Remote Connection of Diagnostic Tool

Master of Science Thesis in Communication Engineering

IRINA – ELENA APETRI
ALI RAZA

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2011
Report No. EX066/2011

Remote Connection of Diagnostic Tool
Irina-Elena Apetri and Ali Raza

Technical report no EX066/2011 Department of Signals and Systems
Chalmers University of Technology
SE-41296 Gothenburg
Sweden
Telephone +46-(0)31 772 1000

Department of Signals and Systems
Gothenburg, Sweden 2011

Contents

List of Tables

List of Figures

Abstract

Most of the electronic systems of today's vehicles are controlled via ECUs (Electronic Control Units). These can be programmed to perform in a specific way and they also store the DTC (Diagnostic Trouble Codes) generated in case of system's malfunction. With the aid of a computer application, the engine mechanic can access these diagnostic codes, identify faults, perform parameter programming and update the ECUs with software patches.

The present thesis was conducted at Volvo Penta Global Aftermarket Technical Support department, Göteborg. Its aim was to implement a prototype for connecting the existing service tool to a remotely located vessel through a wireless communication link and to investigate system's requirements in terms of latency, bandwidth, protocols and software/hardware platforms.

The prototype comprises of three software applications - an Android, a Windows mobile and a desktop application, developed in Eclipse and Visual Studio 2008 IDEs using Java and C++ programming languages. The thesis describes the design, implementation and evaluates the performance of the obtained system which serves the purpose of establishing a logical link between the remote engine and the diagnostic tool.

Acknowledgements

This paper is a master thesis for the Communication Engineering Master program at Chalmers University of Technology. It has been carried out at Volvo Penta, Global Aftermarket Department, Göteborg.

We would like to express our gratitude to our examiner and supervisors, for all their efforts in guiding us throughout the development of this thesis: Per Nestvall, Esbjörn Sjöberg, Adam Ståleby, Petter Andersson, Henk Wymeersch and Jingya Li.

Irina Apetri, Ali Raza, Göteborg, June 2011

Chapter 1

Introduction

1.1 Background of remote diagnostics in marine industry

Remote diagnostic refers to the concept of resolving an issue from distance. It includes connecting, diagnosing and programming the control unit of modern vehicle remotely. According to market analysts Frost & Sullivan, the term remote diagnostics refers to “the ability of accessing, through a wireless network, the vehicle’s performance and trouble codes in case of malfunction and, in case needed, to provide the necessary service support” [1].

Most modern vehicles of today have Electronic Controlled Units (ECUs) installed that control one or more electrical systems or subsystems of that vehicle. ECUs are interconnected via the serial bus and can be programmed to perform in a specific way. Today around 70% of the boat operations are driven by software, a fact that indicates the growing importance of software components. The ECU stores the Diagnostic Trouble Codes (DTCs) which are generated when a fault or break down occurs in the system. With the aid of a computer application, the repair or maintenance mechanic at workshop reads these diagnostic codes, displays status parameters, performs parameter programming and updates the ECUs with software patches.

Even though a local service technician represents a determining factor in problem’s definition, by providing an expert technician real time access to the vehicle electronics eases the process of solving most of the ECU-related problems. Therefore the possibility of a remote connection will not only add a new feature to the current diagnostics tool but may prove to be an efficient method in terms of saving time and money.

1.2 Thesis Objectives and Scope

We conducted our research at Volvo Penta, a leading manufacturer and supplier of complete drive systems for marine and industrial applications. Volvo Penta’s engines are widely used in leisure or commercial vessels and in several industrial off-road applications. As mentioned previously, most engines are now equipped with advanced electronic systems which gather data with the scope of enhancing engine's performance or monitoring its under load behaviour.

Computer based diagnostic is often employed in the service process in order to facilitate the maintenance, problem tracking and repair of an engine equipped with sophisticated electronic systems. The "Volvo Diagnosis Application" (VODIA) is Windows based software developed by Volvo IT. Its purpose is to perform engine diagnostic, troubleshooting, calibration and tests. Given the high on-site usage and mobility requirements of a diagnostic tool, the VODIA was implemented on a mobile platform. This ensured a high flexibility and portability.

Currently, the VODIA tool communicates with the engine via SAE J1708/J1587 data bus and it requires less than one minute for performing a complete engine scan or for accessing a complete history of engine's operation. However, if the engine mechanic is not able to identify the source of a problem, the case must be forwarded to an expert technician; in the case of a serious issue, an on-site visit may be as well required.

Given the new emerging technologies in the area of electronics and mobile communications (embedded systems, 3G/4G) it is now possible to perform the same diagnostic operations remotely, without the need for expensive on-site visits and reducing considerably the time allocated for solving a certain issue. Nevertheless, the benefit justification for a remote diagnostic system holds also for the customers who are permanently looking for inexpensive and reliable service support.

The aim of the project is to develop a prototype for connecting the existing Volvo Penta's service tool to a remotely located vessel through a wireless communication link i.e. GSM/3G. This would provide the expert at Volvo Penta with the possibility of performing diagnostic tests remotely (i.e. read error codes, perform ECU or vessel configuration test etc.) and further help him to effectively solve the customer's issues in a relatively short amount of time. The thesis should also investigate the systems' needs in terms of bandwidth, coverage and protocols as well as the selection of mobile network operator, hardware platforms and specify/execute potential software changes.

The proposed solution should make use, as much as possible, of the existing diagnostic setup with minimal additional hardware requirements. Moreover, the prototype should have a good design and be flexible to future developments.

Chapter 2

Theoretical Background

2.1 Introduction to SAE J1708/J1587

J1587 is an automotive diagnostic protocol for heavy duty and medium duty vehicles which serves the purpose of off-vehicle data communication. It was developed by the Society of Automotive Engineers (SAE) and defines the messages that are transmitted within a J1708 network. With respect to OSI model, SAE J1587 provides the transport and application layers functionality while SAE 1708 serves primarily as physical and media access control layer (see Figure **??**)

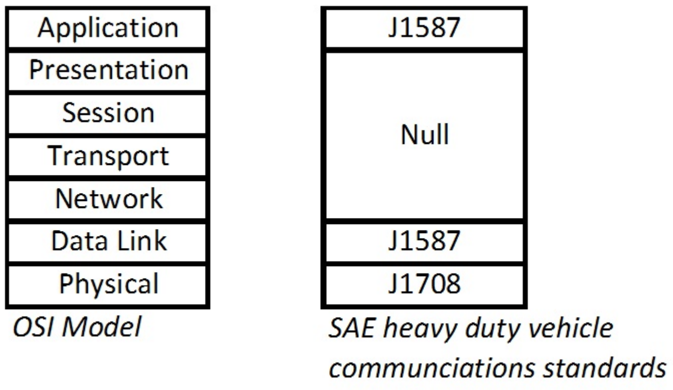


Figure 2.1: OSI reference model and the SAE standards

The format of J1708 messages is shown in Figure 2.2. Within every packet, the first field corresponds to the Message Identification (MID) byte, used to identify uniquely the sending node. The J1708 standard specification defines the MID range of 0-127. This field is followed by maximum 19 bytes of data. Exceptions to this length may be made when the vehicle is not moving. A checksum – computed as the two's complement of the sum of the MID and the data characters – represents the last field of the message.

MID	Data Characters (max 19 bytes)	Checksum

Figure 2.2: The format of a J1708 packet as defined in [2]

Note that there is no byte reserved in the message format for the length of the data characters. The only way for a receiving node to know that a message is completed is by detecting when the bus has been in idle state for at least 10 bit times. J1708 bit time is set to be 104.17 ms ±500ns [2] and this corresponds to a baud rate of 9600 bps. Each message is assigned a priority on a scale of 1 to 8 and the application defines the circumstances in which the priority is assigned. The most critical message has a priority of one.

J1587 is a networking specification which defines message and data format that is used on J1708 network. The specification defines the message format for message identifier (MID) ranging from 128-255. Every transmitter in the system has a unique MID. The message format is shown below.

MID	Parameter #1	Parameter #2	Parameter #3	...	Parameter #n	

Figure 2.3: SAE J1587 message format [3]

MID 128 (engine ECU) and MID 172 (off board diagnostic computer tool) are two frequently used MIDs by engine development departments. The MID is followed by one or more parameters where each parameter block starts with one or two byte Parameter Identifier (PID). The length and content are defined in parameter definitions which are stated in J1587 specification. J1587 transport protocol allows transfer of 17 to 239 bytes of data to be split into 21 byte messages (packets). J1587 uses messages for diagnostic purpose e.g. reading out fuel economy, coolant temperature and diagnostic trouble codes etc. All together there are around 300 parameters defined in the specification.

2.2 RS-232 Standard

Serial port (RS232) standard was first published by Electronic Industries Alliance in early 1960. It specifies the protocol for information exchange, signal voltage, signal timing and mechanical connectors. Typically, every standard PC is equipped with a serial port and provides a bidirectional asynchronous communication link. The data transfer begins with transmitter and receiver first negotiating the actual data bits and baud rate for the communication session. Devices usually transmit using 7 or 8 data bits. After the data bits are transmitted a stop bit is sent. Additionally, an optional bit, called the parity bit, is added and serves the purpose of error detection. The standard supports a data rate of 20,000 bits/sec with usual baud rate limit of 19,200. However commonly used values are 300, 1200, 2400, 9600 and 19200 baud.

2.3 Bluetooth Specifications and Protocol Stack

Bluetooth is a short range radio link referenced under IEEE 802.15.1. It operates in the 2.4 GHz ISM frequency band and enables wireless connectivity between mobile devices. The key features include robustness, low complexity, low power and low cost [4]. It is a packet based protocol with master-slave structure. The current specification allows up to seven slave devices to communicate with a master radio in one device and this configuration is called a piconet. At physical layer the technology uses Frequency Hop Spread Spectrum (FHSS) with 79 frequency channels, each being 1Mhz wide.

While Bluetooth does not have exactly equivalent protocols to TCP/IP or UDP, it has the RFCOMM protocol which provides roughly the same reliability and service as TCP (see Figure 2.4). According to Bluetooth's specification, RFCOMM protocol emulates serial and USB ports over the Logical Link Control and Adaption protocol (L2CAP). It provides reliable data transfer, simultaneous connections and flow control. The transport protocol L2CAP can provide connectionless and connection oriented service and it depends on the application how it configures the connection semantics. Usually connection oriented channels, which are similar to TCP, are used.

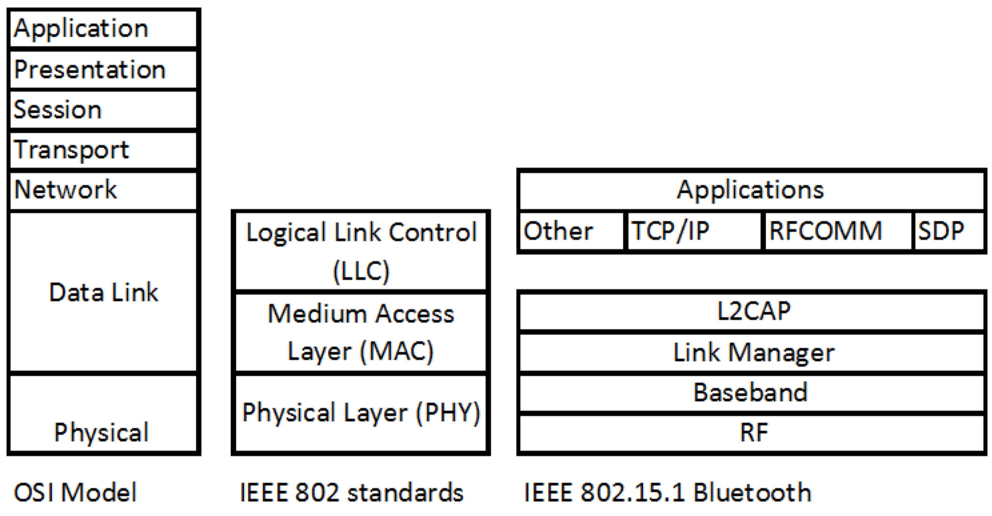


Figure 2.4: OSI reference Model, IEEE 802 family of standards and Bluetooth stack

The major difference form programming's point of view between TCP and RFCOMM is the choice of port number; TCP supports up to 65535 open ports on a single machine whereas RFCOMM supports 30. Due to limited number of ports, a given application may have port collision with other applications. In order to resolve this issue Service Discovery Protocol (SDP) is used, which allows ports to be assigned at run time. The procedure begins with Bluetooth SDP server searching for the services offered by the devices found in its vicinity. Every service is identified by its Universal Unique Identifier (UUID), a 128 bit number assigned to the client and server application at design time.

Currently there are several versions for the Bluetooth specifications. Version 1.0 offers a maximum bandwidth less than 1Mbps for a range of about 10 meters. The range depends on the power class of the radio device e.g. the device can operate at a range of 10 meter or even 100 meter with enhanced transmitters. Version 2 offers a maximum bandwidth of 2-3 Mbps.

Moreover, many different implementation of Bluetooth stack also exist. Bluetooth Protocol Stack provides a software API that is used to configure and initialise the drivers, allowing the device to communicate with other devices over a Bluetooth connection. Some common examples include Widcomm, Microsoft Windows stack, Bluetooth Toshiba stack and BlueZ for Linux. For our applications we used HP iPAQ which has built in Widcomm Bluetooth stack and Android 2.1 which has Linux based BlueZ Bluetooth stack.

2.4 Network Programming

2.4.1 Concepts related to socket programming

Sockets were initially created as part of the BSD UNIX operating system and later appeared in Linux or other BSD systems (also, Microsoft adapted a version known as Windows sockets for their systems). Hence several types of sockets exist, the difference between them being made by the topology of communication and the addresses used.

The most common socket type is called Internet socket and it can be defined as an end-point of a communication link between two processes residing on different stations, across an IP-based computer network. A pair of connected sockets represents a communication interface between the processes or threads and data exchange takes place based on a combination of local and remote IP addresses and port numbers.

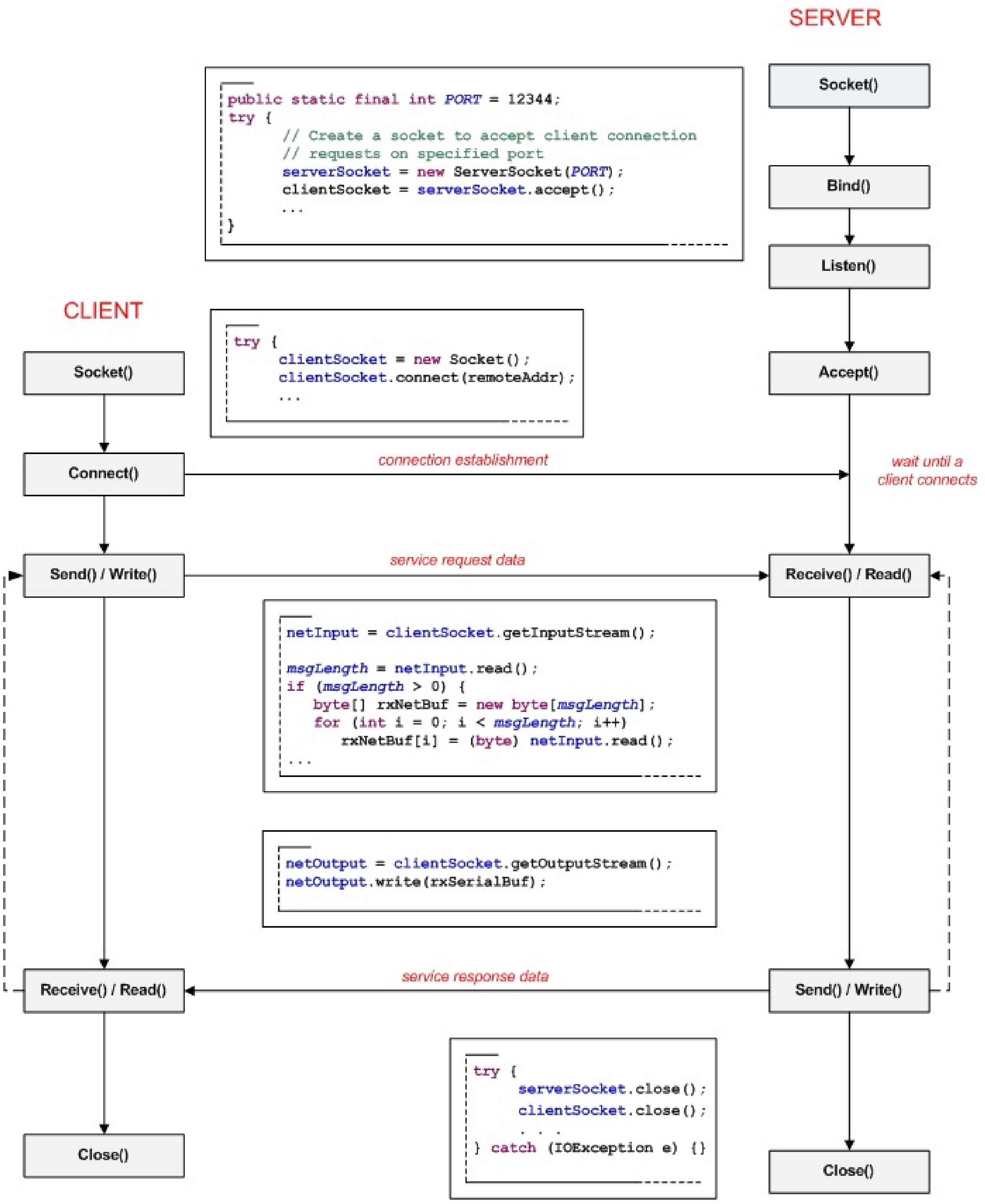
2.4.2 The client server model

All the operations within a network can be regarded as client processes that communicate with server processes. The server process creates a socket, associates to that socket an address and launches a mechanism to listen the connection requests coming from clients. Reversely, the client process creates a socket and asks for permission to connect to the server. In case the server accepts the request and the connection is established, the server and the client can communicate freely.

This model can vary with the nature of the designed applications: it can be symmetric, asymmetric, based on simplex or duplex communication. However, most of the applications are based on the scenario given in Figure 2.5. Note that this scenario is suitable only for TCP applications and the algorithm to create such an application is the following:

- **TCP client:** the first step is to create a socket using the **socket()** system call and to establish a connection to the remote server - identified by an IP address and port number - by calling the **connect()** function. Once the server accepts the connection request, messages may be exchanged between the communicating parties using **send()** and **receive()** or **write()** and **read()** system calls.

- **TCP server**: initially, a socket is created using the same **socket()** function call. Next, the given IP address and the corresponding port are binded to the socket with **bind()** system call. From this moment, the server listens for incoming connection requests. The call of **accept()** function creates a new socket over which the client and the server communicate. Multiple clients can connect to the same port on the server at the same time. Incoming data is distinguished by the client host and port from which it came. No more than one server socket can listen to a particular port at one time. Therefore, since a server may need to handle many connections at once, server programs tend to be multi-threaded. Generally the server socket listening on the port will only accept the connections. It then passes off the actual processing of connections to a separate thread.



```
public static final int PORT = 12344;
try {
        // Create a socket to accept client connection
        // requests on specified port
        serverSocket = new ServerSocket(PORT);
        clientSocket = serverSocket.accept();
        ...
}
```

```
try {
       clientSocket = new Socket();
       clientSocket.connect(remoteAddr);
       ...
```

```
netInput = clientSocket.getInputStream();

msgLength = netInput.read();
if (msgLength > 0) {
    byte[] rxNetBuf = new byte[msgLength];
    for (int i = 0; i < msgLength; i++)
        rxNetBuf[i] = (byte) netInput.read();
...
```

```
netOutput = clientSocket.getOutputStream();
netOutput.write(rxSerialBuf);
```

```
try {
     serverSocket.close();
     clientSocket.close();
     . . .
} catch (IOException e) {}
```

Figure 2.5: Client/Server applications scenario based on connection oriented protocols (Socket stream TCP)[5]

2.4.3 Stream sockets vs. datagram socktes

Different types of sockets correspond to different protocols. The main types of sockets used today are stream sockets and datagram sockets. The stream sockets are secure communication flows (with no errors), full duplex, based on the TCP protocol, which provides a sequential, errorless data transmission. A considerable advantage of TCP is that it retransmits the sequences that were not received correctly; the messages are sent sequentially thus a reordering of the packets is not necessary at the destination.

Datagram sockets are also known as connectionless sockets, use IP addresses for routing, but UDP as transport layer protocol. These sockets do not keep the connection opened during the communication session and they make the transfer packet by packet. Also, UDP sockets do not have to be connected before being used.

A difference between UDP sockets and TCP sockets is that UDP sockets preserve message boundaries. Consequently, receiving a message might be simpler than with TCP sockets. Another difference is that in case of using UDP, there is no guarantee that a message will arrive at destination and it also can be delivered in a different order than they were sent. Thus, in case of a client request, the request might be lost, duplicated, delayed or sent to a different destination [5].

2.5 GPRS/3G Network

With increasing user demand for higher data rate, the mobile network technology has evolved significantly from 2G to 3G and now 4G (LTE).

GPRS is a packet switched service capable of providing data rate of 56kb/sec and up to a maximum of 114kbps. Practically speaking, it offers data rate of 40-60kbps download and 20kbps upload. EDGE, which is an enhancement to GPRS network, uses 8PSK encoding and increases the capacity for GPRS and GSM network by 3 fold. IMT-2000 or 3G offers a minimum of 2Mbps for stationary users and 384 kbps for moving vehicles. According to ITU's (International Telecommunication Union) requirements, 4G, which is an all IP packet switched network, offers a maximum of 100Mbps for high mobility and 1Gbps for low mobility.

Having said this, from our application development point of view it does not matter if the android phone is connected to GPRS or 3G network. In our case where the data rate is at max 38.4kbps the effect is not significant. Latency which is one of the crucial factors will depend on the physical location of the remote vessel rather the type of network connection.

Signal coverage provided by network operators is now well established all over the world, so having an access to the network should not be a problem. Costs are operator dependent and charges may apply only for keeping the connection alive for the application. The data transfer is usually charged for each megabyte of traffic being transferred.

2.5.1 IP Address Issues with GPRS/3G remote monitoring and diagnostics

GPRS/3G communication technology provides an efficient and cost effective solution for remote monitoring and diagnostics without having to hard-wire the devices to central control unit. However, the issue of dynamic IP address needs to be taken into consideration. When a device connects to mobile data network, the operator usually assigns it a private and dynamic IP address. This private IP address means that the device is operating behind a NAT (Network Address Translation) table and is not visible over the Internet; in the case of an application like the one described in this thesis, the communication may be initiated only in one way – thus, the party having assigned the private IP – namely the customer - will be able to contact the service.

Several solutions exist to address this issue:

- **Public static IP address**: Some network operators offer the possibility to assign a static IP address to a specific SIM card. From system's perspective the client and the server both know the IP address of each other thus they can either initiate or wait for a connection request. This option is relatively expensive and some network operator may not provide this facility.

- **Virtual Private Network (VPN)**: This service provides two features: 1) Group mobile devices on a private network 2) Increased security. When the server and the client connect to the mobile network they are assigned a private IP address and placed on the same network segment. This allows them to establish a bidirectional communication link. The VPN tunnel is said to be secured or trusted because the data is encrypted and routed through a path that has specified properties. Mobile virtual network operators (MVNO) are companies that do not have their own telecom infrastructure or licensed frequencies but they offer services of setting up and maintaining a VPN by renting out a small pool of IP addresses that they have acquired from other network operators. Some countries do not have MVNOs and some carriers do not support VPN.

- **Dynamic domain name server (DDNS)**: Having a static IP address is expensive and sometimes the operator does not provide this facility. DDNS offers an alternative solution by assigning a domain name to the device when it registers itself with the server. Also, whenever the IP address of the device changes, the DDNS client residing on the device, sends an update to the server with the new IP address. Thus it allows the control centre to access the device with the same host name while the DDNS does the address translation. The mapping table in DDNS is updated every time it receives a new IP address from the device. The difference between DDNS and DNS is that DDNS takes care of dynamic IP address while DNS operates only on static IP address. Free servers are available with basic functionality but dedicated servers can be used with monthly subscription. Some devices do not support DDNS client but Android provides applications that offer this functionality [6].

Once the device is connected another issue may be a possible 'disconnection' from the mobile network. This may happen due to the following reasons:

- **Timeout**: If the connection is idle for some time, the user is disconnected from the network. The service provider has limited recourses thus, to optimize the bandwidth efficiency, it is important to free system's resources when they are not in use. The idle time is around 10 to 15 minutes but it also depends on the network operator and the time of the day.

- **Quality of Service (QoS)** required: A temporary dropout from the network because either more customers or a customer with higher priority started operating in the cell; to maintain the Quality of Service a temporary dropout from the network may happen.

- **Device losses signal strength**: Wireless channel is time varying thus if a device losses its signal strength or connection to the network the session closes and the connection has to be initialized again.

In our system test and analysis we used data connection SIM card from Tele2 service provider. Tele2 offers a public IP address to the devices connected to their network. As our primary objective was to develop and analyze the remote diagnostic system in terms of latency, bandwidth and coverage we opted for public IP address option because it facilitated us to perform our analysis without setting up our own server or using a third party service.

2.6 Mobile Application Development using Android and Windows CE/Mobile

2.6.1 Basic concepts related to Android framework

Android is an open-source project led by Google. Driven through Apache License, the platform consists of an operating system and a Software Development Kit (SDK) for mobile devices. The platform includes an emulator, debugging utilities, memory and performance profiling tools and a strong community for source code and set of example programmes [7]. The SDK is available to anyone who wants to develop their application to run on Android operating system (OS). The SDK uses Java programming language in combination with Extensible Markup Language (XML). The development framework can be broadly categorized in four layers as shown in Figure 2.6.

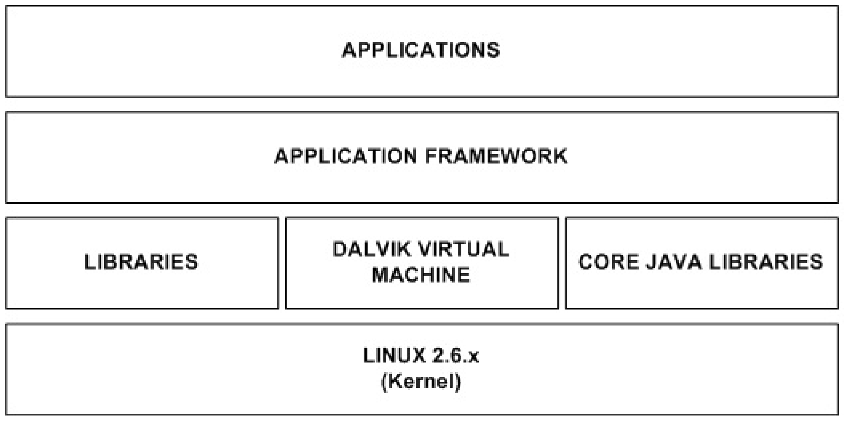


Figure 2.6: Android Framework overview

Application Layer: It is here that the complied version of the application resides. By default every application in Android OS runs as separate Linux process and is assigned a unique user ID at the time of installation. This user ID is verified when the application tries to access system resources and serves the purpose of security and also preventing any application to impact user experience or associated data or system files. To share application space or components with other applications explicit permission has to be provided in the AndroidManifest.xml file (discussed in section 5.1).

Framework Layer: It is here that the building blocks of the application are created using the underlying system libraries and services. The five basic building components of an application are:

- Views: It manages layout and events of user interface when an activity launches.
- Content providers: The API enables to share, store, retrieve, modify and manage application data.
- Resource manager: Deals with application non-code related files e.g. icons, images, audio files and layout. It provides a way to optimize application for range of device configurations.
- Notification mangers: Provide alerts to the application when events happen on the device. It allows the application to bind to certain events that launch an activity if required.
- Activity manager: It manages the life cycle, resources and activity state of the application and is discussed in more detail in the implementation chapter.

Library and Runtime Layer: This layer holds core system libraries (libc, SSL, OpenGL), Google Java libraries and the Dalvik Virtual Machine, a memory-optimized runtime environment, to run multiple instances efficiently.

Linux Kernel: Android uses Linux kernel version 2.6.x for its device drivers, memory management, process management and networking [7].

2.6.2 Basic Concepts related to Windows CE /Mobile .NET framework

Windows Embedded CE is a 32-bit operating system developed for minimalistic computers and embedded systems. Even though at the beginning most of its features were similar to Windows 95, Windows CE is not a scaled-down version of desktop Windows but a distinct OS and kernel. It is currently supported on Intel x86, MIPS, ARM and Hitachi SuperH processors.

