ARM9 Interrupt Latency measurement and storage

Serial Communication with ARM9 and Data Storage for Analysis

Sanjeev Kumarr Gopal

Mtr-Nr: 747763

Hochschule Esslingen

sagogs00@hs-esslingen.de

Project Supervisor - Vikas Agrawal

June 27, 2014

Hochschule Esslingen
University of Applied Sciences

Nah an Mensch und Technik.

Contents

Abstract

The concepts of Serial Communication discussed here is used to receive data from an ARM9 Controller. The received data is further processed. A Database is setup and a table is created. The data received is then stored in the database. The data is then retrieved back from the database based on various criteria for future Analysis. Graphical User Interfaces (GUI's) are then designed for the front end user to receive the data, store the data in database and also to retrieve the data from database. Finally, the GUI's are integrated into one as single application.

I. Introduction

Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel or computer bus. The term "serial" most often refers to the RS232 port on the back of the original IBM PC, often called "the" serial port, and "the" serial cable designed to plug into it, and the many devices designed to be compatible with it. RS-232 is a standard for serial communication transmission of data. It formally defines the signals connecting between a DTE (data terminal equipment) such as a computer terminal, and a DCE (data circuit-terminating equipment, originally defined as data communication equipment), such as a modem. The RS-232 standard is commonly used in computer serial ports. The standard defines the electrical characteristics and timing of signals, the meaning of signals, and the physical size and pinout of connectors.

Use a Database if

1. The information is a large amount that would become unmanageable in spreadsheet form and is related to a particular subject.
2. You want to maintain records for ongoing use.
3. The information is subject to many changes (change of address, pricing changes, etc.).
4. You want to generate reports based on the information.

Focusing on the Project goal, this documentation encompasses the development of GUI's for receiving data from an ARM9 controller through serial communication and retrieving values from database. The document consists of various sections illustrating the literature, softwares and methodology with future scope of improvement.

II. Project Overview

Serial Communication with ARM9 and Data Storage for Analysis project is divided into four parts:
1. Data Reception from ARM9 controller through serial communication
2. Splitting of data and storing them in the database.
3. Retrieving the values from the database using two different criteria.
4. Desin of GUI's for the above 3 parts.

The source code is written on Eclipse platform. Java Swings are used for the designing of GUI's. Oracle 11g Exress Edition database is used and sqldeveloper is the respective database management system.

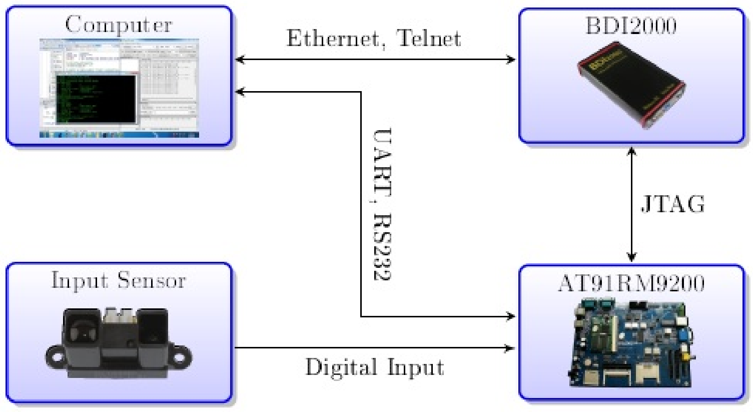


Figure 1: *System Overview*

III. Introduction to Serial Communications

III.1 Overview

The purpose of this application note is to attempt to describe the main elements in Serial Communication. This application note attempts to cover enough technical details of RS232.

III.2 Terminologies

Baudrate - Baud rate indicates the transmission speed on the serial line. With RS232/422/485 interfaces, the baudrate is the same as the transfer rate in bps.

Bit - Binary digit, can have value 0 or 1.

Bitrate- The bitrate defines the number of bits transmitted in a specified amount of time.

bps - Bits per Second (bps) is the rate at which data is transmitted.

CTS - Clear To Send, signal line on the DCE to handshake with the DTE.

DCE - Data Communications Equipment (DTE) is the other endpoint in a serial communication. Typically, it is a modem or serial terminal.

Databits - The number of bits that are used to send a single character over a serial line.

