by episodes:

```
$sock = fsockopen($ip, $port, $errno, $errstr, 30);
if (!$sock) {
        printit("$errstr ($errno)");
        exit(1);
}
// Spawn shell process
$descriptorspec = array(
    0 => array("pipe", "r"),   // stdin is a pipe that the child will read from
    1 => array("pipe", "w"),   // stdout is a pipe that the child will write to
    2 => array("pipe", "w")    // stderr is a pipe that the child will write to
);
$process = proc_open($shell, $descriptorspec, $pipes);
if (!is_resource($process)) {
        printit("ERROR: Can't spawn shell");
        exit(1);
}
```

If nothing blocks the above process, we will have a prompt on the client side to inform users whether this connection is success or not. It is achieved by following codes:

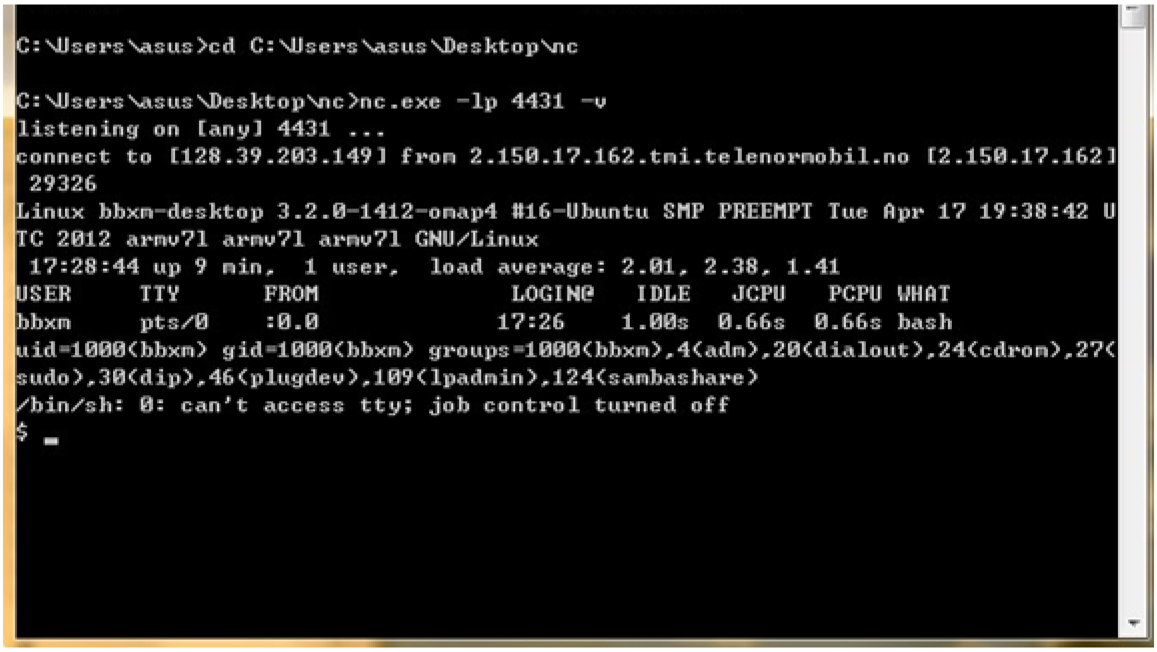
```
stream_set_blocking($pipes[0], 0);
stream_set_blocking($pipes[1], 0);
stream_set_blocking($pipes[2], 0);
stream_set_blocking($sock, 0);
printit("Successfully opened reverse control to $ip:$port");
while (1) {
        // Check for end of TCP connection
        if (feof($sock)) {
                printit("ERROR: Connection terminated");
                break;
        }
        // Check for end of STDOUT
        if (feof($pipes[1])) {
                printit("ERROR: Process terminated");
                break;
        }
```

Once the PHP script is running on the embedded system, the prompt of successfully connection will automatically pop out, which is shown in Figure 27. Then Figure 28 shows that we can enter any Linux commands on the server side as we do in the Ubuntu terminal.



```
Install List
Install List~
on boat all 3.png
on boat.png
process data.png
reversecontroll.sh~
reverse.php
reverse.php~
reverse.sh
s5 pt.png
s5 t & p.png
sniffer_xbee
sniffer_xbee1.c
sniffer_xbee1.c~
store in database.png
TN_101_Customising_FTDI_VID_PID_In_Linux(FT_000081).pdf
Untitled Folder
upload.sh
upload.sh~
bbxm@bbxm-desktop:~/Desktop$ php reverse.php -f
bbxm@bbxm-desktop:~/Desktop$ PHP Notice:  Undefined variable: daemon in /home/bb
xm/Desktop/reverse.php on line 136
Successfully opened reverse shell to 128.39.203.149:4431
```

Figure 27 Reverse control on the embedded system



```
C:\Users\asus>cd C:\Users\asus\Desktop\nc

C:\Users\asus\Desktop\nc>nc.exe -lp 4431 -v
listening on [any] 4431 ...
connect to [128.39.203.149] from 2.150.17.162.tmi.telenormobil.no [2.150.17.162]
 29326
Linux bbxm-desktop 3.2.0-1412-omap4 #16-Ubuntu SMP PREEMPT Tue Apr 17 19:38:42 U
TC 2012 armv7l armv7l armv7l GNU/Linux
 17:28:44 up 9 min,  1 user,  load average: 2.01, 2.38, 1.41
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
bbxm     pts/0    :0.0             17:26    1.00s  0.66s  0.66s bash
uid=1000(bbxm) gid=1000(bbxm) groups=1000(bbxm),4(adm),20(dialout),24(cdrom),27(
sudo),30(dip),46(plugdev),109(lpadmin),124(sambashare)
/bin/sh: 0: can't access tty; job control turned off
$ _
```

Figure 28 Control terminal on the remote server

We have tested this function in three ways of Internet connection of embedded system: cable connection, Wi-Fi connection and 3G/4G Router respectively, more details will be given in Chapter 6. It works well in all scenarios, which the connection is stable and the control performance is effective.

6 Test and Validation

In order to evaluate the availability and performance of the system, we have conducted a number of tests for different scenarios, some of which will be presented in this chapter.

In the first three cases described in the following sections, the tests are concentrated on the critical functionalities and feature. The fourth one is an on-site test on the boat provided by Marex AS, the overall performance of the system is validated in practical use.

6.1 Test Scenarios

For testing the system, we set up a testbed consists of one Actisense NGT-1 USB Gateways, one GPS receiver, one DST110 Depth/Speed/Temperature Triducer and one TLM100 Tank Level Monitor. These devices are connected to the CAN Bus as shown in Figure 29. The testbed is also used in the examples of data filtering mentioned in Section 4.2.

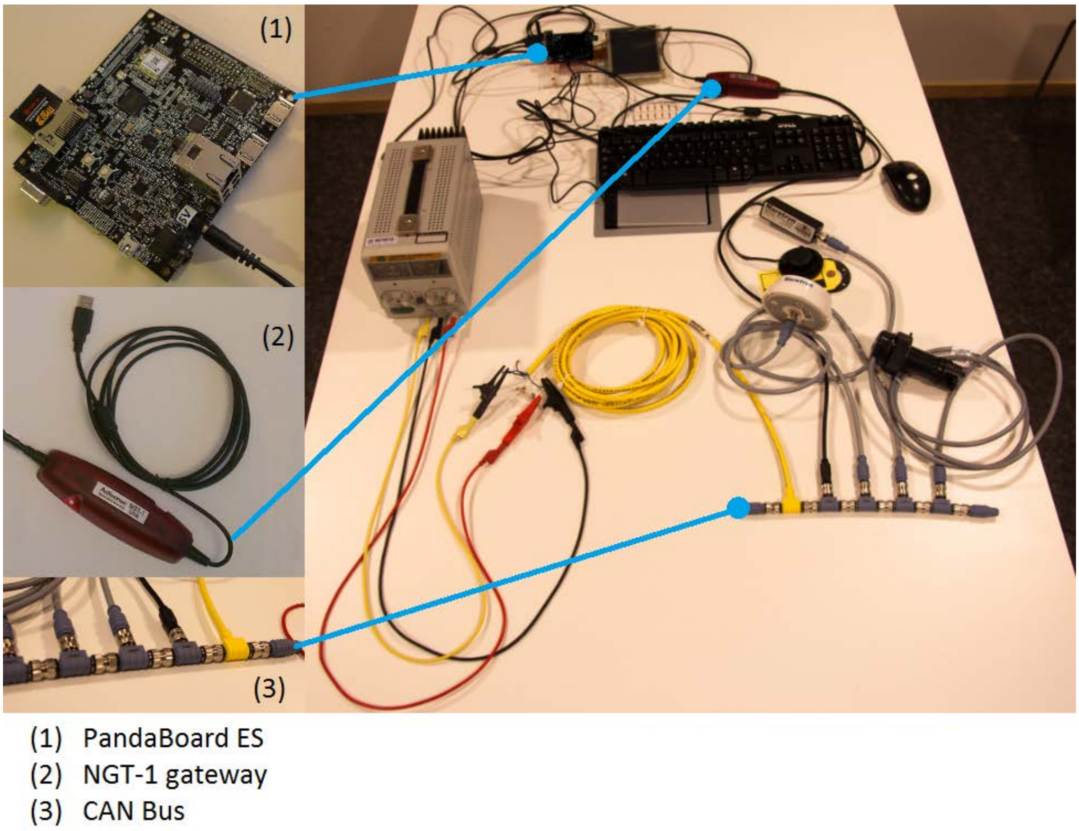


Figure 29 Testbed

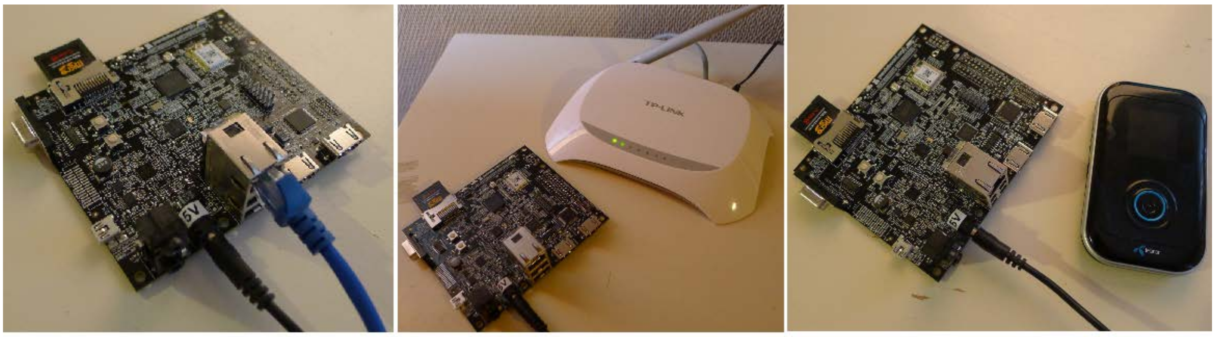
The testbed is used in the following experiments.

- Experiment on access modes
- Experiment on real-time transmission
- Experiment on Local access and control using Android phone
- Experiment on Boat

In the following sections, the test scenarios will be introduced in detail.

6.2 Experiment on Access Modes

An available connection to the Internet is a prerequisite for data transmission. Therefore, the Internet connectivity under different access modes is our first concern.



(1) (2) (3)

(1) Wired connection to the Internet

(2) Wireless connection to the Internet through an AP

(3) Wireless connection to cellular network through an 3G/4G mobile router

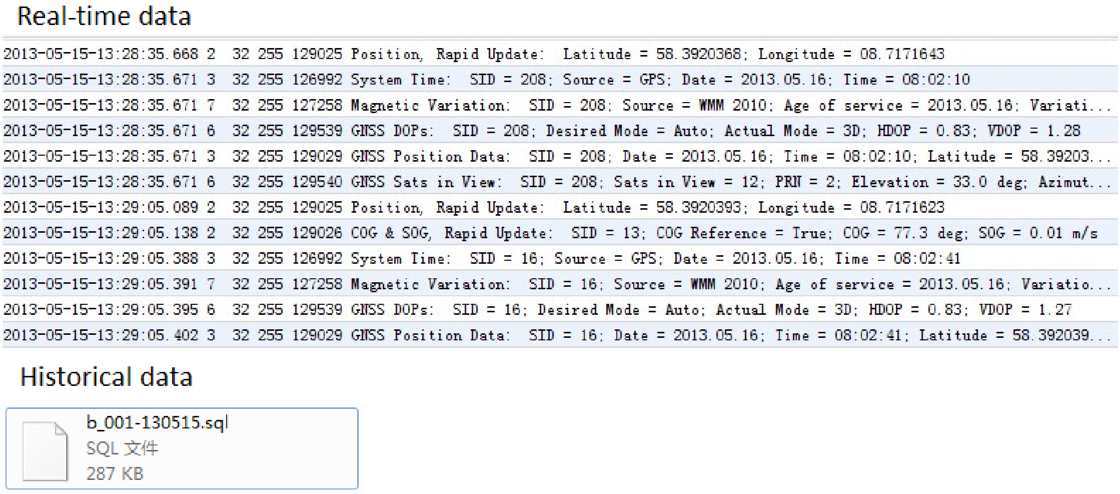
Figure 30 Access modes

Under the access modes shown in Figure 30, all the Internet-dependent activities between the remote server (which has a public IP) and embedded system have been conducted. And the tests have been repeated for three times, the results are listed in Table 7.

Table 7 Internet-dependent functionalities under different access modes

Functionality	Access Mode (1)	Access Mode (2)	Access Mode (3)
Real-time data transmission	succeed	succeed	succeed
Historical data transmission	succeed	succeed	succeed
Reverse control	succeed	succeed	succeed

The real-time data is received in the remote server as well as the historical data in the file *b_001-130515.sql* as shown in Figure 31.