According to [8], “Windows CE conforms to the definition of a real-time operating system, with deterministic interrupt latency. It supports 256 priority levels and uses priority inheritance for dealing with priority inversion. The fundamental unit of execution is the thread. This helps simplifying the interface and improves execution time”. Several platforms have been based on the core Windows CE operating system (Pocket PC 2000/2002, Windows Mobile 2003, Windows Mobile 2003 SE, Windows Mobile 5.0, Windows Mobile 6, Smartphone 2002/2003) along with many industrial devices and embedded systems. Also, unlike other Microsoft operating systems, large parts of Windows CE are offered in source code form not only to vendors but also to the general public.

The 6th release of Windows Embedded OS targets from enterprise specific tools such as industrial controllers to consumer electronics devices. The new redesigned kernel has raised the process limit from 32 (previous versions) to 32000, each process receiving now 2GB of virtual address space (from 32MB). Other new features of Windows Embedded CE 6.0 are mentioned in [8]out of which we underline “the support for multiple radio and 802.11i (WPA2) and 802.11e (QoS) wireless standards as well as the Cellcore Voice components that enable devices to make data connections and initiate voice calls through the cellular networks. Also, Windows Mobile devices are based on Windows CE kernel but due to additional licensing and testing most of the custom devices use Windows CE instead of Windows Mobile” [8].

Regarding the development tools suitable for designing, creating, building, testing and debugging customized Windows Embedded applications, those are provided by Platform Builder together with Visual Studio IDE. Both native (C/C++) and managed (C#, VB.NET) development are supported on Windows CE/Mobile.

Chapter 3

System Analysis

3.1 Current service process at Volvo Penta's Aftermarket Technical Support

The organization has a central support structure and one of industry's largest dealer networks. Whenever a customer experiences a problem with his engine he contacts the closest dealer representative (1st line support level) for the repair. The service process is carried out by a local mechanic who, based on the stated issue, its possible causes, the diagnostic data and his own experience, proposes a suitable solution. Most of the problems find their answer here and the unsuccessful cases are redirected to specific Market Units, companies owned by Volvo Penta that are in charge of all dealerships.

In case the problem is not solved at the Market Units' side (2nd line support level), the case is further forwarded, to the Subject Matter Experts (Back Office). An expert will try to understand the problem, but in most cases he has to request for more information. With the new engine series having extended electronic components, a complete description is even more difficult to obtain. Thus, by having remote access to the vessel, the expert at Volvo Penta will have the possibility to access the diagnostic codes and solve the issue more efficiently.

3.2 New Service Process

We are willing to design a system that will serve as a prototype for a remote diagnostic tool that is to be employed in the service process; the tool may be used between Market Units and the Service Matter Experts in order to allow the latter to perform certain diagnostic tests or parameter programming without the need of an on-site visit.

From a company's perspective, remote service support will allow part of the configuration, support and troubleshooting tasks to be performed without involving expensive on-site visits.

Also, remote diagnostics may decrease the warranty costs and the time allocated for solving a certain issue. Nevertheless, data collection is possible as well as full access to a complete history of events regarding a certain vessel's functioning – which can be further used in the development of new, more reliable products.

Chapter 4

System Design

4.1 Basic Idea

Technical service and support is not always accomplished only through a series of questions and answers between the customer and service engineer, but it often involves sending the service engineer to a customer site for better diagnosis and problem solving. However, the on-site visits required servicing a given product, and implicitly the time and costs associated to them, may be significantly reduced if a remote diagnostic tool is used. This implies that the operations usually performed during an on-site visit (parameters inspection, error codes, performance testing) are to be carried out from a centralized location, over the Internet.

In this section are described and analyzed some of the possible implementation scenarios for the future remote diagnostics tool. The solution chosen to serve as a prototype takes into consideration the current software and hardware constraints and aims at performing close to the tool used today in the service process. Nevertheless, the system is to be built on the current one, employing the already available equipment and software applications while keeping the additional hardware requests to minimum.

The implementation scenarios discussed in the followings are based all on the same idea – the establishment of a logical link between a remote engine and the VODIA tool located at a centralized site. Hence, access to the remote engine is to be provided still through the current diagnostic tool and the steps that are performed during a regular diagnostics procedure will remain unchanged. However, the service request data leaving the VODIA PDA will not reach directly the engine but rather a computer connected to the Internet. The purpose of the computer will be to forward the requests to a device across the Internet (another computer, smart-phone, and/or PDA) which will further convey them to the engine. Since the process is bidirectional, service response data will be forwarded in the same manner from the engine back to the VODIA tool.

4.2 Proposed System Architectures

Several configurations were taken into account for implementing the prototype. A possible solution would have been the one shown in Figure 4.1. At the remote location, the communication between the PC and the engine is made through the Volvo 88890020 interface, using USB or WLAN 802.11b/g standards. The PC may further send the captured information to a central office, where another PC is used to deliver it to the VODIA Tool. From one point of view, this configuration owns the advantage of extra processing power – due to the PCs involved at both ends – and ease of implementation – only simple client/server applications are required -. On the other hand, this configuration is not highly portable since an engine mechanics must always carry a laptop instead of a Pocket PC.

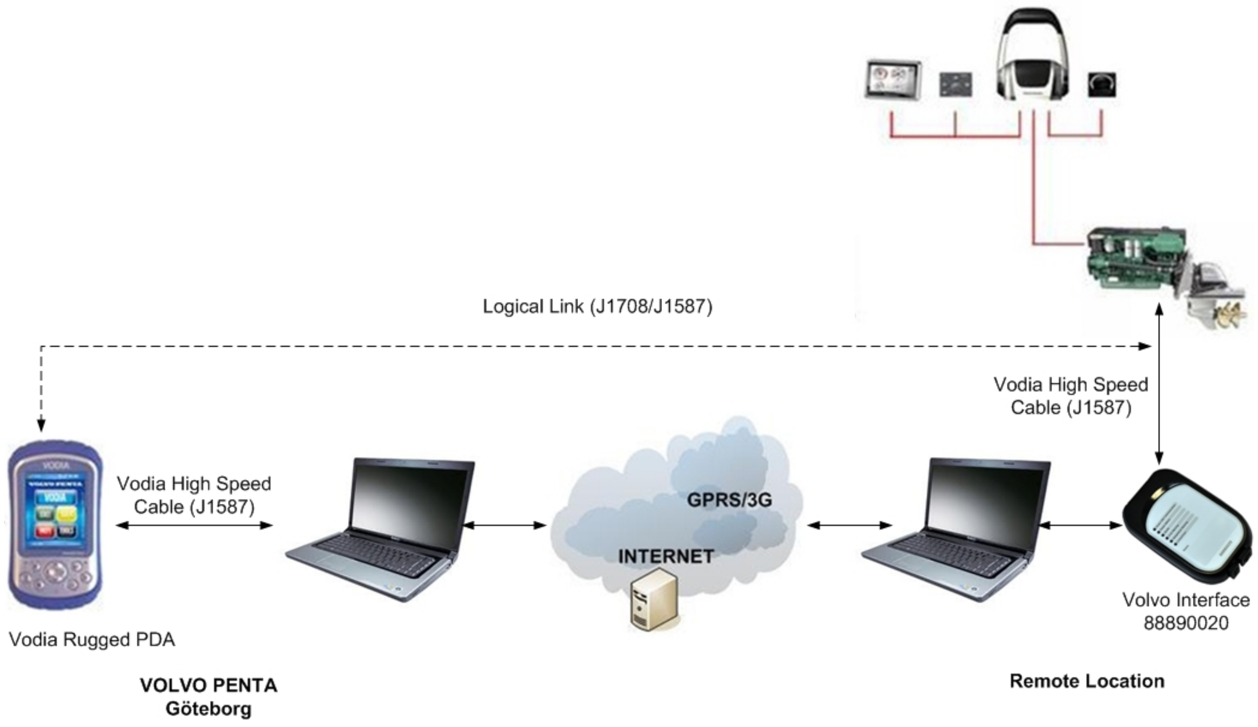


Figure 4.1: Possible architecture for the remote diagnostic system's prototype

In order to fulfill the portability requirement, a configuration like the one in Figure 4.2 was also taken into consideration. Similar to the previous case, the only way of interconnecting the engine and the smartphone is through the interface adapter. Unfortunately, there is no possibility for an Android application to interact with other hardware via USB (there is neither any USB support for input devices or APIs that would enable the communication with other USB peripherals). Also, ad-hoc mode is supported only starting with Android 2.2 (the platform available on the smartphone is Android 2.1, which has no support for ad-hoc mode).

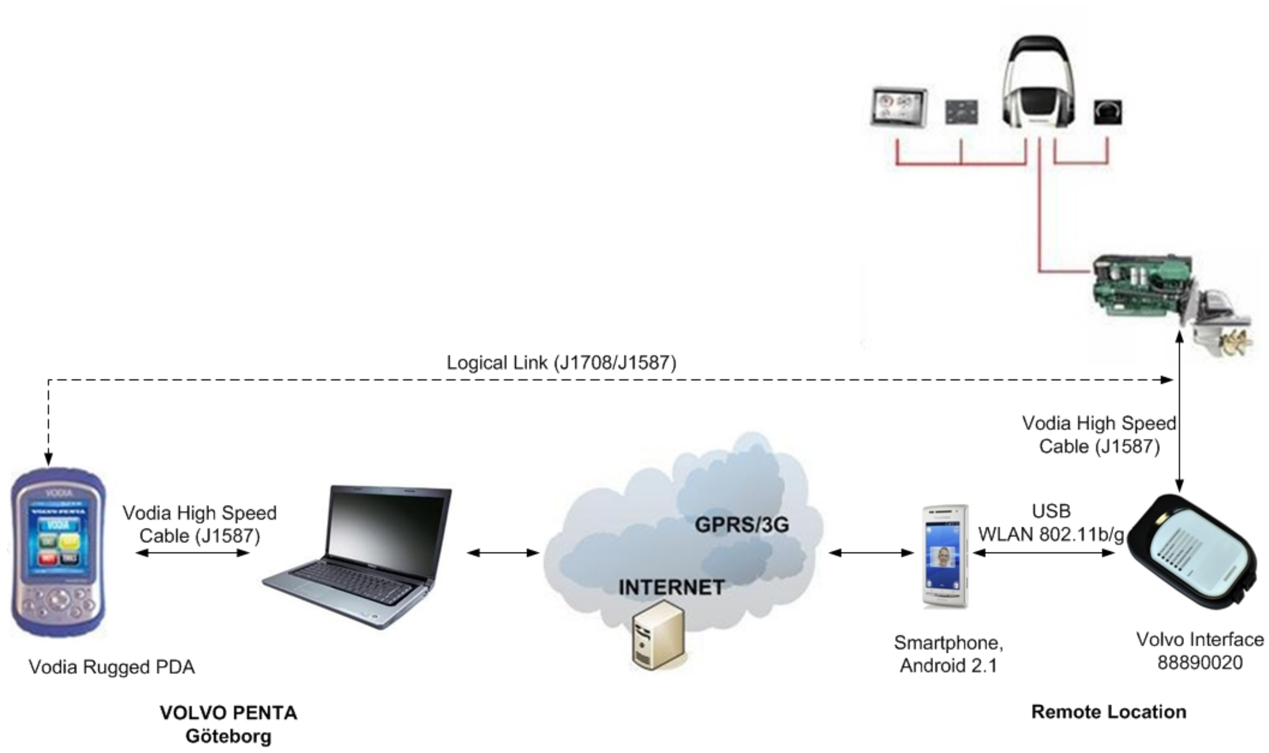


Figure 4.2: Possible architecture for the remote diagnostic system's prototype

Given the limitations raised by the previous configurations, the interface adapter was replaced in the following setup with another PDA. This ensures that the J1587 messages received from the engine will reach the smartphone via Bluetooth wireless interface. Further on, the phone will send the received packets to the remote location. The setup shown in Figure 4.3 applies to a general case scenario and is easy to be implemented. The only constraint is the mobile Internet connectivity.

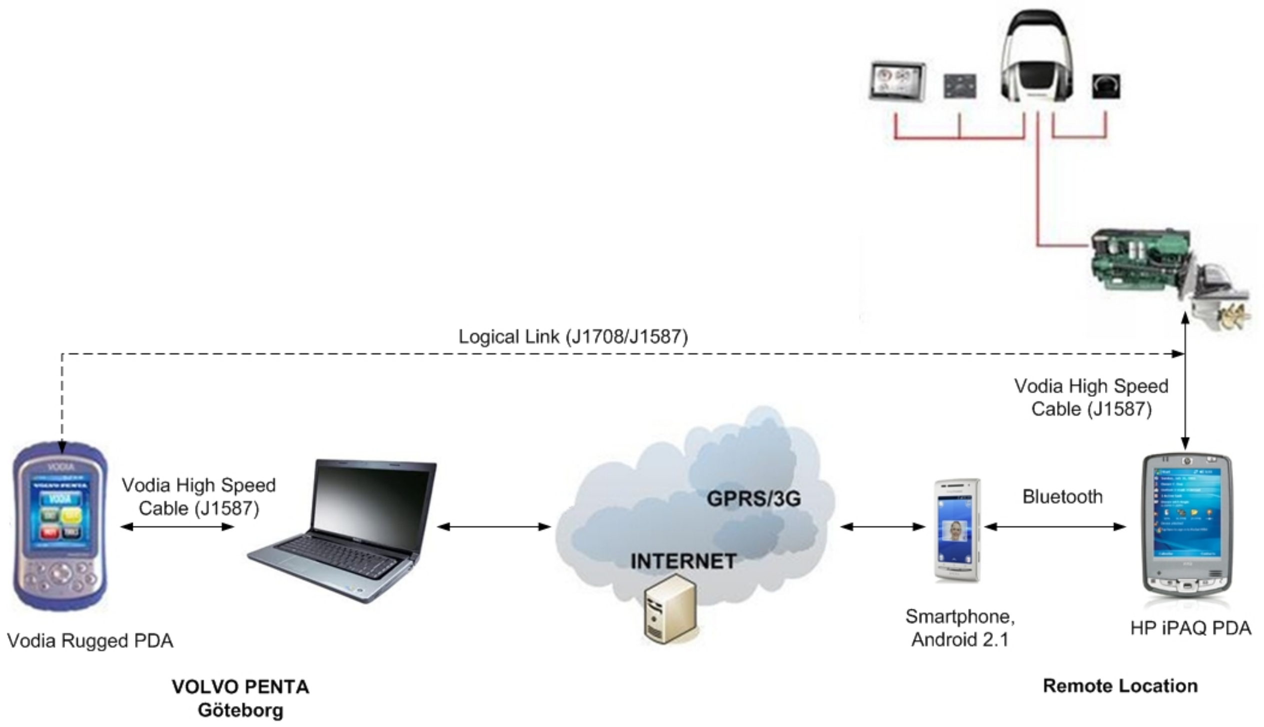


Figure 4.3: Possible architecture for the remote diagnostic system's prototype

4.3 Selected Solution

Three possible set-ups for the remote connectivity of the diagnostics tool were proposed and analyzed in the previous subchapter. The selected solution (Figure 4.3) together with its possible implementation is described in the followings.

As stated before, by creating a logical link between the remote location and the central site, the diagnostic tool will work just as if it had been be directly connected to the engine. However, given the number of intermediary devices that process the received information and the latency introduced by the network, the setup for remote diagnostics is expected to exhibit a certain delay compared to the original one (note that, the current tool needs less than one minute to scan the engine, to identify faults or to access a complete history with engine's operation). Please refer to Chapter 6 for an exact comparison between the time required for the current tool to perform certain tasks and the results obtained by the remote diagnostic system.

4.3.1 Motivation

Out of the presented solutions, the one described above is the most suitable for a general case scenario, where the only way of establishing a data transfer is through the mobile network.

Moreover, we have considered building a prototype that employs the already available device used today in the service process (the VODIA rugged PDA) without requesting for additional expensive hardware.

The smartphone application was developed on the Android platform using Java programming language. The reasoning for preferring Android to other mobile platforms is that even though it is a rather new operating system, by the year 2010 it reached, according to [9], the second position on the smartphone operating system market. Also the broad availability of mobile devices (ranging from high-end products to regular and affordable devices) together with the rich collection of libraries, classes and methods that manage Bluetooth functionality, network applications, network and telephony services, user interface and widget design, make Android a suitable platform for such an application. Nevertheless, there are no restrictions and no special permissions to be required by third parties in order to create and publish Android software.

Starting with Android 2.1, the platform includes full support for the Bluetooth protocol stack. The Bluetooth APIs enable the user to access the whole Bluetooth functionality and let applications "scan for other Bluetooth devices, query the local Bluetooth device adapter for paired Bluetooth devices, establish RFCOMM channels and connect to other devices through service discovery, and, nevertheless transfer data or manage multiple connections" [10].

Also several new functionalities may be added to the present scenario in order to improve the user experience. For example, given that Android 2.2 (or higher) supports tethering, the smart-phone application can be extended to include this facility and share its Internet connection with the PDA. Hence, the PDA may connect to the smartphone either via WiFi (in ad-hoc mode) or via Bluetooth and get access to the Internet (using the 2G/3G mobile connection).

It should be also taken into account that Android 3.1 will provide new capabilities to integrate connected peripherals with applications running on the platform; more specifically, it brings in a fully platform-integrated USB stack and set of APIs that may eliminate the need of intermediary devices to communicate with the engine.

We considered initially the possibility of using the vehicle interface adapter (J1587 to USB /WiFi) to eliminate the need for another PDA application. However, in Android 2.1 there is no support for USB, so it would have been necessary to provide our own low-level functions for the USB implementation. Due to this we preferred to use the PDA at the remote site to solve the physical layer interconnection. Unfortunately, given that the PDA will have two concurrent serial connections (one from engine and another - Bluetooth), it may act as a possible bottleneck – slowing down the data transmission.

For the desktop application running at the central site, we selected Java as the programming language. The main reasons were its platform independency and the existence of a fully functional and well documented API for serial communications (the **javax.comm** provides applications access to RS-232 hardware and its features include: serial port parameters configuration – baud rate, parity -, hardware and software data flow-control, data transfer and asynchronous event notification for data available on the port [11]).

Related to the user interface design – there are several Java GUI alternatives out of which we opted for SWT (Standard Widget Toolkit). "SWT is an open source widget toolkit designed to provide efficient, portable access to the user-interface facilities of the operating systems on which it is implemented" [12]. Also, SWT is entirely written in Java and in order to display GUI elements its implementation accesses the native GUIs of the operating system on which the application is running.

4.3.2 Hardware and software constraints

The devices used in the prototype's design are given below:

Type	CPU	RAM	OS
HP iPAQ PDA	ARM920T PXA27x	56.62 MB	Winows Mobile 5.0
VODIA Rugged PDA	Intel PXA270	107.47 MB	Winows Mobile 6.1
Sony Xperia X8	ARM 11, 600 MHz	256 MB	Android 2.1
PC	Intel Core2Duo P8600, 2.4 GHz	3 GB	Winows Windows 7

Table 4.1: Devices used in the prototype's design

Volvo 9998555 Signal Level Converter – used to connect the PC to the J1708 bus. In case the diagnostics tests are run from a PC, the signal converter ensures a mean of accessing the engine's J1708 bus. Its main purpose is to convert the signal levels of RS-232 to J1708 but it also provides signals that may be used for bus access management. Since J1708 is a bus topology (i.e. only one wire) and RS-232 has separate data transmission lines, it is expected that, once sent, data will be echoed back to the transmitter (PC). Given that Java provides no readily-available mechanism for echo elimination, an algorithm that drops the loopback messages must be implemented.

No software constrains exist for the implementation of the desktop application. The programming language selected was Java and the development environment was Eclipse-based IDE. Same holds for the smartphone application with the specification that currently, Android does not yet provide developers with a USB communication API. Therefore, for the current prototype data transfer between the smartphone and PDA was possible only through the Bluetooth interface.

In the last place, the only constraint related to the HP iPAQ PDA is the Bluetooth protocol stack employed (Widcomm's stack). This implies using the WIDCOMM Bluetooth API and the Microsoft Visual Studio 2005 (or higher) development environment.

4.4 Detailed software architecture

4.4.1 Use Case model

The purpose of the current project is to develop a system for connecting the existing Volvo Penta's service tool to a remotely located vessel through a wireless communication link (GPRS/3G). The main functions of the system are: to establish a connection via the mobile data network between a central office and a remote location and to perform system's diagnostics tests remotely (read error codes, perform ECU or vessel configuration test).

Figure 4.4 shows the block scheme of the system. As it can be seen, it may be divided in two other subsystems – one running at the central location and the other running at the remote site. The first subsystem incorporates the VODIA and the Remote Diagnostic Tool application while the latter contains the Android and the PDA application.

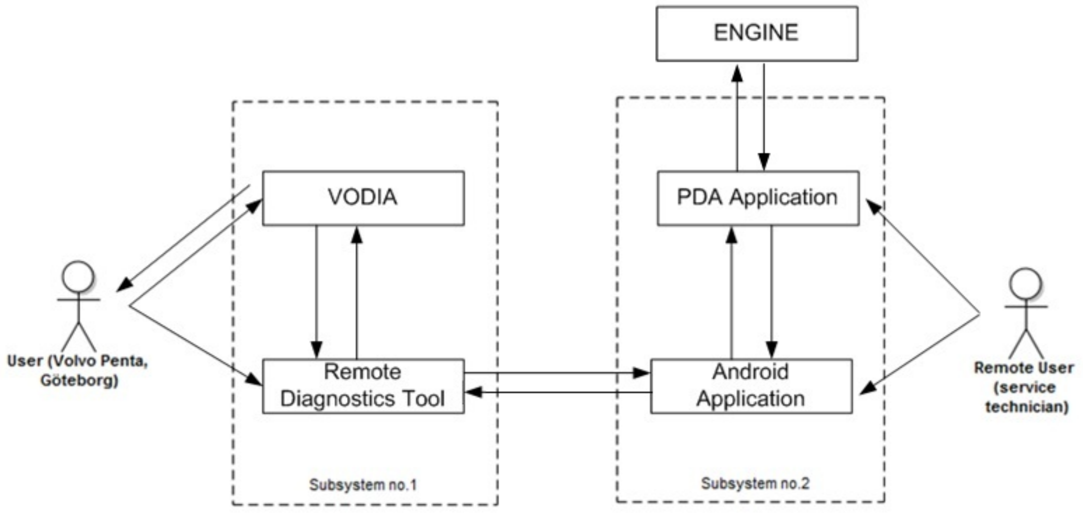


Figure 4.4: Prototype's conceptual architecture (Block diagram)

Hence, at the central location's site, a desktop application running on a PC will be able to communicate with the VODIA tool using the serial interface. The main purpose of the desktop application is to send correctly the data read from the serial interface - SAE J1708/J1587 formatted messages - over the Internet. The communication flow is bidirectional therefore the application has to be capable as well of receiving the data sent over the network by the remote party.

Nevertheless, given the fragmentation and echo that may appear at the serial interface when data is being read, respectively written, the application must provide an algorithm that ensures messages are correctly formatted before being sent over the Internet and that the echo appearing after writing data to the serial port is entirely eliminated.

At the remote location, the engine will communicate with a PDA that runs an application capable of reading and writing the SAE J1708/J1587 messages from and to the Bluetooth interface. Further on, the data delivered to the Bluetooth interface is captured in a smart phone and sent over the network. Conversely, the smart phone will be able to receive that data from the network interface and deliver it to the Bluetooth interface.

Note that at this end, the smart phone's purpose is only to relay the data received from the PDA, over the Internet, to the desktop application and no other processing is to be done here. Unlike this, the PDA will additionally flag the messages received from the engine with an extra byte indicating their exact length. The purpose of this flag is to ensure that at the receiving end, the messages are separated and sent correctly to the VODIA tool.

4.4.2 System's Functions. Detailed description of the use case diagrams

The use cases of each subsystem are presented in the following tables. The main functions of the system are: initialization of the serial communication, selection of the operating mode (client or server) and execution of diagnostic tests remote.

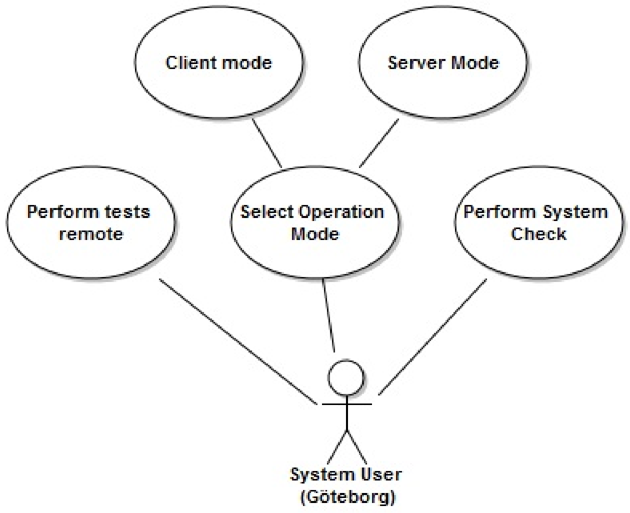


Figure 4.5: Use case diagram of the first subsystem

Use Case 1	**Perform system check**	
Actor	Service user (Type: Human)	
Preconditions	The VODIA tool must be connected (via the signal level converter) to the PC's serial port	
Post-conditions	**Successful completion**	Serial port communictaion was initialized
	Failure condition	Serial port was already opened
Flow of Events	1. The user selectes the option for performing system's check	

Use Case 2	**Select operation mode**	
Actor	Service user (Type: Human)	
Preconditions	Serial port connection must have been initialized successfully	
Post-conditions	**Successful completion**	Selection of the desired operation mode
	Failure condition	-
Flow of Events	1. The user selects the operation mode: server mode (Wait for connection request) or client mode(Connect to client)	

Use Case 3	**Server Mode**	
Actor	Service user (Type: Human)	
Preconditions	"Wait for connection request" option was selected	
Post-conditions	**Successful completion**	Accept a connection request from a remote host
	Failure condition	I/O error occurred when creating the server socket
Flow of Events	1. Start listening for incoming connection requests 2. Once a client is connected, display its IP address	

Use Case 4	**Client Mode**	
Actor	Service user (Type: Human)	
Preconditions	"Connect to client" option was selected	
Post-conditions	**Successful completion**	The connection to the remote host has been established
	Failure condition	I/O error occurred when creating the socket, the IP address of the remote host could not be determined
Flow of Events	1. Enter the IP address of the remote host 2. Connect to the remote server	

Use Case 5	**Perform tests remote**	
Actor	Service user (Type: Human)	
Preconditions	An active connection with the remote location	
Post-conditions	**Successful completion**	Perform diagnostics tests (Read error codes, ECU test, Vessel configuration test)
	Failure condition	Timeout exceeded
Flow of Events	1. The user works only with the VODIA tool	

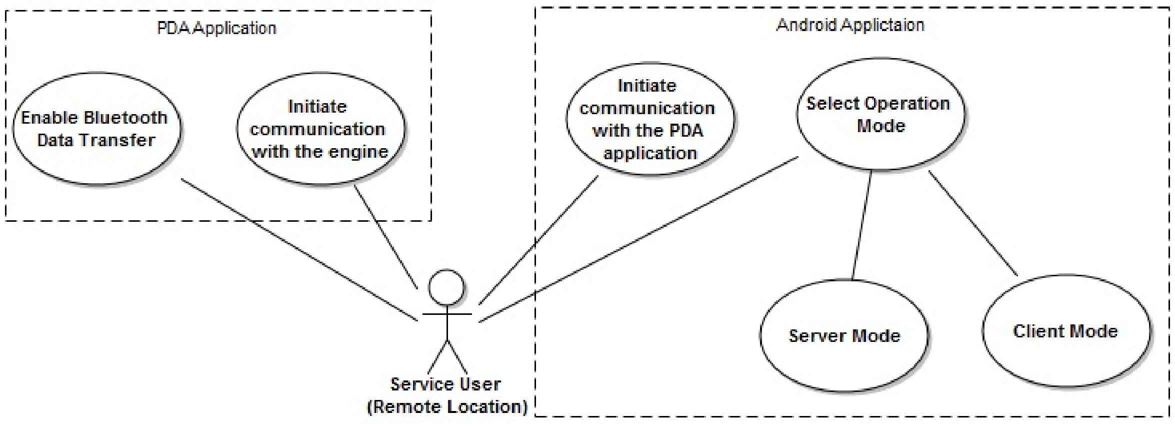

PDA Application
Enable Bluetooth Data Transfer
Initiate communication with the engine
Android Applictaion
Initiate communication with the PDA application
Select Operation Mode
Server Mode
Client Mode
Service User (Remote Location)

Figure 4.6: Use case diagram of the remote subsystem

Use Case 1	**Enable Bluetooth data transfer**	
Actor	Service user (Type: Human)	
Preconditions	Bluetooth device connectivity is enabled	
Post-conditions	**Successful completion**	Open a server connection for an RFCOMM port Listen for a connection attempt
	Failure condition	SCN value already in use Security level could not be assigned Port already opened or port negotiation failed
Flow of Events	1. Enable Bluetooth connectivity (start advertising a given service, wait for a Bluetooth client to connect)	

Use Case 2	**Initiate communication with the engine**	
Actor	Service user (Type: Human)	
Preconditions	Bluetooth device connectivity is enabled	
Post-conditions	**Successful completion**	Serial communication is enabled
	Failure condition	Serial port initialization failed
Flow of Events	1. Activate transfer mode – initiate the communication with the engine	

Use Case 3	**Initiate communication with the PDA application**	
Actor	Service user (Type: Human)	
Preconditions	The “VODIA Remote” application must be started	
Post-conditions	**Successful completion**	Connection with the other Bluetooth device is established
	Failure condition	Bluetooth not available I/O error – connection failure
Flow of Events	1. Setup Bluetooth on the smartphone 2. Discover the available Bluetooth devices 3. Select the Bluetooth device to connect with 4. Initiate connection with the selected device	

Use Case 4	**Select operation mode**	
Actor	Service user (Type: Human)	
Preconditions	Established Bluetooth connection with the PDA	
Post-conditions	**Successful completion**	Selection of the desired operation mode succeeded
	Failure condition	-
Flow of Events	1. The User selects the operation mode (server or client)	

Use Case 5	**Server mode**	
Actor	Service user (Type: Human)	
Preconditions	Established Bluetooth connection with the PDA	
Post-conditions	**Successful completion**	Accept a connection request from a remote host
	Failure condition	I/O error occurred when creating the server socket
Flow of Events	1. The User selects the “Wait for Connection” option (Server Mode) 2. Start listening for incoming connection requests 3. Once a client is connected, display its IP address	

Use Case 6	**Client mode**	
Actor	Service user (Type: Human)	
Preconditions	The connection to the remote host has been established	
Post-conditions	**Successful completion**	Perform diagnostics tests (Read error codes, ECU test, Vessel configuration test)
	Failure condition	I/O error occurred when creating the socket The IP address of the remote host could not be determined
Flow of Events	1. The User selects the “Connect to service” option (Client Mode) 2. Enter the IP address of the remote host 3. Initiate a connection with the remote host	

4.4.3 Class Diagrams

As a result of designing the prototype for the remote service tool, we obtained the class diagrams presented in the following figures. Their analysis gives a complete view upon the component modules, classes and the interactions and relations between them.