DSR - Data Set Ready, signal line on the DCE to handshake with the DTE.

DTE - Data Terminal Equipment (DTE) is one of two endpoints in a serial communication. An example would be a computer.

DTR - Data Terminal Ready, signal line on the DTE to handshake with the DCE.

Parity bit - A bit at the end of every character which is used for error detection. This bit is computed from the databits.

RS232 - The original standard which defined hardware serial communications. It has since been renamed to TIA-232.

Start bit - A bit that indicates the start of a new character on the serial line.

Stop bit - A bit that indicates the end of a character on the serial line.

III.3 DCE and DTE Devices

DTE stands for Data Terminal Equipment, and DCE stands for Data Communications Equipment. These terms are used to indicate the pin-out for the connectors on a device and the direction of the signals on the pins. Your computer is a DTE device, while most other devices such as modem and other serial devices are usually DCE devices.

RS-232 has been around as a standard for decades as an electrical interface between Data Terminal Equipment (DTE) and Data Circuit-Terminating Equipment (DCE) such as modems or DSUs. It appears under different incarnations such as RS-232C, RS-232D, V.24, V.28 or V.10. RS-232 is used for asynchronous data transfer as well as synchronous links such as SDLC, HDLC, Frame Relay and X.25

III.4 RS232

RS-232 (Recommended standard-232) is a standard interface approved by the Electronic Industries Association (EIA) for connecting serial devices. In other words, RS-232 is a longestablished standard that describes the physical interface and protocol for relatively low-speed serial data communication between computers and related devices.

An industry trade group, the Electronic Industries Association (EIA), defined it originally for teletypewriter devices. In 1987, the EIA released a new version of the standard and changed the name to EIA-232-D. Many people, however, still refer to the standard as RS-232C, or just RS- 232.

RS-232 is the interface that your computer uses to talk to and exchange data with your modem and other serial devices. The serial ports on most computers use a subset of the RS- 232C standard.

III.5 Signal Description

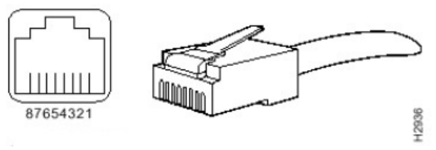


Figure 2: *RJ-45 Connector*

TxD: - This pin carries data from the computer to the serial device

RXD: - This pin carries data from the serial device to the computer

DTR signals: - DTR is used by the computer to signal that it is ready to communicate with the serial device like modem. In other words, DTR indicates to the Dataset (i.e., the modem or DSU/CSU) that the DTE (computer) is ON.

DSR: - Similarly to DTR, Data set ready (DSR) is an indication from the Dataset that it is ON.

DCD: - Data Carrier Detect (DCD) indicates that carrier for the transmit data is ON.

RTS: - This pin is used to request clearance to send data to a modem

CTS: - This pin is used by the serial device to acknowledge the computer's RTS Signal. In most situations, RTS and CTS are constantly on throughout the communication session.

Clock signals (TC, RC, and XTC): - The clock signals are only used for synchronous communications. The modem or DSU extracts the clock from the data stream and provides a steady clock signal to the DTE. Note that the transmit and receive clock signals do not have to be the same, or even at the same baud rate.

CD: - CD stands for Carrier Detect. Carrier Detect is used by a modem to signal that it has made a connection with another modem, or has detected a carrier tone. In other words, this is used by the modem to signal that a carrier signal has been received from a remote modem.

RI: - RI stands for Ring Indicator. A modem toggles(keystroke) the state of this line when an incoming call rings your phone. In other words, this is used by an auto answer modem to signal the receipt of a telephone ring signal

The Carrier Detect (CD) and the Ring Indicator (RI) lines are only available in connections to a modem. Because most modems transmit status information to a PC when either a carrier signal is detected (i.e. when a connection is made to another modem) or when the line is ringing, these two lines are rarely used.

III.6 Limitations of RS-232

RS-232 has some serious shortcomings as an electrical interface.

Firstly, the interface presupposes a common ground between the DTE and DCE. This is a reasonable assumption where a short cable connects a DTE and DCE in the same room, but with longer lines and connections between devices that may be on different electrical busses, this may not be true. We have seen some spectacular electrical events causes by "uncommon grounds".