Real-time data

```
2013-05-15-13:28:35.668 2  32 255 129025 Position, Rapid Update:  Latitude = 58.3920368; Longitude = 08.7171643
2013-05-15-13:28:35.671 3  32 255 126992 System Time:  SID = 208; Source = GPS; Date = 2013.05.16; Time = 08:02:10
2013-05-15-13:28:35.671 7  32 255 127258 Magnetic Variation:  SID = 208; Source = WMM 2010; Age of service = 2013.05.16; Variati...
2013-05-15-13:28:35.671 6  32 255 129539 GNSS DOPs:  SID = 208; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.83; VDOP = 1.28
2013-05-15-13:28:35.671 3  32 255 129029 GNSS Position Data:  SID = 208; Date = 2013.05.16; Time = 08:02:10; Latitude = 58.39203...
2013-05-15-13:28:35.671 6  32 255 129540 GNSS Sats in View:  SID = 208; Sats in View = 12; PRN = 2; Elevation = 33.0 deg; Azimut...
2013-05-15-13:29:05.089 2  32 255 129025 Position, Rapid Update:  Latitude = 58.3920393; Longitude = 08.7171623
2013-05-15-13:29:05.138 2  32 255 129026 COG & SOG, Rapid Update:  SID = 13; COG Reference = True; COG = 77.3 deg; SOG = 0.01 m/s
2013-05-15-13:29:05.388 3  32 255 126992 System Time:  SID = 16; Source = GPS; Date = 2013.05.16; Time = 08:02:41
2013-05-15-13:29:05.391 7  32 255 127258 Magnetic Variation:  SID = 16; Source = WMM 2010; Age of service = 2013.05.16; Variatio...
2013-05-15-13:29:05.395 6  32 255 129539 GNSS DOPs:  SID = 16; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.83; VDOP = 1.27
2013-05-15-13:29:05.402 3  32 255 129029 GNSS Position Data:  SID = 16; Date = 2013.05.16; Time = 08:02:41; Latitude = 58.392039...
```

Historical data

b_001-130515.sql
SQL 文件
287 KB

Figure 31 Data received in the remote server

The results indicate that available connections to the Internet are established and the Internet-dependent features function correctly under all the three access modes.

6.3 Experiment on Real-Time Transmission

Under the Access Mode (3) described in the last section, we conduct tests on the real-time transmission as follows.

Step 1: Start the remote server with connection to the Internet.

Step 2: Make sure the IP address of the server is globally routable.

Step 3: Start the embedded system with connection to the cellular network by using a mobile router.

Step 4: Connect the embedded system to the NMEA 2000 network and then launch the data collection system.

Step 5: Observe the messages added to the remote database.

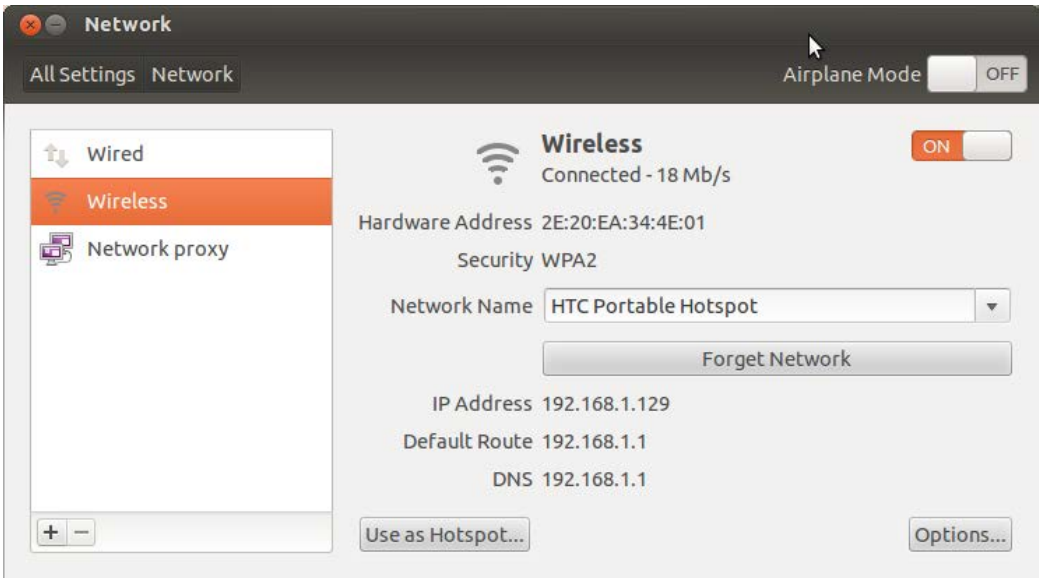
After repeating the above steps for 20 times, it is observed that the filtered data on the embedded system can be transmitted to the remote server through cellular network.

6.4 Experiment on Local Access and Control using Android Phone

By using a smart phone with an SSH client installed, boat owners can easily log in the embedded system on their boats to perform any operations by following the steps.

Step 1: Enable the Wi-Fi access point (hotspot) on the phone after setting the SSID and password.

Step 2: Connect the embedded system to the access point through Wi-Fi and note down the IP address.



Airplane Mode OFF

Wireless ON

Connected - 18 Mb/s

Hardware Address 2E:20:EA:34:4E:01

Security WPA2

Network Name HTC Portable Hotspot

IP Address 192.168.1.129

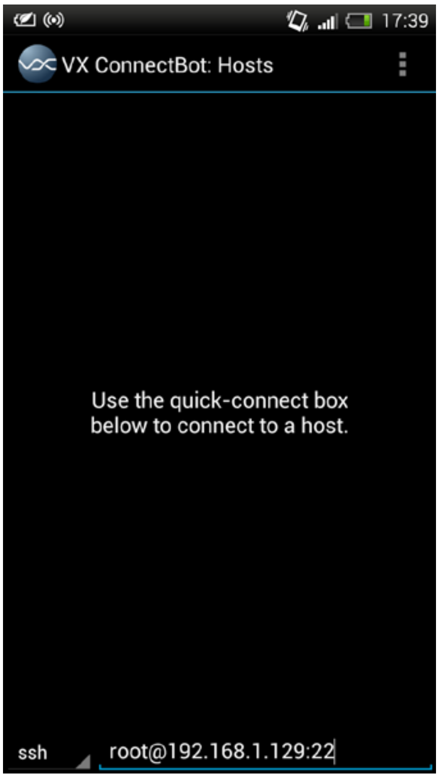
Default Route 192.168.1.1

DNS 192.168.1.1

Figure 32 Connect the embedded system to mobile phone through Wi-Fi

Step 3: Start a SSH client application on the phone (In this example, we use VX ConnectBot for Android which is free and can be found in the application market) and log in by typing:

<username>@<IP of the embedded system>:<port number of SSH>

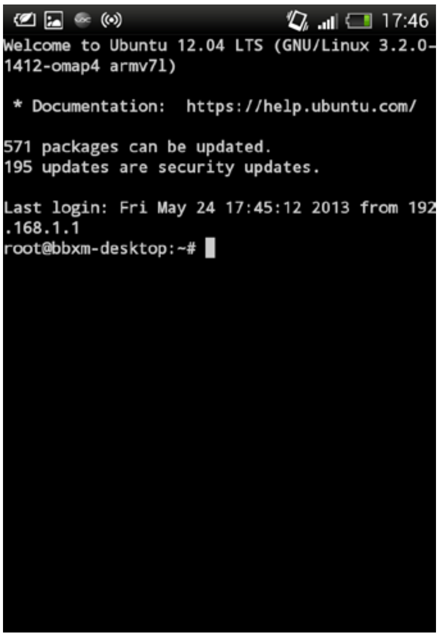


Use the quick-connect box below to connect to a host.

ssh root@192.168.1.129:22

Figure 33 Login screen

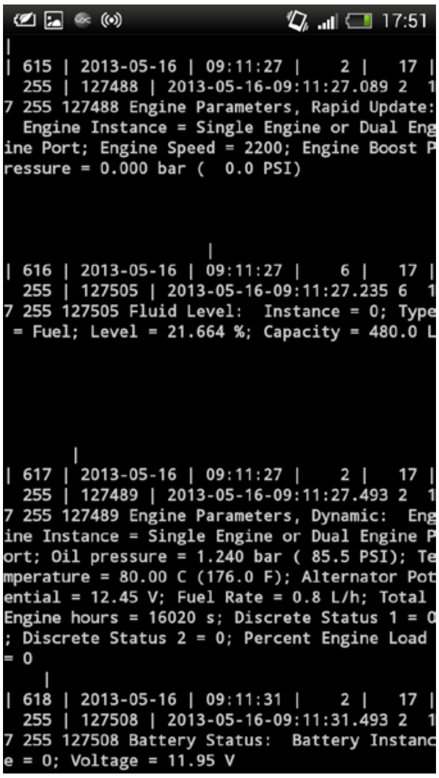
Step 4: After inputting the password, the login information will be displayed on the phone.



```
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-
1412-omap4 armv7l)

 * Documentation:  https://help.ubuntu.com/

571 packages can be updated.
195 updates are security updates.

Last login: Fri May 24 17:45:12 2013 from 192
.168.1.1
root@bbxm-desktop:~# 
```

Figure 34 Log in the embedded system

Now we have obtained the control on the embedded system and can access the data in the local database or perform FTP uploading. Through SSH, all operations can be performed by inputting corresponding commands on the phone in the same way as they are done on the PandaBoard ES.



```
|
| 615 | 2013-05-16 | 09:11:27 |    2 |   17 |
  255 | 127488 | 2013-05-16-09:11:27.089 2  1
7 255 127488 Engine Parameters, Rapid Update:
  Engine Instance = Single Engine or Dual Eng
ine Port; Engine Speed = 2200; Engine Boost P
ressure = 0.000 bar (  0.0 PSI)



                    |
| 616 | 2013-05-16 | 09:11:27 |    6 |   17 |
  255 | 127505 | 2013-05-16-09:11:27.235 6  1
7 255 127505 Fluid Level:  Instance = 0; Type
 = Fuel; Level = 21.664 %; Capacity = 480.0 L




      |
| 617 | 2013-05-16 | 09:11:27 |    2 |   17 |
  255 | 127489 | 2013-05-16-09:11:27.493 2  1
7 255 127489 Engine Parameters, Dynamic:  Eng
ine Instance = Single Engine or Dual Engine P
ort; Oil pressure = 1.240 bar ( 85.5 PSI); Te
mperature = 80.00 C (176.0 F); Alternator Pot
ential = 12.45 V; Fuel Rate = 0.8 L/h; Total
Engine hours = 16020 s; Discrete Status 1 = 0
; Discrete Status 2 = 0; Percent Engine Load
= 0
    |
| 618 | 2013-05-16 | 09:11:31 |    2 |   17 |
  255 | 127508 | 2013-05-16-09:11:31.493 2  1
7 255 127508 Battery Status:  Battery Instanc
e = 0; Voltage = 11.95 V
```

Figure 35   Local access

6.5 Experiment on Boat

For estimating the system performance in practical use, we implemented our data collection system on a leisure boat and take an on-site test at sea.

The remote server is placed in the campus of University of Agder in Grimstad while the boat sails from Fevik to Arendal and go back to Fevik.

The embedded system is connected to a NMEA 2000 network through the NGT-1 gateway and Telenor's cellular network via the 3G/4G router. And in the NMEA 2000 network, there are other two devices: a GPS receiver and an engine.

During the voyage, we can see the collected data on the screen as illustrated in Figure 36. The data are generated by either the GPS receiver or the engine. Meanwhile, we contact our partner who is monitoring the remote server in the campus by mobile phone for verifying whether the data transmission is successfully performed.

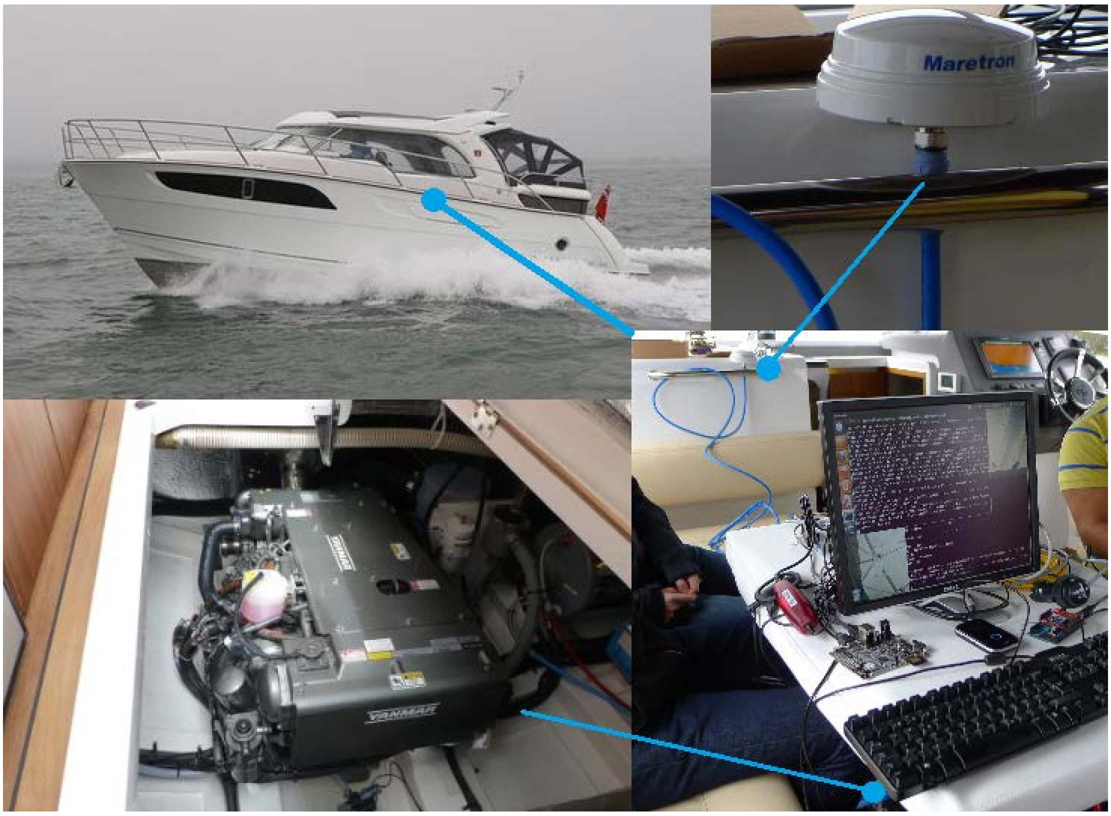
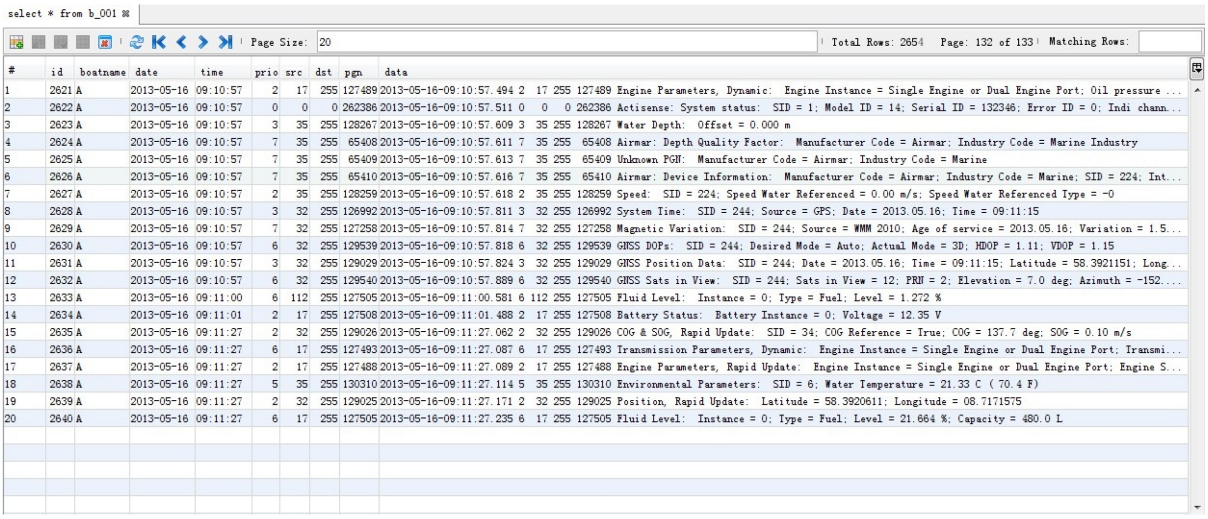


Figure 36 Real-life experiment

The trip took about 60 minutes, during which there is only one failed attempt to transmit the real-time data to the remote server. It occurs when the mobile router disconnects from Telenor’s network. Then right after the recovering of connection, the real-time data transmission resumes.