4.4.3.1 Android Application

The class diagram of the Android application is given in Figure 4.7. The entry point in the application is defined in the VodiaActivity class which is inherited from the android.app.Activity. Basically, this class takes care of creating the UI and establishing the main points of interaction with the user. As a subclass of Activity, it is mandatory to implement the **onCreate()** method. Here is where the activity is initialized and also where the layout for the activity's UI is defined. The purpose of the **onClick()** method is to retrieve the widgets with which the user needs to interact or to take further actions based on their selection.

Another method implemented in this class is **populateSpinner()** – whose purpose is to set up Bluetooth and to perform a complete scan for other available Bluetooth devices. Once a new device is found, it is added to a spinner and displayed to the user. This method is called whenever the user wishes to refresh the list of available devices within the local area.

In **getNetConnection()** method, depending on the selected operation mode, are instantiated objects of **VodiaNetClient** or **VodiaNetServer** type; they are employed when dealing with or serving a remote connection.

VodiaNetClient and **VodiaNetServer** classes extend the Thread class and override the run method. They contain the functionality of the TCP client respectively, TCP server that are due to operate on smartphone's side. Within the run() method, new objects of InputStream and OutputStream type are instantiated in order to allow the delivery and reception of data over the Internet and Bluetooth wireless interface.

For creating the **RemoteVodiaUtil** the singleton design pattern was used. The Singleton pattern is one of the most common design patters used in Java. “Its aim is to encapsulate the creation of an object in order to maintain control over it. This not only ensures that only one is created, but also allows lazy instantiation; that is, the instantiation of the object can be delayed until it is actually needed” [13].

The methods implemented by this class are: **connectBtSocket()** - within which the BluetoothDevice object that represents the remote device is obtained; further on, this object is used in order to acquire a Bluetooth socket and to initiate a connection. The **getLocalIpAddres()** method is used to retrieve the IP address assigned to the local device. This method is called when the application operates as a TCP server, waiting for incoming connection requests. Note that in most cases the assigned IP corresponds to a private one, not visible over the Internet. A call to **getSocket()** creates a new socket, binds it to a certain port and connects it to the given address.

For the AdapterModel class, the Adapter design pattern was used. The aim of this class is only to wrap the data in order to interface it with the spinner component.

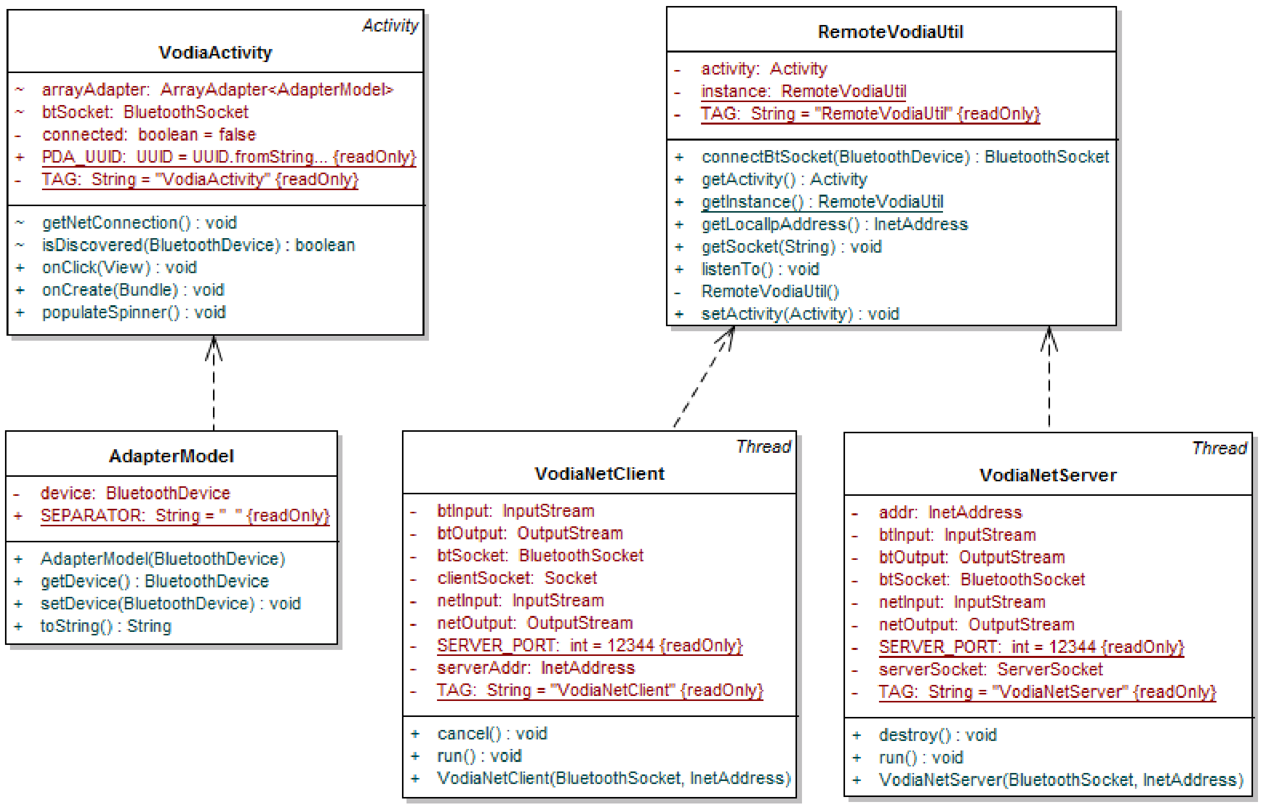

Activity
VodiaActivity
~ arrayAdapter: ArrayAdapter<AdapterModel>
~ btSocket: BluetoothSocket
- connected: boolean = false
+ PDA_UUID: UUID = UUID.fromString... {readOnly}
- TAG: String = "VodiaActivity" {readOnly}
~ getNetConnection() : void
~ isDiscovered(BluetoothDevice) : boolean
+ onClick(View) : void
+ onCreate(Bundle) : void
+ populateSpinner() : void
RemoteVodiaUtil
- activity: Activity
- instance: RemoteVodiaUtil
- TAG: String = "RemoteVodiaUtil" {readOnly}
+ connectBtSocket(BluetoothDevice) : BluetoothSocket
+ getActivity() : Activity
+ getInstance() : RemoteVodiaUtil
+ getLocalIpAddress() : InetAddress
+ getSocket(String) : void
+ listenTo() : void
- RemoteVodiaUtil()
+ setActivity(Activity) : void
AdapterModel
- device: BluetoothDevice
+ SEPARATOR: String = " " {readOnly}
+ AdapterModel(BluetoothDevice)
+ getDevice() : BluetoothDevice
+ setDevice(BluetoothDevice) : void
+ toString() : String
Thread
VodiaNetClient
- btInput: InputStream
- btOutput: OutputStream
- btSocket: BluetoothSocket
- clientSocket: Socket
- netInput: InputStream
- netOutput: OutputStream
- SERVER_PORT: int = 12344 {readOnly}
- serverAddr: InetAddress
- TAG: String = "VodiaNetClient" {readOnly}
+ cancel() : void
+ run() : void
+ VodiaNetClient(BluetoothSocket, InetAddress)
Thread
VodiaNetServer
- addr: InetAddress
- btInput: InputStream
- btOutput: OutputStream
- btSocket: BluetoothSocket
- netInput: InputStream
- netOutput: OutputStream
- SERVER_PORT: int = 12344 {readOnly}
- serverSocket: ServerSocket
- TAG: String = "VodiaNetServer" {readOnly}
+ destroy() : void
+ run() : void
+ VodiaNetServer(BluetoothSocket, InetAddress)

Figure 4.7: The class diagram of the Android application

4.4.3.2 PDA Application

To the already existing code that dealt with reading the J1587 packets from the engine only one class was added. The **RFCommPort** class inherits the **CRfCommPort** class (Widcomm SDK) and overrides two of its virtual methods: **OnEventReceived()** and **OnDataReceived()**. The aforementioned methods are used in order to notify the Bluetooth server that an event, respectively, data has been received from the remote side.

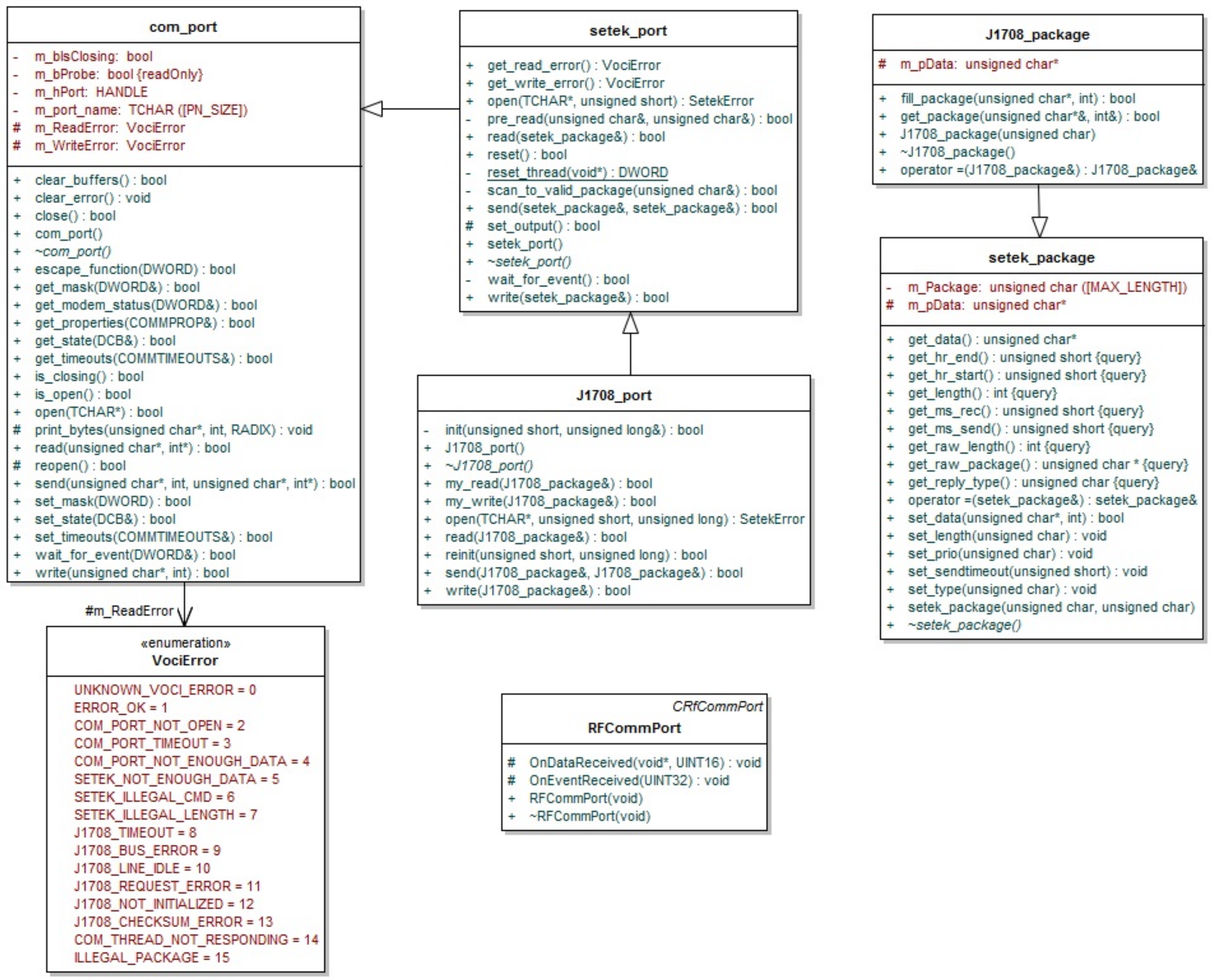

com_port
- m_bIsClosing: bool
- m_bProbe: bool {readOnly}
- m_hPort: HANDLE
- m_port_name: TCHAR ([PN_SIZE])
# m_ReadError: VociError
# m_WriteError: VociError
+ clear_buffers() : bool
+ clear_error() : void
+ close() : bool
+ com_port()
+ ~com_port()
+ escape_function(DWORD) : bool
+ get_mask(DWORD&) : bool
+ get_modem_status(DWORD&) : bool
+ get_properties(COMMPROP&) : bool
+ get_state(DCB&) : bool
+ get_timeouts(COMMTIMEOUTS&) : bool
+ is_closing() : bool
+ is_open() : bool
+ open(TCHAR*) : bool
# print_bytes(unsigned char*, int, RADIX) : void
+ read(unsigned char*, int*) : bool
# reopen() : bool
+ send(unsigned char*, int, unsigned char*, int*) : bool
+ set_mask(DWORD) : bool
+ set_state(DCB&) : bool
+ set_timeouts(COMMTIMEOUTS&) : bool
+ wait_for_event(DWORD&) : bool
+ write(unsigned char*, int) : bool
#m_ReadError
«enumeration»
VociError
UNKNOWN_VOCI_ERROR = 0
ERROR_OK = 1
COM_PORT_NOT_OPEN = 2
COM_PORT_TIMEOUT = 3
COM_PORT_NOT_ENOUGH_DATA = 4
SETEK_NOT_ENOUGH_DATA = 5
SETEK_ILLEGAL_CMD = 6
SETEK_ILLEGAL_LENGTH = 7
J1708_TIMEOUT = 8
J1708_BUS_ERROR = 9
J1708_LINE_IDLE = 10
J1708_REQUEST_ERROR = 11
J1708_NOT_INITIALIZED = 12
J1708_CHECKSUM_ERROR = 13
COM_THREAD_NOT_RESPONDING = 14
ILLEGAL_PACKAGE = 15
setek_port
+ get_read_error() : VociError
+ get_write_error() : VociError
+ open(TCHAR*, unsigned short) : SetekError
- pre_read(unsigned char&, unsigned char&) : bool
+ read(setek_package&) : bool
+ reset() : bool
- reset_thread(void*) : DWORD
- scan_to_valid_package(unsigned char&) : bool
+ send(setek_package&, setek_package&) : bool
# set_output() : bool
+ setek_port()
+ ~setek_port()
- wait_for_event() : bool
+ write(setek_package&) : bool
J1708_port
- init(unsigned short, unsigned long&) : bool
+ J1708_port()
+ ~J1708_port()
+ my_read(J1708_package&) : bool
+ my_write(J1708_package&) : bool
+ open(TCHAR*, unsigned short, unsigned long) : SetekError
+ read(J1708_package&) : bool
+ reinit(unsigned short, unsigned long) : bool
+ send(J1708_package&, J1708_package&) : bool
+ write(J1708_package&) : bool
CRfCommPort
RFCommPort
# OnDataReceived(void*, UINT16) : void
# OnEventReceived(UINT32) : void
+ RFCommPort(void)
+ ~RFCommPort(void)
J1708_package
# m_pData: unsigned char*
+ fill_package(unsigned char*, int) : bool
+ get_package(unsigned char*&, int&) : bool
+ J1708_package(unsigned char)
+ ~J1708_package()
+ operator =(J1708_package&) : J1708_package&
setek_package
- m_Package: unsigned char ([MAX_LENGTH])
# m_pData: unsigned char*
+ get_data() : unsigned char*
+ get_hr_end() : unsigned short {query}
+ get_hr_start() : unsigned short {query}
+ get_length() : int {query}
+ get_ms_rec() : unsigned short {query}
+ get_ms_send() : unsigned short {query}
+ get_raw_length() : int {query}
+ get_raw_package() : unsigned char * {query}
+ get_reply_type() : unsigned char {query}
+ operator =(setek_package&) : setek_package&
+ set_data(unsigned char*, int) : bool
+ set_length(unsigned char) : void
+ set_prio(unsigned char) : void
+ set_sendtimeout(unsigned short) : void
+ set_type(unsigned char) : void
+ setek_package(unsigned char, unsigned char)
+ ~setek_package()

Figure 4.8: The class diagram for the PDA application

4.4.3.3 Service Application -“Remote Diagnostic Tool”

The class diagram for the “Remote Diagnostics Tool” is presented in Figure4.9. **VodiaRemoteWindow** is the principal class of the application and it defines two methods - the **main()** method - principal entry point in the application - and **createShell()** – that defines the top-level Shell of the user interface.

VodiaRemoteUtil class integrates methods for initializing the serial communication, for initiating and closing a remote connection and for checking the validity of the IP address entered by the user. To create this class, we used the Singleton design pattern.

VodiaTcpServer and **VodiaTcpClient** include the whole functionality of the TCP server and client. Both classes extend the Thread class (java.lang.Thread) and implement the **run()** method. Within **run()** the input and output streams associated with client’s socket are obtained and data is sent and received using the **read()** and **write()** methods. The classes implement as well the **SerialPortEventListener** interface and override the **serialEvent()** method. This method contains the routine to be followed in case data is available at the serial interface.

Note that for the Android and the service desktop application programs only Transmission Control Protocol (TCP) was used. At the lowest level, computer networks provide unreliable packet delivery: as mentioned in Chapter 2.4.3, packets may be lost (e.g. in case of congestion, network hardware failures or when transmission errors occur), may arrive in a different order, delayed or duplicated. At the application level, in general, a high data transfer takes place between two devices, thus a connectionless protocol such as User Datagram Protocol (UDP), adds extra constraints to these applictaions: they must have built-in error detection and recovery mechanisms in order to handle entirely the problem of reliability. Obviously, the present software programs may be easily extended to support, beside TCP, UDP; but, unless the afore-mentioned measure is considered the applications may fail dramatically when used over the global data network.

IPAddressCheck - the UI required a mechanism for preventing the errors that may occur due to the data entered by the user. Since SWT provides no special control for distinguishing between proper and improper user inputs, an efficient way to validate the entered IP address was with the aid of regular expressions. A regular expression – also known as regex – represents a specific kind of text pattern that may be used to simplify the text processing tasks. Hence, the string entered by the user is simply matched against an IP address pattern. If the matching succeeds, the flow continues with initiating the connection with the remote site, otherwise a corresponding error message is displayed.

Throughout the application’s execution, users must be provided with appropriate feedback regarding the current state of the system and the events that take place. The **VodiaRemoteDebug** class is a singleton created with the purpose of logging and displaying the communication status, the sequence of messages exchanged between the remote entities and possible errors that may appear during a communication session. Given that all debug messages, warnings, exceptions or errors had to be displayed to the user, the approach of using a singleton pattern seemed suitable; thus, we avoided instantiating new objects of the **VodiaRemoteDebug** type in every class where we had log messages to display and nevertheless, ensured that resources are more efficiently used.

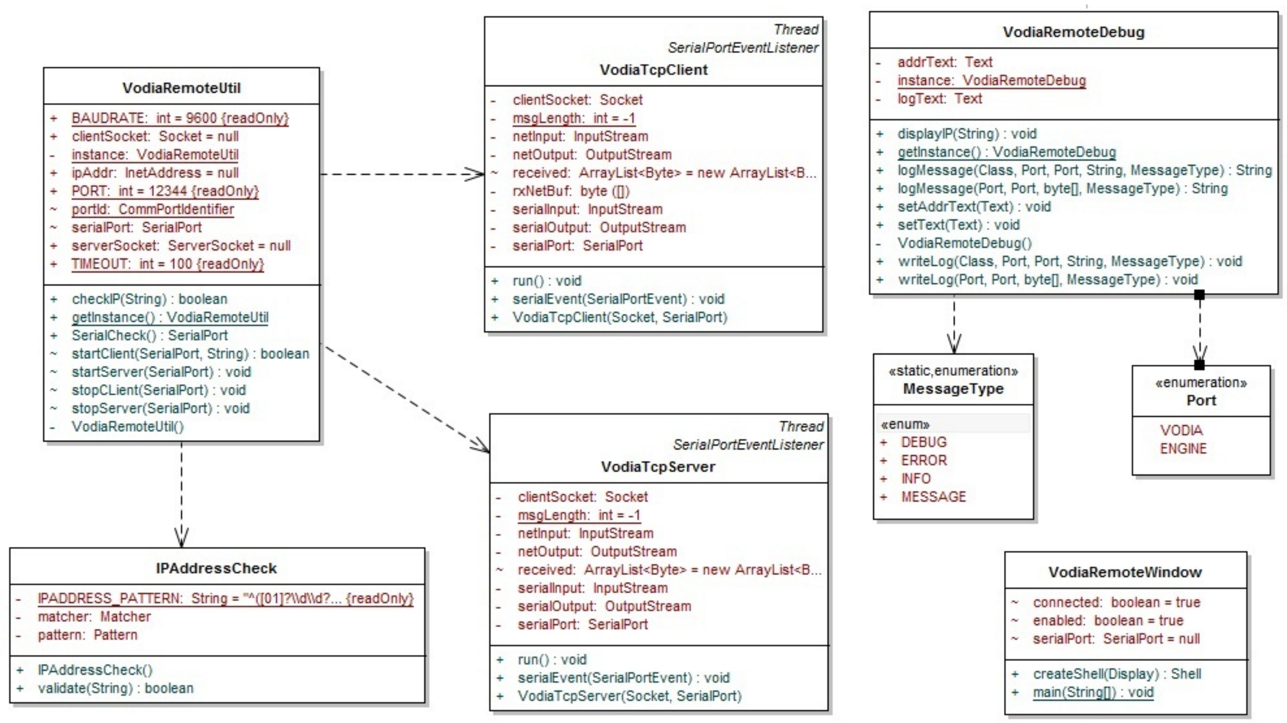

VodiaRemoteUtil
+ BAUDRATE: int = 9600 {readOnly}
+ clientSocket: Socket = null
- instance: VodiaRemoteUtil
+ ipAddr: InetAddress = null
+ PORT: int = 12344 {readOnly}
~ portId: CommPortIdentifier
~ serialPort: SerialPort
+ serverSocket: ServerSocket = null
+ TIMEOUT: int = 100 {readOnly}
+ checkIP(String) : boolean
+ getInstance() : VodiaRemoteUtil
+ SerialCheck() : SerialPort
~ startClient(SerialPort, String) : boolean
~ startServer(SerialPort) : void
~ stopCLient(SerialPort) : void
~ stopServer(SerialPort) : void
- VodiaRemoteUtil()
Thread
SerialPortEventListener
VodiaTcpClient
- clientSocket: Socket
- msgLength: int = -1
- netInput: InputStream
- netOutput: OutputStream
~ received: ArrayList<Byte> = new ArrayList<B...
- rxNetBuf: byte ([])
- serialInput: InputStream
- serialOutput: OutputStream
- serialPort: SerialPort
+ run() : void
+ serialEvent(SerialPortEvent) : void
+ VodiaTcpClient(Socket, SerialPort)
VodiaRemoteDebug
- addrText: Text
- instance: VodiaRemoteDebug
- logText: Text
+ displayIP(String) : void
+ getInstance() : VodiaRemoteDebug
+ logMessage(Class, Port, Port, String, MessageType) : String
+ logMessage(Port, Port, byte[], MessageType) : String
+ setAddrText(Text) : void
+ setText(Text) : void
- VodiaRemoteDebug()
+ writeLog(Class, Port, Port, String, MessageType) : void
+ writeLog(Port, Port, byte[], MessageType) : void
«static,enumeration»
MessageType
«enum»
+ DEBUG
+ ERROR
+ INFO
+ MESSAGE
«enumeration»
Port
VODIA
ENGINE
Thread
SerialPortEventListener
VodiaTcpServer
- clientSocket: Socket
- msgLength: int = -1
- netInput: InputStream
- netOutput: OutputStream
~ received: ArrayList<Byte> = new ArrayList<B...
- serialInput: InputStream
- serialOutput: OutputStream
- serialPort: SerialPort
+ run() : void
+ serialEvent(SerialPortEvent) : void
+ VodiaTcpServer(Socket, SerialPort)
IPAddressCheck
- IPADDRESS_PATTERN: String = "^([01]?\\d\\d?... {readOnly}
- matcher: Matcher
- pattern: Pattern
+ IPAddressCheck()
+ validate(String) : boolean
VodiaRemoteWindow
~ connected: boolean = true
~ enabled: boolean = true
~ serialPort: SerialPort = null
+ createShell(Display) : Shell
+ main(String[]) : void

Figure 4.9: The class diagram of the “Remote Diagnostic Tool” application

Chapter 5

Implementation

This section outlines the steps followed to implement each module of the remote diagnostic tool. It is also given a detailed description of the problems together with the solutions to remedy them and algorithms we provided for an optimization of the communication process. As described previously we aim at establishing a logical link between the current diagnosis application running at a central location and a remote engine (see Figure 4.4 for the prototype's conceptual architecture). The project's development had been carried out in three steps.

A first step consisted in writing the Android application that will serve the purpose of transferring the J1587 messages received from the PDA to the remote location, via GPRS/3G. The application should be able to scan for the available Bluetooth devices and establish a connection with the selected one. If the Bluetooth connection succeeds, the user may further select the desired operation mode, namely, server or client. In Server mode he waits to be contacted by a central service and in client mode he initiates the connection request himself. Note that in a real-life scenario, the server mode may be used provided that the user is assigned a public IP. However this is not always the case – depending on the network operator the users may be assigned private IP addresses that are not visible over the Internet.

The second step was to develop a PDA application aimed at reading the J1708 messages from the engine and sending them, via Bluetooth, to the Android phone. The PDA application and the Android one were designed to work in a Bluetooth client-server fashion; the Bluetooth client functionality was assigned to the smartphone while the server part was taken care of by the PDA. The Bluetooth server implemented here is able to assign a service channel number, to setup a service in the local device and then wait for clients to attempt a connection. The client also has the possibility to enable the data transfer mode in order to allow messages to be exchanged between the engine and the PDA.

The last part was to implement a Java desktop application able to convey the data received from the VODIA tool to the Android phone and conversely, from the Android phone to VODIA. Thus, its functionality includes both a TCP server and a TCP client and a routine for reading and writing data to the serial port.