Secondly, a signal on a single line is impossible to screen effectively for noise. By screening the entire cable one can reduce the influence of outside noise, but internally generated noise remains a problem. As the baud rate and line length increase, the effect of capacitance between the cables introduces serious crosstalk until a point is reached where the data itself is unreadable.

III.7 USB to RS232 DB9 Serial Adapter

Figure 3: *PL-2303 USB to Serial adapter*

The PL-2303 USB to Serial adapter is your smart and convenient accessory for connecting RS-232 serial devices to your USB-equipped Windows host computer. It provides a bridge connection with a standard DB 9-pin male serial port connector in one end and a standard Type-A USB plug connector on the other end. You simply attach the serial device onto the serial port of the cable and plug the USB connector into your PC USB port. It allows a simple and easy way of adding serial connections to your PC without having to go thru inserting a serial card and traditional port configuration.

This USB to Serial adapter is ideal for connecting modems, cellular phones, PDAs, digital cameras, card readers and other serial devices to your computer. It provides serial connections up to 1Mbps of data transfer rate. And since USB does not require any IRQ resource, more devices can be attached to the system without the previous hassles of device and resource conflicts.

Finally, the PL-2303 USB to Serial adapter is a fully USB Specification compliant device and therefore supports advanced power management such as suspend and resume operations as well as remote wakeup. The PL-2303 USB Serial cable adapter is designed to work on all Windows operating systems.

IV. Graphical User Interfaces

IV.1 Introduction to Java Swings

Swing, which is an extension library to the AWT, includes new and improved components that enhance the look and functionality of GUIs. Swing can be used to build Standalone swing gui Apps as well as Servlets and Applets. It employs a model/view design architecture. Swing is more portable and more flexible than AWT.

Swing Model/view design: The âĂIJview partâĂİ of the MV design is implemented with a component object and the UI object. The âĂIJmodel partâĂİ of the MV design is implemented by a model object and a change listener object.

Swing is built on top of AWT and is entirely written in Java, using AWTâĂŹs lightweight component support. In particular, unlike AWT, t he architecture of Swing components makes it easy to customize both their appearance and behavior. Components from AWT and Swing can be mixed, allowing you to add Swing support to existing AWT-based programs. For example, swing components such as JSlider, JButton and JCheckbox could be used in the same program with standard AWT labels, textfields and scrollbars. You could subclass the existing Swing UI, model, or change listener classes without having to reinvent the entire implementation. Swing also has the ability to replace these objects on-the-fly.

- 100% Java implementation of components
- Pluggable Look & Feel
- Lightweight components
- **Uses MVC Architecture**
  Model represents the data
  View as a visual representation of the data
  Controller takes input and translates it to changes in data
- **Three parts**
  Component set (subclasses of JComponent)
  Support classes
  Interfaces

In Swing, classes that represent GUI components have names beginning with the letter J. Some examples are JButton, JLabel, and JSlider. Altogether there are more than 250 new classes and 75 interfaces in Swing - twice as many as in AWT.

Java swing class hierarchy:

The class JComponent, descended directly from Container, is the root class for most of Swing's user interface components.

JPanel is Swing's version of the AWT class Panel and uses the same default layout, FlowLayout. JPanel is descended directly from JComponent.

JFrame is Swing's version of Frame and is descended directly from that class. The components added to the frame are referred to as its contents; these are managed by the contentPane. To add a component to a JFrame, we must use its contentPane instead.

JLabel, descended from JComponent, is used to create text labels.

JTextField allows editing of a single line of text. New features include the ability to justify the text left, right, or center, and to set the text's font.

JButton is a component the user clicks to trigger a specific action.

JComboBox is like a drop down box. You can click a drop-down arrow and select an option from a list. For example, when the component has focus, pressing a key that corresponds to the first character in some entry's name selects that entry. A vertical scrollbar is used for longer lists.

JSeperator creates a new separator with the specified horizontal or vertical orientation.