The data transmitted by the embedded system was successfully received and stored by the remote server, part of which is shown in Figure 37.



select * from b_001

Page Size: 20 Total Rows: 2654 Page: 132 of 133 Matching Rows:

#	id	boatname	date	time	prio	src	dst	pgn	data
1	2621	A	2013-05-16	09:10:57	2	17	255	127489	2013-05-16-09:10:57.494 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pressure ...
2	2622	A	2013-05-16	09:10:57	0	0	0	262386	2013-05-16-09:10:57.511 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi chann...
3	2623	A	2013-05-16	09:10:57	3	35	255	128267	2013-05-16-09:10:57.609 3 35 255 128267 Water Depth: Offset = 0.000 m
4	2624	A	2013-05-16	09:10:57	7	35	255	65408	2013-05-16-09:10:57.611 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
5	2625	A	2013-05-16	09:10:57	7	35	255	65409	2013-05-16-09:10:57.613 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
6	2626	A	2013-05-16	09:10:57	7	35	255	65410	2013-05-16-09:10:57.616 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 224; Int...
7	2627	A	2013-05-16	09:10:57	2	35	255	128259	2013-05-16-09:10:57.618 2 35 255 128259 Speed: SID = 224; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
8	2628	A	2013-05-16	09:10:57	3	32	255	126992	2013-05-16-09:10:57.811 3 32 255 126992 System Time: SID = 244; Source = GPS; Date = 2013.05.16; Time = 09:11:15
9	2629	A	2013-05-16	09:10:57	7	32	255	127258	2013-05-16-09:10:57.814 7 32 255 127258 Magnetic Variation: SID = 244; Source = WMM 2010; Age of service = 2013.05.16; Variation = 1.5...
10	2630	A	2013-05-16	09:10:57	6	32	255	129539	2013-05-16-09:10:57.818 6 32 255 129539 GNSS DOPs: SID = 244; Desired Mode = Auto; Actual Mode = 3D; HDOP = 1.11; VDOP = 1.15
11	2631	A	2013-05-16	09:10:57	3	32	255	129029	2013-05-16-09:10:57.824 3 32 255 129029 GNSS Position Data: SID = 244; Date = 2013.05.16; Time = 09:11:15; Latitude = 58.3921151; Long...
12	2632	A	2013-05-16	09:10:57	6	32	255	129540	2013-05-16-09:10:57.889 6 32 255 129540 GNSS Sats in View: SID = 244; Sats in View = 12; PRN = 2; Elevation = 7.0 deg; Azimuth = -152....
13	2633	A	2013-05-16	09:11:00	6	112	255	127505	2013-05-16-09:11:00.581 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 1.272 %
14	2634	A	2013-05-16	09:11:01	2	17	255	127508	2013-05-16-09:11:01.488 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.35 V
15	2635	A	2013-05-16	09:11:27	2	32	255	129026	2013-05-16-09:11:27.062 2 32 255 129026 COG & SOG, Rapid Update: SID = 34; COG Reference = True; COG = 137.7 deg; SOG = 0.10 m/s
16	2636	A	2013-05-16	09:11:27	6	17	255	127493	2013-05-16-09:11:27.087 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Transmi...
17	2637	A	2013-05-16	09:11:27	2	17	255	127488	2013-05-16-09:11:27.089 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engine S...
18	2638	A	2013-05-16	09:11:27	5	35	255	130310	2013-05-16-09:11:27.114 5 35 255 130310 Environmental Parameters: SID = 6; Water Temperature = 21.33 C (70.4 F)
19	2639	A	2013-05-16	09:11:27	2	32	255	129025	2013-05-16-09:11:27.171 2 32 255 129025 Position, Rapid Update: Latitude = 58.3920611; Longitude = 08.7171575
20	2640	A	2013-05-16	09:11:27	6	17	255	127505	2013-05-16-09:11:27.235 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 21.664 %; Capacity = 480.0 L

Figure 37 Real-time data in real-life experiment

From the figure above, we can see the state of the engine (ID is 17), GPS coordinates as well as other information. For more information collected in this experiment, please refer to Appendix E.

The system performance shown in this experiment is inspiring as it gives evidence that this marine data collection system has the potential to be put into commercial use.

To summarize, the results of the tests described in this chapter reflect that the system has good availability and stability, which strengthen our confidence to make it a commercial product.

7 Discussions

There are several topics in which we are interested. During the system development, we come across difficulties which even lead to changes in design. After repeated attempt and modification, the difficulties are solved and the design goal is fulfilled. In addition, the experience we have gained may be helpful to improve our work in the future.

7.1 Operating System

In this thesis, we have tested different versions of Ubuntu as shown in Table 8. Statistically, Linux systems operate better on PandaBoard ES than on BeagleBoard-xM. When we install Quantal Quetzal or Oneiric Ocelot on BeagleBoard-xM, there are occasional freezes after booting the system. Only Precise Pangolin operates stably. When installing Precise Pangolin server edition on PandaBoard ES, we cannot log into the system. The only available version is Ubuntu 12.04 Precise Pangolin desktop edition.

Table 8 Compatibility of operating system on the embedded system

Platform	Distribution ID	Version	Code Name	Kernel Version	Compatibility
PandaBoard ES	Ubuntu for OMAP4 Desktop	12.04	Precise Pangolin	3.2.0	Yes
PandaBoard ES	Ubuntu for OMAP4 Server	12.04	Precise Pangolin	3.2.0	No
BeagleBoard-xM	Ubuntu for OMAP3 Desktop	11.10	Oneiric Ocelot	3.0.42	No
BeagleBoard-xM	Ubuntu for OMAP3 Desktop	12.04	Precise Pangolin	3.2.0	Yes
BeagleBoard-xM	Ubuntu for OMAP3 Desktop	12.10	Quantal Quetzal	3.5.0	No

7.2 Wi-Fi Stability on BeagleBoard-xM

Since there is no integrated Wi-Fi chip on BeagleBoard-xM, we must use a USB Wi-Fi adapter instead. However, we find that USB Wi-Fi adapters are not well supported in Ubuntu system. The BeagleBoard-xM will be disconnected from access point irregularly and it is unpredictable. We use Belkin and Edimax's Wi-Fi adapter with Realtek RTL8188CU. By checking on the Internet, the same problem also happens on Broadcom and D-Link's products.

Therefore, for connection stability, we recommend to use PandaBoard ES as the hardware platform.

7.3 Device ID Assignment

After many tests, we find that in a CAN Bus network the same Device ID is always assigned to a specified device even if the network topology is changed. Take the GPS Receiver for example, its ID in the CAN Bus network is always 32 regardless of system reboot or the port in which it is installed. Such quasistatic distribution facilitates to set up a mapping between devices and their Device IDs.

7.4 Portability of Software

Our software is programmed in C, therefore, it has excellent portability, i.e., the software can be used in any Linux device as long as there are USB interfaces on that device.

7.5 IP Address Allocation

As the IP address of the remote server is required for establishing connection, it should be consistent and routable. I.e., a static and public address is required for the remote server so that the embedded system can transmit data to the dedicated server.

8 Conclusions and Future Work

In this project, we have developed a marine electronics data collection system for NMEA 2000-compliant electronics on boats. Based on other’s work, we make our own contributions and create a multifunctional data collection system. Furthermore, we integrate various technologies on it for adding advanced features. And the prototype of the system has been implemented and tested in both library and real-life experiments. The test results indicate that the system is functional and stable in practical use. It is very possible that the system will be used commercially and brings actual value to both manufacturers and owners.

8.1 Contributions

Our main contributions in this thesis include:

- Design and develop the system architecture
- Implement data filtering
- Data storage and management in MySQL database
- Automatic data transmission via the Internet
- Data publishing through web service
- Data and transmission security
- Local control and data browse by using SSH
- Reverse control over the embedded system

All the features above make the system an integral solution for marine data collection and transmission, which has been tested in real-life experiments.

8.2 Future Work

For the future work, we need to further simplify the operation by providing a graphical user interface (GUI) for the system, so that users can perform tasks by simply clicking a button instead of inputting a command.

Furthermore, we can invoke Google Map API in the remote server side, so that boats can be located with knowing the real-time GPS coordinates. Or with a set of historical GPS data, we can even illustrate the sailing route on the map and estimate the current location.

In addition, we may adopt satellite communication as a new Internet access mode in the future so that the transmission range will be greatly extended.

References

[1] NMEA 2000 and CANbus, http://www.whichmarineelectronics.com/?p=5, [access on Feb. 14, 2013]

[2] Hella KGaA Hueck and Co., Teknisk informasjon, http://www.hellanor.no/filestore/PDF_filer/Teknisk_informasjon/Hella/Elektronikk/can_bus.pdf, Nov. 27, 2003 [access on May 19, 2013]

[3] *International Telecommunication Union*, ITU-T Recommendation G.984.1 Gigabit-capable Passive Optical Networks (GPON): General characteristics, March, 2003

[4] UN: Six billion mobile phone subscriptions in the world, http://www.bbc.co.uk/news/technology-19925506, [access on May 20, 2013]

[5] RESTful Web service, http://www.oracle.com/technetwork/articles/javase/index-137171.html [access on April 21, 2013]

[6] Iyappan, P.; Arvind, K. S.; Geetha, N.; Vanitha, S., "Pluggable Encryption Algorithm In Secure Shell(SSH) Protocol,"*Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on*, vol., no., pp.808,813, 16-18 Dec. 2009

[7] Suresh, V. M.; Karthikeswaran, D.; Sudha, V. M.; Chandraseker, D.M., "Web server load balancing using SSL back-end forwarding method," *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on* , vol., no., pp.822,827, 30-31 March 2012

[8] AES Encryption, http://townsendsecurity.com/sites/default/files/AES_Introduction.pdf

[9] Ubuntu 12.04.1 LTS (Precise Pangolin), http://cdimage.ubuntu.com/releases/12.04/release/, [access on March 17, 2013]

[10] Welcome to CANboat, https://github.com/canboat/canboat/wiki/CANboat, [access on January 20, 2013]

[11] NMEA 2000 Appendix B – PGNs (NMEA Network Messages) Full Database, http://www.nmea.org/store/index.asp?show=pdet&pid=325&cid=7, [access on January 23, 2013]

[12] Frank Cassidy, NMEA 2000 Explained - The Latest Word, March 2, 1999

[13] Using SSL for Secure Connections, http://dev.mysql.com/doc/refman/5.5/en/ssl-connections.html, [access on April 10, 2013]

[14] Configuring Serv-U with an SSL Certificate, http://www.rhinosoft.com/KnowledgeBase/kbarticle.asp?RefNo=1053&prod=rs, [access on April 19, 2013]

[15] Future Technology Devices International Ltd., Technical Note TN_101 Implementing Custom FTDI VID and PID Codes using Linux, Oct. 30, 2008.

Appendix A Installing NGT-1 gateway on embedded system

The guide for installing the NGT-1 Gateway on a Linux embedded system is shown as follows.

Step 1 Connect Actisense NGT-1 to the embedded system through USB interface. At this stage, the virtual COM (ttyUSB) for NGT-1 has not yet been created.

Step 2 Add product information to ftdi_sio by typing the following command in Ubuntu terminal:

```
sudo modprobe ftdi_sio vendor=0x0403 product=0xd9aa
```

Step 3 Add ftdi_sio to the system modules for the change to take effect.

```
echo ftdi_sio >>/etc/modules
```

Step 4 Create a configuration file for the NGT-1 gateway:

```
echo options ftdi_sio vendor=0x0403 product=0xd9aa >>/etc/modprobe.d/actisense.conf
```

Step 5 The virtual COM port (/dev/ttyUSB0) can be found under the directory /dev by using the command below:

```
ls -al dev/ttyUSB*
```

Step 6 If you do not log in as a root user, please remember to add your username to the *dialout* group, so that all these changes will take effective in your account.

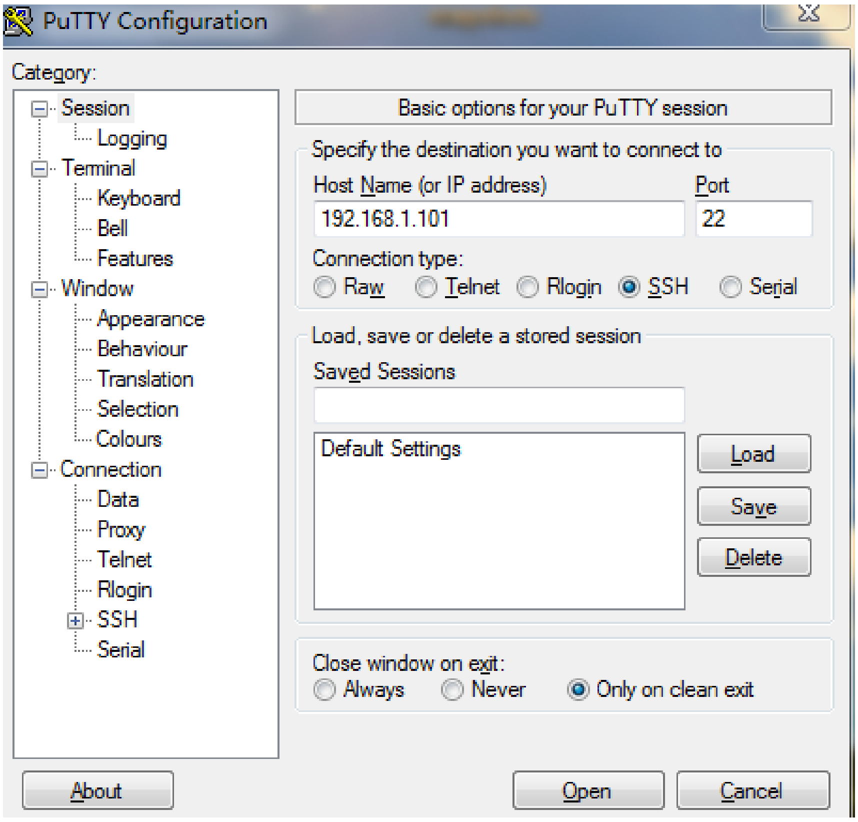
```
sudo usermod -aG dialout <your-username>
```

BRLTTY is a Linux/Unix-console access application for blind users. Please uninstall it first if it is installed on the system, or else the virtual COM port will not be accessible [15].