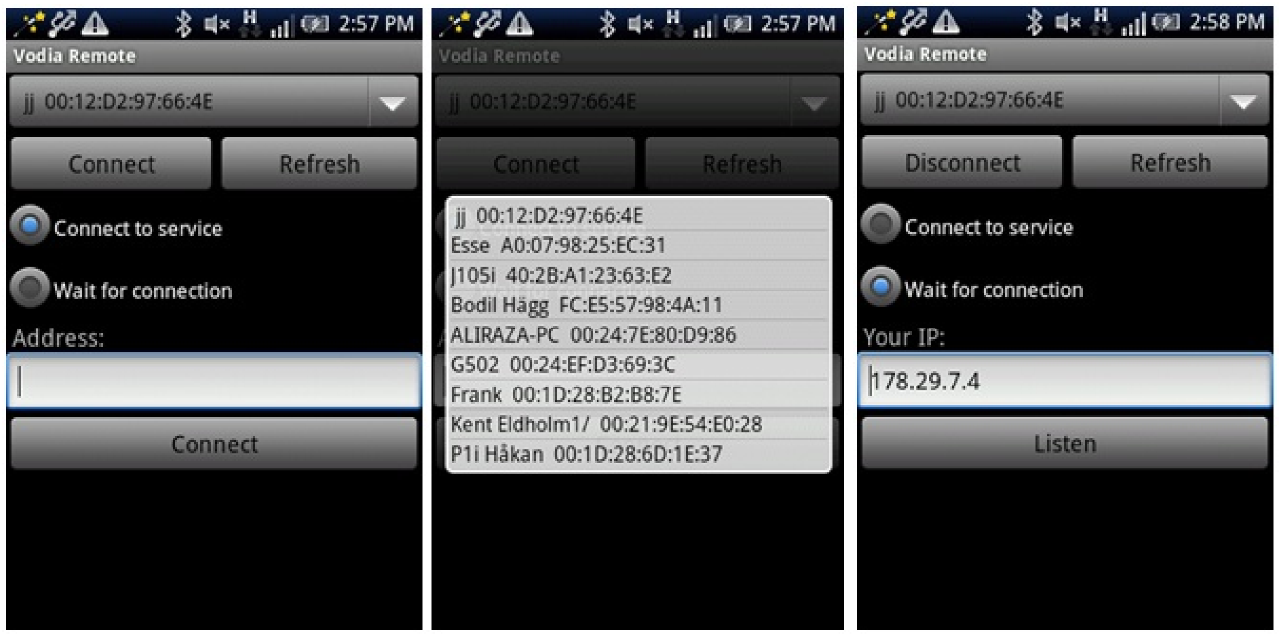
5.1 Android Application

The application was developed in **Eclipse IDE** using **Android 2.1** platform and **Java** programming language. The objective was to develop an application having functionality to connect to the PDA as a Bluetooth client and also to act as a stream-based client/server for remote access over the 3G/GPRS link.

Android framework comes with an open source **Linux** based Bluetooth stack – **BlueZ** - which supports, among others, the Widcomm (Broadcom) and Microsoft Bluetooth stack. The application expects the phone to have an active data connection in order to be able to support remote connectivity. To allow the application to open network sockets, permission has to be added in the **AndroidManifest.xml** file as discussed briefly in this section.

Android has a **Logcat** utility to print and monitor debugging information. The logging utility class android.util.Log is available in the **Android SDK**. Common logging messages include, errors, warnings, informational, debug and verbose each having a certain degree of severity. Log filters can be created by defining a static tag in the application and later, applying the filter to output specific data. We used static string tags for debugging and named them according to the class in which the error message/exception appeared: **VodiaActivity, VodiaNetClient, VodiaNetServer, RemoteVodiaUtil**. For displaying short, unobtrusive messages to the user – e.g. to indicate possible problems or errors that occurred – simple Toast widgets were used.

To begin with, the first step is to develop a basic User Interface (UI) that will help the user to interact with the application. Figure **??** shows snap shots of the screen at different connection phases.



Your IP: 178.29.7.4

Figure 5.1: The user interface of the Android application at different connection phases. The first picture displays the general view, the second - the Bluetooth devices available in the local area and the third – the enabling of the server operation mode.

Building an Android application involves several basic steps:

1. **Designing a Layout and inserting widgets**: Layout class which is inherited from the ViewGroup base class organizes the application screen contents e.g. buttons, text controls, images etc. It is also referred to as layout resource and it is stored as XML file in /res/layout resource directory of the application. Basically, it provides a template for the interface screen. The UI in figure above shows a simple Linear layout with nested Table layout that holds a spinner widget, a "Connect" and, respectively, a "Refresh" push button. A RadioGroup widget is used to set the mode of the application to either client or server. Also the UI contains an editable text box to enter the IP address of the remote server (when acting as a client) or to display the device's IP address (when the operation mode is set to server). Lastly, the UI contains the button "Connect" which initiates the corresponding routine depending on the selected operating mode.

2. **Adding Permission to AndroidManifest.xml**: It is mandatory for every Android application to have a manifest file. The file outlines essential features and modules that are needed by the Android system before the app launches e.g. security characteristics, activities, intent receivers, content and service providers etc. In other words, AndroidManifest.xml acts as a development descriptor for the application. In this .xml file we add the permission for Bluetooth and Internet access.

3. **Defining classes and implementing the Callback functions**: When application launches, it begins with an Activity that is presented to the user. Figure 5.2shows the state path of a typical activity in Android. The colored ovals represent the major states of the Activity while the rectangles indicate the methods called in order for the Activity to move from one state to another [14].

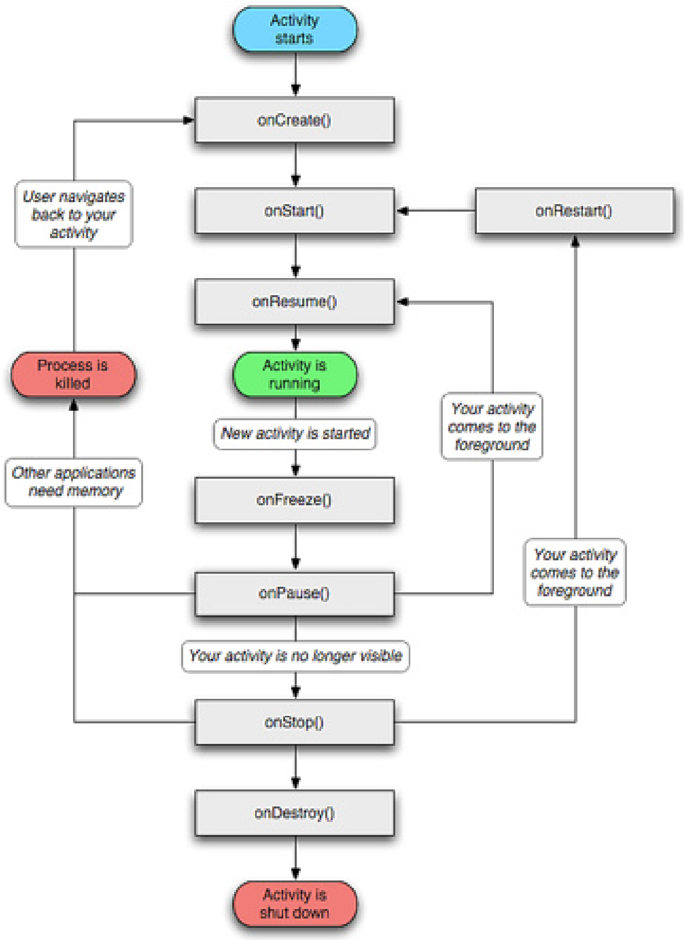

Activity starts
onCreate()
User navigates back to your activity
onStart()
onRestart()
onResume()
Process is killed
Activity is running
New activity is started
Your activity comes to the foreground
Other applications need memory
onFreeze()
Your activity comes to the foreground
onPause()
Your activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Figure 5.2: The Activity life cycle[14]

The class diagram for Android application is shown in Figure 4.7. VodiaActivity which is a subclass of Activity implements the **onCreate()** method. In this method it is necessary to load and display the layout resource on the screen. The spinner which was added in the UI design is referenced and binded to arrayAdapter which describes item's layout.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    this.getWindow().setSoftInputMode(
        WindowManager.LayoutParams.SOFT_INPUT_STATE_ALWAYS_HIDDEN);
    Spinner ss = (Spinner) findViewById(R.id.spinner1);
    arrayAdapter = new ArrayAdapter<AdapterModel>(this,
        android.R.layout.simple_spinner_item);
    ss.setAdapter(arrayAdapter);
}
```

The method **onClick()** is called when user interacts with any of the widgets, by initiating the processing required for a particular action. The user is firstly expected to press the 'Refresh' button. This will call the **populateSpinner()** method that performs an inquiry and scan after Bluetooth devices available in its vicinity. In this method, after getting an instance of the local Bluetooth adapter, it is verified whether the device is enabled; if not, an activity is started to enable it.

```
BluetoothAdapter adapter = BluetoothAdapter.getDefaultAdapter();
if (!adapter.isEnabled()) {
    Intent enableBtIntent = new Intent(
        BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableBtIntent, -1);
}
```

The next step is to start discovery phase which involves an inquiry scan of 12 seconds in order to find the remote, discoverable devices. An object of the **BroadcastReceiver** class is instantiated; its purpose is to respond to the broadcast announcements and to receive the action of an Intent object. We are only interested in the event corresponding to newly found remote devices therefore, **BroadcastReceiver** is registered only for this event. If the "Intent" object delivered to **onReceive()** method matches a new remote device found, the method adds it as a new item in the spinner.

```
if (adapter.startDiscovery()) {
final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device = intent
            .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            if (!isDiscovered(device))
                arrayAdapter.add(new AdapterModel(device));
        }
    }
};
```

The method **isDiscovered()**, employed in the discovery phase, ensures that the devices found are not added twice in the item list. It returns a Boolean value – "false" – if a device that has not been previously added to the spinner and "true" otherwise.

Once the spinner is populated, the user may select the device he/she wishes to connect to. The 'Connect' button initiates the routine for connect to the remote device.

```
Spinner spinner = (Spinner) findViewById(R.id.spinner1);
AdapterModel selectedItem = (AdapterModel) spinner.getSelectedItem();
btSocket = RemoteVODIAUtil.getInstance().connectBtSocket(
        selectedItem.getDevice());
if (btSocket == null)
    Toast.makeText(this, R.string.err_conn_bt_msg, Toast.LENGTH_LONG).show();
```

The method **connectBtSocket()** is implemented in the **RemoteVodiaUtil** class and takes a **BluetoothDevice** class instance as an input. The method createRfcommSocket-ToServiceRecord performs SDP lookup of the given UUID and creates an RFCOMM Bluetooth socket. An exception occurs if the device is not available or permission to access Bluetooth is not provided in the manifest file. Note that both the PDA and the Android app must use the same UUID. The next step is to connect Bluetooth socket to the RFCOMM channel using the 'connect' call, which is blocking and returns an exception if an outgoing connection is not established. If the connection is successfully established 'Connect' button text is changed to 'Disconnect' and the connected Bluetooth socket is returned.

```
btSocket = btDevice.createRfcommSocketToServiceRecord(VODIAActivity.PDA_UUID);
BluetoothAdapter.getDefaultAdapter().cancelDiscovery();
btSocket.connect();
```

Another method implemented in this class is **getNetConnection()**. Once the Bluetooth connection is successfully established, the next phase is to create a socket and to connect, over the Internet, to a remote station (or, wait for an incoming connection attempt). There are two modes of operation, either as client or as a server. When RadioButton is set to initiate a server session the text box displays the IP address of the local device and the **VodiaNetServer** thread is initiated. The remote site uses the displayed IP (local device's IP) address to connect to the server.

```
InetAddress addr = RemoteVODIAUtil.getInstance().getLocalIpAddress();
EditText et = (EditText) findViewById(R.id.editText1);
et.setText(addr.getHostAddress());
VODIANetServer thread = new VODIANetServer(btSocket, addr);
thread.start();
```

For obtaining the local IP address, the method **getLocalIpAddress()** implemented in **RemoteVodiaUtil** class is called. Its aim is to scan the system's available interfaces and to return the Internet address of the connected interface.

```
public InetAddress getLocalIpAddress() {
...
for (Enumeration<NetworkInterface> en = NetworkInterface
    .getNetworkInterfaces(); en.hasMoreElements();) {
    NetworkInterface intf = en.nextElement();
    for (Enumeration<InetAddress> enumIpAddr = intf
            .getInetAddresses(); enumIpAddr.hasMoreElements();) {
        InetAddress inetAddress = enumIpAddr.nextElement();
        if (!inetAddress.isLoopbackAddress())
            return inetAddress;
```

If the client mode is selected, the user must type in the editable text box the IP address of the server he wants to connect to. The address is read and converted to String and is used as an input along with connected Bluetooth socket to **VodiaNetClient** thread.

```
InetAddress[] serviceAddr = InetAddress.getAllByName(et.getText().toString());
VodiaNetClient thread = new VodiaNetClient(btSocket,serviceAddr[0]);
thread.start();
```

The class **VodiaNetServer** which extends Thread class sets up a TCP server for remote client to connect to. The constructor of the class takes as parameters a connected Bluetooth socket and the IP address of the local device. In the **run()** method, the first task here was to create a server socket with the port number and the IP address of the local device. The server socket is set to listen to 12344 (arbitary port number) where it blocks and wait for the client to make a connection request.

```
serverSocket = new ServerSocket(SERVER_PORT, 2, addr);
InetAddress ipAddr = serverSocket.getInetAddress();
while (!endListening) {
    clientSocket = serverSocket.accept();
    if (clientSocket != null)
        endListening = true;
```

After the client is connected the input and output streams to write and read data to/from sockets are aquired. The Java method **InputSteam.avalaible()** is used to dynamically set the size of the buffer in which data is read. This method returns a -1 if no data is available. When the returned buffer size is greater that 0 the method **InputStream.read()** is used in order to read the bytes and store them in the buffer. If the data is read from the Bluetooth input stream, it is written to the network output stream and vice versa. To write to a socket the method **OutputStream.write()** was used. The thread once initiated runs in an infinite loop.

```
while (true) {
    byte[] rxBtBuf = new byte[btInput.available()];
    if (rxBtBuf.length > 0) {
        btInput.read(rxBtBuf);
        if (rxBtBuf != null)
            netOutput.write(rxBtBuf);
        byte[] rxBuf = new byte[netInput.available()];
        if (rxBuf.length > 0)
            netInput.read(rxBuf);
        if (rxBuf != null)
            btOutput.write(rxBuf);
    }
```

VodiaNetClient class extends as well the Thread class and overrides the **run()** method. Its purpose is to implement the TCP client functionality. The routines for creating the socket, for reading and writing the data are similar to the ones presented for the **VodiaNetServer** class.

5.2 PDA Application

This application was developed using C++ programming language in **Visual Studio 2008 IDE**. Several classes used for the physical layer interconnection between the engine and the PDA were already available. An initial step in the implementation process was to identify the possible changes and additions that are to be made in the original functions and classes in order to incorporate the Bluetooth functionality. Thus, we eliminated parts of the code that were communicating with other layers and modified the functions aimed at initializing the serial communication with the engine, reading and writing data to the communication interface.

Given that the application residing on the smartphone implements the Bluetooth client-side mechanism, the PDA will implement the server-side. Due to the fact that the Bluetooth protocol stack implemented on the HP iPAQ PDA is the Widcomm (Broadcom) one, the application was developed using the Widcomm Bluetooth for Windows CE SDK.

Broadcom offers a fully functional API for Bluetooth programming, providing programmers direct access to L2CAP (Logical Link Control and Adaptation Protocol), RFCOMM protocols and OBEX (Object Exchange Protocol), SDP(Service Discovery Protocol), SPP (Serial Port Profile), LAP (LAN Access Point) profiles. Unlike the Bluetooth Windows programming, Broadcom's development kit ensures a higher level of abstraction by hiding the technical details; also, it is not MFC dependent, although it works with it, and uses pure virtual methods for callbacks. The steps followed to implement the Bluetooth server are shown in Figure 5.3.

To begin with, an object of class **CRfCommIf** was instantiated. In order to get a SCN (Secure Channel Number) assigned, the function **AssignScnValue()** was call. Its scope is to assign new UUID (Universally Unique Identifier), a 128-bit value used to identify uniquely a Bluetooth application/service. To generate the UUID value we used the *GuidGen* tool available in Visual Studio and we ensured that this value matches the one from the Android application.

The next step was to setup the service in the local Bluetooth device – by creating an object of the **CSdpService** type and executing the methods: **AddServiceClassIdList(), AddServiceName(), AddRFCommProtocolDescriptor(), MakePublicBrowsable()**. To serve this purpose, these functions were called in a separate method – **createServiceRecord()** – that returns a Boolean value in case all the mentioned methods executed successfully and service set up succeeded. Note that the **AddServiceClassIdList()** was the first function called inside the function. Its aim is to add a service class ID list attribute to the service record [9]. In our case, the application has only one GUID in the service class ID list.

A call to **CRfCommIf** method OpenServer starts the server and waits for a connection attempt from a client. Once a Bluetooth client connects, the function **OnEventReceived()** is called. The *BtIfDefinitions.h* file contains the values for several event codes, out of which only PORT_EV_CONNECTED, and PORT_EV_CONNECT_ERR (used to detect a connection request and a disconnect) are of interest for the present application. Similarly, **OnDataReceived()** intercepts the PORT_EV_RXCHAR (data character received) events and the available data may be read into a corresponding buffer.

The methods mentioned above are provided by the **CRfCommPort** class. They are defined as pure virtual methods, so in order to be able to control the RFCOMM connections the server application must provide a derived class in which their real method equivalent is given. The class implemented solely with this purpose is **RFCommPort** class. It contains an empty constructor and overrides the aforementioned methods.

Once the application is notified that a client connected, by a PORT_EV_CONNECTED event, the client's unique address is retrieved and displayed on the PDA's screen. Same event is raised when the client disconnects from the server. Moreover, if data is available on the Bluetooth interface, the PORT_EV_RXCHAR is raised. The data is read into a buffer and further sent to the serial interface.

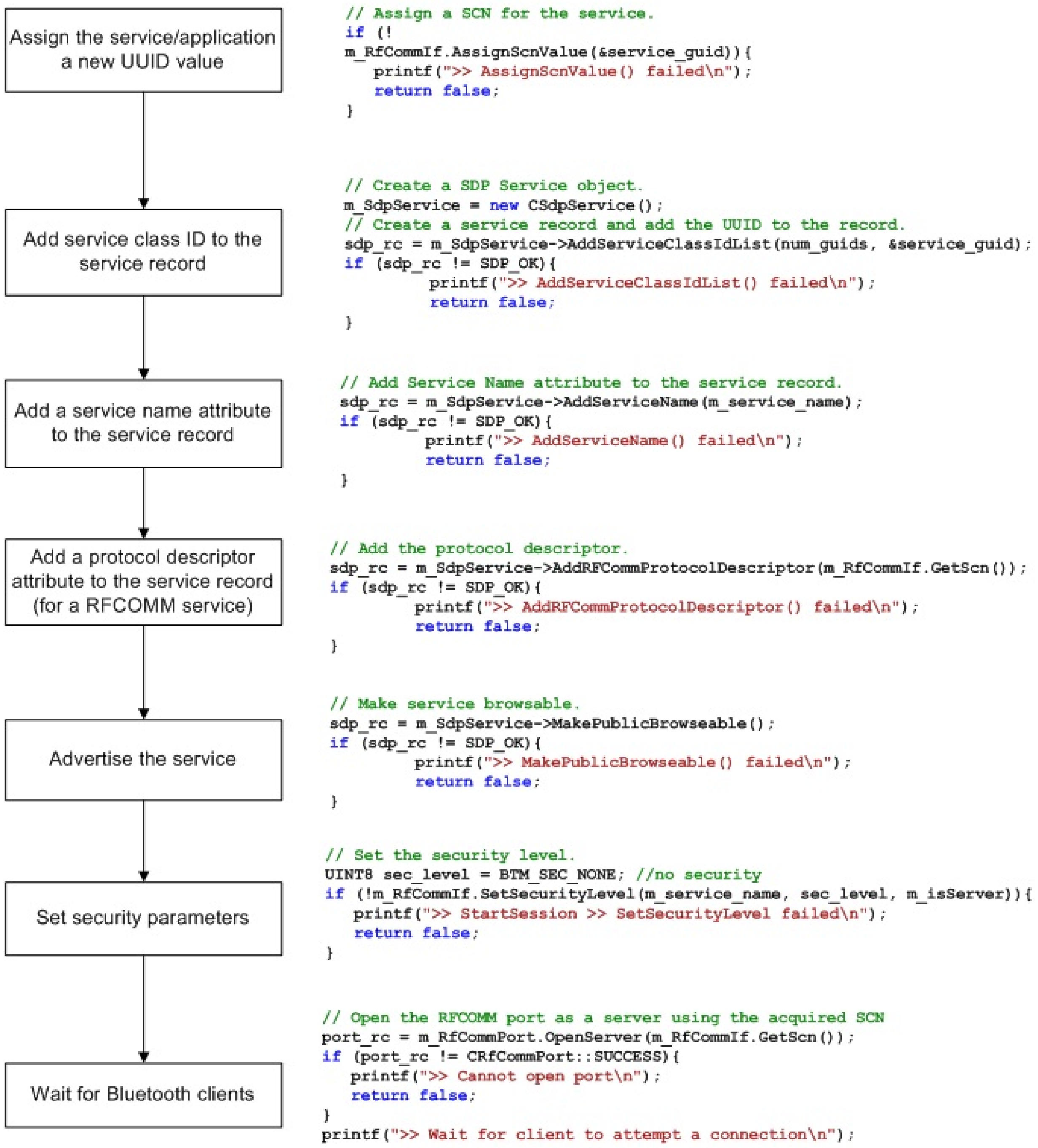
In order to be able to read the data available on the serial interface (engine replies), we kept a main thread – **Thread_Receive** – in which, we listen for events on the serial interface. Once an event appears, it is verified whether it corresponds to a "data received character" event, case in which the function **SendDataToBT()** is called. Inside this method, the engine data (reply) is written to the RFCOMM port using the **CRfCommPort::Write()** function.

```
bool SendDataToBT(unsigned char *data, int length)
{
    UINT16 written=0;
    port_rc = m_RfCommPort.Write((void *) data, length, &written);
    if (CRfCommPort::SUCCESS != port_rc)
        {
            printf("Write failed, with code %i\n", port_rc);
            return false;
        }
    return true;
}
```

The application contains as well a function for closing down the communication session. Within **StopSession()**, the RFCOMM port is closed and the service is no longer advertised.

```
bool StopSession()
{
    // Close the RFCOMM port.
    //
    port_rc = m_RfCommPort.Close();
    if (port_rc != CRfCommPort::SUCCESS && port_rc != CRfCommPort::NOT_OPENED)
    {
        printf(">> StopSession >> Port Close() failed\n");
        return false;
    }
    // Delete the SDP object (if created).  This
    // removes the service from the 'available' list.
    //
    if (m_SdpService)
    {
        delete m_SdpService;
        m_SdpService = NULL;
    }
    printf(">> StopSession >> Port closed\n");
    return true;
}
```

In order to properly convey data between the engine and the smartphone, the PDA application follows a simple convention and the steps to be performed by the user (for activating/deactivating the communication session) appear in a logical and natural order. Given its final purpose, the application does not need a complex user interface and it is essential that the instructions to use the system are intuitive, being based on recognition rather than recall; also the visibility of the system's status is only partially ensured – the user can see if the status of transfer mode, can read the Bluetooth address of the connected device; however, this can be further improved in the sense that more appropriate feedback can be provided to the user at reasonable times (i.e. dialog boxes that announce that the Bluetooth connection went down or that there is no data exchanged between the engine and the smartphone). We considered the system is simple enough to be used without any help or documentation, but in case needed, this feature can be easily added.


Assign the service/application a new UUID value
Add service class ID to the service record
Add a service name attribute to the service record
Add a protocol descriptor attribute to the service record (for a RFCOMM service)
Advertise the service
Set security parameters
Wait for Bluetooth clients

```
// Assign a SCN for the service.
if (!
m_RfCommIf.AssignScnValue(&service_guid)){
   printf(">> AssignScnValue() failed\n");
   return false;
}

// Create a SDP Service object.
m_SdpService = new CSdpService();
// Create a service record and add the UUID to the record.
sdp_rc = m_SdpService->AddServiceClassIdList(num_guids, &service_guid);
if (sdp_rc != SDP_OK){
        printf(">> AddServiceClassIdList() failed\n");
        return false;
}

// Add Service Name attribute to the service record.
sdp_rc = m_SdpService->AddServiceName(m_service_name);
if (sdp_rc != SDP_OK){
        printf(">> AddServiceName() failed\n");
        return false;
}

// Add the protocol descriptor.
sdp_rc = m_SdpService->AddRFCommProtocolDescriptor(m_RfCommIf.GetScn());
if (sdp_rc != SDP_OK){
        printf(">> AddRFCommProtocolDescriptor() failed\n");
        return false;
}

// Make service browsable.
sdp_rc = m_SdpService->MakePublicBrowseable();
if (sdp_rc != SDP_OK){
        printf(">> MakePublicBrowseable() failed\n");
        return false;
}

// Set the security level.
UINT8 sec_level = BTM_SEC_NONE; //no security
if (!m_RfCommIf.SetSecurityLevel(m_service_name, sec_level, m_isServer)){
   printf(">> StartSession >> SetSecurityLevel failed\n");
   return false;
}

// Open the RFCOMM port as a server using the acquired SCN
port_rc = m_RfCommPort.OpenServer(m_RfCommIf.GetScn());
if (port_rc != CRfCommPort::SUCCESS){
   printf(">> Cannot open port\n");
   return false;
}
printf(">> Wait for client to attempt a connection\n");
```

Figure 5.3: Steps for setting up a Bluetooth server application

5.3 Service Application

The service application resides at the central location and interacts with the VODIA Tool via the serial bus interface. The application was developed in Eclipse IDE using Java programming language.

5.3.1 Main classes

As discussed in Chapter 2.3, using sockets we can design both client and enhancementserver programs. To begin with, we designed a server that creates a socket, associates to the socket an address and launches a mechanism to listen for incomming connection requests. Regarding the used protocol, we have chosen to work only with TCP socket based programming.

Although UDP, which is a connectionless oriented transport protocol, may gain extra credits due to its effciency and flexibility, we must take into consideration that TCP, which is a connection oreiented protocol, offers a reliable byte-stream service. This aspect is critical for remote diagnostics applications where undelivered, duplicated or delayed messages may cause timeouts and generate successive retransmissions, increasing considerably the amount of time required to perform a test.

Moreover, we must take into account that messages sent via UDP might arrive at their destination in a different order than they were sent; this is a disadvantage considering that the server might waste precious time dealing with loss or reordering. However, if desired, the application may be extended easily to support multiple transmission protocols given that Java offers a clear distinction between TCP and UDP and separate classes are defined for both protocols.

So, in order to implement the TCP server, a common scenario was used (see Figure 2.5). The server's scope was to set up an endpoint for clients to connect and passively wait for connections. Initially a **ServerSocket** instance was created to listen on a specific port (in our case, we selected a port number that is not already dedicated to other services, namely 12344). If port binding succeeds, the server may start accepting connection requests. Once a connection is requested and successfully established, a **Socket** object is returned and binded to the same local port. From now on, the server and the client may communicate over this newly created socket.

The routine for creating a new server socket and launching a mechanism for listening after incomming connection requests is contained in the **VodiaRemoteUtil** class. Once a client is connected to the socket, a new instance of the **VodiaTcpServer** class is created in order to handle the client. Also **VodiaRemoteUtil** class contains the routine for closing the server socket and the socket associated to the client.

```
serverSocket = new ServerSocket(PORT);
    clientSocket = serverSocket.accept();
    InetAddress ipAddr = clientSocket.getInetAddress();
    VodiaTcpServer thread = new VodiaTcpServer(clientSocket,serialPort);
thread.start();
```

Since there is only one serial port it is not possible to handle multiple client connections simultaneously, therefore the server accepts only one connection request. However, a different application logic would be to accept several clients, enqueue them and serve them based on their arrival or priority in the queue.

The interaction with the client takes place in the **VodiaTcpServer** class.This class extends the **Thread** class (java.lang.Thread) and implements the **run()** method. Withing this method, data exchange takes place once the input and output streams associated with the client's socket are obtained.

Once having completed the server functionality, the next step was to access the serial port for data reading and writing. Using the **Java Communication API** the user may access the serial port in a platform-independent manner. The two main classes defined within the Communictaion API are **SerialPort** and **ParallelPort**; both of them inherit an abstract class – **CommPort** – and define the two types of ports that may be accessed using the given API (see Figure 5.4 for the structure of the Java Comm API).

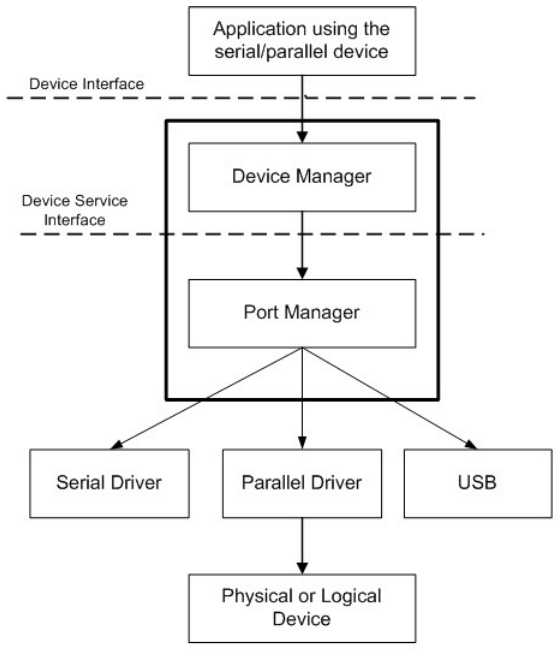


Figure 5.4: The structure of the Java Communication API [11]

Given that the constructor of the SerialPort class is not public, in order to obtain an instance of this class a static method must be employed. To obtain the list containing the available ports on the system we used the static method **CommPortIdentifier.getPortIdentifiers()**. Once a port is selected, it may be converted from a **CommPortIdentifier** object to the **CommPort** type it actually represents and its parameters (baudrate, start and stop bits, data bits, flow control) may be set accordingly. Similarly to the previous case, reading and writing are prossible once the stream input and output objects were obtained.