IV.2 Design of GUI

The following code (a part of complete code) is used for designing the GUI of this project.

```
private void initComponents()
{
        aButton = new javax.swing.JButton();
        bButton = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jSeparator1 = new javax.swing.JSeparator();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jComboBox1 = new javax.swing.JComboBox();
        jLabel4 = new javax.swing.JLabel();
        Start = new javax.swing.JTextField();
        jLabel5 = new javax.swing.JLabel();
        End = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Latency Application");

        aButton.setText("Receive Data");

        bButton.setText("Stop Data");
        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jLabel1.setText("Serial Communication");

        jLabel2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jLabel2.setText("Latency History");

        jLabel3.setText("Search By :");

        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[]{
            "Date", "Latency"}));

        jLabel4.setText("Start :");
}
```

The GUI looks like below.

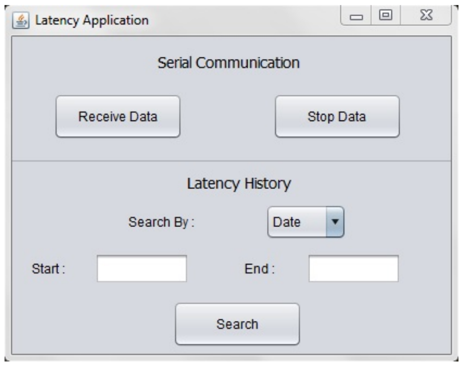


Figure 4: *Graphical User Interface*

IV.3 Threading Concepts

In the above section, there are *Receive Data* and *Stop Data* buttons. They call different methods and when *Receive Data* button is pressed, it calls a method that has a functionality which runs on an infinite loop. So, in this case, the press of *Stop Data* button is never recognized. For overcoming this error, concept of threading is used.

Thread:

A thread is a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time.

The thread provides a Runnable object. The Runnable interface defines a single method *run*, meant to contain the code executed in the thread. The Runnable object is then passed to the *Thread constructor*.

The following code is written for threading concept.

```
public void start() {
        if (thread == null) {
            thread = new Thread(this);
            thread.start();
        }
    }

    public void stop() {
        thread.stop();
        thread = null;
    }

    public void run() {
        while (thread != null) {

            ReadData();
        }
        thread = null;
    }
```

IV.4 Nimbus Look and Feel

Pluggable look and feel is a mechanism used in the Java Swing widget toolkit allowing to change the look and feel of the graphical user interface at runtime.

Swing allows an application to specialize the look and feel of widgets by modifying the default (via runtime parameters), deriving from an existing one, by creating one from scratch, or, beginning with J2SE 5.0, by using the skinnable synth look and feel, which is configured with an XML property file. The look and feel can be changed at runtime.

Nimbus is a polished cross-platform look and feel introduced in the Java SE 6 Update 10 (6u10) release [1]. For backwards compatibility, Metal is still the default Swing look and feel, but you can change to Nimbus by adding the following code to the event-dispatching thread before creating the graphical user interface (GUI):

```
import javax.swing.UIManager.*;
try {
    for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels()) {
        if ("Nimbus".equals(info.getName())) {
            UIManager.setLookAndFeel(info.getClassName());
            break;
        }
    }
} catch (Exception e) {
    // If Nimbus is not available, you can set the GUI to another look and feel.
}
```

The first line of code retrieves the list of all installed look and feel implementations for the platform and then iterates through the list to determine if Nimbus is available. If so, Nimbus is set as the look and feel.

V. Data Reception from ARM9 Processor

V.1 Overview

This section explains, how the data that is being continuously sent by the ARM9 Controller can be received using the concept of serial communication that was explained in one of the previous sections.

We use **Java Simple Serial Connector** - a Library for working with serial ports.

V.2 Prerequisites

There are two prerequisites before one can receive data from the ARM9 Controller.

- Firstly, *jssc library* should be downloaded, installed and added to the eclipse environment. It can be done using the following steps.
  - Download the jSSC-2.8.0-Release [2].
  - Extract all file somewhere.
  - In Eclipse: Project Properties -> Java Build path -> Libraries
  - Press Add external jars and choose jssc.jar
- Secondly, *PL-2303 USB to Serial adapter* should be downloaded and installed in order to detect the RS232 adapter [3].

V.3 Data Format

We receive fixed 34 bytes of data in a String format. Each character is of 1 byte and also the spaces in-between are considered as a byte.

The format of the data is as below.

DD/MM/YYYY HH:MM:SS Value Unit\n

- DD/MM/YYYY - Date on which data is sent.
- HH:MM:SS - Time at which data is sent.
- Value - Measured Interrupt Latency value.
- Unit - Unit of the Latency value(us).