Appendix B Remote control on embedded system through SSH

Initially, we use PuTTY on a Windows laptop to enable remote control on the embedded system via SSH, which requires that the embedded system should be reachable, i.e., a public IP address is required. In order to make connection between our server and embedded system, the following steps are performed:

Step 1: As shown in Figure 38, enter the IP address of the embedded system, which is in the same LAN or has a public IP address.



Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.101 Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Figure 38 PuTTY configuration

Step 2: Once it successfully connects to the embedded system, a control terminal will pop up, in which we need to enter the password of remote embedded system.

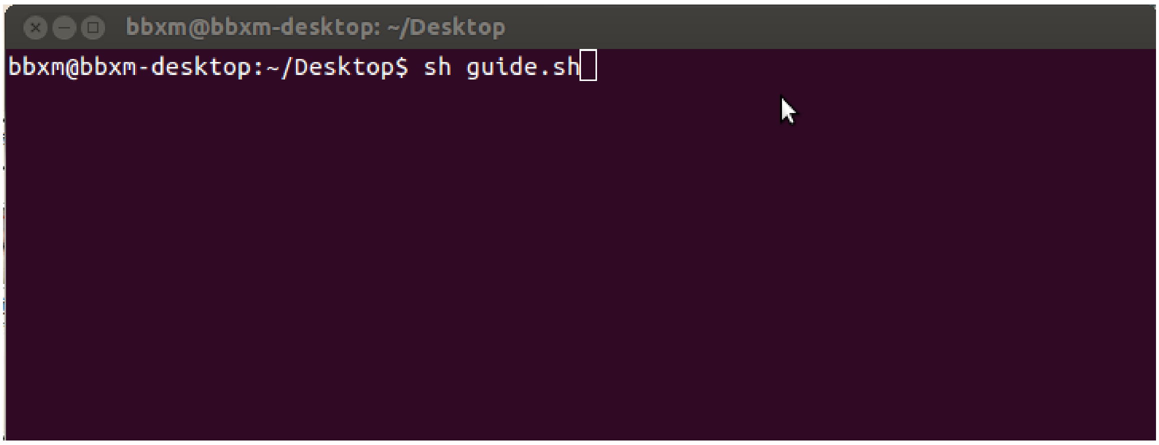
Step 3: After the password is verified, we can type any command to control the remote embedded system.

However, in practical scenario, the cellular network is often selected as the access mode when sailing at sea. In cellular networks, the IP address of the embedded system is usually a dynamic address assigned by the operator. It is difficulty for the remote server side to obtain the address. In addition, operators of cellular networks may restrict the SSH access in terms of security consideration. Consequently, SSH is not a proper approach for our case.

Appendix C Improvement on user experience

For the users who are not familiar with the Ubuntu terminal operations, it might be difficult to memorize the commands. Thus we develop a shell script to simplify the operation. When a user intends to manually launch data collection and transmission on the embedded system or through local control, he should follow the following steps.

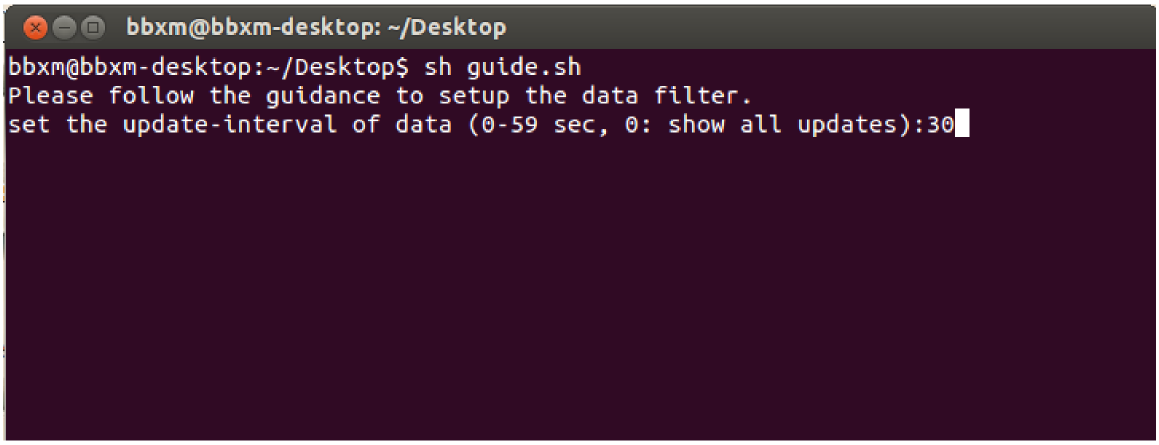
Step 1: Open the terminal and go to the directory of the shell script and type the command in Figure 39 to launch it.



```
bbxm@bbxm-desktop:~/Desktop$ sh guide.sh
```

Figure 39 Execute the shell script

Step 2: Set message update interval according to the actual demand.

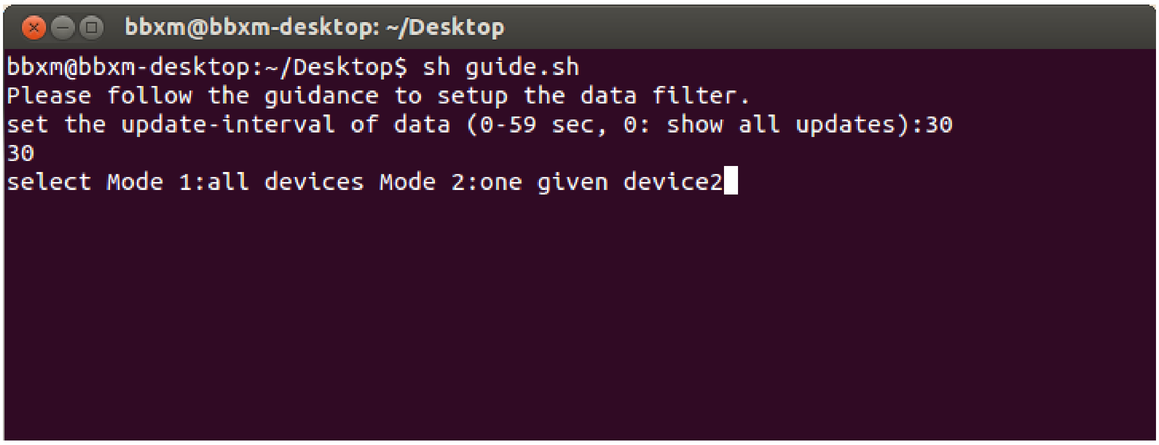


```
bbxm@bbxm-desktop:~/Desktop$ sh guide.sh
Please follow the guidance to setup the data filter.
set the update-interval of data (0-59 sec, 0: show all updates):30
```

Figure 40 Set message update interval

Step 2: Select filtering mode.

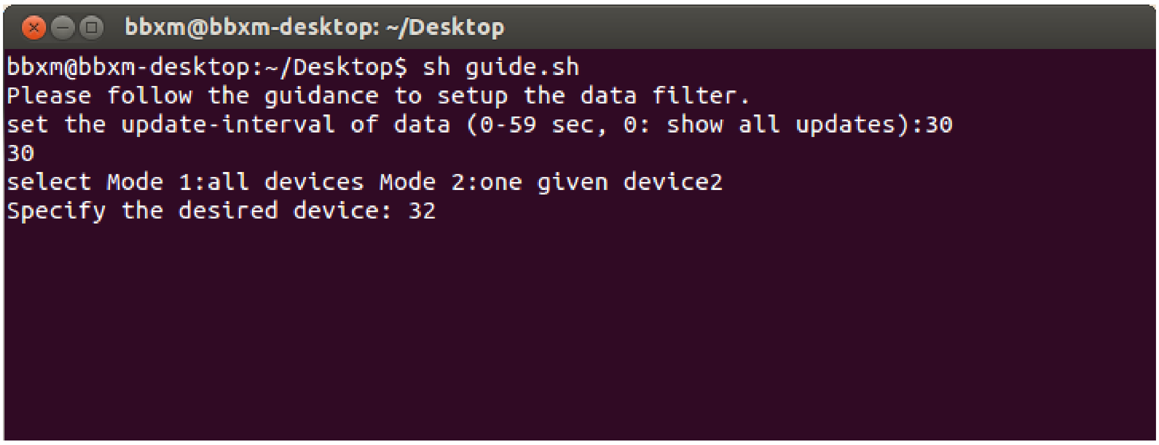
As shown in Figure 41, mode 1 is used for collecting message from all the electronics in the NMEA 2000 network while mode 2 is adopted for data collection from a specific device. If mode 1 is selected, the data collection and transmission will start imediately. Or else go to Step 3 if mode 2 is selected.



```
bbxm@bbxm-desktop:~/Desktop$ sh guide.sh
Please follow the guidance to setup the data filter.
set the update-interval of data (0-59 sec, 0: show all updates):30
30
select Mode 1:all devices Mode 2:one given device2
```

Figure 41 Select filtering mode

Step 3: Enter the device ID of the device, in which the user is interested.



```
bbxm@bbxm-desktop:~/Desktop$ sh guide.sh
Please follow the guidance to setup the data filter.
set the update-interval of data (0-59 sec, 0: show all updates):30
30
select Mode 1:all devices Mode 2:one given device2
Specify the desired device: 32
```

Figure 42 Specify device ID

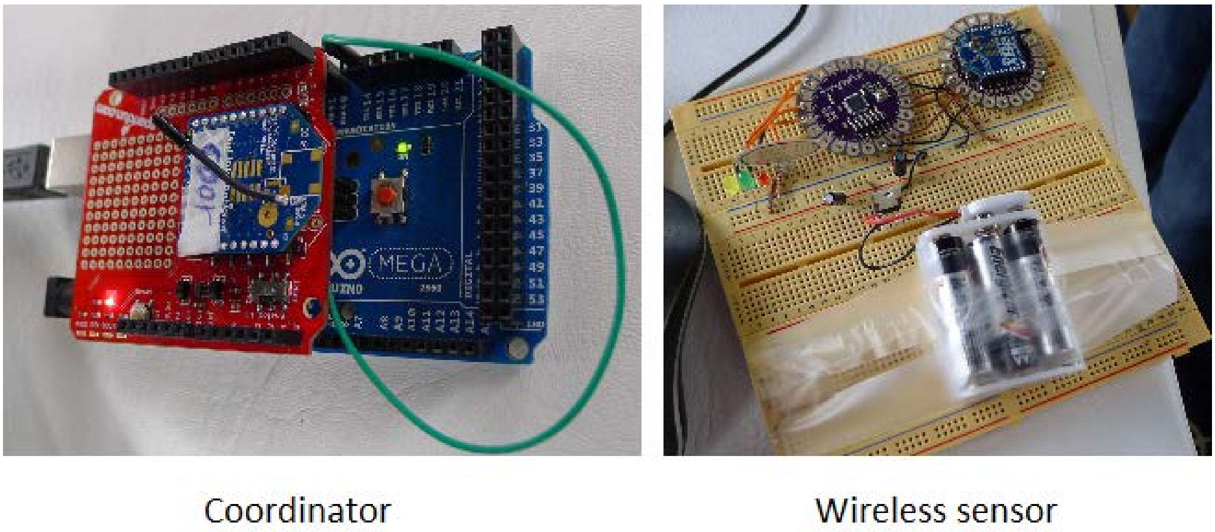
Then only the data from the selected device will be transmitted to the remote server and stored in both sides.

This script can also be used in the scenarios of local control by using SSH.

Appendix D Integration with other systems

The data collection and transmission system can be integrated with other systems: the wireless sensor network (WSN) and the graphical user interface (GUI) for the remote server.

The WSN system developed by another two master students is used for collecting data from wireless sensors installed on boats. A coordinator needs to be connected to the embedded system in order to receive the data from the sensors through wireless transmission.



Coordinator

Wireless sensor

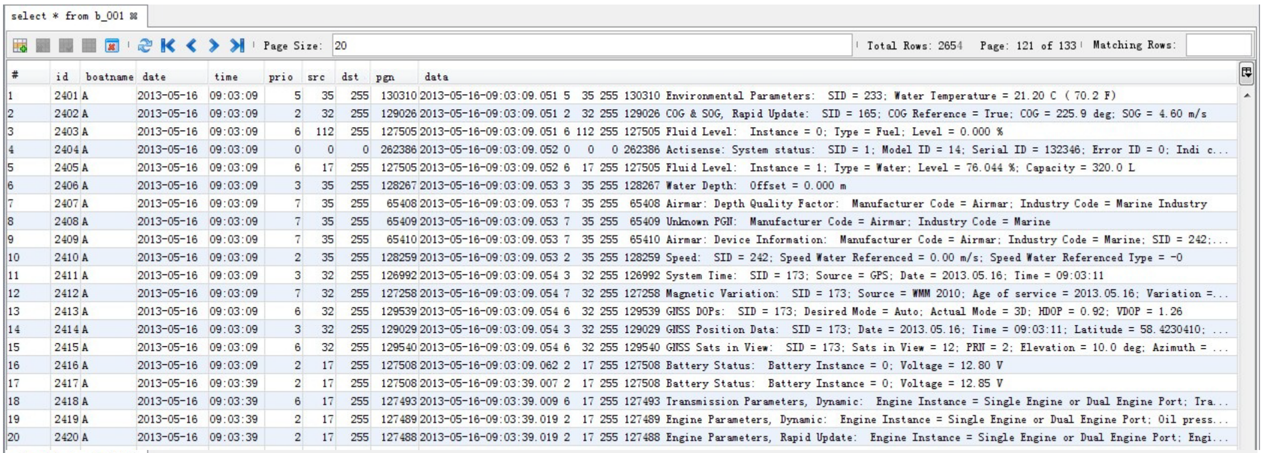
Figure 43 coordinator and wireless sensor

The GUI for the remote server is an enhancement to the remote server. It provides more user-friendly interface and better performance on data search.

Appendix E Results of real-life experiment

We have real-life experiments on a Marex 320 boat on 16 May 2013. There are five devices connected to the CAN Bus: a NGT-1 gateway (ID is 0), a GPS receiver (ID is 32), an engine (ID is 17), a tank level monitor (ID is 35) and a DST110 Triducer (ID is 112). As the tank level monitor and DST110 Triducer are not installed properly on the boat, so the data from them are not accurate.