```
portId = CommPortIdentifier.getPortIdentifier(defaultPort);
// open serial port and set parameters
CommPort commPort = null;
commPort = portId.open("VODIARemoteDesktop", TIMEOUT * 1000);
SerialPort serialPort = (SerialPort) commPort;
serialPort.setSerialPortParams(BAUDRATE, SerialPort.DATABITS_8,
            SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
   serialPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
```

Reading and writing data to the serial port can be done in two ways - *synchronously* and *asynchronously*. Synchronous communication assumes the existence of a communication protocol between the Java application and the external device and knowledge of the exact order of commands and messages exchanged between the two entities. This does not apply to our case, since the aim of our application is simply data relaying and no interpretation takes place here. Hence, since we have no previous knowledge related to the order at which reading and writing operations take place, the application must be notified only when an event appears on the serial port.

The API offers as well the option for an asynchronous communication by allowing the instantiation of objects of **Observer** (or Listener) type that raise notifications whenever the state of the communication port has changed. In a general case, every Observer pattern has a subject and multiple observer entities. The subject keeps a list (name and address) of each registered observer with it so, once an event happens, it goes through its list and notifies all the observers interested in that kind of event.

In order to be notified of the events which occur on the serial port, the class that contains the server's functionality – **VodiaTcpServer** – must beforehand implement the **SerialPortEventListener** interface, thus override the method **serialEvent (SerialPortEvent event)**. Nevertheless, the class must add an instance of itself to the serial port's observer list such that serial events are received when particular conditions occur. Note that a serial port is permitted to have only one observer (listener), so whenever more than one listener attempts to add itself, a java.util.TooManyListenersException will be thrown. For our case, it was enough to instruct the serial port to notify its listener only when data is available at the serial interface and for this, the **notifyOnDataAvailable()** method was employed.

Also, the **serialEvent(SerialPortEvent event)** method contains the routine to be followed if the condition that led to firing an event on the interface matches the DATA_AVAILABLE one. Therefore, whenever data was available on the serial port, we read it in a buffer and further send it over the network.

Two problems occurred when testing the application. Both of them were related to the fact that data was sometimes received fragmented both from the serial interface and from the network. When these fragments reached the engine (or the VODIA application) they were being discarded since they did not correspond to valid J1587 messages. This led to timeouts and retransmission in the actual diagnostic tool which increased considerably the time required to perform a test.

A correct formatting of the messages received over the network could have been made provided that a field in the received packet indicates its length. Since in a J1587 packet there is no field for the size, the only solution to overcome this fragmentation issue was to add (at the PDA's side) a new byte indicating the dimension of the packet. At the receiver, the bytes received were accumulated in a buffer until their number reached the value indicated by the flag, and only after that sent over to the VODIA tool.

```
while (true) {
    msgLength = netInput.read(); //the first byte indicates the length
    if (msgLength > 0) {
        byte[] rxNetBuf = new byte[msgLength];
        for (int i = 0; i < msgLength; i++)
            rxNetBuf[i] = (byte) netInput.read();
            serialOutput.write(rxNetBuf);
            msgLength = -1;
        }
}
```

A similar problem occurred when data was read from the serial interface. However, the previous method employed to retrieve correctly the message could not be used here. The approaches used are described in the followings.

Since the minimum length of the messages sent from the VODIA tool to the engine is usually 4 bytes and the first byte always equals 172, a possible solution was to drop all fragments containing less than 4 bytes or having a different value as a start byte. However, this ensured only that fragments were not further conveyed to the engine and did not decrease the time necessary to perform a test.

Another solution would have been to add a three (or less) bytes message to the previous received one. This method proved to be inefficient since long messages could have been as well fragmented into more than three parts or, in case of larger packets, fragments might have been made up from more than 4 bytes.

Given that there is no field within a J1587 message to indicate its length, the solution to this issue came from that fact that fragments arrive at the port with a 16[ms] delay apart from each other. Therefore, by using a dynamic array (Array List) to accumulate the received data bytes for a given time duration – by stopping the execution of a thread - , we could reduce significantly the number of corrupted messages and decrease the test's execution time.

```
if (serialInput.available() > 0) {
    serialInput.read(rxSerialBuf);
    if (received.isEmpty()) {
        for (int i = 0; i < rxSerialBuf.length; i++)
            received.add(rxSerialBuf[i]);
            Thread s = new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    sleep(20);
                    byte[] toSend = new byte[received.size()];
                    for (int i = 0; i < toSend.length; i++)
                        toSend[i] = received.get(i);
                        received.removeAll(received);
                        if (((int) toSend[0] & 0xFF) == 172)
                            netOutput.write(toSend);
                        } catch (Exception e) {}
                }});
            s.start();
        } else {
            for (int i = 0; i < rxSerialBuf.length; i++)
                received.add(rxSerialBuf[i]);
                return;
        }
}
```

In order for the central service to be able to contact the customer, a new class that implements the client functionality was added. The methods implemented in this class, as well as the approaches used to format correctly the J1587 packets are identical to those employed in the **VodiaTcpServer** class. Hence, the initial step in creating the TCP client was to create a socket and to connect to a remote address. Once connection is established, data transfer may take place freely.

```
InetAddress[] serviceAddr = InetAddress.getAllByName(strIp);
SocketAddress remoteAddr = new InetSocketAddress(serviceAddr[0],PORT);
clientSocket = new Socket();
clientSocket.connect(remoteAddr);
VODIATcpClient thread = new VODIATcpClient(clientSocket, serialPort);
thread.start();
```

As mentioned previously, users must be provided with appropriate feedback regarding the current state of the system and the events that take place. The **VodiaRemoteDebug** class is a singleton created with the purpose of logging and displaying the communication status, the sequence of messages exchanged between the remote entities and possible errors that may appear during a communication session. Given that all debug messages, warnings, exceptions or errors had to be displayed to the user, the approach of using a singleton pattern seemed suitable; thus, we avoided instantiating new objects of the **VodiaRemoteDebug** type in every class where we had log messages to display and nevertheless, ensured that resources are more efficiently used.

To implement the singleton pattern, it is required to make the default constructor of the class private (to prevent the instantiation of the object by other classes) and to add a static method that returns an instance to the singleton.

```
private static VODIARemoteDebug instance;
    private VODIARemoteDebug() {
        instance = this;
    }
    public static VODIARemoteDebug getInstance() {
        if (instance == null)
            new VODIARemoteDebug();
        return instance;
    }
```

5.3.2 The User Interface

Regarding the interaction between the user and the application, we excluded from the beginning the possibility of having command line control upon the execution of the program and opted for the default interaction style used by most personal computers, namely **WIMP** (Windows, Icons, Menu, Pointing Devices). Hence, objects and actions are continuously represented by visual objects and the user has to deal with push buttons and option selections instead of complex command line syntax.

As mentioned in Chapter 4, there are several Java GUI alternatives out of which we selected **SWT** (Standard Widget Toolkit) because it provides access to the native GUIs of the operating system on which the application is running and because **SWT** applications exhibit a greater responsiveness compared to Swing or AWT.

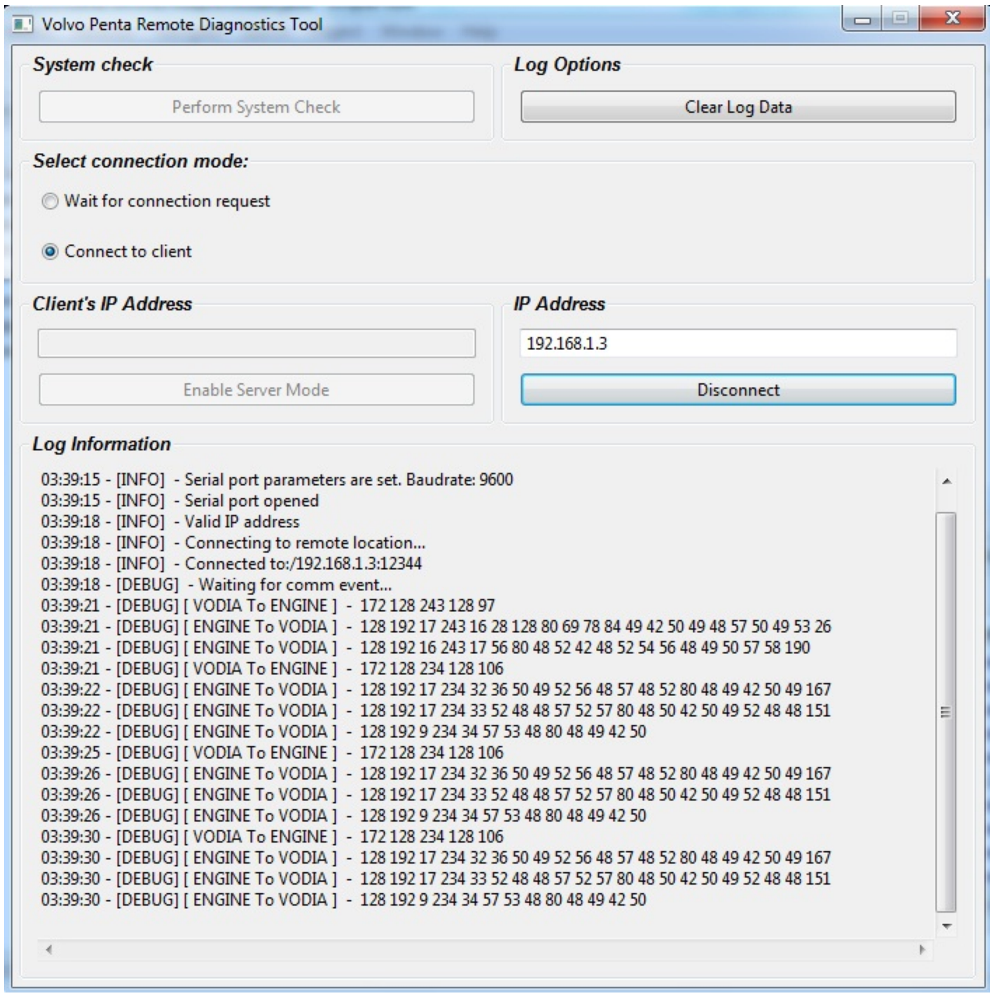
The **VodiaRemoteWindow** class contains the **main()** method and the implementation of the UI. In **main()** is instantiated a new object of **Display** type (that will act as a local resources manager, managing the connections between the SWT and the underlying OS); also, all the needed widgets are attached to a **Shell** object. In case the application is exited, the OS' resources used by the application need to be disposed explicitly. This was taken care of by calling the **dispose()** method implemented by the Display class.

```
public static void main(String[] args) {
    Display display = new Display();
    Shell shell = new VODIARemoteWindow().createShell(display);
    shell.open();
    while (!shell.isDisposed()) {
        if (!display.readAndDispatch())
            display.sleep();
    }
    display.dispose();
    System.exit(0);
}
```

The user interface designed for the desktop application is displayed in Figure 5.5. The main form is divided into several groups: **System check, Log options, Connection mode, IP Address, Client's IP Address** and **Log Information.** The user must first initiate the serial communication between the PC and the VODIA, by performing a system check; once this was executed successfully, the selection of the connection mode is possible. Thus, the application may act either as a TCP server (case in which the server mode must be enabled) - waiting for clients to connect - , or as a TCP client - initiating connections with a remote station at a given host address - . Also, a text area gives the user the possibility to assess the evolution of the system by displaying all the messages (error, debug or information messages) generated at different steps in the setup phase or during communication with the remote side.

The final design is aesthetic and minimalist: it does not contain redundant or useless information that would hamper visibility and control of sensitive information; the operating paradigm can be easily understood and the configuration sequence (serial port initialization, selection of operation mode, connection with the remote station) follows a natural and logical order, every change that takes place having an immediate visual effect.

Error prevention was done by limiting the user's access to some parts of the UI unless the steps for system's configuration were followed accordingly or their execution did not finalized with an error. For example, in case the user does not perform the “System Check” (which in fact initializes the communication between the PC and the VODIA Tool) or if this fails (the serial port is already owned by another application), the user can neither select the operating mode of the system nor establish a connection with a remote party. Also a control of the user data had to be performed: for example, after the user is prompted for address of the remote host, it is checked whether this corresponds to a valid IP; if not, a corresponding error message is displayed and the connection request to the remote host is not initiated.



Select connection mode: Wait for connection request [✓ Connect to client]

IP Address: 192.168.1.3

```
03:39:15 - [INFO]  - Serial port parameters are set. Baudrate: 9600
03:39:15 - [INFO]  - Serial port opened
03:39:18 - [INFO]  - Valid IP address
03:39:18 - [INFO]  - Connecting to remote location...
03:39:18 - [INFO]  - Connected to:/192.168.1.3:12344
03:39:18 - [DEBUG]  - Waiting for comm event...
03:39:21 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
03:39:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
03:39:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
03:39:21 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
03:39:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
03:39:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
03:39:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
03:39:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
03:39:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
03:39:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
03:39:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
03:39:30 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
03:39:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
03:39:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
03:39:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
```

Figure 5.5: The user interface of the desktop application

Chapter 6

System Tests and Results

As mentioned in the previous chapter the prototype of the remote diagnostics tool is made of three separate modules: PDA application (implemented in Visual Studio 2008 using C++ programming language), smartphone application and a desktop application (both implemented using Java programming language in Eclipse IDE).

In order to simplify the final integration, to reduce possible issues that may appear and to ensure that finally, the whole system works coherently, each unit underwent a separate testing. The hardware setups employed during each unit's test, the results obtained and the encountered issues are mentioned below. Nevertheless, the system's functionality was tested both over a WLAN network and over the mobile network. The logs containing the messages exchanged between the VODIA and the remote engine during different phases of the communication session were analyzed and compared with the J1587 Navigator application's logs. In the end, a comparative analysis of the prototype's performance with respect to the current diagnostic tool is made.

Note that the tests and results discussed in the following sections were performed using VODIA version 4.4.0 with read timeout value of 4000[ms], write timeout of 4000[ms] and the number of retries to 4 (for some particular messages the number of retransmissions might be higher).

6.1 Testing of the Android application

The functionality of the smartphone application was tested against a C Bluetooth server and a stream based server and client, over a WLAN network. Hence, we could verify whether enabling discoverability, searching after other Bluetooth devices and connection to a remote device work properly and whether the smartphone can exchange simple byte arrays with the Bluetooth server (see Figure 6.1). It must be mentioned that, the Bluetooth server was written in C using the BlueZ development library and ran under Linux Ubuntu distribution.

Moreover, it was tested if the application works as expected provided that, besides the

Bluetooth functionality, the stream based server (or client) was also running - more specifically, arrays of bytes were being sent from the TCP server (client) to the smartphone and then conveyed to the Bluetooth server; reversely, data was sent from the Bluetooth server to the smartphone and further to the TCP server (client). The integrity of received data was checked at both ends.

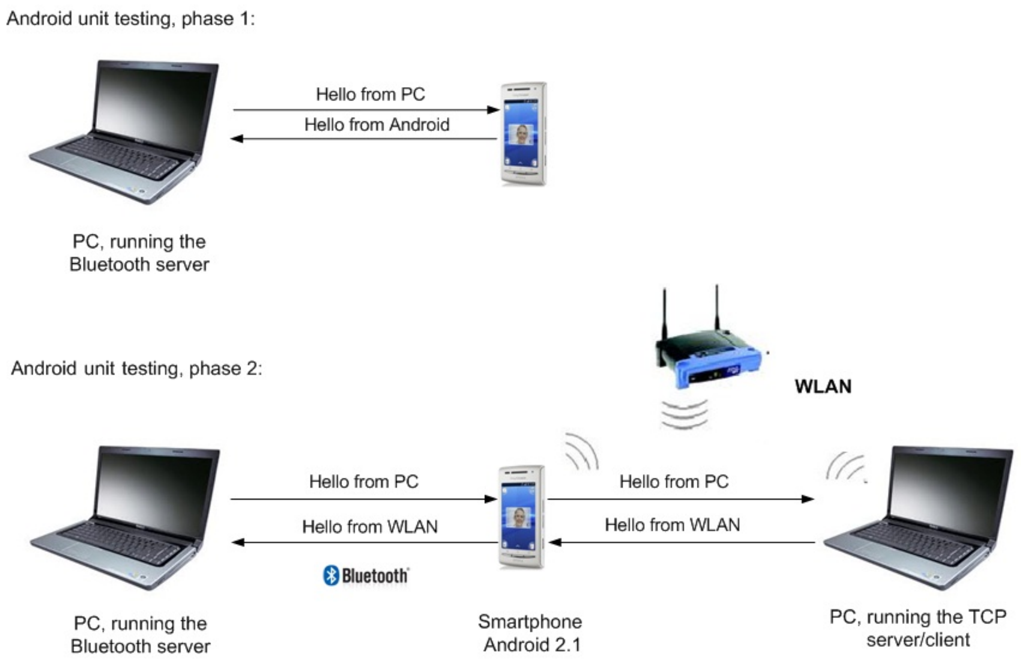

Android unit testing, phase 1:
Hello from PC
Hello from Android
PC, running the Bluetooth server
Android unit testing, phase 2:
WLAN
Hello from PC
Hello from WLAN
Bluetooth
Hello from PC
Hello from WLAN
PC, running the Bluetooth server
Smartphone Android 2.1
PC, running the TCP server/client

Figure 6.1: Hardware setup used for testing the Bluetooth and TCP server/client functionality on the smartphone

6.2 Testing of the PDA application

A first step was to test the Bluetooth server functionality, namely to verify if the application can accept incoming connection requests, display correctly the address of the connected device and transmit/receive hardcoded messages to/from the smartphone. For this, a similar setup to the one shown in Figure 6.2 (Phase 1) was used; the messages exchanged were checked against the data values they contained. The next step consisted in verifying the communication with the engine.

The setups used for this purpose are shown in Figure 6.2 and Figure 6.3. It must be mentioned that the application running on the PC – J1587 Navigator - can perform some diagnostic tests similar to VODIA e.g. querying the ECU for component ID, software version, error codes etc. This was an important tool for in the testing procedure, helping us to identify the correct format of the exchanged messages.

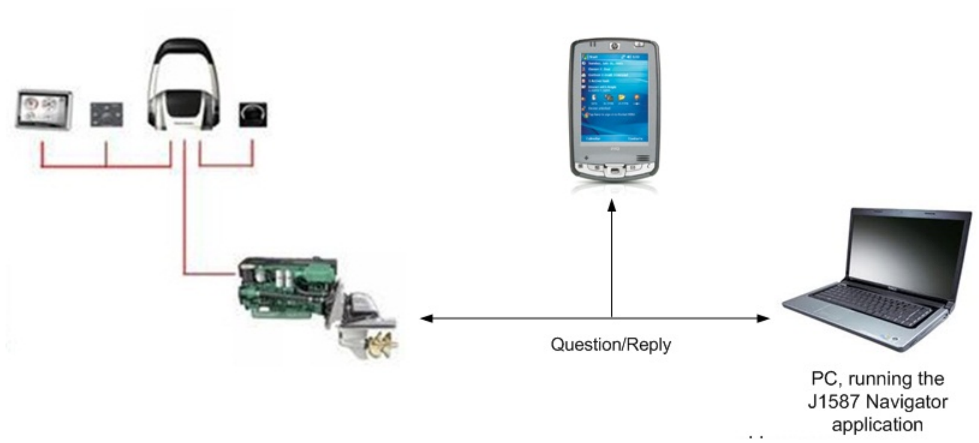


Figure 6.2: Hardware setup used for testing the PDA's serial communication with the engine

As seen in Figure 6.2 the engine is connected in parallel to the Navigator and to PDA application. The idea was to send out J1587 questions from the Navigator and to read the replies from the engine both on the PDA and on the PC. Using this setup we could verify that the read functionality of the application i.e. data values read by the application should be exactly the same as the values shown in the Navigator's logs.

So, once the serial initialization is done, the PDA application waits for a serial port event. In case this event corresponds to a data received event (**EV_RXCHAR**), the incoming data is read in a buffer and displayed on the console. By comparing this data with the Navigator logs we observed that apart from serial port data, 2 extra bytes were added to the package. These two bytes corresponded to an intermediary interface (communication jacket) that exists between the PDA and the engine and were not actually part of the original reply. Also, at the remote side, when the reply will be fed back to the VODIA application, these bytes will be again added, thus it is redundant to send them further in the communication chain.

In order to test the “write” functionality we hardcoded an arbitrary question (J1587 message) and wrote it to serial port. In addition to the data bytes, we added two other fields - Timeout and Priority – to which we had set random values (1000[ms] for timeout and 4 for priority).

In conclusion, we checked if the serial port initialization was performed correctly as well as the integrity of data values that were read in the buffer. Furthermore, the write functionality to serial port was also tested. The Navigator tool served the purpose of comparing the data sent and received to/from the engine.

The last phase for verifying the PDA's functionality was to connect the devices as shown in Figure 6.3. At that point, the flow of events was the following: the PDA sends out a question to the engine (the question was hardcoded in the application), the engine's response is read by the application and further forwarded to Android application over the Bluetooth link. The Android application reads the incoming data stream from Bluetooth and relays it to TCP server/client running on PC. The response read in the server/client application at PC was verified against the Navigator tool.

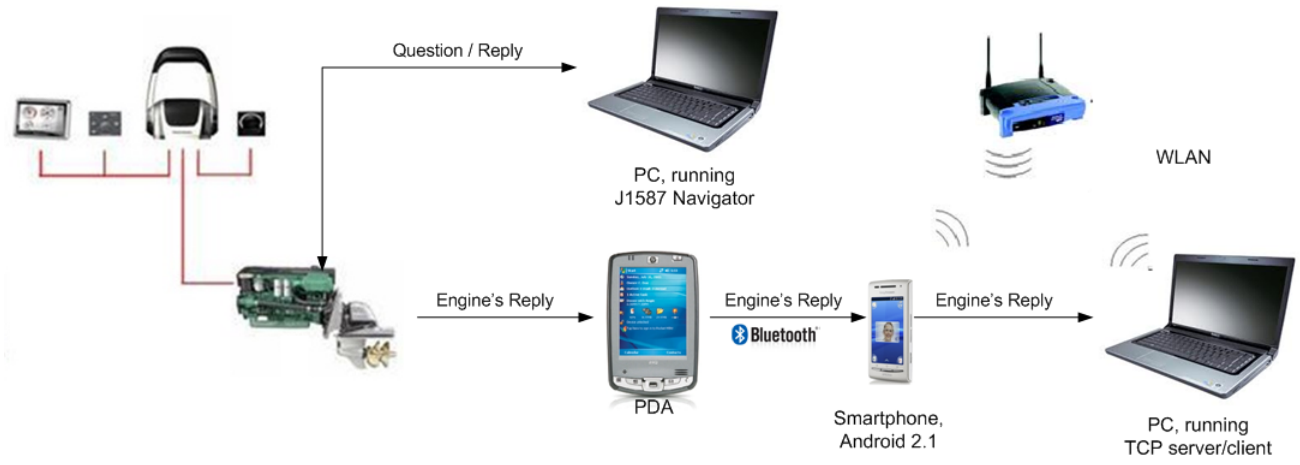


Figure 6.3: Hardware setup used for testing the PDA application

6.3 Testing of the desktop application

This part consisted in verifying whether data is read and written correctly over the serial port. The application residing on the PC is aimed at reading the incoming messages from the VODIA tool and sending them further over the Internet. Also, once data is received over the network interface this is conveyed to the serial port. In order to connect the PC to the J1708 bus a signal level converter was used; this ensured the conversion of J1708's signal levels to RS-232 (please refer to Chapter 4.3.2 for more details related to the hardware devices used). In case the computer does not have a serial port, a USB to RS-232 adapter is also required.

At this point two problems were identified. The first was related to the appearance of echo once data was being written on the serial interface. The latter problem consisted in the reception of fragmented messages. Both issues were dealt with using the mechanisms described in Chapter 5. In particular, solving the latter issue led to a significant decrease of the time required to perform a remote test – Table 6.4 – 1 shows a comparison between the recorded time durations required to perform several diagnostic tests before and after fragmentation was eliminated.

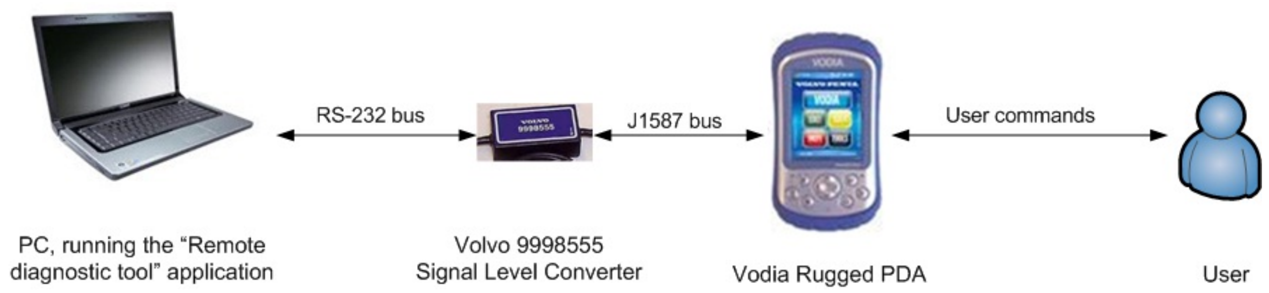

RS-232 bus
J1587 bus
User commands
PC, running the "Remote diagnostic tool" application
Volvo 9998555 Signal Level Converter
Vodia Rugged PDA
User

Figure 6.4: Hardware setup used for testing the "Remote diagnostic tool" – desktop application

During the testing phase, we ran the "Remote diagnostic tool" application on a PC with serial port and we noticed that the message fragmentation was caused by the USB to serial (RS-232) adapter; note that not all adapters have a totally correct functioning and the outcome when programmed, might be different. For example some will fragment the data (like in our case) while others might buffer it for a certain time.

Hence, if the application is running on a PC with serial port, the routines described in Chapter 5 are no longer needed: once data is read into the reception buffer it may be directly sent over the Internet. Otherwise, if the application runs on a PC without serial port, the routine must be kept in order to eliminate the fragmentation introduced by the adapter.

6.4 Prototype's testing and results

The purpose of the project was to build a prototype for remote diagnostics using as much as possible the already available hardware and software platform. Moreover, the project should prove that diagnostic tests may be performed irrespective of location, provided that a similar set-up as the one shown in Figure 6.5 is used. Once the prototype was build, it was equally important to optimize the system such that it holds comparable performance with the present tool employed in the service process (VODIA version 4.4.0). The obtained results were logged and compared with the Navigator's logs. The system must be able to perform successfully the remote log-in and several diagnostic tests (reading of error codes, ECU test and Vessel Configuration Test).

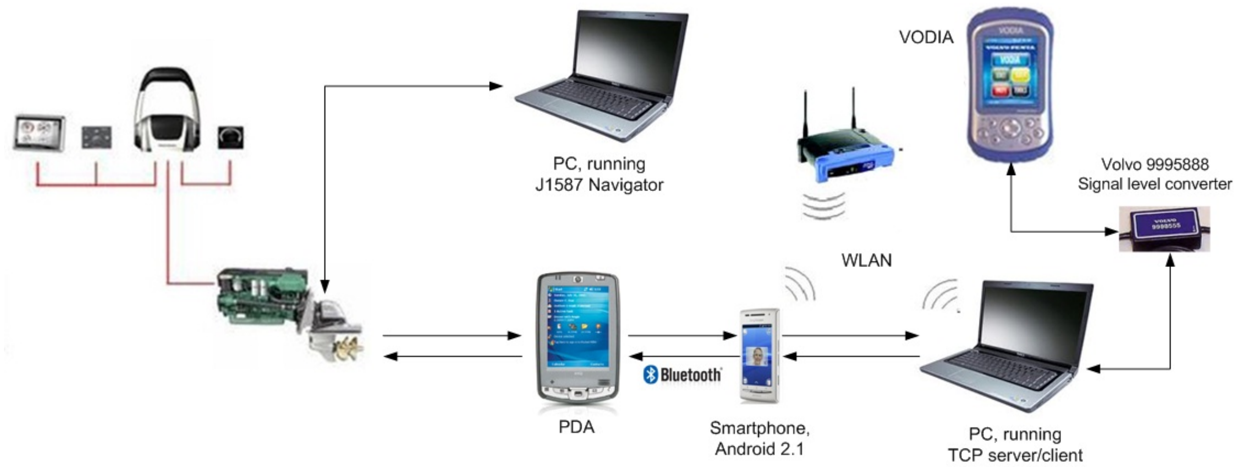


Figure 6.5: Hardware setup used for prototype's testing

The hardware setup used for testing the prototype is displayed in Figure 6.5. The testing was performed both over a WLAN network and over the mobile network. Hence, the service application will read data serially from the VODIA tool and will forward the question, over the network, to the smartphone. The Android application will receive the incoming data and send it, over Bluetooth, to the PDA. The PDA writes the incoming J1587 packet (question) to the serial port. The response from the engine will be relayed back to the service application in a similar manner. When the response is received, the service application will convey it to VODIA. This setup will establish a logical link between VODIA tool and remote engine.

Latency is one of the factors that have a major impact on system's performance. Given that certain time-out values and number of retries are set within the VODIA, it is important to ensure that the round trip time does not exceed those values. Moreover, in case badly formatted messages (questions) reach the engine, they will be discarded and once the resend timer expires, the diagnostic tool attempts another retransmission. Inherently, the time required for a specific test to be performed increases with the number of retires and the performance of the system decreases considerably (note that every time, the VODIA application waits 4[s] to obtain an answer from the engine; in case the question is not replied within the given time, the tool retransmits the question and the procedure repeats until the appropriate answer is received).

Table 6.1 shows the time required for each test to be performed successfully, before and after message fragmentation was eliminated. The obtained values are reasonable given the high number of messages dropped because of their bad format and implicitly the high number of retransmissions that took place.

Diagnostic Test	**VODIA [s]**	**VODIA Remote (Case 1) [s]**	**VODIA Remote (Case 2) [s]**
Log In	40	130	107
Error Codes	25	152	73
ECU Information Test			
MID128 Engine Control Unit	2	80	45
MID164 Helm Control Unit	3	60	6
MID187 Powertrain Control Unit	2	60	4
Vessel Configuration Test	24	180	150

Table 6.1: The time required for the log in phase and several diagnostic tests (WLAN). The PC on which the application was tested had no serial port: thus the USB to RS232 serial adapter was used to connect the signal level converter with the PC. *Case 1: fragmentation was not eliminated; *Case 2: fragmentation was eliminated using the routines described in Chapter 5.

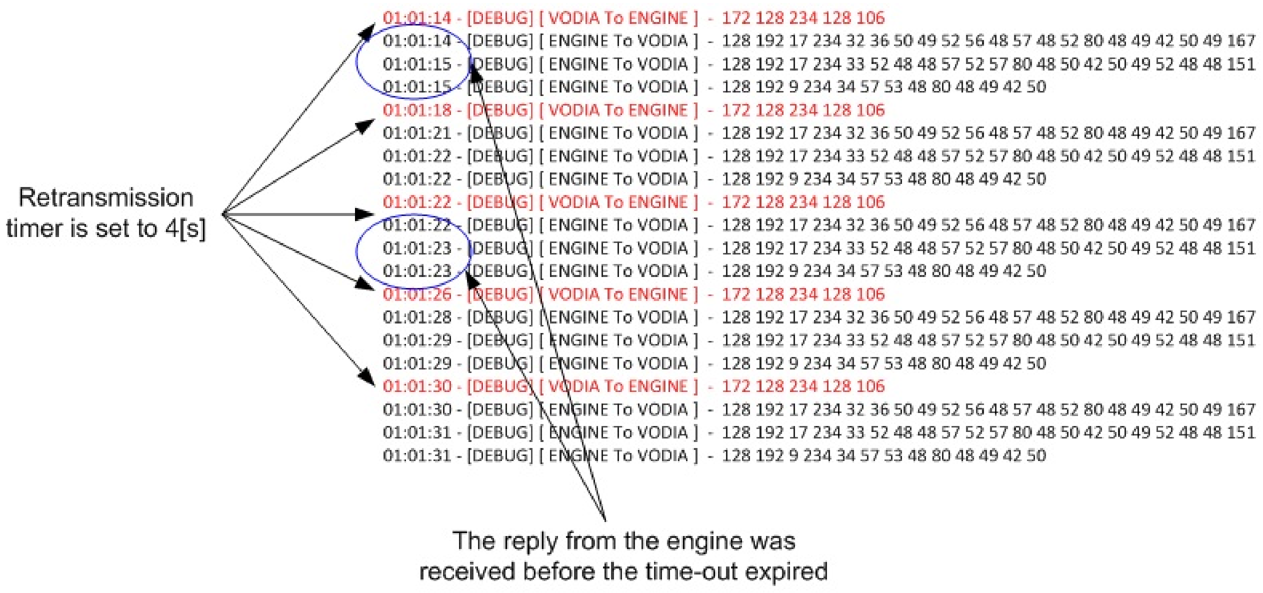
Table 6.2shows the latency results obtained for several diagnostic tests performed when the system was connected over the WLAN network and over the mobile network. As a comparison, the table includes the latency values obtained for the case when the VODIA tool is connected directly to the engine and when it is connected over the mobile network.

Diagnostic Test	**VODIA [s]**	**VODIA Remote (WLAN) [s]**	**VODIA Remote (mobile network) [s]**
Log In	40	107	102
Error Codes	25	73	83
ECU Information Test			
MID128 Engine Control Unit	10	45	47
MID164 Helm Control Unit	3	6	11
MID187 Powertrain Control Unit	2	4	9
Vessel Configuration Test	24	150	168

Table 6.2: Comparison between the times required by VODIA to perform remotely (over a WLAN network and over a mobile network) and the current setup used in the service process (VODIA). Note: the application was tested on a PC without serial port so the USB to RS232 serial adapter was also used with routines described in Chapter 5.

Also, in Appendix A are presented the logs containing the sequence of messages and time duration required for each test to be performed, for the case when VODIA is directly connected to the engine. Appendix B shows the logs extracted from the Java desktop application for the case when VODIA is operating remotely. Note that the sequence of messages is the same and that the time required for engine's reply to reach the PC is around 1[s].

As an observation, the amount of time required for a diagnostic test to be performed is still large compared to the original setup. The difference comes from the fact that one particular question is repeatedly asked, even though the answer appears to be given before the timeout expires (see Figure 6.6). The explanation for this behavior may be that either the reply reaches the VODIA tool after the timeout (less possible since the baud rate for the serial link is 9600[bps] and is sufficiently high in order to transport 56 bytes of data without any noticeable delay!) or the intermediary device that ensures the signal's level conversion from serial to J1708 introduces a certain latency (but again, if this is the cause, all the packets reaching VODIA should be delayed with the same time – which does not happen).


Retransmission timer is set to 4[s]
The reply from the engine was received before the time-out expired