V.4 Port Configuration

The COM port has certain parameters that have to be configured with same values for both the sender and the receiver in order to enable a successful communication. There are four parameters that are to be configured.

- **Baud Rate** - Baud is synonymous to symbols per second or pulses per second. It is the unit of symbol rate, also known as baud or modulation rate; the number of distinct symbol changes (signaling events) made to the transmission medium per second in a digitally modulated signal or a line code. Baud is related to but should not be confused with gross bit rate expressed as bits per second, bps, b/s, bit/s or bits/s. However, though technically incorrect, in the case of modem manufacturers baud commonly refers to bits per second. They make a distinction by also using the term characters per second (CPS).

- **Data Bits** - The number of data bits in each character can be 5 (for Baudot code), 6 (rarely used), 7 (for true ASCII), 8 (for most kinds of data, as this size matches the size of a byte), or 9 (rarely used). 8 data bits are almost universally used in newer applications. 5 or 7 bits generally only make sense with older equipment such as teleprinters.

  Most serial communications designs send the data bits within each byte LSB (Least significant bit) first. This standard is also referred to as "little endian." Also possible, but rarely used, is "big endian" or MSB (Most Significant Bit) first serial communications; this was used, for example, by the IBM 2741 printing terminal. (See Bit numbering for more about bit ordering.) The order of bits is not usually configurable within the serial port interface. To communicate with systems that require a different bit ordering than the local default, local software can re-order the bits within each byte just before sending and just after receiving.

- **Stop Bits** - Stop bits sent at the end of every character allow the receiving signal hardware to detect the end of a character and to resynchronise with the character stream. Electronic devices usually use one stop bit. If slow electromechanical teleprinters are used, one-and-one half or two stop bits are required.

- **Parity** - Parity is a method of detecting errors in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of 1 bits in each character, including the parity bit, is always odd or always even. If a byte is received with the wrong number of 1s, then it must have been corrupted. However, an even number of errors can pass the parity check.

  Electromechanical teleprinters were arranged to print a special character when received data contained a parity error, to allow detection of messages damaged by line noise. A single parity bit does not allow implementation of error correction on each character, and communication protocols working over serial data links will have higher-level mechanisms to ensure data validity and request retransmission of data that has been incorrectly received.

  The parity bit in each character can be set to none (N), odd (O), even (E), mark (M), or space (S). None means that no parity bit is sent at all. Mark parity means that the parity bit is always set to the mark signal condition (logical 1) and likewise space parity always sends the parity bit in the space signal condition. Aside from uncommon applications that use the 9th (parity) bit for some form of addressing or special signalling, mark or space parity is uncommon, as it adds no error detection information. Odd parity is more useful than even, since it ensures that at least one state transition occurs in each character, which makes it more reliable. The most common parity setting, however, is "none", with error detection handled by a communication protocol.

In our case, the parameter values are,

- Baud Rate - 115200
- Data bits - 8
- Stop Bits - 1
- Parity - 0 (No parity)

V.5 Steps involved

The following are the steps involved for receiving the data.

- Open the port.
- Set the parameters (Baud rate, Data bits, Stop bits, Parity).
- Read bytes from serial port [4].
- Close the port.

The code for the above steps is as below.

```
public static void ReadData() {
            try {
                serialPort.openPort();// Open serial port
                System.out.println("Port Open");
                serialPort.setParams(115200, 8, 1, 0);// Set params.
                System.out.println("Parameters Set");
                while (true) {
                 byte[] buffer = serialPort.readBytes(34);// Read 34 bytes from serial port
                } else {
                    break;
                }
                }
                serialPort.closePort();// Close serial port
                System.out.println("Port Closed");
            }
             catch (SerialPortException ex) {
                System.out.println(ex);

            }
}
```

VI. Storage/Retrieval of Data

VI.1 Introduction to Database

A database is a collection of data that is related to a particular topic or purpose. As an example, employee records in a filing cabinet, a collection of sales leads in a notebook, are examples of collections of data or databases.