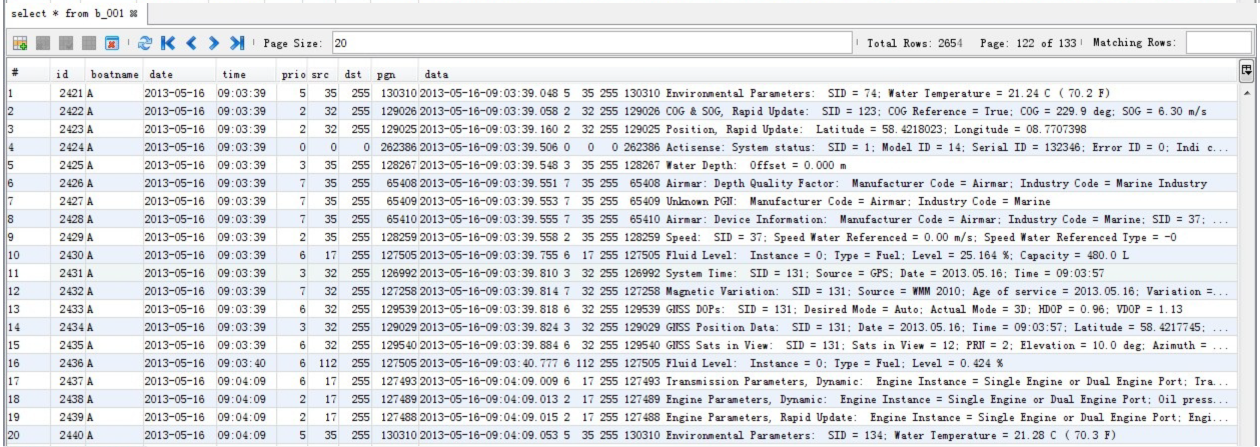
The following is part of the collected data stored in the remote server.



select * from b_001

Page Size: 20 Total Rows: 2654 Page: 121 of 133 Matching Rows:

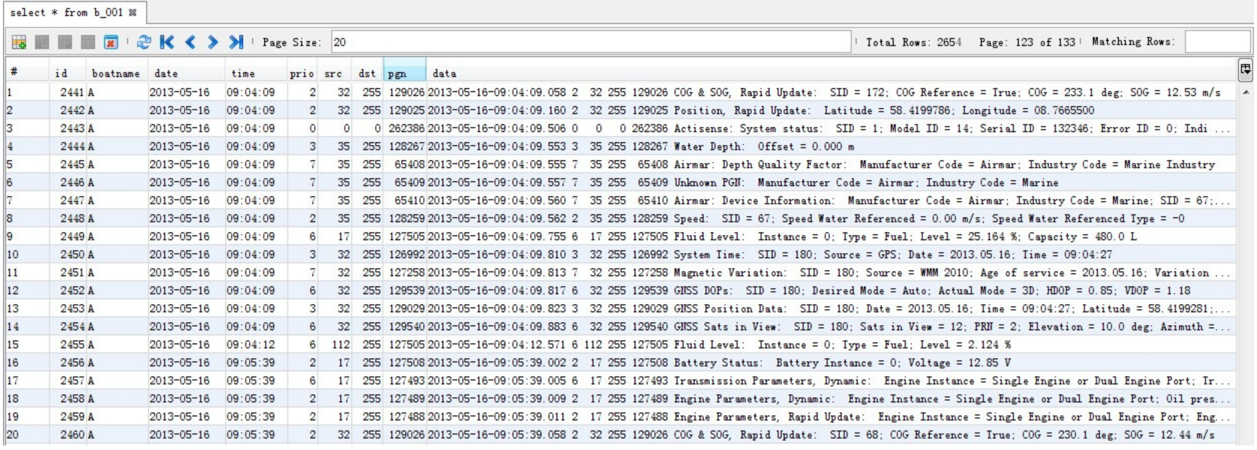
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2401	A	2013-05-16	09:03:09	5	35	255	130310	2013-05-16-09:03:09.051 5 35 255 130310 Environmental Parameters: SID = 233; Water Temperature = 21.20 C (70.2 F)
2	2402	A	2013-05-16	09:03:09	2	32	255	129026	2013-05-16-09:03:09.051 2 32 255 129026 COG & SOG, Rapid Update: SID = 165; COG Reference = True; COG = 225.9 deg; SOG = 4.60 m/s
3	2403	A	2013-05-16	09:03:09	6	112	255	127505	2013-05-16-09:03:09.051 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 0.000 %
4	2404	A	2013-05-16	09:03:09	0	0	0	262386	2013-05-16-09:03:09.052 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi c...
5	2405	A	2013-05-16	09:03:09	6	17	255	127505	2013-05-16-09:03:09.052 6 17 255 127505 Fluid Level: Instance = 1; Type = Water; Level = 76.044 %; Capacity = 320.0 L
6	2406	A	2013-05-16	09:03:09	3	35	255	128267	2013-05-16-09:03:09.053 3 35 255 128267 Water Depth: Offset = 0.000 m
7	2407	A	2013-05-16	09:03:09	7	35	255	65408	2013-05-16-09:03:09.053 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
8	2408	A	2013-05-16	09:03:09	7	35	255	65409	2013-05-16-09:03:09.053 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
9	2409	A	2013-05-16	09:03:09	7	35	255	65410	2013-05-16-09:03:09.053 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 242;...
10	2410	A	2013-05-16	09:03:09	2	35	255	128259	2013-05-16-09:03:09.053 2 35 255 128259 Speed: SID = 242; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
11	2411	A	2013-05-16	09:03:09	3	32	255	126992	2013-05-16-09:03:09.054 3 32 255 126992 System Time: SID = 173; Source = GPS; Date = 2013.05.16; Time = 09:03:11
12	2412	A	2013-05-16	09:03:09	7	32	255	127258	2013-05-16-09:03:09.054 7 32 255 127258 Magnetic Variation: SID = 173; Source = WMM 2010; Age of service = 2013.05.16; Variation =...
13	2413	A	2013-05-16	09:03:09	6	32	255	129539	2013-05-16-09:03:09.054 6 32 255 129539 GNSS DOPs: SID = 173; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.92; VDOP = 1.26
14	2414	A	2013-05-16	09:03:09	3	32	255	129029	2013-05-16-09:03:09.054 3 32 255 129029 GNSS Position Data: SID = 173; Date = 2013.05.16; Time = 09:03:11; Latitude = 58.4230410; ...
15	2415	A	2013-05-16	09:03:09	6	32	255	129540	2013-05-16-09:03:09.054 6 32 255 129540 GNSS Sats in View: SID = 173; Sats in View = 12; PRN = 2; Elevation = 10.0 deg; Azimuth = ...
16	2416	A	2013-05-16	09:03:09	2	17	255	127508	2013-05-16-09:03:09.062 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.80 V
17	2417	A	2013-05-16	09:03:39	2	17	255	127508	2013-05-16-09:03:39.007 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.85 V
18	2418	A	2013-05-16	09:03:39	6	17	255	127493	2013-05-16-09:03:39.009 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tra...
19	2419	A	2013-05-16	09:03:39	2	17	255	127489	2013-05-16-09:03:39.019 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil press...
20	2420	A	2013-05-16	09:03:39	2	17	255	127488	2013-05-16-09:03:39.019 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engi...



select * from b_001

Page Size: 20 Total Rows: 2654 Page: 122 of 133 Matching Rows:

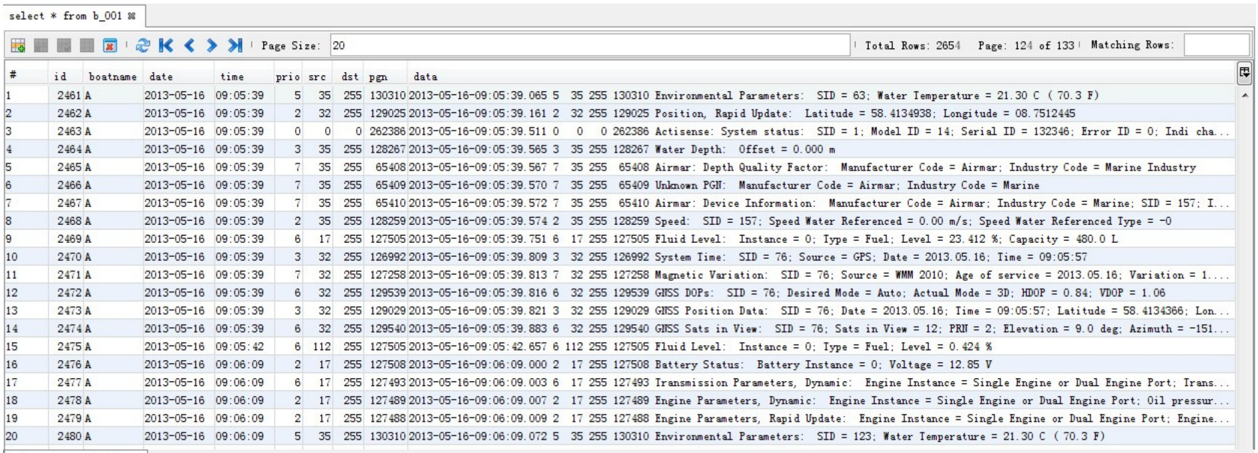
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2421	A	2013-05-16	09:03:39	5	35	255	130310	2013-05-16-09:03:39.048 5 35 255 130310 Environmental Parameters: SID = 74; Water Temperature = 21.24 C (70.2 F)
2	2422	A	2013-05-16	09:03:39	2	32	255	129026	2013-05-16-09:03:39.058 2 32 255 129026 COG & SOG, Rapid Update: SID = 123; COG Reference = True; COG = 229.9 deg; SOG = 6.30 m/s
3	2423	A	2013-05-16	09:03:39	2	32	255	129025	2013-05-16-09:03:39.160 2 32 255 129025 Position, Rapid Update: Latitude = 58.4218023; Longitude = 08.7707398
4	2424	A	2013-05-16	09:03:39	0	0	0	262386	2013-05-16-09:03:39.506 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi c...
5	2425	A	2013-05-16	09:03:39	3	35	255	128267	2013-05-16-09:03:39.548 3 35 255 128267 Water Depth: Offset = 0.000 m
6	2426	A	2013-05-16	09:03:39	7	35	255	65408	2013-05-16-09:03:39.551 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
7	2427	A	2013-05-16	09:03:39	7	35	255	65409	2013-05-16-09:03:39.553 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
8	2428	A	2013-05-16	09:03:39	7	35	255	65410	2013-05-16-09:03:39.555 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 37; ...
9	2429	A	2013-05-16	09:03:39	2	35	255	128259	2013-05-16-09:03:39.558 2 35 255 128259 Speed: SID = 37; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
10	2430	A	2013-05-16	09:03:39	6	17	255	127505	2013-05-16-09:03:39.755 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 25.164 %; Capacity = 480.0 L
11	2431	A	2013-05-16	09:03:39	3	32	255	126992	2013-05-16-09:03:39.810 3 32 255 126992 System Time: SID = 131; Source = GPS; Date = 2013.05.16; Time = 09:03:57
12	2432	A	2013-05-16	09:03:39	7	32	255	127258	2013-05-16-09:03:39.814 7 32 255 127258 Magnetic Variation: SID = 131; Source = WMM 2010; Age of service = 2013.05.16; Variation =...
13	2433	A	2013-05-16	09:03:39	6	32	255	129539	2013-05-16-09:03:39.818 6 32 255 129539 GNSS DOPs: SID = 131; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.96; VDOP = 1.13
14	2434	A	2013-05-16	09:03:39	3	32	255	129029	2013-05-16-09:03:39.824 3 32 255 129029 GNSS Position Data: SID = 131; Date = 2013.05.16; Time = 09:03:57; Latitude = 58.4217745; ...
15	2435	A	2013-05-16	09:03:39	6	32	255	129540	2013-05-16-09:03:39.884 6 32 255 129540 GNSS Sats in View: SID = 131; Sats in View = 12; PRN = 2; Elevation = 10.0 deg; Azimuth = ...
16	2436	A	2013-05-16	09:03:40	6	112	255	127505	2013-05-16-09:03:40.777 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 0.424 %
17	2437	A	2013-05-16	09:04:09	6	17	255	127493	2013-05-16-09:04:09.009 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tra...
18	2438	A	2013-05-16	09:04:09	2	17	255	127489	2013-05-16-09:04:09.013 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil press...
19	2439	A	2013-05-16	09:04:09	2	17	255	127488	2013-05-16-09:04:09.015 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engi...
20	2440	A	2013-05-16	09:04:09	5	35	255	130310	2013-05-16-09:04:09.053 5 35 255 130310 Environmental Parameters: SID = 134; Water Temperature = 21.28 C (70.3 F)



select * from b_001

Page Size: 20 Total Rows: 2654 Page: 123 of 133 Matching Rows:

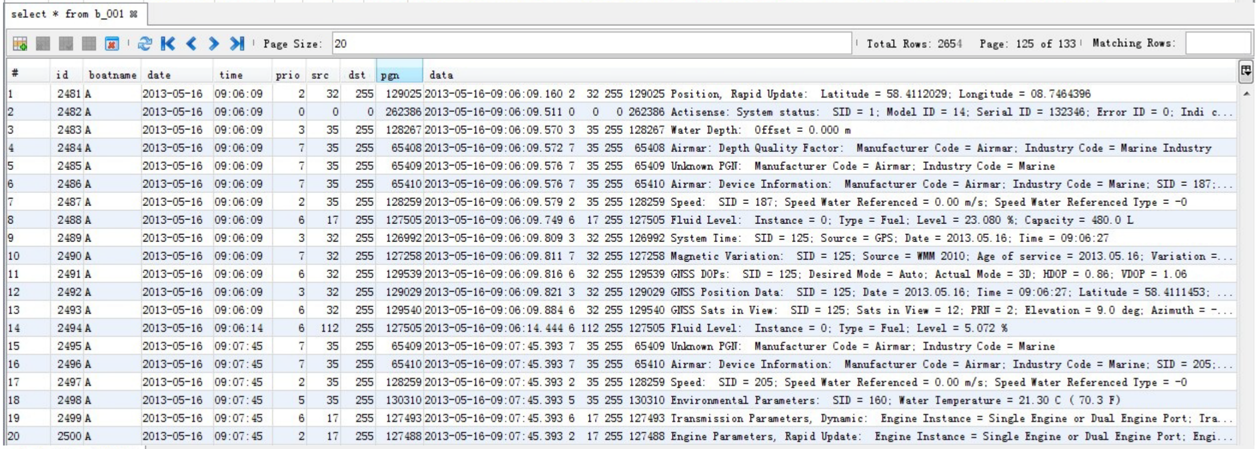
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2441	A	2013-05-16	09:04:09	2	32	255	129026	2013-05-16-09:04:09.058 2 32 255 129026 COG & SOG, Rapid Update: SID = 172; COG Reference = True; COG = 233.1 deg; SOG = 12.53 m/s
2	2442	A	2013-05-16	09:04:09	2	32	255	129025	2013-05-16-09:04:09.160 2 32 255 129025 Position, Rapid Update: Latitude = 58.4199786; Longitude = 08.7665500
3	2443	A	2013-05-16	09:04:09	0	0	0	262386	2013-05-16-09:04:09.506 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi ...
4	2444	A	2013-05-16	09:04:09	3	35	255	128267	2013-05-16-09:04:09.553 3 35 255 128267 Water Depth: Offset = 0.000 m
5	2445	A	2013-05-16	09:04:09	7	35	255	65408	2013-05-16-09:04:09.555 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
6	2446	A	2013-05-16	09:04:09	7	35	255	65409	2013-05-16-09:04:09.557 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
7	2447	A	2013-05-16	09:04:09	7	35	255	65410	2013-05-16-09:04:09.560 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 67;...
8	2448	A	2013-05-16	09:04:09	2	35	255	128259	2013-05-16-09:04:09.562 2 35 255 128259 Speed: SID = 67; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
9	2449	A	2013-05-16	09:04:09	6	17	255	127505	2013-05-16-09:04:09.755 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 25.164 %; Capacity = 480.0 L
10	2450	A	2013-05-16	09:04:09	3	32	255	126992	2013-05-16-09:04:09.810 3 32 255 126992 System Time: SID = 180; Source = GPS; Date = 2013.05.16; Time = 09:04:27
11	2451	A	2013-05-16	09:04:09	7	32	255	127258	2013-05-16-09:04:09.813 7 32 255 127258 Magnetic Variation: SID = 180; Source = WMM 2010; Age of service = 2013.05.16; Variation ...
12	2452	A	2013-05-16	09:04:09	6	32	255	129539	2013-05-16-09:04:09.817 6 32 255 129539 GNSS DOPs: SID = 180; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.85; VDOP = 1.18
13	2453	A	2013-05-16	09:04:09	3	32	255	129029	2013-05-16-09:04:09.823 3 32 255 129029 GNSS Position Data: SID = 180; Date = 2013.05.16; Time = 09:04:27; Latitude = 58.4199281;...
14	2454	A	2013-05-16	09:04:09	6	32	255	129540	2013-05-16-09:04:09.883 6 32 255 129540 GNSS Sats in View: SID = 180; Sats in View = 12; PRN = 2; Elevation = 10.0 deg; Azimuth =...
15	2455	A	2013-05-16	09:04:12	6	112	255	127505	2013-05-16-09:04:12.571 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 2.124 %
16	2456	A	2013-05-16	09:05:39	2	17	255	127508	2013-05-16-09:05:39.002 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.85 V
17	2457	A	2013-05-16	09:05:39	6	17	255	127493	2013-05-16-09:05:39.005 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tr...
18	2458	A	2013-05-16	09:05:39	2	17	255	127489	2013-05-16-09:05:39.009 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pres...
19	2459	A	2013-05-16	09:05:39	2	17	255	127488	2013-05-16-09:05:39.011 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Eng...
20	2460	A	2013-05-16	09:05:39	2	32	255	129026	2013-05-16-09:05:39.058 2 32 255 129026 COG & SOG, Rapid Update: SID = 68; COG Reference = True; COG = 230.1 deg; SOG = 12.44 m/s



select * from b_001

Page Size: 20 | Total Rows: 2654 Page: 124 of 133 | Matching Rows:

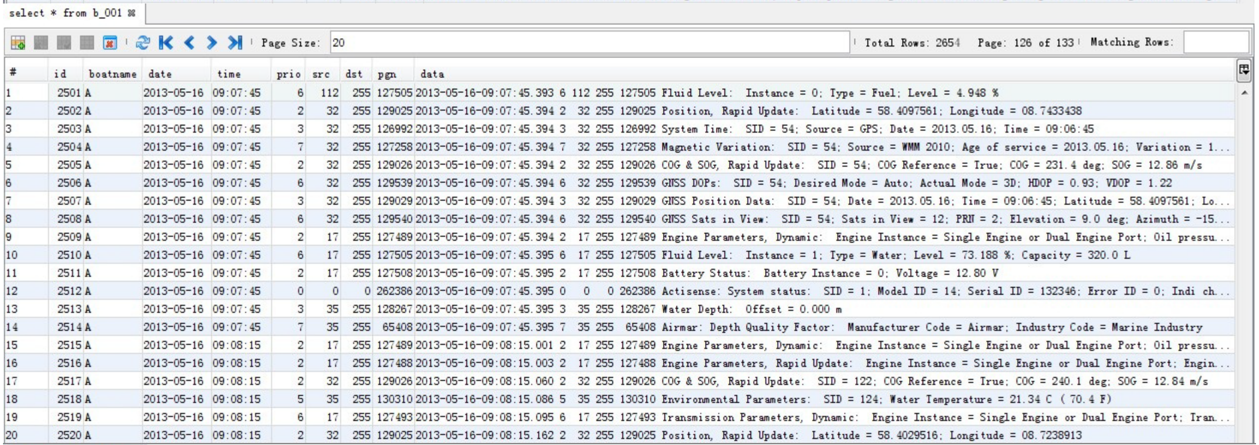
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2461	A	2013-05-16	09:05:39	5	35	255	130310	2013-05-16-09:05:39.065 5 35 255 130310 Environmental Parameters: SID = 63; Water Temperature = 21.30 C (70.3 F)
2	2462	A	2013-05-16	09:05:39	2	32	255	129025	2013-05-16-09:05:39.161 2 32 255 129025 Position, Rapid Update: Latitude = 58.4134938; Longitude = 08.7512445
3	2463	A	2013-05-16	09:05:39	0	0	0	262386	2013-05-16-09:05:39.511 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi cha...
4	2464	A	2013-05-16	09:05:39	3	35	255	128267	2013-05-16-09:05:39.565 3 35 255 128267 Water Depth: Offset = 0.000 m
5	2465	A	2013-05-16	09:05:39	7	35	255	65408	2013-05-16-09:05:39.567 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
6	2466	A	2013-05-16	09:05:39	7	35	255	65409	2013-05-16-09:05:39.570 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
7	2467	A	2013-05-16	09:05:39	7	35	255	65410	2013-05-16-09:05:39.572 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 157; I...
8	2468	A	2013-05-16	09:05:39	2	35	255	128259	2013-05-16-09:05:39.574 2 35 255 128259 Speed: SID = 157; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
9	2469	A	2013-05-16	09:05:39	6	17	255	127505	2013-05-16-09:05:39.751 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 23.412 %; Capacity = 480.0 L
10	2470	A	2013-05-16	09:05:39	3	32	255	126992	2013-05-16-09:05:39.809 3 32 255 126992 System Time: SID = 76; Source = GPS; Date = 2013.05.16; Time = 09:05:57
11	2471	A	2013-05-16	09:05:39	7	32	255	127258	2013-05-16-09:05:39.813 7 32 255 127258 Magnetic Variation: SID = 76; Source = WMM 2010; Age of service = 2013.05.16; Variation = 1....
12	2472	A	2013-05-16	09:05:39	6	32	255	129539	2013-05-16-09:05:39.816 6 32 255 129539 GNSS DOPs: SID = 76; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.84; VDOP = 1.06
13	2473	A	2013-05-16	09:05:39	3	32	255	129029	2013-05-16-09:05:39.821 3 32 255 129029 GNSS Position Data: SID = 76; Date = 2013.05.16; Time = 09:05:57; Latitude = 58.4134366; Lon...
14	2474	A	2013-05-16	09:05:39	6	32	255	129540	2013-05-16-09:05:39.883 6 32 255 129540 GNSS Sats in View: SID = 76; Sats in View = 12; PRN = 2; Elevation = 9.0 deg; Azimuth = -151...
15	2475	A	2013-05-16	09:05:42	6	112	255	127505	2013-05-16-09:05:42.657 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 0.424 %
16	2476	A	2013-05-16	09:06:09	2	17	255	127508	2013-05-16-09:06:09.000 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.85 V
17	2477	A	2013-05-16	09:06:09	6	17	255	127493	2013-05-16-09:06:09.003 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Trans...
18	2478	A	2013-05-16	09:06:09	2	17	255	127489	2013-05-16-09:06:09.007 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pressur...
19	2479	A	2013-05-16	09:06:09	2	17	255	127488	2013-05-16-09:06:09.009 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engine...
20	2480	A	2013-05-16	09:06:09	5	35	255	130310	2013-05-16-09:06:09.072 5 35 255 130310 Environmental Parameters: SID = 123; Water Temperature = 21.30 C (70.3 F)



select * from b_001

Page Size: 20 | Total Rows: 2654 Page: 125 of 133 | Matching Rows:

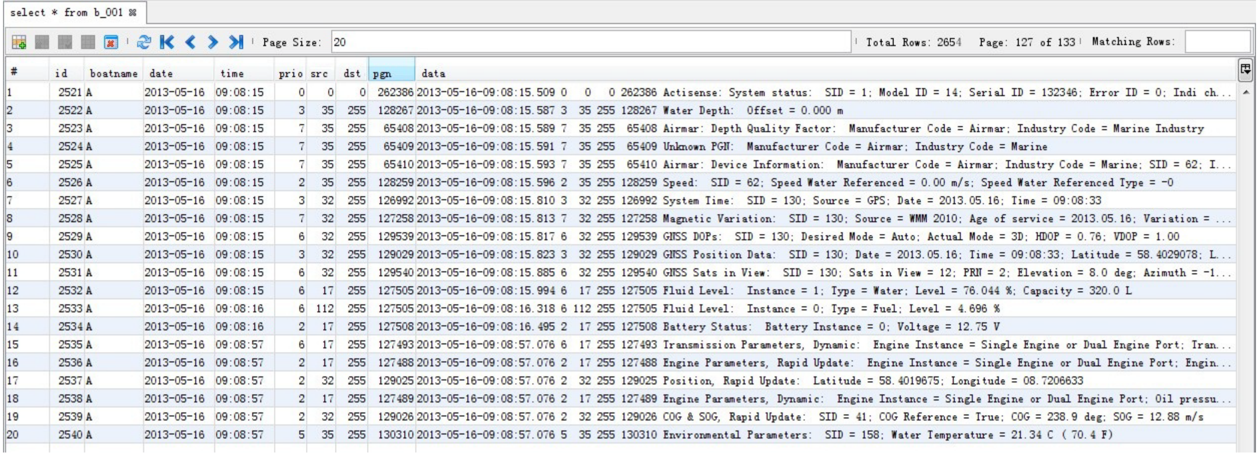
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2481	A	2013-05-16	09:06:09	2	32	255	129025	2013-05-16-09:06:09.160 2 32 255 129025 Position, Rapid Update: Latitude = 58.4112029; Longitude = 08.7464396
2	2482	A	2013-05-16	09:06:09	0	0	0	262386	2013-05-16-09:06:09.511 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi c...
3	2483	A	2013-05-16	09:06:09	3	35	255	128267	2013-05-16-09:06:09.570 3 35 255 128267 Water Depth: Offset = 0.000 m
4	2484	A	2013-05-16	09:06:09	7	35	255	65408	2013-05-16-09:06:09.572 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
5	2485	A	2013-05-16	09:06:09	7	35	255	65409	2013-05-16-09:06:09.576 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
6	2486	A	2013-05-16	09:06:09	7	35	255	65410	2013-05-16-09:06:09.576 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 187;...
7	2487	A	2013-05-16	09:06:09	2	35	255	128259	2013-05-16-09:06:09.579 2 35 255 128259 Speed: SID = 187; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
8	2488	A	2013-05-16	09:06:09	6	17	255	127505	2013-05-16-09:06:09.749 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 23.080 %; Capacity = 480.0 L
9	2489	A	2013-05-16	09:06:09	3	32	255	126992	2013-05-16-09:06:09.809 3 32 255 126992 System Time: SID = 125; Source = GPS; Date = 2013.05.16; Time = 09:06:27
10	2490	A	2013-05-16	09:06:09	7	32	255	127258	2013-05-16-09:06:09.811 7 32 255 127258 Magnetic Variation: SID = 125; Source = WMM 2010; Age of service = 2013.05.16; Variation =...
11	2491	A	2013-05-16	09:06:09	6	32	255	129539	2013-05-16-09:06:09.816 6 32 255 129539 GNSS DOPs: SID = 125; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.86; VDOP = 1.06
12	2492	A	2013-05-16	09:06:09	3	32	255	129029	2013-05-16-09:06:09.821 3 32 255 129029 GNSS Position Data: SID = 125; Date = 2013.05.16; Time = 09:06:27; Latitude = 58.4111453; ...
13	2493	A	2013-05-16	09:06:09	6	32	255	129540	2013-05-16-09:06:09.884 6 32 255 129540 GNSS Sats in View: SID = 125; Sats in View = 12; PRN = 2; Elevation = 9.0 deg; Azimuth = -...
14	2494	A	2013-05-16	09:06:14	6	112	255	127505	2013-05-16-09:06:14.444 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 5.072 %
15	2495	A	2013-05-16	09:07:45	7	35	255	65409	2013-05-16-09:07:45.393 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
16	2496	A	2013-05-16	09:07:45	7	35	255	65410	2013-05-16-09:07:45.393 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 205;...
17	2497	A	2013-05-16	09:07:45	2	35	255	128259	2013-05-16-09:07:45.393 2 35 255 128259 Speed: SID = 205; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
18	2498	A	2013-05-16	09:07:45	5	35	255	130310	2013-05-16-09:07:45.393 5 35 255 130310 Environmental Parameters: SID = 160; Water Temperature = 21.30 C (70.3 F)
19	2499	A	2013-05-16	09:07:45	6	17	255	127493	2013-05-16-09:07:45.393 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tra...
20	2500	A	2013-05-16	09:07:45	2	17	255	127488	2013-05-16-09:07:45.393 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engi...



select * from b_001

Page Size: 20 | Total Rows: 2654 Page: 126 of 133 | Matching Rows:

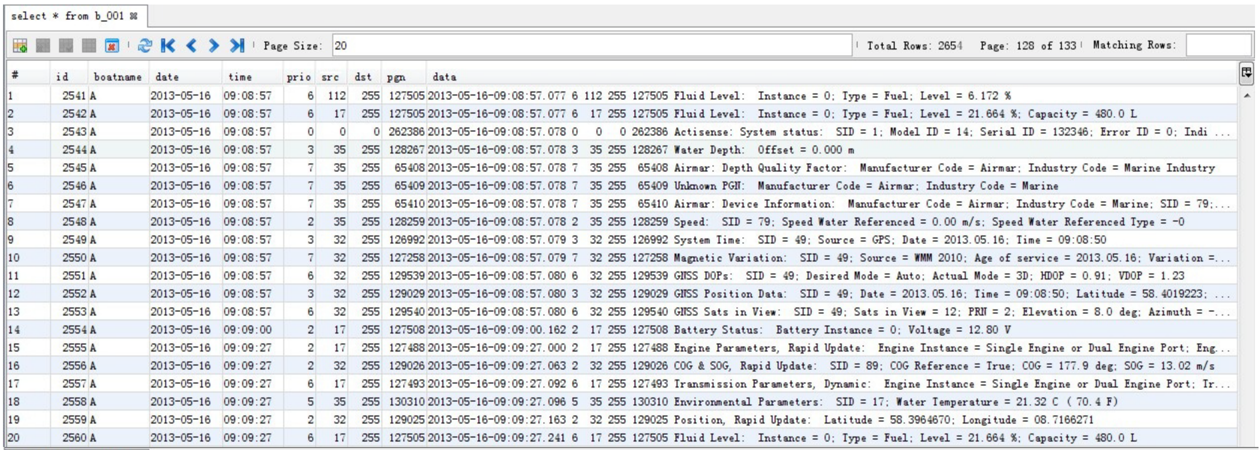
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2501	A	2013-05-16	09:07:45	6	112	255	127505	2013-05-16-09:07:45.393 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 4.948 %
2	2502	A	2013-05-16	09:07:45	2	32	255	129025	2013-05-16-09:07:45.394 2 32 255 129025 Position, Rapid Update: Latitude = 58.4097561; Longitude = 08.7433438
3	2503	A	2013-05-16	09:07:45	3	32	255	126992	2013-05-16-09:07:45.394 3 32 255 126992 System Time: SID = 54; Source = GPS; Date = 2013.05.16; Time = 09:06:45
4	2504	A	2013-05-16	09:07:45	7	32	255	127258	2013-05-16-09:07:45.394 7 32 255 127258 Magnetic Variation: SID = 54; Source = WMM 2010; Age of service = 2013.05.16; Variation = 1...
5	2505	A	2013-05-16	09:07:45	2	32	255	129026	2013-05-16-09:07:45.394 2 32 255 129026 COG & SOG, Rapid Update: SID = 54; COG Reference = True; COG = 231.4 deg; SOG = 12.86 m/s
6	2506	A	2013-05-16	09:07:45	6	32	255	129539	2013-05-16-09:07:45.394 6 32 255 129539 GNSS DOPs: SID = 54; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.93; VDOP = 1.22
7	2507	A	2013-05-16	09:07:45	3	32	255	129029	2013-05-16-09:07:45.394 3 32 255 129029 GNSS Position Data: SID = 54; Date = 2013.05.16; Time = 09:06:45; Latitude = 58.4097561; Lo...
8	2508	A	2013-05-16	09:07:45	6	32	255	129540	2013-05-16-09:07:45.394 6 32 255 129540 GNSS Sats in View: SID = 54; Sats in View = 12; PRN = 2; Elevation = 9.0 deg; Azimuth = -15...
9	2509	A	2013-05-16	09:07:45	2	17	255	127489	2013-05-16-09:07:45.394 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pressu...
10	2510	A	2013-05-16	09:07:45	6	17	255	127505	2013-05-16-09:07:45.395 6 17 255 127505 Fluid Level: Instance = 1; Type = Water; Level = 73.188 %; Capacity = 320.0 L
11	2511	A	2013-05-16	09:07:45	2	17	255	127508	2013-05-16-09:07:45.395 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.80 V
12	2512	A	2013-05-16	09:07:45	0	0	0	262386	2013-05-16-09:07:45.395 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi ch...
13	2513	A	2013-05-16	09:07:45	3	35	255	128267	2013-05-16-09:07:45.395 3 35 255 128267 Water Depth: Offset = 0.000 m
14	2514	A	2013-05-16	09:07:45	7	35	255	65408	2013-05-16-09:07:45.395 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
15	2515	A	2013-05-16	09:08:15	2	17	255	127489	2013-05-16-09:08:15.001 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pressu...
16	2516	A	2013-05-16	09:08:15	2	17	255	127488	2013-05-16-09:08:15.003 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engin...
17	2517	A	2013-05-16	09:08:15	2	32	255	129026	2013-05-16-09:08:15.060 2 32 255 129026 COG & SOG, Rapid Update: SID = 122; COG Reference = True; COG = 240.1 deg; SOG = 12.84 m/s
18	2518	A	2013-05-16	09:08:15	5	35	255	130310	2013-05-16-09:08:15.086 5 35 255 130310 Environmental Parameters: SID = 124; Water Temperature = 21.34 C (70.4 F)
19	2519	A	2013-05-16	09:08:15	6	17	255	127493	2013-05-16-09:08:15.095 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tran...
20	2520	A	2013-05-16	09:08:15	2	32	255	129025	2013-05-16-09:08:15.162 2 32 255 129025 Position, Rapid Update: Latitude = 58.4029516; Longitude = 08.7238913



select * from b_001

Page Size: 20 | Total Rows: 2654 Page: 127 of 133 | Matching Rows:

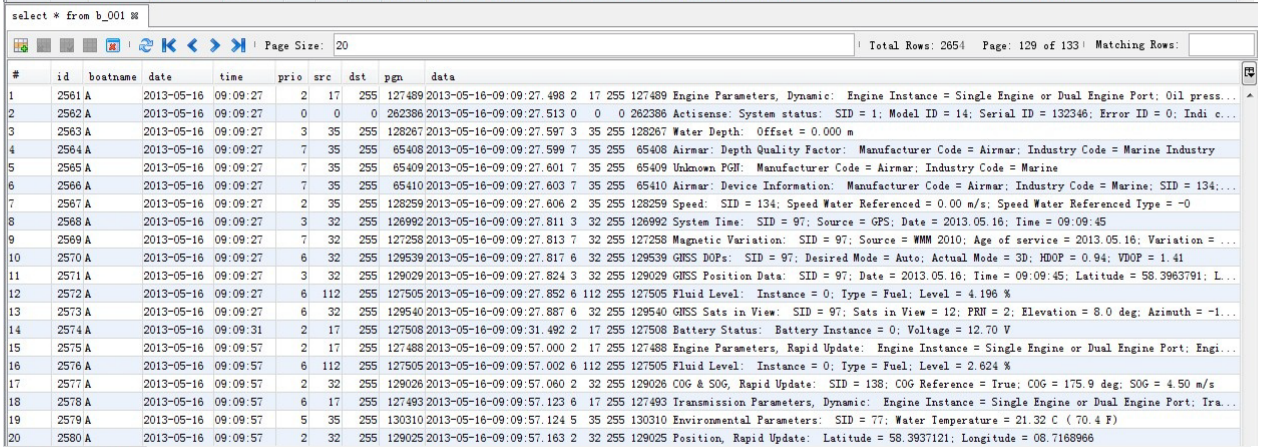
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2521	A	2013-05-16	09:08:15	0	0	0	262386	2013-05-16-09:08:15.509 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi ch...
2	2522	A	2013-05-16	09:08:15	3	35	255	128267	2013-05-16-09:08:15.587 3 35 255 128267 Water Depth: Offset = 0.000 m
3	2523	A	2013-05-16	09:08:15	7	35	255	65408	2013-05-16-09:08:15.589 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
4	2524	A	2013-05-16	09:08:15	7	35	255	65409	2013-05-16-09:08:15.591 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
5	2525	A	2013-05-16	09:08:15	7	35	255	65410	2013-05-16-09:08:15.593 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 62; I...
6	2526	A	2013-05-16	09:08:15	2	35	255	128259	2013-05-16-09:08:15.596 2 35 255 128259 Speed: SID = 62; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
7	2527	A	2013-05-16	09:08:15	3	32	255	126992	2013-05-16-09:08:15.810 3 32 255 126992 System Time: SID = 130; Source = GPS; Date = 2013.05.16; Time = 09:08:33
8	2528	A	2013-05-16	09:08:15	7	32	255	127258	2013-05-16-09:08:15.813 7 32 255 127258 Magnetic Variation: SID = 130; Source = WMM 2010; Age of service = 2013.05.16; Variation = ...
9	2529	A	2013-05-16	09:08:15	6	32	255	129539	2013-05-16-09:08:15.817 6 32 255 129539 GNSS DOPs: SID = 130; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.76; VDOP = 1.00
10	2530	A	2013-05-16	09:08:15	3	32	255	129029	2013-05-16-09:08:15.823 3 32 255 129029 GNSS Position Data: SID = 130; Date = 2013.05.16; Time = 09:08:33; Latitude = 58.4029078; L...
11	2531	A	2013-05-16	09:08:15	6	32	255	129540	2013-05-16-09:08:15.885 6 32 255 129540 GNSS Sats in View: SID = 130; Sats in View = 12; PRN = 2; Elevation = 8.0 deg; Azimuth = -1...
12	2532	A	2013-05-16	09:08:15	6	17	255	127505	2013-05-16-09:08:15.994 6 17 255 127505 Fluid Level: Instance = 1; Type = Water; Level = 76.044 %; Capacity = 320.0 L
13	2533	A	2013-05-16	09:08:16	6	112	255	127505	2013-05-16-09:08:16.318 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 4.696 %
14	2534	A	2013-05-16	09:08:16	2	17	255	127508	2013-05-16-09:08:16.495 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.75 V
15	2535	A	2013-05-16	09:08:57	6	17	255	127493	2013-05-16-09:08:57.076 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tran...
16	2536	A	2013-05-16	09:08:57	2	17	255	127488	2013-05-16-09:08:57.076 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engin...
17	2537	A	2013-05-16	09:08:57	2	32	255	129025	2013-05-16-09:08:57.076 2 32 255 129025 Position, Rapid Update: Latitude = 58.4019675; Longitude = 08.7206633
18	2538	A	2013-05-16	09:08:57	2	17	255	127489	2013-05-16-09:08:57.076 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pressu...
19	2539	A	2013-05-16	09:08:57	2	32	255	129026	2013-05-16-09:08:57.076 2 32 255 129026 COG & SOG, Rapid Update: SID = 41; COG Reference = True; COG = 238.9 deg; SOG = 12.88 m/s
20	2540	A	2013-05-16	09:08:57	5	35	255	130310	2013-05-16-09:08:57.076 5 35 255 130310 Environmental Parameters: SID = 158; Water Temperature = 21.34 C (70.4 F)



select * from b_001

Page Size: 20 Total Rows: 2654 Page: 128 of 133 Matching Rows:

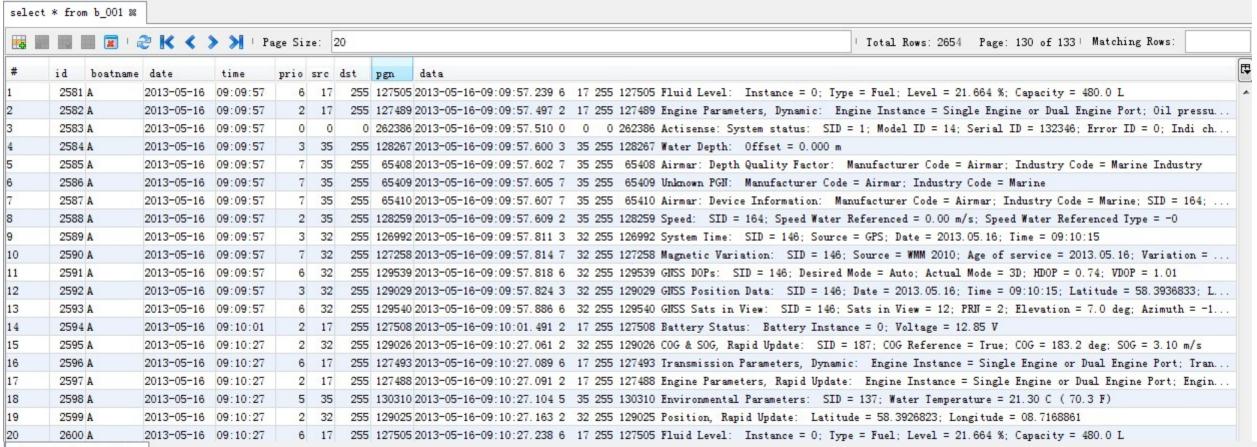
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2541	A	2013-05-16	09:08:57	6	112	255	127505	2013-05-16-09:08:57.077 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 6.172 %
2	2542	A	2013-05-16	09:08:57	6	17	255	127505	2013-05-16-09:08:57.077 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 21.664 %; Capacity = 480.0 L
3	2543	A	2013-05-16	09:08:57	0	0	0	262386	2013-05-16-09:08:57.078 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi ...
4	2544	A	2013-05-16	09:08:57	3	35	255	128267	2013-05-16-09:08:57.078 3 35 255 128267 Water Depth: Offset = 0.000 m
5	2545	A	2013-05-16	09:08:57	7	35	255	65408	2013-05-16-09:08:57.078 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
6	2546	A	2013-05-16	09:08:57	7	35	255	65409	2013-05-16-09:08:57.078 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
7	2547	A	2013-05-16	09:08:57	7	35	255	65410	2013-05-16-09:08:57.078 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 79;...
8	2548	A	2013-05-16	09:08:57	2	35	255	128259	2013-05-16-09:08:57.078 2 35 255 128259 Speed: SID = 79; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
9	2549	A	2013-05-16	09:08:57	3	32	255	126992	2013-05-16-09:08:57.079 3 32 255 126992 System Time: SID = 49; Source = GPS; Date = 2013.05.16; Time = 09:08:50
10	2550	A	2013-05-16	09:08:57	7	32	255	127258	2013-05-16-09:08:57.079 7 32 255 127258 Magnetic Variation: SID = 49; Source = WMM 2010; Age of service = 2013.05.16; Variation =...
11	2551	A	2013-05-16	09:08:57	6	32	255	129539	2013-05-16-09:08:57.080 6 32 255 129539 GNSS DOPs: SID = 49; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.91; VDOP = 1.23
12	2552	A	2013-05-16	09:08:57	3	32	255	129029	2013-05-16-09:08:57.080 3 32 255 129029 GNSS Position Data: SID = 49; Date = 2013.05.16; Time = 09:08:50; Latitude = 58.4019223; ...
13	2553	A	2013-05-16	09:08:57	6	32	255	129540	2013-05-16-09:08:57.080 6 32 255 129540 GNSS Sats in View: SID = 49; Sats in View = 12; PRN = 2; Elevation = 8.0 deg; Azimuth = -...
14	2554	A	2013-05-16	09:09:00	2	17	255	127508	2013-05-16-09:09:00.162 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.80 V
15	2555	A	2013-05-16	09:09:27	2	17	255	127488	2013-05-16-09:09:27.000 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Eng...
16	2556	A	2013-05-16	09:09:27	2	32	255	129026	2013-05-16-09:09:27.063 2 32 255 129026 COG & SOG, Rapid Update: SID = 89; COG Reference = True; COG = 177.9 deg; SOG = 13.02 m/s
17	2557	A	2013-05-16	09:09:27	6	17	255	127493	2013-05-16-09:09:27.092 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tr...
18	2558	A	2013-05-16	09:09:27	5	35	255	130310	2013-05-16-09:09:27.096 5 35 255 130310 Environmental Parameters: SID = 17; Water Temperature = 21.32 C (70.4 F)
19	2559	A	2013-05-16	09:09:27	2	32	255	129025	2013-05-16-09:09:27.163 2 32 255 129025 Position, Rapid Update: Latitude = 58.3964670; Longitude = 08.7166271
20	2560	A	2013-05-16	09:09:27	6	17	255	127505	2013-05-16-09:09:27.241 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 21.664 %; Capacity = 480.0 L



select * from b_001

Page Size: 20 Total Rows: 2654 Page: 129 of 133 Matching Rows:

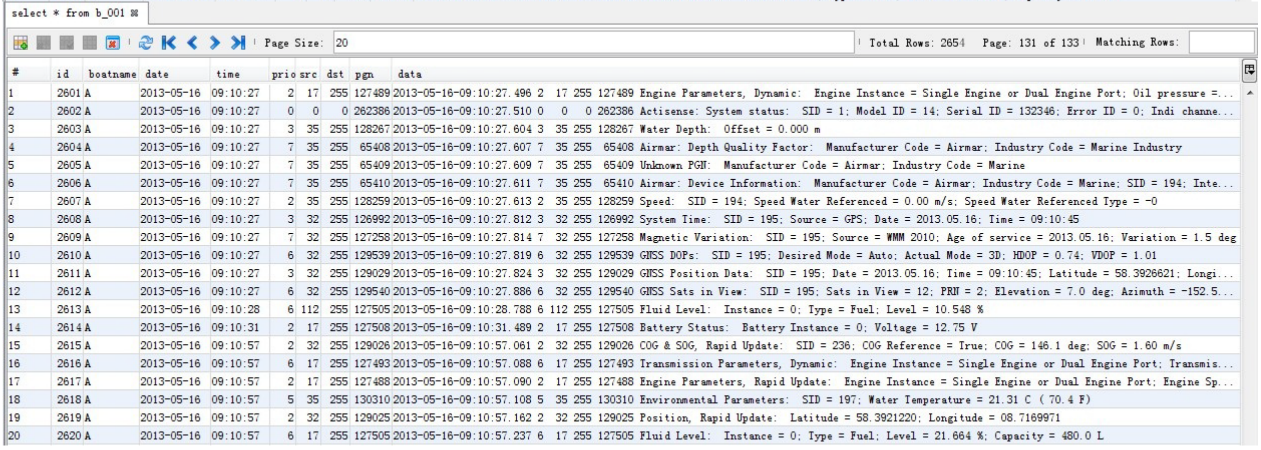
#	id	boatname	date	time	prio	src	dst	pgn	data
1	2561	A	2013-05-16	09:09:27	2	17	255	127489	2013-05-16-09:09:27.498 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil press...
2	2562	A	2013-05-16	09:09:27	0	0	0	262386	2013-05-16-09:09:27.513 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi c...
3	2563	A	2013-05-16	09:09:27	3	35	255	128267	2013-05-16-09:09:27.597 3 35 255 128267 Water Depth: Offset = 0.000 m
4	2564	A	2013-05-16	09:09:27	7	35	255	65408	2013-05-16-09:09:27.599 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
5	2565	A	2013-05-16	09:09:27	7	35	255	65409	2013-05-16-09:09:27.601 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
6	2566	A	2013-05-16	09:09:27	7	35	255	65410	2013-05-16-09:09:27.603 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 134;...
7	2567	A	2013-05-16	09:09:27	2	35	255	128259	2013-05-16-09:09:27.606 2 35 255 128259 Speed: SID = 134; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
8	2568	A	2013-05-16	09:09:27	3	32	255	126992	2013-05-16-09:09:27.811 3 32 255 126992 System Time: SID = 97; Source = GPS; Date = 2013.05.16; Time = 09:09:45
9	2569	A	2013-05-16	09:09:27	7	32	255	127258	2013-05-16-09:09:27.813 7 32 255 127258 Magnetic Variation: SID = 97; Source = WMM 2010; Age of service = 2013.05.16; Variation = ...
10	2570	A	2013-05-16	09:09:27	6	32	255	129539	2013-05-16-09:09:27.817 6 32 255 129539 GNSS DOPs: SID = 97; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.94; VDOP = 1.41
11	2571	A	2013-05-16	09:09:27	3	32	255	129029	2013-05-16-09:09:27.824 3 32 255 129029 GNSS Position Data: SID = 97; Date = 2013.05.16; Time = 09:09:45; Latitude = 58.3963791; L...
12	2572	A	2013-05-16	09:09:27	6	112	255	127505	2013-05-16-09:09:27.852 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 4.196 %
13	2573	A	2013-05-16	09:09:27	6	32	255	129540	2013-05-16-09:09:27.887 6 32 255 129540 GNSS Sats in View: SID = 97; Sats in View = 12; PRN = 2; Elevation = 8.0 deg; Azimuth = -1...
14	2574	A	2013-05-16	09:09:31	2	17	255	127508	2013-05-16-09:09:31.492 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.70 V
15	2575	A	2013-05-16	09:09:57	2	17	255	127488	2013-05-16-09:09:57.000 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engi...
16	2576	A	2013-05-16	09:09:57	6	112	255	127505	2013-05-16-09:09:57.002 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 2.624 %
17	2577	A	2013-05-16	09:09:57	2	32	255	129026	2013-05-16-09:09:57.060 2 32 255 129026 COG & SOG, Rapid Update: SID = 138; COG Reference = True; COG = 175.9 deg; SOG = 4.50 m/s
18	2578	A	2013-05-16	09:09:57	6	17	255	127493	2013-05-16-09:09:57.123 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tra...
19	2579	A	2013-05-16	09:09:57	5	35	255	130310	2013-05-16-09:09:57.124 5 35 255 130310 Environmental Parameters: SID = 77; Water Temperature = 21.32 C (70.4 F)
20	2580	A	2013-05-16	09:09:57	2	32	255	129025	2013-05-16-09:09:57.163 2 32 255 129025 Position, Rapid Update: Latitude = 58.3937121; Longitude = 08.7168966



select * from b_001

Page Size: 20 Total Rows: 2654 Page: 130 of 133 Matching Rows:

#	id	boatname	date	time	prio	src	dst	pgn	data
1	2581	A	2013-05-16	09:09:57	6	17	255	127505	2013-05-16-09:09:57.239 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 21.664 %; Capacity = 480.0 L
2	2582	A	2013-05-16	09:09:57	2	17	255	127489	2013-05-16-09:09:57.497 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pressu...
3	2583	A	2013-05-16	09:09:57	0	0	0	262386	2013-05-16-09:09:57.510 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi ch...
4	2584	A	2013-05-16	09:09:57	3	35	255	128267	2013-05-16-09:09:57.600 3 35 255 128267 Water Depth: Offset = 0.000 m
5	2585	A	2013-05-16	09:09:57	7	35	255	65408	2013-05-16-09:09:57.602 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
6	2586	A	2013-05-16	09:09:57	7	35	255	65409	2013-05-16-09:09:57.605 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
7	2587	A	2013-05-16	09:09:57	7	35	255	65410	2013-05-16-09:09:57.607 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 164; ...
8	2588	A	2013-05-16	09:09:57	2	35	255	128259	2013-05-16-09:09:57.609 2 35 255 128259 Speed: SID = 164; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
9	2589	A	2013-05-16	09:09:57	3	32	255	126992	2013-05-16-09:09:57.811 3 32 255 126992 System Time: SID = 146; Source = GPS; Date = 2013.05.16; Time = 09:10:15
10	2590	A	2013-05-16	09:09:57	7	32	255	127258	2013-05-16-09:09:57.814 7 32 255 127258 Magnetic Variation: SID = 146; Source = WMM 2010; Age of service = 2013.05.16; Variation = ...
11	2591	A	2013-05-16	09:09:57	6	32	255	129539	2013-05-16-09:09:57.818 6 32 255 129539 GNSS DOPs: SID = 146; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.74; VDOP = 1.01
12	2592	A	2013-05-16	09:09:57	3	32	255	129029	2013-05-16-09:09:57.824 3 32 255 129029 GNSS Position Data: SID = 146; Date = 2013.05.16; Time = 09:10:15; Latitude = 58.3936833; L...
13	2593	A	2013-05-16	09:09:57	6	32	255	129540	2013-05-16-09:09:57.886 6 32 255 129540 GNSS Sats in View: SID = 146; Sats in View = 12; PRN = 2; Elevation = 7.0 deg; Azimuth = -1...
14	2594	A	2013-05-16	09:10:01	2	17	255	127508	2013-05-16-09:10:01.491 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.85 V
15	2595	A	2013-05-16	09:10:27	2	32	255	129026	2013-05-16-09:10:27.061 2 32 255 129026 COG & SOG, Rapid Update: SID = 187; COG Reference = True; COG = 183.2 deg; SOG = 3.10 m/s
16	2596	A	2013-05-16	09:10:27	6	17	255	127493	2013-05-16-09:10:27.089 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Tran...
17	2597	A	2013-05-16	09:10:27	2	17	255	127488	2013-05-16-09:10:27.091 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engin...
18	2598	A	2013-05-16	09:10:27	5	35	255	130310	2013-05-16-09:10:27.104 5 35 255 130310 Environmental Parameters: SID = 137; Water Temperature = 21.30 C (70.3 F)
19	2599	A	2013-05-16	09:10:27	2	32	255	129025	2013-05-16-09:10:27.163 2 32 255 129025 Position, Rapid Update: Latitude = 58.3926823; Longitude = 08.7168861
20	2600	A	2013-05-16	09:10:27	6	17	255	127505	2013-05-16-09:10:27.238 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 21.664 %; Capacity = 480.0 L



select * from b_001

Page Size: 20 Total Rows: 2654 Page: 131 of 133 Matching Rows:

#	id	boatname	date	time	prio	src	dst	pgn	data
1	2601	A	2013-05-16	09:10:27	2	17	255	127489	2013-05-16-09:10:27.496 2 17 255 127489 Engine Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Oil pressure =...
2	2602	A	2013-05-16	09:10:27	0	0	0	262386	2013-05-16-09:10:27.510 0 0 0 262386 Actisense: System status: SID = 1; Model ID = 14; Serial ID = 132346; Error ID = 0; Indi channe...
3	2603	A	2013-05-16	09:10:27	3	35	255	128267	2013-05-16-09:10:27.604 3 35 255 128267 Water Depth: Offset = 0.000 m
4	2604	A	2013-05-16	09:10:27	7	35	255	65408	2013-05-16-09:10:27.607 7 35 255 65408 Airmar: Depth Quality Factor: Manufacturer Code = Airmar; Industry Code = Marine Industry
5	2605	A	2013-05-16	09:10:27	7	35	255	65409	2013-05-16-09:10:27.609 7 35 255 65409 Unknown PGN: Manufacturer Code = Airmar; Industry Code = Marine
6	2606	A	2013-05-16	09:10:27	7	35	255	65410	2013-05-16-09:10:27.611 7 35 255 65410 Airmar: Device Information: Manufacturer Code = Airmar; Industry Code = Marine; SID = 194; Inte...
7	2607	A	2013-05-16	09:10:27	2	35	255	128259	2013-05-16-09:10:27.613 2 35 255 128259 Speed: SID = 194; Speed Water Referenced = 0.00 m/s; Speed Water Referenced Type = -0
8	2608	A	2013-05-16	09:10:27	3	32	255	126992	2013-05-16-09:10:27.812 3 32 255 126992 System Time: SID = 195; Source = GPS; Date = 2013.05.16; Time = 09:10:45
9	2609	A	2013-05-16	09:10:27	7	32	255	127258	2013-05-16-09:10:27.814 7 32 255 127258 Magnetic Variation: SID = 195; Source = WMM 2010; Age of service = 2013.05.16; Variation = 1.5 deg
10	2610	A	2013-05-16	09:10:27	6	32	255	129539	2013-05-16-09:10:27.819 6 32 255 129539 GNSS DOPs: SID = 195; Desired Mode = Auto; Actual Mode = 3D; HDOP = 0.74; VDOP = 1.01
11	2611	A	2013-05-16	09:10:27	3	32	255	129029	2013-05-16-09:10:27.824 3 32 255 129029 GNSS Position Data: SID = 195; Date = 2013.05.16; Time = 09:10:45; Latitude = 58.3926621; Longi...
12	2612	A	2013-05-16	09:10:27	6	32	255	129540	2013-05-16-09:10:27.886 6 32 255 129540 GNSS Sats in View: SID = 195; Sats in View = 12; PRN = 2; Elevation = 7.0 deg; Azimuth = -152.5...
13	2613	A	2013-05-16	09:10:28	6	112	255	127505	2013-05-16-09:10:28.788 6 112 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 10.548 %
14	2614	A	2013-05-16	09:10:31	2	17	255	127508	2013-05-16-09:10:31.489 2 17 255 127508 Battery Status: Battery Instance = 0; Voltage = 12.75 V
15	2615	A	2013-05-16	09:10:57	2	32	255	129026	2013-05-16-09:10:57.061 2 32 255 129026 COG & SOG, Rapid Update: SID = 236; COG Reference = True; COG = 146.1 deg; SOG = 1.60 m/s
16	2616	A	2013-05-16	09:10:57	6	17	255	127493	2013-05-16-09:10:57.088 6 17 255 127493 Transmission Parameters, Dynamic: Engine Instance = Single Engine or Dual Engine Port; Transmis...
17	2617	A	2013-05-16	09:10:57	2	17	255	127488	2013-05-16-09:10:57.090 2 17 255 127488 Engine Parameters, Rapid Update: Engine Instance = Single Engine or Dual Engine Port; Engine Sp...
18	2618	A	2013-05-16	09:10:57	5	35	255	130310	2013-05-16-09:10:57.108 5 35 255 130310 Environmental Parameters: SID = 197; Water Temperature = 21.31 C (70.4 F)
19	2619	A	2013-05-16	09:10:57	2	32	255	129025	2013-05-16-09:10:57.162 2 32 255 129025 Position, Rapid Update: Latitude = 58.3921220; Longitude = 08.7169971
20	2620	A	2013-05-16	09:10:57	6	17	255	127505	2013-05-16-09:10:57.237 6 17 255 127505 Fluid Level: Instance = 0; Type = Fuel; Level = 21.664 %; Capacity = 480.0 L