```
01:01:14 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:14 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:18 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:22 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:23 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:23 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:30 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
```

Figure 6.6: One of the issues that appeared when testing the prototype: the question "172 128 234 128 106" was being sent several times even though the reply from the engine was received before the time-out expired. Note that the logs were taken from the Java application, and the time values correspond to the moment the packets reached the desktop computer not the VODIA tool. Based on SAE J1587 specifications, the message corresponds to off-board diagnostics, component specific request from Engine #1.

To confirm the fact that the retransmissions were not due to a small read and write time-out values set within the VODIA tool, we modified these values to 7000[ms]. As expected, this did not solve the aforementioned issue. The solution for this was to add a time spacing of several milliseconds between each message sent over the serial interface. This reduced the number of retransmission and improved the latency exhibited by the system. The cause of this problem might be the high data rate of the serial link compared to the J1708 bus. The next table displays the time values obtained during a final evaluation of the system.

Diagnostic Test	**VODIA [s]**	**VODIA Remote (WLAN) [s]**	**VODIA Remote (mobile network) [s]**
Log In	40	51	80
Error Codes	25	34	47
ECU Information Test			
MID128 Engine Control Unit	2	10	17
MID164 Helm Control Unit	3	6	14
MID187 Powertrain Control Unit	2	4	7
Vessel Configuration Test	24	70	78

Table 6.3: Comparison between the times required by VODIA to perform remotely (over a WLAN network and over a mobile network) and the current setup used in the service process (VODIA). Note: the application was tested on a PC with serial port.

The time duration necessary for the diagnostic tests to be performed remotely depends not only on the inherent behavior of the devices that compose the system (their computational power, the design of the software applications they are running) but also on the overall network's performances. For a complete evaluation of the present prototype, several parameters – such as latency and throughput – need also to be measured.

Latency can be defined as the delay that occurs in a packet switched network when a single packet or datagram travels from one designated point to another. In some cases, the latency is referred to as the round trip time and represents the delay that occurs when a packet travels from a host to a destination and back. Several tools are available for calculating the round trip time between arbitrary hosts on the Internet, out of which Ping is the most frequently used one.

The Ping utility uses ICMP (Internet Control Message Protocol) echo request messages to contact other devices on the network e.g. servers, switches routers etc. Generally the purpose of "ping" is to test whether a certain destination is reachable and responding but it can also serve as a relatively accurate way of measuring the latency. Hence, once the user invokes the "ping" command, an ICMP echo request packet is sent to the given destination. If the target host is up and running it will send back an ICMP echo reply message (the reply contains in fact a copy of the data sent in the request). The results of this command show a statistical summary of received packets, maximum, minimum and average round trip time.

In order to measure the latency we sent 100 ICMP echo-request packets of 21 bytes. Note that in the case of the mobile network, the results of "ping" are highly dependent on the existing network traffic at that time instant. The estimate is more accurate for the WLAN case, where no additional traffic exists. So, the average round-trip time obtained for the WLAN was of 5[ms] while for the mobile network was of 280[ms].

As an example, when VODIA was operated over the mobile network, a rough estimate of the inherent delay that occurred in the Login phase when the tests were performed, was, on average, 14[s] (Login phase had 33 questions (average 6 bytes each), 60 replies (average 20 bytes each) (see Appendix C) and we considered the one-way latency to be of 140[ms]). When operated over the WLAN, the delay during the Log-in phase was around 1[s].

Chapter 7

Conclusions and future developments

This chapter summarizes the most important aspects related to the development of the "Remote connection of diagnostics tool" project. Several theoretical concepts essential for understanding the project's requirements, together with the most relevant phases from the actual development of the applications are briefly recalled here.

7.1 Conclusions

The main task of the thesis was to develop a prototype for the remote connection of the VODIA diagnostic tool. Initially, we investigated several possible configurations that are suitable for such a system, their main advantages and drawbacks; from the presented set-ups we selected the one that could be implemented using the actual software and hardware platforms.

In total, three applications were developed – a PDA, an Android and a Java desktop application – in order to serve the purpose of establishing a communication link between the engine and the diagnostic tool. For each application, we defined the use-cases, established the level of interaction with the user and designed suitable user interfaces. All three applications are based on the client/server model. This type of architecture is relatively easy to be developed, highly flexible, manageable and modular.

During the developing and testing phases, several problems were encountered. One was related to the presence of the echo on the serial interface – since Java was not offering a mechanism to automatically remove echo, we had to implement a mechanism for eliminating the loop backed messages. Also, in some cases, the messages read from the serial or network interface were fragmented, and they were discarded both by the engine and by the diagnostic tool. In order to solve this issue, separate routines were added at the PDA's and desktop's application side – one involved adding an extra field indicating the message length and the other consisted in simply waiting a certain time interval before sending the serially read data over the network.

The resulted system was initially tested over the WLAN network and over the Internet. A comparison between the performances of the currently used diagnostic tool and the remote prototype is made. The improvements that can be made in order to obtain smaller latencies are mentioned in the "Future developments" section.

Remote diagnostics holds many advantages not only in the field of marine industry. Warranty costs may be significantly reduced and a visible improvement in the aftermarket service support can be attained. On the other hand, introducing this new feature may hold a high cost and its true benefit may be acknowledged in a long time by the customers. Nevertheless, given the peace at which mobile platforms evolve, the constituent technologies of such a system will be always threatened by obsolescence [1].

7.2 Future developments

Taking into consideration the actual system's complexity and the relatively high number of devices in the transmission chain and, implicitly the number of software applications involved, several enhancements and optimizations are to be considered. This section presents some of the most relevant improvements that may be brought to the current system.

An important change that may contribute to a more efficient communication between the remote sides consists in eliminating one of the "proxies" (or relays) used in the present set-up, namely the PDA. Note that, at PDA's side there are two concurrent serial connections (specifically, the RFCOMM interface and the serial one) that may act as a bottleneck, increasing the processing time. Given that Google announced the new Android 3.4 OS that will include full support for USB and a completely functional API to support USB application development, the PDA may be eliminated completely from the transmission chain and replaced with the Volvo Vehicle interface Adapter (see Chapter 4.2, Figure 4.2).

Also, at the smartphone's side, the Bluetooth server functionality may also be added. However, in case it is not wished for the user to perform too many operations (refresh the list with available Bluetooth, selecting the device he wishes to connect to), this feature may be masked or hidden and, once the application is launched into execution the connection with the other Bluetooth device may be done automatically.

In case the PDA is further kept, the current application requires a better user interface. Currently, the UI is very simple but it does not have a very good error prevention mechanism and the user has no aid for recognizing, diagnose and helping the app recover from a certain error. Also, the user cannot see whether data transfer actually takes place (basically there is no visibility of the system's status), the help and documentation were not added. Moreover, if the PDAs are equipped with WLAN modules, for the case when an access point (AP) is available in the area, data may be transferred directly over the Internet without the extra need of a smartphone.

For the desktop application, an immediate enhancement would be the implementation of a client queuing mechanism. . Once a client experiences a problem he contacts and connects to the central office's server. The technician may then handle each client one by one. Moreover, such a mechanism can be easily extended to support priorities (the clients may be served based on the gravity of the issue they are facing or simply by the services they requested).

Another nice feature would be a messaging system. When a client connects to the service, he should be able to give a brief description of the problem or simply interact with the service technician using a messaging mechanism.

Currently, data is sent over Internet in plain text. To prevent spoofing or masquerading, data should be sent through a secure channel (SSH, SSL, VPN). In the end, in order to reduce the time associated with reading and writing data to the serial interface, the diagnostics may be done directly using the PC.

Bibliography

[1] Valsan, Anil. The Europeean Vehicle Diagnostics System Market, 2002, www.frost.com/prod/servlet/cpo/2658558.pps

[2] Truck And Bus Low Speed Communication Network Committee, SAE J1708 - Serial Data Communications Between Microcomputer Systems in Heavy - Duty Vehicle Applications, 2010.

[3] Truck And Bus Low Speed Communication Network Committee, SAE J1587 - Electronic Data Interchange Between Microcomputer Systems in Heavy-Duty Vehicle Applications, 2008.

[4] Bluetooth. Specification of the Bluetooth System, 2002, http://www.bluetooth.com/pdf/Bluetooth_11_Specifications_Book.pdf.

[5] Dobrota, Virgirl. Digital networks in telecommunications, Volume III: OSI and TCP/IP, Cluj-Napoca : Mediamira, 2002.

[6] Android Applictaion - DynDns, http://www.appbrain.com/app/dyndns/org.l6n.dyndns

[7] Android Developers Webpage, http://developer.android.com/guide/basics/what-is-android.html.

[8] Embedded .NET User Group, Introduction to MS Embedded Technologies, http://www.embedded.net.nz/_layouts/viewlsts.aspx?BaseType=1

[9] Gartner Inc., http://www.gartner.com/it/page.jsp?id=1543014

[10] Android Developers Webpage, http://developer.android.com/guide/topics/wireless/bluetooth.html

[11] Java Communictaion API, http://www.oracle.com/technetwork/java/index-jsp-141752.html

[12] Eclipse Foundation, SWT (Standard Widget Toolkit), http://www.eclipse.org/swt/

[13] IBM developer works, Java Design Patterns,
http://www.freejavaguide.com/java-design-patterns.pdf

[14] Android Developers Webpage,
http://developer.android.com/guide/topics/fundamentals/activities.html

Appendix A

Navigator Logs for different diagnostic tests

Test Description: Log-in, Duration: 39.67 [s]

```
  0,000 - [VODIA - ENGINE] 172 128 243 228 253
  1,042 - [VODIA - ENGINE] 172 128 243 228 253
  6,119 - [VODIA - ENGINE] 172 128 243 128 097
  6,280 - [ENGINE - VODIA] 128 192 017 243 016 028 128 080 069 078 084 049 042 050 049 048 057 050 049 053 026
  6,311 - [ENGINE - VODIA] 128 192 016 243 017 056 080 048 052 042 048 052 054 056 048 049 050 057 058 190
  6,339 - [VODIA - ENGINE] 172 128 234 128 106
  6,504 - [ENGINE - VODIA] 128 192 017 234 032 036 050 049 052 056 048 057 048 052 080 048 049 042 050 049 167
  6,536 - [ENGINE - VODIA] 128 192 017 234 033 052 048 048 057 052 057 080 048 050 042 050 049 052 048 048 151
  6,553 - [ENGINE - VODIA] 128 192 009 234 034 057 053 048 080 048 049 042 050
  9,956 - [VODIA - ENGINE] 172 000 243 097
10,505 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
10,530 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
10,547 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
10,841 - [ENGINE - VODIA] 164 192 017 243 016 028 164 086 079 076 086 079 042 050 049 052 053 054 053 053 156
10,872 - [ENGINE - VODIA] 164 192 016 243 017 057 080 048 049 042 052 048 048 049 048 048 049 054 058 174
11,017 - [ENGINE - VODIA] 128 192 017 243 016 028 128 080 069 078 084 049 042 050 049 048 057 050 049 053 026
11,041 - [ENGINE - VODIA] 128 192 016 243 017 056 080 048 052 042 048 052 054 056 048 049 050 057 058 190
13,984 - [VODIA - ENGINE] 172 000 243 097
14,536 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
14,561 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
14,579 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
14,873 - [ENGINE - VODIA] 164 192 017 243 016 028 164 086 079 076 086 079 042 050 049 052 053 054 053 053 156
14,903 - [ENGINE - VODIA] 164 192 016 243 017 057 080 048 049 042 052 048 048 049 048 048 049 054 058 174
15,080 - [ENGINE - VODIA] 128 192 017 243 016 028 128 080 069 078 084 049 042 050 049 048 057 050 049 053 026
15,104 - [ENGINE - VODIA] 128 192 016 243 017 056 080 048 052 042 048 052 054 056 048 049 050 057 058 190
15,206 - [VODIA - ENGINE] 172 128 234 187 047
15,816 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
15,842 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
15,865 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
15,893 - [VODIA - ENGINE] 172 128 234 164 070
16,248 - [ENGINE - VODIA] 164 192 017 234 032 036 050 049 054 049 048 052 050 053 080 048 049 042 050 049 138
16,280 - [ENGINE - VODIA] 164 192 017 234 033 053 050 051 048 055 048 080 048 050 042 050 049 053 050 051 118
16,304 - [ENGINE - VODIA] 164 192 009 234 034 048 055 050 080 048 050 042 018
16,332 - [VODIA - ENGINE] 172 128 234 128 106
16,488 - [ENGINE - VODIA] 128 192 017 234 032 036 050 049 052 056 048 057 048 052 080 048 049 042 050 049 167
16,514 - [ENGINE - VODIA] 128 192 017 234 033 052 048 048 057 052 057 080 048 050 042 050 049 052 048 048 151
16,537 - [ENGINE - VODIA] 128 192 009 234 034 057 053 048 080 048 049 042 050
20,948 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
20,984 - [ENGINE - VODIA] 187 254 172 001 226 184
21,018 - [VODIA - ENGINE] 172 254 187 001 007 147
21,064 - [ENGINE - VODIA] 187 254 172 001 009 145
21,101 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
21,172 - [ENGINE - VODIA] 187 254 172 012 231 002 068 101 102 097 117 108 116 048 048 048 081
21,211 - [VODIA - ENGINE] 172 254 187 004 058 001 000 109 239
21,257 - [ENGINE - VODIA] 187 254 172 002 232 001 176
21,376 - [VODIA - ENGINE] 172 254 187 004 211 128 230 187 163
21,414 - [ENGINE - VODIA] 187 254 172 007 211 230 004 002 016 240 004 209
21,494 - [VODIA - ENGINE] 172 254 187 003 105 083 001 219
21,536 - [ENGINE - VODIA] 187 254 172 001 009 145
21,566 - [VODIA - ENGINE] 172 128 243 164 061
22,000 - [ENGINE - VODIA] 164 192 017 243 016 028 164 086 079 076 086 079 042 050 049 052 053 054 053 053 156
22,032 - [ENGINE - VODIA] 164 192 016 243 017 057 080 048 049 042 052 048 048 049 048 048 049 054 058 174
22,060 - [VODIA - ENGINE] 172 128 234 164 070
22,640 - [ENGINE - VODIA] 164 192 017 234 032 036 050 049 054 049 048 052 050 053 080 048 049 042 050 049 138
22,673 - [ENGINE - VODIA] 164 192 017 234 033 053 050 051 048 055 048 080 048 050 042 050 049 053 050 051 118
22,696 - [ENGINE - VODIA] 164 192 009 234 034 048 055 050 080 048 050 042 018
22,799 - [VODIA - ENGINE] 172 128 237 164 067
```

```
23,184 - [ENGINE - VODIA] 164 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 094
23,218 - [VODIA - ENGINE] 172 128 194 164 110
23,319 - [ENGINE - VODIA] 164 194 000 154
23,357 - [VODIA - ENGINE] 172 254 164 004 212 128 194 164 244
23,593 - [ENGINE - VODIA] 164 254 172 009 212 194 006 087 249 001 094 248 004 098
23,633 - [VODIA - ENGINE] 172 254 187 003 105 083 000 220
23,673 - [ENGINE - VODIA] 187 254 172 001 009 145
23,726 - [VODIA - ENGINE] 172 128 247 128 093
23,763 - [ENGINE - VODIA] 128 247 004 000 000 000 000 133
26,215 - [VODIA - ENGINE] 172 128 237 128 103
26,343 - [ENGINE - VODIA] 128 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 130
26,511 - [VODIA - ENGINE] 172 128 194 128 146
30,540 - [VODIA - ENGINE] 172 128 194 128 146
33,244 - [ENGINE - VODIA] 128 192 017 194 032 036 164 165 001 105 165 001 174 165 001 110 165 001 100 165 223
33,272 - [ENGINE - VODIA] 128 192 017 194 033 001 106 165 001 091 226 007 105 229 001 174 229 001 110 229 065
33,295 - [ENGINE - VODIA] 128 192 009 194 034 001 100 229 001 106 229 001 056
33,332 - [VODIA - ENGINE] 172 254 128 004 212 128 194 128 060
37,367 - [VODIA - ENGINE] 172 254 128 004 212 128 194 128 060
38,360 - [ENGINE - VODIA] 128 254 172 015 212 194 012 096 181 001 097 181 001 216 249 002 097 245 001 206
38,687 - [VODIA - ENGINE] 172 254 187 004 058 001 000 181 167
38,752 - [ENGINE - VODIA] 187 254 172 002 232 001 176
38,941 - [VODIA - ENGINE] 172 128 237 187 044
39,304 - [ENGINE - VODIA] 187 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 071
39,433 - [VODIA - ENGINE] 172 128 194 187 087
39,541 - [ENGINE - VODIA] 187 194 000 131
39,576 - [VODIA - ENGINE] 172 254 187 004 212 128 194 187 198
39,674 - [ENGINE - VODIA] 187 254 172 003 212 194 000 002
```

Test Description: ECU Test, MID 187 Powertrain Control Unit , Duration: 2.094 [s]

```
0,000 - [VODIA - ENGINE] 172 128 243 187 038
0,577 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
0,609 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
0,628 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
0,653 - [VODIA - ENGINE] 172 128 234 187 047
1,281 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
1,313 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
1,330 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
1,386 - [VODIA - ENGINE] 172 128 237 187 044
1,729 - [ENGINE - VODIA] 187 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 071
1,833 - [VODIA - ENGINE] 172 254 187 003 042 004 000 106
1,940 - [ENGINE - VODIA] 187 254 172 004 236 004 000 000 167
2,051 - [VODIA - ENGINE] 172 254 187 003 042 002 000 108
2,094 - [ENGINE - VODIA] 187 254 172 004 236 002 000 000 169
```

Test Description: ECU Test, MID 128 Engine Control Unit, Duration: 9.817 [s]

```
0,000 - [VODIA - ENGINE] 172 128 243 128 097
0,141 - [ENGINE - VODIA] 128 192 017 243 016 028 128 080 069 078 084 049 042 050 049 048 057 050 049 053 026
0,165 - [ENGINE - VODIA] 128 192 016 243 017 056 080 048 052 042 048 052 054 056 048 049 050 057 058 190
0,193 - [VODIA - ENGINE] 172 128 234 128 106
0,333 - [ENGINE - VODIA] 128 192 017 234 032 036 050 049 052 056 048 057 048 052 080 048 049 042 050 049 167
0,358 - [ENGINE - VODIA] 128 192 017 234 033 052 048 048 057 052 057 080 048 050 042 050 049 052 048 048 151
0,382 - [ENGINE - VODIA] 128 192 009 234 034 057 053 048 080 048 049 042 050
0,492 - [VODIA - ENGINE] 172 128 237 128 103
0,589 - [ENGINE - VODIA] 128 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 130
0,762 - [VODIA - ENGINE] 172 254 128 003 042 004 000 165
0,864 - [ENGINE - VODIA] 128 254 172 004 236 004 000 002 224
0,899 - [VODIA - ENGINE] 172 254 128 003 042 004 001 164
```

```
1,004 - [ENGINE - VODIA] 128 254 172 015 236 004 001 002 050 049 053 051 050 048 048 053 080 048 049 145
1,046 - [VODIA - ENGINE] 172 254 128 003 042 004 002 163
5,080 - [VODIA - ENGINE] 172 254 128 003 042 004 002 163
9,117 - [VODIA - ENGINE] 172 254 128 003 042 004 002 163
9,228 - [ENGINE - VODIA] 128 254 172 015 236 004 002 002 050 049 053 055 054 055 052 051 080 048 049 127
9,309 - [VODIA - ENGINE] 172 254 128 003 049 001 000 161
9,360 - [ENGINE - VODIA] 128 254 172 004 240 001 000 000 225
9,434 - [VODIA - ENGINE] 172 254 128 003 049 002 000 160
9,480 - [ENGINE - VODIA] 128 254 172 004 240 002 000 000 224
9,595 - [VODIA - ENGINE] 172 254 128 003 042 002 000 167
9,697 - [ENGINE - VODIA] 128 254 172 004 236 002 000 000 228
9,766 - [VODIA - ENGINE] 172 128 247 128 093
9,817 - [ENGINE - VODIA] 128 247 004 000 000 000 000 133
```

Test Description: ECU Test, MID 164 Helm Control unit, Duration: 2.7 [s]

```
0,000 - [VODIA - ENGINE] 172 128 243 164 061
0,312 - [ENGINE - VODIA] 164 192 017 243 016 028 164 086 079 076 086 079 042 050 049 052 053 054 053 053 156
0,336 - [ENGINE - VODIA] 164 192 016 243 017 057 080 048 049 042 052 048 048 049 048 048 049 054 058 174
0,364 - [VODIA - ENGINE] 172 128 234 164 070
0,696 - [ENGINE - VODIA] 164 192 017 234 032 036 050 049 054 049 048 052 050 053 080 048 049 042 050 049 138
0,721 - [ENGINE - VODIA] 164 192 017 234 033 053 050 051 048 055 048 080 048 050 042 050 049 053 050 051 118
0,744 - [ENGINE - VODIA] 164 192 009 234 034 048 055 050 080 048 050 042 018
0,856 - [VODIA - ENGINE] 172 128 237 164 067
1,081 - [ENGINE - VODIA] 164 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 094
1,249 - [VODIA - ENGINE] 172 254 164 003 042 004 000 129
1,324 - [ENGINE - VODIA] 164 254 172 004 236 004 000 004 186
1,359 - [VODIA - ENGINE] 172 254 164 003 042 004 001 128
1,496 - [ENGINE - VODIA] 164 254 172 015 236 004 001 004 050 049 053 054 048 049 054 050 080 048 049 102
1,533 - [VODIA - ENGINE] 172 254 164 003 042 004 002 127
1,688 - [ENGINE - VODIA] 164 254 172 015 236 004 002 004 050 049 053 054 048 048 056 057 080 048 049 093
1,725 - [VODIA - ENGINE] 172 254 164 003 042 004 003 126
1,880 - [ENGINE - VODIA] 164 254 172 015 236 004 003 004 050 049 053 052 049 056 054 053 080 048 049 091
1,918 - [VODIA - ENGINE] 172 254 164 003 042 004 004 125
2,104 - [ENGINE - VODIA] 164 254 172 015 236 004 004 004 050 049 054 049 051 056 048 054 080 048 049 095
2,183 - [VODIA - ENGINE] 172 254 164 003 049 001 000 125
2,220 - [ENGINE - VODIA] 164 254 172 004 240 001 000 000 189
2,284 - [VODIA - ENGINE] 172 254 164 003 049 002 000 124
2,317 - [ENGINE - VODIA] 164 254 172 004 240 002 000 000 188
2,377 - [VODIA - ENGINE] 172 254 164 003 049 003 000 123
2,412 - [ENGINE - VODIA] 164 254 172 004 240 003 000 000 187
2,483 - [VODIA - ENGINE] 172 254 164 003 049 004 000 122
2,534 - [ENGINE - VODIA] 164 254 172 004 240 004 000 000 186
2,648 - [VODIA - ENGINE] 172 254 164 003 042 002 000 131
2,700 - [ENGINE - VODIA] 164 254 172 004 236 002 000 000 192
```

Test Description: Back Button ECU, Duration: 2.09 [s]