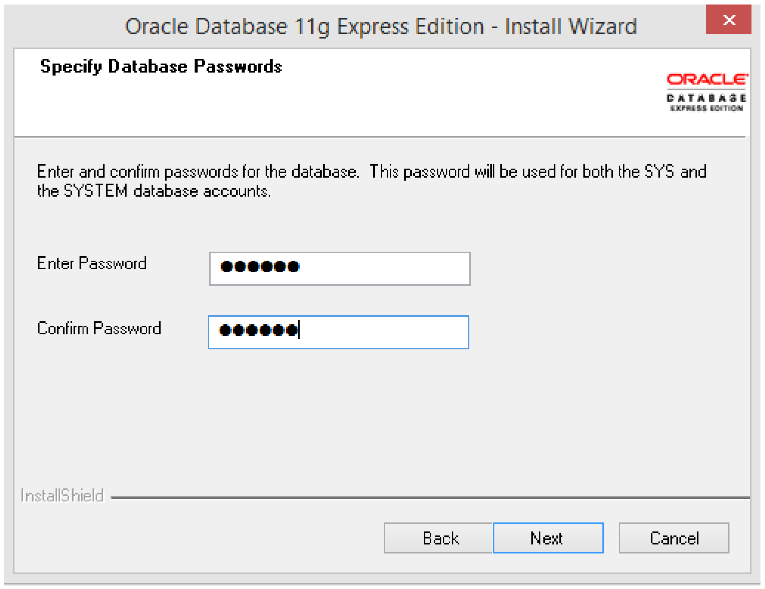
A database management system (DBMS) is a system that stores and retrieves information in a database. It is used to help you organize your data according to a subject, so that it is easy to track and verify your data, and you can store information about how different subjects are related, so that it makes it easy to bring related data together.

Oracle database 11g Express Edition is used.

Installation Guides:

1. Download the Oracle 11g Express Edition [5].

2. At the time of installation, the Database password should be specified as **system**.


Oracle Database 11g Express Edition - Install Wizard

Specify Database Passwords

Enter and confirm passwords for the database. This password will be used for both the SYS and the SYSTEM database accounts.

Enter Password ●●●●●●

Confirm Password ●●●●●●

Figure 5: *Database Password screen*

3. A new database has to be created using the below steps.

- Select Get Started from windows.

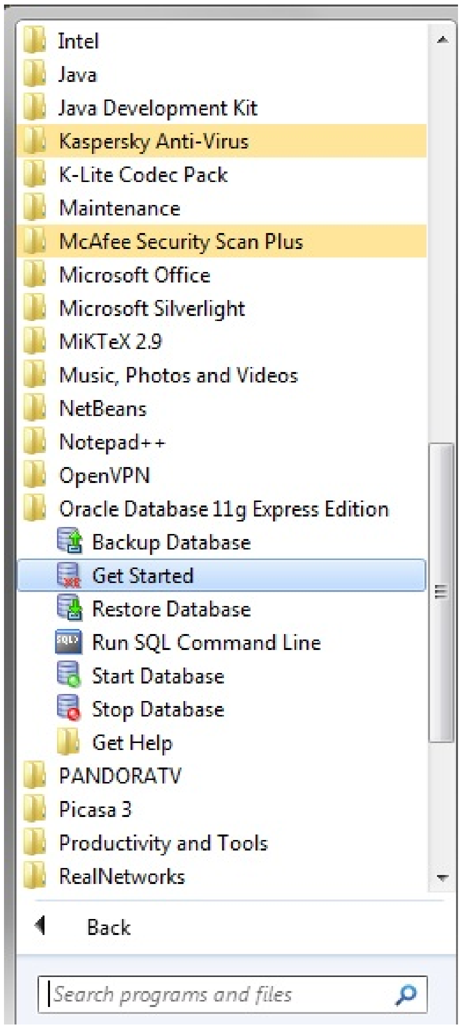
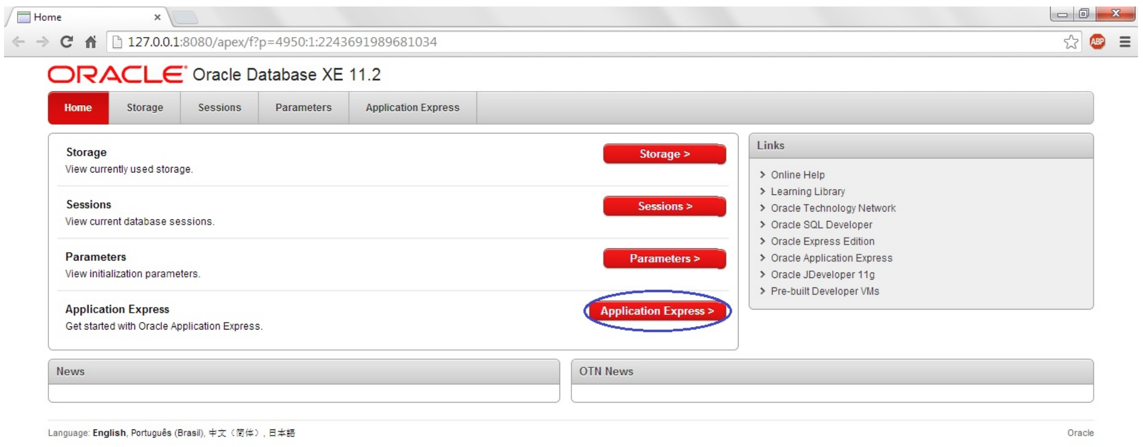


Figure 6: *Getting started with Database*

- Click on **Application Express**, enter **sanjeev** in all the fields and click on **Create Workspace**.



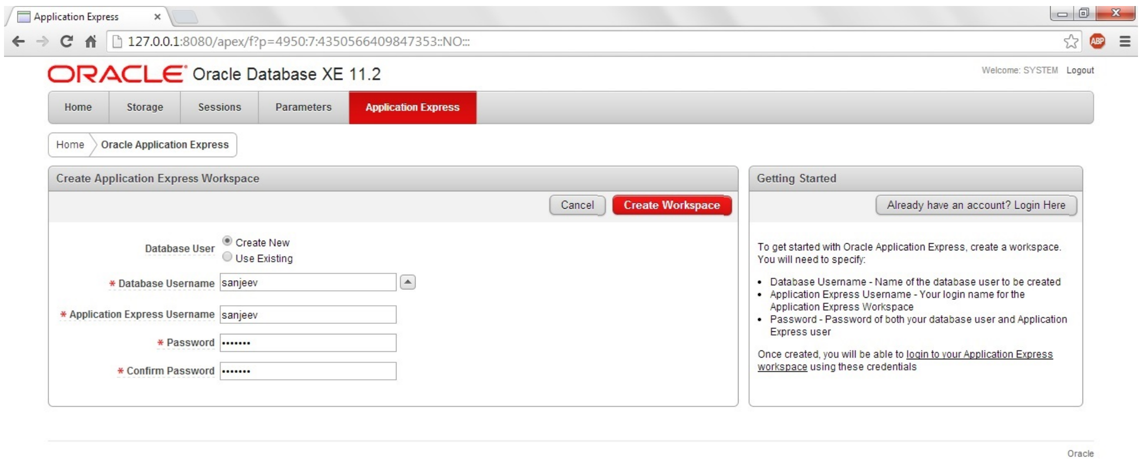
Storage
View currently used storage.

Sessions
View current database sessions.

Parameters
View initialization parameters.

Application Express
Get started with Oracle Application Express.

Figure 7: *Setting up a new Database*


Application Express
127.0.0.1:8080/apex/f?p=4950:7:4350566409847353::NO:::
ORACLE Oracle Database XE 11.2
Welcome: SYSTEM Logout
Home
Storage
Sessions
Parameters
Application Express
Home
Oracle Application Express
Create Application Express Workspace
Cancel
Create Workspace
Getting Started
Already have an account? Login Here
Oracle

Database User	(● Create New) Use Existing
* Database Username	sanjeev
* Application Express Username	sanjeev
* Password	•••••••
* Confirm Password	•••••••

To get started with Oracle Application Express, create a workspace. You will need to specify:

- Database Username - Name of the database user to be created
- Application Express Username - Your login name for the Application Express Workspace
- Password - Password of both your database user and Application Express user

Once created, you will be able to login to your Application Express workspace using these credentials

Figure 8: *Entering the fields for new Database*

The database is created and ready to be used. ojdbc14-10.2.0.4.0 library should be installed in eclipse [6].