```
0,000 - [VODIA - ENGINE] 172 128 243 187 038
0,570 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
0,595 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
0,613 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
0,639 - [VODIA - ENGINE] 172 128 234 187 047
1,242 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
1,267 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
1,290 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
1,636 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
1,682 - [ENGINE - VODIA] 187 254 172 001 226 184
1,715 - [VODIA - ENGINE] 172 254 187 001 007 147
```

```
1,763 - [ENGINE - VODIA] 187 254 172 001 009 145
1,794 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
1,870 - [ENGINE - VODIA] 187 254 172 012 231 002 068 101 102 097 117 108 116 048 048 048 081
1,909 - [VODIA - ENGINE] 172 254 187 004 058 001 000 109 239
1,957 - [ENGINE - VODIA] 187 254 172 002 232 001 176
2,054 - [VODIA - ENGINE] 172 254 187 003 105 083 000 220
2,090 - [ENGINE - VODIA] 187 254 172 001 009 145
```

Test Description: Error Codes, Duration: 23.224 [s]

```
 0,000 - [VODIA - ENGINE] 172 128 243 128 097
 0,165 - [ENGINE - VODIA] 128 192 017 243 016 028 128 080 069 078 084 049 042 050 049 048 057 050 049 053 026
 0,197 - [ENGINE - VODIA] 128 192 016 243 017 056 080 048 052 042 048 052 054 056 048 049 050 057 058 190
 0,225 - [VODIA - ENGINE] 172 128 234 128 106
 0,357 - [ENGINE - VODIA] 128 192 017 234 032 036 050 049 052 056 048 057 048 052 080 048 049 042 050 049 167
 0,383 - [ENGINE - VODIA] 128 192 017 234 033 052 048 048 057 052 057 080 048 050 042 050 049 052 048 048 151
 0,405 - [ENGINE - VODIA] 128 192 009 234 034 057 053 048 080 048 049 042 050
 3,351 - [VODIA - ENGINE] 172 128 243 164 061
 3,822 - [ENGINE - VODIA] 164 192 017 243 016 028 164 086 079 076 086 079 042 050 049 052 053 054 053 053 156
 3,853 - [ENGINE - VODIA] 164 192 016 243 017 057 080 048 049 042 052 048 048 049 048 048 049 054 058 174
 3,881 - [VODIA - ENGINE] 172 128 234 164 070
 4,462 - [ENGINE - VODIA] 164 192 017 234 032 036 050 049 054 049 048 052 050 053 080 048 049 042 050 049 138
 4,494 - [ENGINE - VODIA] 164 192 017 234 033 053 050 051 048 055 048 080 048 050 042 050 049 053 050 051 118
 4,518 - [ENGINE - VODIA] 164 192 009 234 034 048 055 050 080 048 050 042 018
 4,552 - [VODIA - ENGINE] 172 128 243 187 038
 5,094 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
 5,126 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
 5,137 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
 5,163 - [VODIA - ENGINE] 172 128 234 187 047
 5,798 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
 5,823 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
 5,847 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
 5,938 - [VODIA - ENGINE] 172 128 247 128 093
 5,977 - [ENGINE - VODIA] 128 247 004 000 000 000 000 133
 6,051 - [VODIA - ENGINE] 172 128 194 128 146
10,085 - [VODIA - ENGINE] 172 128 194 128 146
12,217 - [ENGINE - VODIA] 128 192 017 194 032 036 164 165 001 105 165 001 174 165 001 110 165 001 100 165 223
12,245 - [ENGINE - VODIA] 128 192 017 194 033 001 106 165 001 091 226 007 105 229 001 174 229 001 110 229 065
12,268 - [ENGINE - VODIA] 128 192 009 194 034 001 100 229 001 106 229 001 056
12,299 - [VODIA - ENGINE] 172 195 003 128 164 229 133
12,362 - [ENGINE - VODIA] 128 196 015 164 229 086 079 076 086 079 255 255 255 255 255 255 255 255 150
12,394 - [VODIA - ENGINE] 172 195 003 128 105 229 192
12,459 - [ENGINE - VODIA] 128 196 015 105 229 086 079 076 086 079 255 255 255 255 255 255 255 255 209
12,491 - [VODIA - ENGINE] 172 195 003 128 174 229 123
12,555 - [ENGINE - VODIA] 128 196 015 174 229 086 079 076 086 079 255 255 255 255 255 255 255 255 140
12,586 - [VODIA - ENGINE] 172 195 003 128 110 229 187
12,651 - [ENGINE - VODIA] 128 196 015 110 229 086 079 076 086 079 255 255 255 255 255 255 255 255 204
12,683 - [VODIA - ENGINE] 172 195 003 128 100 229 197
12,747 - [ENGINE - VODIA] 128 196 015 100 229 086 079 076 086 079 255 255 255 255 255 255 255 255 214
12,779 - [VODIA - ENGINE] 172 195 003 128 106 229 191
12,843 - [ENGINE - VODIA] 128 196 015 106 229 086 079 076 086 079 255 255 255 255 255 255 255 255 208
12,875 - [VODIA - ENGINE] 172 195 003 128 091 226 209
12,938 - [ENGINE - VODIA] 128 196 015 091 226 086 079 076 086 079 255 255 255 255 255 255 255 255 226
12,970 - [VODIA - ENGINE] 172 195 003 128 105 229 192
13,035 - [ENGINE - VODIA] 128 196 015 105 229 086 079 076 086 079 255 255 255 255 255 255 255 255 209
13,068 - [VODIA - ENGINE] 172 195 003 128 174 229 123
13,130 - [ENGINE - VODIA] 128 196 015 174 229 086 079 076 086 079 255 255 255 255 255 255 255 255 140
13,162 - [VODIA - ENGINE] 172 195 003 128 110 229 187
13,228 - [ENGINE - VODIA] 128 196 015 110 229 086 079 076 086 079 255 255 255 255 255 255 255 255 204
```

```
13,259 - [VODIA - ENGINE] 172 195 003 128 100 229 197
13,322 - [ENGINE – VODIA] 128 196 015 100 229 086 079 076 086 079 255 255 255 255 255 255 255 255 214
13,354 - [VODIA - ENGINE] 172 195 003 128 106 229 191
13,419 - [ENGINE - VODIA] 128 196 015 106 229 086 079 076 086 079 255 255 255 255 255 255 255 255 208
14,020 - [VODIA - ENGINE] 172 254 128 004 212 128 194 128 060
18,056 - [VODIA - ENGINE] 172 254 128 004 212 128 194 128 060
20,661 - [ENGINE - VODIA] 128 254 172 015 212 194 012 096 181 001 097 181 001 216 249 002 097 245 001 206
20,707 - [VODIA - ENGINE] 172 254 128 006 212 195 003 128 096 245 097
20,766 - [ENGINE - VODIA] 128 192 017 254 016 020 172 018 212 196 015 096 245 086 079 076 086 079 255 255 063
20,788 - [ENGINE - VODIA] 128 192 008 254 017 255 255 255 255 255 255 175
20,823 - [VODIA - ENGINE] 172 254 128 006 212 195 003 128 097 245 096
20,885 - [ENGINE - VODIA] 128 192 017 254 016 020 172 018 212 196 015 097 245 086 079 076 086 079 255 255 062
20,900 - [ENGINE - VODIA] 128 192 008 254 017 255 255 255 255 255 255 175
20,936 - [VODIA - ENGINE] 172 254 128 006 212 195 003 128 216 249 229
21,005 - [ENGINE - VODIA] 128 192 017 254 016 020 172 018 212 196 015 216 249 086 079 076 086 079 255 255 195
21,020 - [ENGINE - VODIA] 128 192 008 254 017 255 255 255 255 255 255 175
21,057 - [VODIA - ENGINE] 172 254 128 006 212 195 003 128 097 245 096
21,125 - [ENGINE - VODIA] 128 192 017 254 016 020 172 018 212 196 015 097 245 086 079 076 086 079 255 255 062
21,141 - [ENGINE - VODIA] 128 192 008 254 017 255 255 255 255 255 255 175
21,908 - [VODIA - ENGINE] 172 128 194 164 110
21,987 - [ENGINE - VODIA] 164 194 000 154
22,022 - [VODIA - ENGINE] 172 254 164 004 212 128 194 164 244
22,158 - [ENGINE - VODIA] 164 254 172 009 212 194 006 087 249 001 094 248 004 098
22,199 - [VODIA - ENGINE] 172 254 164 006 212 195 003 164 087 249 030
22,389 - [ENGINE - VODIA] 164 192 017 254 016 020 172 018 212 196 015 087 249 086 079 076 086 079 255 255 032
22,412 - [ENGINE - VODIA] 164 192 008 254 017 255 255 255 255 255 255 139
22,447 - [VODIA - ENGINE] 172 254 164 006 212 195 003 164 094 248 024
22,645 - [ENGINE - VODIA] 164 192 017 254 016 020 172 018 212 196 015 094 248 086 079 076 086 079 255 255 026
22,668 - [ENGINE - VODIA] 164 192 008 254 017 255 255 255 255 255 255 139
23,001 - [VODIA - ENGINE] 172 128 194 187 087
23,092 - [ENGINE - VODIA] 187 194 000 131
23,127 - [VODIA - ENGINE] 172 254 187 004 212 128 194 187 198
23,224 - [ENGINE - VODIA] 187 254 172 003 212 194 000 002
```

Test Description: Error Codes Back Button, Duration: 2.053[s]

```
0,000 - [VODIA - ENGINE] 172 128 243 187 038
0,532 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
0,558 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
0,575 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
0,601 - [VODIA - ENGINE] 172 128 234 187 047
1,204 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
1,230 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
1,253 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
1,601 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
1,645 - [ENGINE - VODIA] 187 254 172 001 226 184
1,677 - [VODIA - ENGINE] 172 254 187 001 007 147
1,727 - [ENGINE - VODIA] 187 254 172 001 009 145
1,757 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
1,833 - [ENGINE - VODIA] 187 254 172 012 231 002 068 101 102 097 117 108 116 048 048 048 081
1,871 - [VODIA - ENGINE] 172 254 187 004 058 001 000 109 239
1,919 - [ENGINE - VODIA] 187 254 172 002 232 001 176
2,016 - [VODIA - ENGINE] 172 254 187 003 105 083 000 220
2,053 - [ENGINE - VODIA] 187 254 172 001 009 145
```

Test Description: Vessel Configuration Test, Duration: 21.036 [s]

```
0,000 - [VODIA - ENGINE] 172 128 243 128 097
0,140 - [ENGINE - VODIA] 128 192 017 243 016 028 128 080 069 078 084 049 042 050 049 048 057 050 049 053 026
```

```
0,164 - [ENGINE - VODIA] 128 192 016 243 017 056 080 048 052 042 048 052 054 056 048 049 050 057 058 190
0,193 - [VODIA - ENGINE] 172 128 234 128 106
0,332 - [ENGINE - VODIA] 128 192 017 234 032 036 050 049 052 056 048 057 048 052 080 048 049 042 050 049 167
0,357 - [ENGINE - VODIA] 128 192 017 234 033 052 048 048 057 052 057 080 048 050 042 050 049 052 048 048 151
0,380 - [ENGINE - VODIA] 128 192 009 234 034 057 053 048 080 048 049 042 050
0,418 - [VODIA - ENGINE] 172 128 243 187 038
0,972 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
  0,997 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
  1,015 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
  1,043 - [VODIA - ENGINE] 172 128 234 187 047
  1,644 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
  1,676 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
  1,693 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
  3,827 - [VODIA - ENGINE] 172 128 243 128 097
  3,980 - [ENGINE - VODIA] 128 192 017 243 016 028 128 080 069 078 084 049 042 050 049 048 057 050 049 053 026
  4,005 - [ENGINE - VODIA] 128 192 016 243 017 056 080 048 052 042 048 052 054 056 048 049 050 057 058 190
  4,033 - [VODIA - ENGINE] 172 128 234 128 106
  4,172 - [ENGINE - VODIA] 128 192 017 234 032 036 050 049 052 056 048 057 048 052 080 048 049 042 050 049 167
  4,197 - [ENGINE - VODIA] 128 192 017 234 033 052 048 048 057 052 057 080 048 050 042 050 049 052 048 048 151
  4,221 - [ENGINE - VODIA] 128 192 009 234 034 057 053 048 080 048 049 042 050
  4,258 - [VODIA - ENGINE] 172 128 243 187 038
  4,812 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
  4,837 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
  4,855 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
  4,881 - [VODIA - ENGINE] 172 128 234 187 047
  5,485 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
  5,516 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
  5,533 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
  5,653 - [VODIA - ENGINE] 172 128 237 128 103
  5,772 - [ENGINE - VODIA] 128 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 130
  5,829 - [VODIA - ENGINE] 172 128 243 128 097
  5,996 - [ENGINE - VODIA] 128 192 017 243 016 028 128 080 069 078 084 049 042 050 049 048 057 050 049 053 026
  6,020 - [ENGINE - VODIA] 128 192 016 243 017 056 080 048 052 042 048 052 054 056 048 049 050 057 058 190
  6,048 - [VODIA - ENGINE] 172 128 234 128 106
  6,187 - [ENGINE - VODIA] 128 192 017 234 032 036 050 049 052 056 048 057 048 052 080 048 049 042 050 049 167
  6,212 - [ENGINE - VODIA] 128 192 017 234 033 052 048 048 057 052 057 080 048 050 042 050 049 052 048 048 151
  6,236 - [ENGINE - VODIA] 128 192 009 234 034 057 053 048 080 048 049 042 050
  6,322 - [VODIA - ENGINE] 172 128 237 128 103
  6,444 - [ENGINE - VODIA] 128 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 130
  6,716 - [VODIA - ENGINE] 172 254 128 003 042 004 000 165
  6,815 - [ENGINE - VODIA] 128 254 172 004 236 004 000 002 224
  6,850 - [VODIA - ENGINE] 172 254 128 003 042 004 001 164
  6,954 - [ENGINE - VODIA] 128 254 172 015 236 004 001 002 050 049 053 051 050 048 048 053 080 048 049 145
  6,992 - [VODIA - ENGINE] 172 254 128 003 042 004 002 163
  7,114 - [ENGINE - VODIA] 128 254 172 015 236 004 002 002 050 049 053 055 054 055 052 051 080 048 049 127
  7,205 - [VODIA - ENGINE] 172 254 128 003 049 001 000 161
  7,247 - [ENGINE - VODIA] 128 254 172 004 240 001 000 000 225
  7,330 - [VODIA - ENGINE] 172 254 128 003 049 002 000 160
  7,367 - [ENGINE - VODIA] 128 254 172 004 240 002 000 000 224
  7,415 - [VODIA - ENGINE] 172 128 194 128 146
11,448 - [VODIA - ENGINE] 172 128 194 128 146
12,160 - [ENGINE - VODIA] 128 192 017 194 032 036 164 165 001 105 165 001 174 165 001 110 165 001 100 165 223
12,187 - [ENGINE - VODIA] 128 192 017 194 033 001 106 165 001 091 226 007 105 229 001 174 229 001 110 229 065
12,211 - [ENGINE - VODIA] 128 192 009 194 034 001 100 229 001 106 229 001 056
12,251 - 172 254 128 004 212 128 194 128 060
16,286 - 172 254 128 004 212 128 194 128 060
17,021 - [ENGINE - VODIA] 128 254 172 015 212 194 012 096 181 001 097 181 001 216 249 002 097 245 001 206
17,073 - [VODIA - ENGINE] 172 128 243 187 038
```

```
17,612 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
17,644 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
17,655 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
17,681 - [VODIA - ENGINE] 172 128 234 187 047
18,283 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
18,308 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
18,332 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
18,409 - [VODIA - ENGINE] 172 128 237 187 044
18,763 - [ENGINE - VODIA] 187 237 017 086 086 032 032 032 048 048 048 056 052 056 032 032 032 032 032 032 071
18,965 - [VODIA - ENGINE] 172 254 187 003 042 004 000 106
19,070 - [ENGINE - VODIA] 187 254 172 004 236 004 000 000 167
19,117 - [VODIA - ENGINE] 172 128 194 187 087
19,193 - [ENGINE - VODIA] 187 194 000 131
19,228 - [VODIA - ENGINE] 172 254 187 004 212 128 194 187 198
19,326 - [ENGINE - VODIA] 187 254 172 003 212 194 000 002
19,609 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
19,659 - [ENGINE - VODIA] 187 254 172 001 226 184
19,691 - [VODIA - ENGINE] 172 254 187 001 007 147
19,739 - [ENGINE - VODIA] 187 254 172 001 009 145
19,771 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
19,847 - [ENGINE - VODIA] 187 254 172 012 231 002 068 101 102 097 117 108 116 048 048 048 081
19,880 - [VODIA - ENGINE] 172 254 187 004 058 001 000 121 227
19,932 - [ENGINE - VODIA] 187 254 172 002 232 000 177
20,192 - [VODIA - ENGINE] 172 254 187 004 058 001 000 103 245
20,245 - [ENGINE - VODIA] 187 254 172 002 232 000 177
20,458 - [VODIA - ENGINE] 172 254 187 004 058 001 000 124 224
20,508 - [ENGINE - VODIA] 187 254 172 002 232 000 177
20,706 - [VODIA - ENGINE] 172 254 187 004 058 001 000 123 225
20,773 - [ENGINE - VODIA] 187 254 172 002 232 000 177
20,975 - [VODIA - ENGINE] 172 254 187 004 058 001 000 116 232
21,036 - [ENGINE - VODIA] 187 254 172 002 232 000 177
```

Test Description: Vessel Configuration Test Back Button, Duration: 2.164 [s]

```
0,000 - [VODIA - ENGINE] 172 128 243 187 038
0,555 - [ENGINE - VODIA] 187 192 017 243 032 031 187 086 079 076 086 079 042 050 049 052 054 057 048 054 091
0,587 - [ENGINE - VODIA] 187 192 017 243 033 050 080 048 049 042 048 048 050 050 050 056 055 050 058 120 010
0,605 - [ENGINE - VODIA] 187 192 004 243 034 004 025 079
0,631 - [VODIA - ENGINE] 172 128 234 187 047
1,259 - [ENGINE - VODIA] 187 192 017 234 032 036 050 049 054 049 048 052 052 048 080 048 049 042 050 049 118
1,284 - [ENGINE - VODIA] 187 192 017 234 033 053 050 051 049 048 055 080 048 050 042 050 049 053 052 051 092
1,308 - [ENGINE - VODIA] 187 192 009 234 034 051 053 053 080 048 049 042 248
1,652 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
1,708 - [ENGINE - VODIA] 187 254 172 001 226 184
1,740 - [VODIA - ENGINE] 172 254 187 001 007 147
1,788 - [ENGINE - VODIA] 187 254 172 001 009 145
1,818 - [VODIA - ENGINE] 172 254 187 004 057 002 069 070 209
1,895 - [ENGINE - VODIA] 187 254 172 012 231 002 068 101 102 097 117 108 116 048 048 048 081
1,935 - [VODIA - ENGINE] 172 254 187 004 058 001 000 109 239
1,981 - [ENGINE - VODIA] 187 254 172 002 232 001 176
2,106 - [VODIA - ENGINE] 172 254 187 003 105 083 000 220
2,164 - [ENGINE - VODIA] 187 254 172 001 009 145
```

Appendix B

Java desktop application logs for different diagnostic tests

Test Description: Log In , Time Interval: 01:00:47 - 01:02:46, Duration: 59 [s]

```
01:00:47 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 228 253
01:00:48 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 228 253
01:00:54 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
01:00:55 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
01:00:55 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
01:00:55 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:00:56 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:00:57 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:00:57 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:00:57 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:00:58 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:00:59 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:00:59 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:00:59 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:00 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:01 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:01 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:07 - [DEBUG] [ VODIA To ENGINE ]  -  172 0 243 97
01:01:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
01:01:10 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
01:01:10 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
01:01:10 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
01:01:10 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
01:01:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
01:01:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
01:01:11 - [DEBUG] [ VODIA To ENGINE ]  -  172 0 243 97
01:01:12 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
01:01:12 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
01:01:12 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
01:01:12 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
01:01:12 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
01:01:13 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
01:01:13 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
01:01:14 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:14 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:18 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:22 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:23 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:23 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:30 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
```

```
01:01:38 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
01:01:40 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
01:01:40 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
01:01:40 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
01:01:40 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
01:01:41 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
01:01:42 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
01:01:42 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
01:01:46 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:48 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:49 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:49 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:50 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:51 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:52 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:52 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:54 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:01:55 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:01:56 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:01:56 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:01:58 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:02:00 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:02:01 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:02:01 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:02:02 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
01:02:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
01:02:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
01:02:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
01:02:07 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
01:02:09 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
01:02:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
01:02:10 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
01:02:10 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
01:02:11 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
01:02:11 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 109 239
01:02:11 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
01:02:11 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 211 128 230 187 163
01:02:12 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 7 211 230 4 2 16 240 4 209
01:02:12 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 1 219
01:02:12 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
01:02:12 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 164 61
01:02:13 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
01:02:14 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
01:02:14 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 164 70
01:02:15 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 32 36 50 49 54 49 48 52 50 53 80 48 49 42 50 49 138
01:02:15 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 33 53 50 51 48 55 48 80 48 50 42 50 49 53 50 51 118
01:02:15 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 9 234 34 48 55 50 80 48 50 42 18
01:02:15 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 164 67
01:02:16 - [DEBUG] [ ENGINE To VODIA ]  -  164 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 94
01:02:16 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 164 110
01:02:17 - [DEBUG] [ ENGINE To VODIA ]  -  164 194 0 154
01:02:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 4 212 128 194 164 244
01:02:17 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 9 212 194 6 87 249 1 94 248 4 98
01:02:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 0 220
01:02:18 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
01:02:18 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 181 167
01:02:19 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
01:02:19 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 187 44
```

```
01:02:20 - [DEBUG] [ ENGINE To VODIA ]  -  187 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 71
01:02:20 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 187 87
01:02:20 - [DEBUG] [ ENGINE To VODIA ]  -  187 194 2 231 60 94
01:02:21 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 212 128 194 187 198
01:02:21 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 9 212 194 6 200 57 232 114 226 121 64
01:02:21 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 247 128 93
01:02:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 247 4 0 0 0 0 133
01:02:24 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 128 103
01:02:25 - [DEBUG] [ ENGINE To VODIA ]  -  128 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 130
01:02:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
01:02:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
01:02:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 16 21 164 165 1 91 162 7 105 165 1 174 165 1 110 165 4
01:02:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 17 1 100 165 1 106 165 1 201
01:02:31 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
01:02:32 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 16 21 164 165 1 91 162 7 105 165 1 174 165 1 110 165 4
01:02:33 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 17 1 100 165 1 106 165 1 201
01:02:35 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
01:02:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
01:02:41 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 12 212 194 9 96 181 1 97 181 1 216 249 2 43
01:02:46 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 12 212 194 9 96 181 1 97 181 1 216 249 2 43
```

ECU – MID 187 Powertrain Control Unit, Time Interval: 02:07:58 - 02:08:10, Duration: 12 [s]

```
02:07:58 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
02:08:01 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
02:08:02 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
02:08:02 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
02:08:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
02:08:07 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
02:08:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
02:08:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
02:08:08 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 187 44
02:08:09 - [DEBUG] [ ENGINE To VODIA ]  -  187 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 71
02:08:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 42 4 0 106
02:08:10 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 4 236 4 0 0 167
02:08:10 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 42 2 0 108
02:08:10 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 4 236 2 0 0 169
```

ECU – MID 128 Engine Control Unit, Time Interval: 02:10:33 - 02:11:24, Duration: 51 [s]

```
02:10:33 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
02:10:36 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
02:10:37 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
02:10:37 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:10:38 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:10:38 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:10:38 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:10:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:10:42 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:10:43 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:10:43 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:10:45 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:10:46 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:10:46 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:10:46 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:10:49 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:10:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:10:51 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:10:51 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
```

```
02:10:53 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:10:54 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:10:54 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:10:54 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:10:57 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:10:58 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:10:59 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:10:59 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:11:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:11:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:11:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:11:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:11:05 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:11:08 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:11:09 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:11:09 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:11:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:11:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:11:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:11:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:11:13 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:11:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:11:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:11:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:11:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 128 103
02:11:18 - [DEBUG] [ ENGINE To VODIA ]  -  128 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 130
02:11:18 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 0 165
02:11:19 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 236 4 0 2 224
02:11:19 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 1 164
02:11:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 236 4 1 2 50 49 53 51 50 48 48 53 80 48 49 145
02:11:20 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 2 163
02:11:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 236 4 2 2 50 49 53 55 54 55 52 51 80 48 49 127
02:11:21 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 49 1 0 161
02:11:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 240 1 0 0 225
02:11:22 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 49 2 0 160
02:11:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 240 2 0 0 224
02:11:22 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 2 0 167
02:11:23 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 236 2 0 0 228
02:11:23 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 247 128 93
02:11:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 247 4 0 0 0 0 133
```

ECU – MID 164, Helm Control Unit, Time Interval: 02:12:23 - 02:12:38, Duration: 15 [s]

```
02:12:23 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 164 61
02:12:27 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
02:12:27 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 164 61
02:12:27 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
02:12:28 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
02:12:28 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
02:12:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 164 70
02:12:29 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 32 36 50 49 54 49 48 52 50 53 80 48 49 42 50 49 138
02:12:29 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 33 53 50 51 48 55 48 80 48 50 42 50 49 53 50 51 118
02:12:29 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 9 234 34 48 55 50 80 48 50 42 18
02:12:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 164 67
02:12:32 - [DEBUG] [ ENGINE To VODIA ]  -  164 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 94
02:12:32 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 0 129
02:12:32 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 236 4 0 4 186
02:12:32 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 1 128
02:12:33 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 15 236 4 1 4 50 49 53 54 48 49 54 50 80 48 49 102
```

```
02:12:33 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 2 127
02:12:34 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 15 236 4 2 4 50 49 53 54 48 48 56 57 80 48 49 93
02:12:34 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 3 126
02:12:34 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 15 236 4 3 4 50 49 53 52 49 56 54 53 80 48 49 91
02:12:34 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 4 125
02:12:35 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 15 236 4 4 4 50 49 54 49 51 56 48 54 80 48 49 95
02:12:35 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 49 1 0 125
02:12:36 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 240 1 0 0 189
02:12:36 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 49 2 0 124
02:12:36 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 240 2 0 0 188
02:12:36 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 49 3 0 123
02:12:37 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 240 3 0 0 187
02:12:37 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 49 4 0 122
02:12:38 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 240 4 0 0 186
02:12:38 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 2 0 131
02:12:38 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 236 2 0 0 192
```

Back Button ECU Information Test, Time Interval: 02:13:19 - 02:13:27, Duration: 8 [s]

```
02:13:19 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
02:13:22 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
02:13:22 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
02:13:22 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
02:13:22 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
02:13:23 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
02:13:24 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
02:13:24 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
02:13:24 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
02:13:25 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
02:13:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
02:13:25 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
02:13:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
02:13:26 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
02:13:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 109 239
02:13:26 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
02:13:27 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 0 220
02:13:27 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
```

Error Codes, Time Interval: 02:18:08 - 02:19:42, Duration: 94 [s]

```
02:18:08 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
02:18:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
02:18:12 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
02:18:12 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:12 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:13 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:13 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:18:16 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:17 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:17 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:17 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:18:20 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:18:24 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
```

```
02:18:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:18:32 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:32 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:33 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:33 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:18:36 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:36 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:37 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:37 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:18:40 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:42 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:43 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:43 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:18:44 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:45 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:45 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:45 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:18:48 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
02:18:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
02:18:51 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
02:18:51 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
02:19:00 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
02:19:02 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
02:19:03 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
02:19:03 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
02:19:03 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
02:19:04 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
02:19:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
02:19:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
02:19:05 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 164 61
02:19:06 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
02:19:07 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
02:19:07 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 164 70
02:19:08 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 32 36 50 49 54 49 48 52 50 53 80 48 49 42 50 49 138
02:19:08 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 33 53 50 51 48 55 48 80 48 50 42 50 49 53 50 51 118
02:19:09 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 9 234 34 48 55 50 80 48 50 42 18
02:19:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 247 128 93
02:19:09 - [DEBUG] [ ENGINE To VODIA ]  -  128 247 4 0 0 0 0 133
02:19:10 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
02:19:12 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 16 21 164 165 1 91 162 7 105 165 1 174 165 1 110 165 4
02:19:13 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 17 1 100 165 1 106 165 1 201
02:19:13 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 164 229 133
02:19:13 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 164 229 86 79 76 86 79 255 255 255 255 255 255 255 255 150
02:19:14 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 91 226 209
02:19:14 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 91 226 86 79 76 86 79 255 255 255 255 255 255 255 255 226
02:19:14 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 105 229 192
02:19:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 105 229 86 79 76 86 79 255 255 255 255 255 255 255 255 209
02:19:15 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 174 229 123
02:19:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 174 229 86 79 76 86 79 255 255 255 255 255 255 255 255 140
02:19:16 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 110 229 187
02:19:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 110 229 86 79 76 86 79 255 255 255 255 255 255 255 255 204
02:19:16 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 100 229 197
02:19:17 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 100 229 86 79 76 86 79 255 255 255 255 255 255 255 255 214
02:19:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 106 229 191
02:19:18 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 106 229 86 79 76 86 79 255 255 255 255 255 255 255 255 208
```

```
02:19:18 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
02:19:22 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
02:19:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 12 212 194 9 96 181 1 97 181 1 216 249 2 43
02:19:24 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 96 245 97
02:19:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 12 212 194 9 96 181 1 97 181 1 216 249 2 43
02:19:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 96 245 97
02:19:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 96 245 86 79 76 86 79 255 255 63
02:19:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
02:19:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 97 245 96
02:19:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 97 245 86 79 76 86 79 255 255 62
02:19:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
02:19:30 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 216 249 229
02:19:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 216 249 86 79 76 86 79 255 255 195
02:19:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
02:19:32 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 187 87
02:19:32 - [DEBUG] [ ENGINE To VODIA ]  -  187 194 2 231 60 94
02:19:32 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 187 231 252 240
02:19:33 - [DEBUG] [ ENGINE To VODIA ]  -  187 196 15 231 252 86 79 76 86 79 255 255 255 255 255 255 255 255 1
02:19:33 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 212 128 194 187 198
02:19:34 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 9 212 194 6 200 57 232 114 226 121 64
02:19:34 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 6 212 195 3 187 200 249 127
02:19:35 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 254 16 20 172 18 212 196 15 200 249 86 79 76 86 79 255 255 152
02:19:36 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 8 254 17 255 255 255 255 255 255 116
02:19:36 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 6 212 195 3 187 232 242 102
02:19:36 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 254 16 20 172 18 212 196 15 232 242 86 79 76 86 79 255 255 127
02:19:37 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 8 254 17 255 255 255 255 255 255 116
02:19:37 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 6 212 195 3 187 226 249 101
02:19:38 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 254 16 20 172 18 212 196 15 226 249 86 79 76 86 79 255 255 126
02:19:38 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 8 254 17 255 255 255 255 255 255 116
02:19:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 164 110
02:19:39 - [DEBUG] [ ENGINE To VODIA ]  -  164 194 0 154
02:19:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 4 212 128 194 164 244
02:19:40 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 9 212 194 6 87 249 1 94 248 4 98
02:19:40 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 6 212 195 3 164 87 249 30
02:19:41 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 254 16 20 172 18 212 196 15 87 249 86 79 76 86 79 255 255 32
02:19:41 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 8 254 17 255 255 255 255 255 255 139
02:19:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 6 212 195 3 164 94 248 24
02:19:42 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 254 16 20 172 18 212 196 15 94 248 86 79 76 86 79 255 255 26
02:19:42 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 8 254 17 255 255 255 255 255 255 139
```

Replay Error Codes, Time Interval: 02:21:02 - 02:21:43, Duration: 41 [s]

```
02:21:02 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 247 128 93
02:21:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 247 128 93
02:21:06 - [DEBUG] [ ENGINE To VODIA ]  -  128 247 4 0 0 0 0 133
02:21:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
02:21:07 - [DEBUG] [ ENGINE To VODIA ]  -  128 247 4 0 0 0 0 133
02:21:10 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
02:21:12 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 16 21 164 165 1 91 162 7 105 165 1 174 165 1 110 165 4
02:21:12 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 17 1 100 165 1 106 165 1 201
02:21:12 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 164 229 133
02:21:14 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 16 21 164 165 1 91 162 7 105 165 1 174 165 1 110 165 4
02:21:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 17 1 100 165 1 106 165 1 201
02:21:16 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 164 229 133
02:21:17 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 164 229 86 79 76 86 79 255 255 255 255 255 255 255 255 150
02:21:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 91 226 209
02:21:17 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 91 226 86 79 76 86 79 255 255 255 255 255 255 255 255 226
02:21:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 105 229 192
02:21:18 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 105 229 86 79 76 86 79 255 255 255 255 255 255 255 255 209
```

```
02:21:18 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 174 229 123
02:21:19 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 174 229 86 79 76 86 79 255 255 255 255 255 255 255 255 140
02:21:23 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 110 229 187
02:21:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 110 229 86 79 76 86 79 255 255 255 255 255 255 255 255 204
02:21:24 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 100 229 197
02:21:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 100 229 86 79 76 86 79 255 255 255 255 255 255 255 255 214
02:21:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 106 229 191
02:21:25 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 106 229 86 79 76 86 79 255 255 255 255 255 255 255 255 208
02:21:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
02:21:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 12 212 194 9 96 181 1 97 181 1 216 249 2 43
02:21:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 96 245 97
02:21:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 96 245 86 79 76 86 79 255 255 63
02:21:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
02:21:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 97 245 96
02:21:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 97 245 86 79 76 86 79 255 255 62
02:21:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
02:21:30 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 216 249 229
02:21:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 216 249 86 79 76 86 79 255 255 195
02:21:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
02:21:32 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 187 87
02:21:33 - [DEBUG] [ ENGINE To VODIA ]  -  187 194 2 231 60 94
02:21:33 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 187 231 252 240
02:21:33 - [DEBUG] [ ENGINE To VODIA ]  -  187 196 15 231 252 86 79 76 86 79 255 255 255 255 255 255 255 255 1
02:21:34 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 212 128 194 187 198
02:21:34 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 9 212 194 6 200 57 232 114 226 121 64
02:21:34 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 6 212 195 3 187 200 249 127
02:21:35 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 254 16 20 172 18 212 196 15 200 249 86 79 76 86 79 255 255 152
02:21:36 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 8 254 17 255 255 255 255 255 255 116
02:21:36 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 6 212 195 3 187 232 242 102
02:21:37 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 254 16 20 172 18 212 196 15 232 242 86 79 76 86 79 255 255 127
02:21:37 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 8 254 17 255 255 255 255 255 255 116
02:21:37 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 6 212 195 3 187 226 249 101
02:21:38 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 254 16 20 172 18 212 196 15 226 249 86 79 76 86 79 255 255 126
02:21:39 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 8 254 17 255 255 255 255 255 255 116
02:21:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 164 110
02:21:40 - [DEBUG] [ ENGINE To VODIA ]  -  164 194 0 154
02:21:40 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 4 212 128 194 164 244
02:21:40 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 9 212 194 6 87 249 1 94 248 4 98
02:21:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 6 212 195 3 164 87 249 30
02:21:41 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 254 16 20 172 18 212 196 15 87 249 86 79 76 86 79 255 255 32
02:21:42 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 8 254 17 255 255 255 255 255 255 139
02:21:42 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 6 212 195 3 164 94 248 24
02:21:43 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 254 16 20 172 18 212 196 15 94 248 86 79 76 86 79 255 255 26
02:21:43 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 8 254 17 255 255 255 255 255 255 139
```

Back Button (Error Codes), Time Interval: 02:22:56 – 02:23:08, Duration: 12 [s]

```
02:22:56 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
02:22:59 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
02:23:00 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
02:23:00 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
02:23:04 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
02:23:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
02:23:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
02:23:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
02:23:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
02:23:06 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
02:23:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
02:23:06 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
```

```
02:23:07 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
02:23:07 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
02:23:07 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 109 239
02:23:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
02:23:08 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 0 220
02:23:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
```

Vessel Configuration Test, Time Interval: 10:27:00 – 10:28:45, Duration: 105 [s]

```
10:27:00 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:27:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:27:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:27:02 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:04 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:04 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:10 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:15 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:19 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:19 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:23 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:23 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:27 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:31 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:32 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:32 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:35 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:35 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:36 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:36 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:39 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:40 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:40 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:43 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:27:44 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:27:44 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:27:44 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:27:44 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:27:45 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:27:46 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
```

```
10:27:46 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:27:48 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:27:48 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:27:49 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:27:49 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:49 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:53 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:53 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:54 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:54 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:27:57 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:27:58 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:27:58 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:27:58 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:28:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:28:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:28:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:28:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:28:05 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:28:06 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:28:06 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:28:06 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:28:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:28:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:28:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:28:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:28:13 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:28:14 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:28:14 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:28:14 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:28:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:28:18 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:28:18 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:28:18 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:28:21 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:28:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:28:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:28:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:28:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:28:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:28:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:28:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:28:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:28:30 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:28:31 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:28:31 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:28:31 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:28:32 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:28:32 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:28:32 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:28:33 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 187 44
10:28:33 - [DEBUG] [ ENGINE To VODIA ]  -  187 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 71
10:28:34 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:28:35 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:28:35 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:28:35 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
```

```
10:28:35 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:28:36 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:28:37 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:28:37 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:28:37 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 187 44
10:28:38 - [DEBUG] [ ENGINE To VODIA ]  -  187 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 71
10:28:38 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 42 4 0 106
10:28:39 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 4 236 4 0 0 167
10:28:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 187 87
10:28:39 - [DEBUG] [ ENGINE To VODIA ]  -  187 194 2 231 60 94
10:28:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 212 128 194 187 198
10:28:40 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 9 212 194 6 200 57 232 114 226 121 64
10:28:40 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:28:41 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
10:28:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
10:28:41 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:28:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:28:42 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
10:28:42 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 121 227
10:28:43 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:28:43 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 103 245
10:28:43 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:28:43 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 124 224
10:28:44 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:28:44 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 123 225
10:28:45 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:28:45 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 116 232
10:28:45 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
```

Back Button Vessel Configuration Test, Time Interval : 10:29:55 – 10:30:03, Duration: 8 [s]

```
10:29:55 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:29:58 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:29:58 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:29:58 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:29:58 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:29:59 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:30:00 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:30:00 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:30:00 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:30:01 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
10:30:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
10:30:01 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:30:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:30:02 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
10:30:02 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 109 239
10:30:02 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
10:30:02 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 0 220
10:30:03 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
```

Appendix C

Final Java desktop application logs for different diagnostic tests

Test Description: Log In , Time Interval: 10:26:00 - 10:27:20, Duration: 80 [s]

```
10:26:00 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:26:01 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:26:01 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:26:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:26:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:26:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:26:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:26:04 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:26:04 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:26:05 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:26:05 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:26:13 - [DEBUG] [ VODIA To ENGINE ]  -  172 0 243 97
10:26:15 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:26:16 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:26:16 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:26:16 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
10:26:16 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
10:26:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:26:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:26:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 0 243 97
10:26:18 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:26:19 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:26:19 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:26:19 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
10:26:19 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
10:26:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:26:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:26:20 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:26:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:26:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:26:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:26:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 164 61
10:26:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:26:27 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
10:26:27 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
10:26:27 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 164 70
10:26:28 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:26:28 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:26:28 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:26:29 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 32 36 50 49 54 49 48 52 50 53 80 48 49 42 50 49 138
10:26:29 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 33 53 50 51 48 55 48 80 48 50 42 50 49 53 50 51 118
10:26:29 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 9 234 34 48 55 50 80 48 50 42 18
10:26:31 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 164 70
10:26:32 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 32 36 50 49 54 49 48 52 50 53 80 48 49 42 50 49 138
10:26:33 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 33 53 50 51 48 55 48 80 48 50 42 50 49 53 50 51 118
10:26:33 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 9 234 34 48 55 50 80 48 50 42 18
10:26:33 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:26:34 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:26:35 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:26:35 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:26:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:26:40 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
```

```
10:26:40 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
10:26:41 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:26:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:26:41 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
10:26:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 109 239
10:26:42 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
10:26:42 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 211 128 230 187 163
10:26:43 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 7 211 230 4 2 16 240 4 209
10:26:43 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 1 219
10:26:43 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:26:43 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 164 61
10:26:44 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
10:26:45 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
10:26:45 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 164 70
10:26:46 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 32 36 50 49 54 49 48 52 50 53 80 48 49 42 50 49 138
10:26:46 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 33 53 50 51 48 55 48 80 48 50 42 50 49 53 50 51 118
10:26:46 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 9 234 34 48 55 50 80 48 50 42 18
10:26:46 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 164 67
10:26:47 - [DEBUG] [ ENGINE To VODIA ]  -  164 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 94
10:26:47 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 164 110
10:26:48 - [DEBUG] [ ENGINE To VODIA ]  -  164 194 0 154
10:26:48 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 4 212 128 194 164 244
10:26:49 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 9 212 194 6 87 249 1 94 248 4 98
10:26:49 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 0 220
10:26:49 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:26:49 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 181 167
10:26:50 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
10:26:50 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 187 44
10:26:51 - [DEBUG] [ ENGINE To VODIA ]  -  187 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 71
10:26:51 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 187 87
10:26:52 - [DEBUG] [ ENGINE To VODIA ]  -  187 194 0 131
10:26:52 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 212 128 194 187 198
10:26:53 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 3 212 194 0 2
10:26:53 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 247 128 93
10:26:54 - [DEBUG] [ ENGINE To VODIA ]  -  128 247 4 0 0 0 0 133
10:26:56 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 128 103
10:26:57 - [DEBUG] [ ENGINE To VODIA ]  -  128 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 130
10:26:57 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:27:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:27:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 32 36 164 165 1 105 165 1 174 165 1 110 165 1 100 165 223
10:27:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 33 1 27 181 1 91 226 7 105 229 1 174 229 1 110 229 128
10:27:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 34 1 100 229 1 27 245 1 119
10:27:05 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:27:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:27:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 32 36 164 165 1 105 165 1 174 165 1 110 165 1 100 165 223
10:27:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 33 1 27 181 1 91 226 7 105 229 1 174 229 1 110 229 128
10:27:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 34 1 100 229 1 27 245 1 119
10:27:11 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
10:27:15 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
10:27:15 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 212 194 12 96 181 1 97 181 1 216 249 2 97 245 1 206
10:27:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 212 194 12 96 181 1 97 181 1 216 249 2 97 245 1 206
```

Test Description: Error Codes, Time Interval: 10:29:23 - 10:30:10, Duration: 47 [s]

```
10:29:23 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:29:25 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:29:25 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:29:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:29:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:29:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:29:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:29:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:29:30 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:29:31 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:29:31 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:29:31 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:29:32 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:29:33 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:29:33 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:29:33 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 164 61
10:29:34 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
10:29:35 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
10:29:35 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 164 70
10:29:36 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 32 36 50 49 54 49 48 52 50 53 80 48 49 42 50 49 138
10:29:36 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 33 53 50 51 48 55 48 80 48 50 42 50 49 53 50 51 118
10:29:36 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 9 234 34 48 55 50 80 48 50 42 18
10:29:37 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 247 128 93
10:29:37 - [DEBUG] [ ENGINE To VODIA ]  -  128 247 4 0 0 0 0 133
10:29:37 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:29:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:29:43 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 32 36 164 165 1 105 165 1 174 165 1 110 165 1 100 165 223
10:29:43 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 33 1 27 181 1 91 226 7 105 229 1 174 229 1 110 229 128
10:29:44 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 34 1 100 229 1 27 245 1 119
10:29:44 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 164 229 133
10:29:44 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 164 229 86 79 76 86 79 255 255 255 255 255 255 255 255 150
10:29:44 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 105 229 192
10:29:45 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 105 229 86 79 76 86 79 255 255 255 255 255 255 255 255 209
10:29:45 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 174 229 123
10:29:46 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 174 229 86 79 76 86 79 255 255 255 255 255 255 255 255 140
10:29:46 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 110 229 187
10:29:46 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 110 229 86 79 76 86 79 255 255 255 255 255 255 255 255 204
10:29:47 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 100 229 197
10:29:47 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 100 229 86 79 76 86 79 255 255 255 255 255 255 255 255 214
10:29:47 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 27 245 254
10:29:48 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 27 245 86 79 76 86 79 255 255 255 255 255 255 255 255 15
10:29:48 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 91 226 209
10:29:49 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 91 226 86 79 76 86 79 255 255 255 255 255 255 255 255 226
10:29:49 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 105 229 192
10:29:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 105 229 86 79 76 86 79 255 255 255 255 255 255 255 255 209
10:29:50 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 174 229 123
10:29:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 174 229 86 79 76 86 79 255 255 255 255 255 255 255 255 140
10:29:50 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 110 229 187
10:29:51 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 110 229 86 79 76 86 79 255 255 255 255 255 255 255 255 204
10:29:51 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 100 229 197
10:29:52 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 100 229 86 79 76 86 79 255 255 255 255 255 255 255 255 214
10:29:52 - [DEBUG] [ VODIA To ENGINE ]  -  172 195 3 128 27 245 254
```

```
10:29:53 - [DEBUG] [ ENGINE To VODIA ]  -  128 196 15 27 245 86 79 76 86 79 255 255 255 255 255 255 255 255 15
10:29:53 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
10:29:57 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
10:29:58 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 212 194 12 96 181 1 97 181 1 216 249 2 97 245 1 206
10:29:58 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 96 245 97
10:29:59 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 96 245 86 79 76 86 79 255 255 63
10:29:59 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
10:30:00 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 97 245 96
10:30:00 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 97 245 86 79 76 86 79 255 255 62
10:30:01 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
10:30:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 216 249 229
10:30:01 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 216 249 86 79 76 86 79 255 255 195
10:30:02 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
10:30:02 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 6 212 195 3 128 97 245 96
10:30:03 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 254 16 20 172 18 212 196 15 97 245 86 79 76 86 79 255 255 62
10:30:04 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 8 254 17 255 255 255 255 255 255 175
10:30:04 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 187 87
10:30:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 194 0 131
10:30:05 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 212 128 194 187 198
10:30:06 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 3 212 194 0 2
10:30:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 164 110
10:30:06 - [DEBUG] [ ENGINE To VODIA ]  -  164 194 0 154
10:30:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 4 212 128 194 164 244
10:30:07 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 9 212 194 6 87 249 1 94 248 4 98
10:30:07 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 6 212 195 3 164 87 249 30
10:30:08 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 254 16 20 172 18 212 196 15 87 249 86 79 76 86 79 255 255 32
10:30:09 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 8 254 17 255 255 255 255 255 255 139
10:30:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 6 212 195 3 164 94 248 24
10:30:10 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 254 16 20 172 18 212 196 15 94 248 86 79 76 86 79 255 255 26
10:30:10 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 8 254 17 255 255 255 255 255 255 139
```

Test Description: Back button Error Codes, Time Interval: 10:31:05 - 10:31:14, Duration: 9 [s]

```
10:31:05 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:31:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:31:09 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:31:09 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:31:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:31:10 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:31:11 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:31:11 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:31:11 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:31:12 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
10:31:12 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
10:31:12 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:31:12 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:31:13 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
10:31:13 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 109 239
10:31:14 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
10:31:14 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 0 220
10:31:14 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
```

ECU Test, MID 128 Engine Control Unit, Time Interval: 10:32:16 – 10:32:33, Duration: 17 [s]

```
10:32:16 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:32:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:32:20 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:32:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:32:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:32:21 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:32:21 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:32:22 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:32:23 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:32:23 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:32:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:32:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:32:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:32:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:32:27 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 128 103
10:32:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 130
10:32:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 0 165
10:32:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 236 4 0 2 224
10:32:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 1 164
10:32:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 236 4 1 2 50 49 53 51 50 48 48 53 80 48 49 145
10:32:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 2 163
10:32:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 236 4 2 2 50 49 53 55 54 55 52 51 80 48 49 127
10:32:30 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 49 1 0 161
10:32:30 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 240 1 0 0 225
10:32:30 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 49 2 0 160
10:32:31 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 240 2 0 0 224
10:32:31 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 2 0 167
10:32:32 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 236 2 0 0 228
10:32:32 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 247 128 93
10:32:33 - [DEBUG] [ ENGINE To VODIA ]  -  128 247 4 0 0 0 0 133
```

ECU Test, MID 164 Helm Control unit, Time Interval:10:33:17 – 10:33:31, Duration: 14 [s]

```
10:33:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 164 61
10:33:20 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 243 16 28 164 86 79 76 86 79 42 50 49 52 53 54 53 53 156
10:33:20 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 16 243 17 57 80 48 49 42 52 48 48 49 48 48 49 54 58 174
10:33:20 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 164 70
10:33:21 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 32 36 50 49 54 49 48 52 50 53 80 48 49 42 50 49 138
10:33:22 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 17 234 33 53 50 51 48 55 48 80 48 50 42 50 49 53 50 51 118
10:33:22 - [DEBUG] [ ENGINE To VODIA ]  -  164 192 9 234 34 48 55 50 80 48 50 42 18
10:33:22 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 164 67
10:33:23 - [DEBUG] [ ENGINE To VODIA ]  -  164 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 94
10:33:23 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 0 129
10:33:24 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 236 4 0 4 186
10:33:24 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 1 128
10:33:25 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 15 236 4 1 4 50 49 53 54 48 49 54 50 80 48 49 102
10:33:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 2 127
10:33:26 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 15 236 4 2 4 50 49 53 54 48 48 56 57 80 48 49 93
10:33:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 3 126
10:33:26 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 15 236 4 3 4 50 49 53 52 49 56 54 53 80 48 49 91
10:33:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 4 4 125
10:33:27 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 15 236 4 4 4 50 49 54 49 51 56 48 54 80 48 49 95
```

```
10:33:27 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 49 1 0 125
10:33:28 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 240 1 0 0 189
10:33:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 49 2 0 124
10:33:29 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 240 2 0 0 188
10:33:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 49 3 0 123
10:33:29 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 240 3 0 0 187
10:33:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 49 4 0 122
10:33:30 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 240 4 0 0 186
10:33:30 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 164 3 42 2 0 131
10:33:31 - [DEBUG] [ ENGINE To VODIA ]  -  164 254 172 4 236 2 0 0 192
```

ECU – MID 187 Powertrain Control Unit, Time Interval: 10:34:10 - 10:34:17, Duration: 7 [s]

```
10:34:10 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:34:12 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:34:13 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:34:13 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:34:13 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:34:14 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:34:15 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:34:15 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:34:15 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 187 44
10:34:16 - [DEBUG] [ ENGINE To VODIA ]  -  187 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 71
10:34:16 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 42 4 0 106
10:34:17 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 4 236 4 0 0 167
10:34:17 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 42 2 0 108
10:34:17 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 4 236 2 0 0 169
```

Back Button ECU Information Test, Time Interval: 10:35:00 - 10:35:09, Duration: 9 [s]

```
10:35:00 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:35:02 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:35:03 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:35:03 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:35:03 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:35:04 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:35:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:35:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:35:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:35:06 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
10:35:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
10:35:07 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:35:07 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:35:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
10:35:08 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 109 239
10:35:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
10:35:08 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 0 220
10:35:09 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
```

Test Description: Vessel Configuration Test, Time Interval: 10:35:36 - 10:36:54, Duration: 78 [s]

```
10:35:36 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:35:38 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:35:39 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:35:39 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
```

```
10:35:40 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:35:40 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:35:40 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:35:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:35:42 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:35:42 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:35:42 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:35:42 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:35:44 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:35:44 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:35:44 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:35:47 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:35:48 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:35:48 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:35:48 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:35:49 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:35:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:35:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:35:50 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:35:51 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:35:52 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:35:52 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:35:52 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:35:53 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:35:54 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:35:54 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:35:54 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 128 103
10:35:55 - [DEBUG] [ ENGINE To VODIA ]  -  128 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 130
10:35:55 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:35:56 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:35:57 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:35:57 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:35:57 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:35:58 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:35:59 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:35:59 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:35:59 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 187 44
10:36:00 - [DEBUG] [ ENGINE To VODIA ]  -  187 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 71
10:36:00 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 42 4 0 106
10:36:01 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 4 236 4 0 0 167
10:36:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 187 87
10:36:02 - [DEBUG] [ ENGINE To VODIA ]  -  187 194 0 131
10:36:02 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 212 128 194 187 198
10:36:03 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 3 212 194 0 2
10:36:03 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:36:04 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
10:36:04 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
10:36:04 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:36:04 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:36:05 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
10:36:05 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 121 227
10:36:06 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:36:06 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 103 245
```

```
10:36:07 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:36:07 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 124 224
10:36:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:36:08 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 123 225
10:36:08 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:36:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 116 232
10:36:09 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 0 177
10:36:09 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 128 97
10:36:10 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 243 16 28 128 80 69 78 84 49 42 50 49 48 57 50 49 53 26
10:36:11 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 16 243 17 56 80 48 52 42 48 52 54 56 48 49 50 57 58 190
10:36:11 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:36:12 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:36:12 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:36:12 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:36:15 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:36:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:36:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:36:16 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:36:19 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:36:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:36:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:36:20 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:36:23 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 128 106
10:36:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 32 36 50 49 52 56 48 57 48 52 80 48 49 42 50 49 167
10:36:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 234 33 52 48 48 57 52 57 80 48 50 42 50 49 52 48 48 151
10:36:24 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 234 34 57 53 48 80 48 49 42 50
10:36:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 237 128 103
10:36:25 - [DEBUG] [ ENGINE To VODIA ]  -  128 237 17 86 86 32 32 32 48 48 48 56 52 56 32 32 32 32 32 32 130
10:36:25 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 0 165
10:36:26 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 236 4 0 2 224
10:36:26 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 1 164
10:36:27 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 236 4 1 2 50 49 53 51 50 48 48 53 80 48 49 145
10:36:27 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 42 4 2 163
10:36:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 236 4 2 2 50 49 53 55 54 55 52 51 80 48 49 127
10:36:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 49 1 0 161
10:36:28 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 240 1 0 0 225
10:36:28 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 3 49 2 0 160
10:36:29 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 4 240 2 0 0 224
10:36:29 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:36:33 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:36:36 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 32 36 164 165 1 105 165 1 174 165 1 110 165 1 100 165 223
10:36:37 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 33 1 27 181 1 91 226 7 105 229 1 174 229 1 110 229 128
10:36:37 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 34 1 100 229 1 27 245 1 119
10:36:37 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:36:41 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 194 128 146
10:36:45 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 32 36 164 165 1 105 165 1 174 165 1 110 165 1 100 165 223
10:36:45 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 17 194 33 1 27 181 1 91 226 7 105 229 1 174 229 1 110 229 128
10:36:45 - [DEBUG] [ ENGINE To VODIA ]  -  128 192 9 194 34 1 100 229 1 27 245 1 119
10:36:45 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
10:36:49 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 128 4 212 128 194 128 60
10:36:50 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 212 194 12 96 181 1 97 181 1 216 249 2 97 245 1 206
10:36:54 - [DEBUG] [ ENGINE To VODIA ]  -  128 254 172 15 212 194 12 96 181 1 97 181 1 216 249 2 97 245 1 206
```

Vessel Configuration Test Back Button, Time Interval: 10:37:55-10:38:03, Duration: 8 [s]

```
10:37:55 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 243 187 38
10:37:57 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 32 31 187 86 79 76 86 79 42 50 49 52 54 57 48 54 91
10:37:58 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 243 33 50 80 48 49 42 48 48 50 50 50 56 55 50 58 120 10
10:37:58 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 4 243 34 4 25 79
10:37:58 - [DEBUG] [ VODIA To ENGINE ]  -  172 128 234 187 47
10:37:59 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 32 36 50 49 54 49 48 52 52 48 80 48 49 42 50 49 118
10:38:00 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 17 234 33 53 50 51 49 48 55 80 48 50 42 50 49 53 52 51 92
10:38:00 - [DEBUG] [ ENGINE To VODIA ]  -  187 192 9 234 34 51 53 53 80 48 49 42 248
10:38:00 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:38:01 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 226 184
10:38:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 1 7 147
10:38:01 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
10:38:01 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 57 2 69 70 209
10:38:02 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 12 231 2 68 101 102 97 117 108 116 48 48 48 81
10:38:02 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 4 58 1 0 109 239
10:38:03 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 2 232 1 176
10:38:03 - [DEBUG] [ VODIA To ENGINE ]  -  172 254 187 3 105 83 0 220
10:38:03 - [DEBUG] [ ENGINE To VODIA ]  -  187 254 172 1 9 145
```