VI.2 Table creation

The database is ready, but for storing the data in the database, we need a table to be created. The table can be created using the following sql query in a sqldeveloper [7].

```
create table sanjeev.latency
(
TIMESTAMP TIMESTAMP,
LATENCY NUMBER,
UNIT VARCHAR2(25)
);
```

The table is created and looks like below.

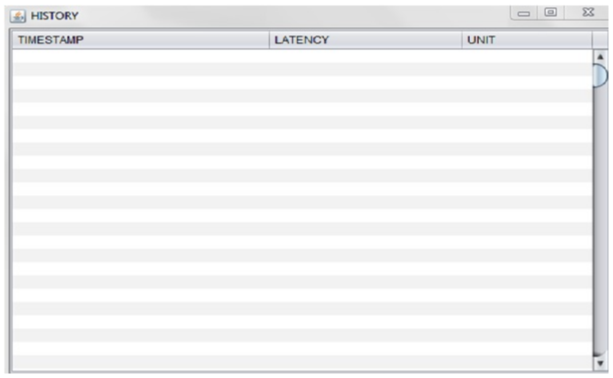


Figure 9: *Table **Latency** in the Database*

VI.3 Splitting of Data

The string is separated by "spaces" in-between.

The command used for splitting the data is as below.

```
String[] data = received_data.split(" ");
```

The substrings are divided into different fields as below.

```
data[0] + data[1] = timestamp
data[2] = latency
data[3] = unit (us)
```

It can be clearly understood seeing the below figure.

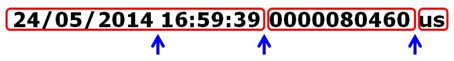


Figure 10: *Data Separation*

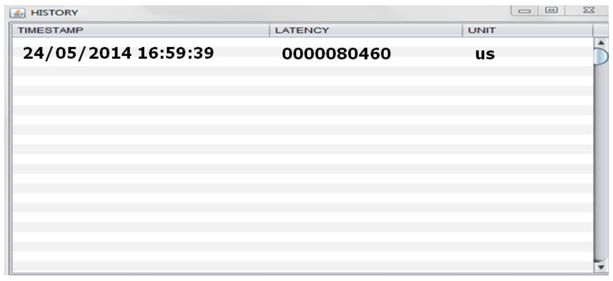
VI.4 Writing values to Database

The following sql query is used to write the values into the database.

```
"insert into sanjeev.latency values (TO_DATE('"
+ timestamp + "', 'dd/mm/yyyy hh24:mi:ss'),'"
+ Integer.parseInt(latency) + "','" + unit + "')";
```

- Timestamp is converted into **Date** Format.
- Latency value is converted into **Integer** Format.
- Unit is used in the same Format(VARCHAR).

The values are stored into the database and the following figure clarifies better.



```
TIMESTAMP              LATENCY       UNIT
24/05/2014 16:59:39    0000080460    us
```

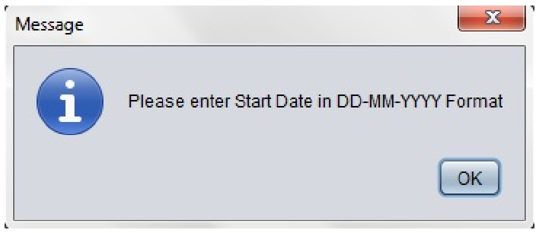
Figure 11: *Data is stored in the Database*

VI.5 Retrieving values from Database

There are two search criteria.

- **By Date:**

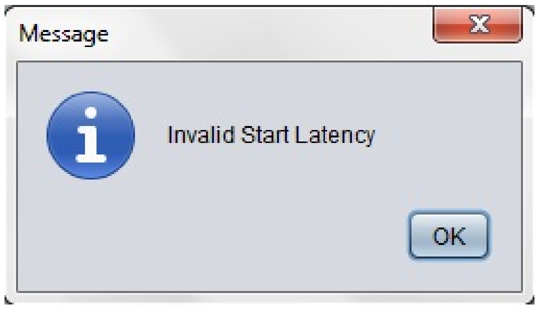
  The Format is *DD-MM-YYYY*. If any other format is given, it throws an exception.


Message

Please enter Start Date in DD-MM-YYYY Format

Figure 12: *Exception for Invalid Date Format*

- **By Latency:**
  The Format is *Integer*. If any other format is given, it throws an exception as well.

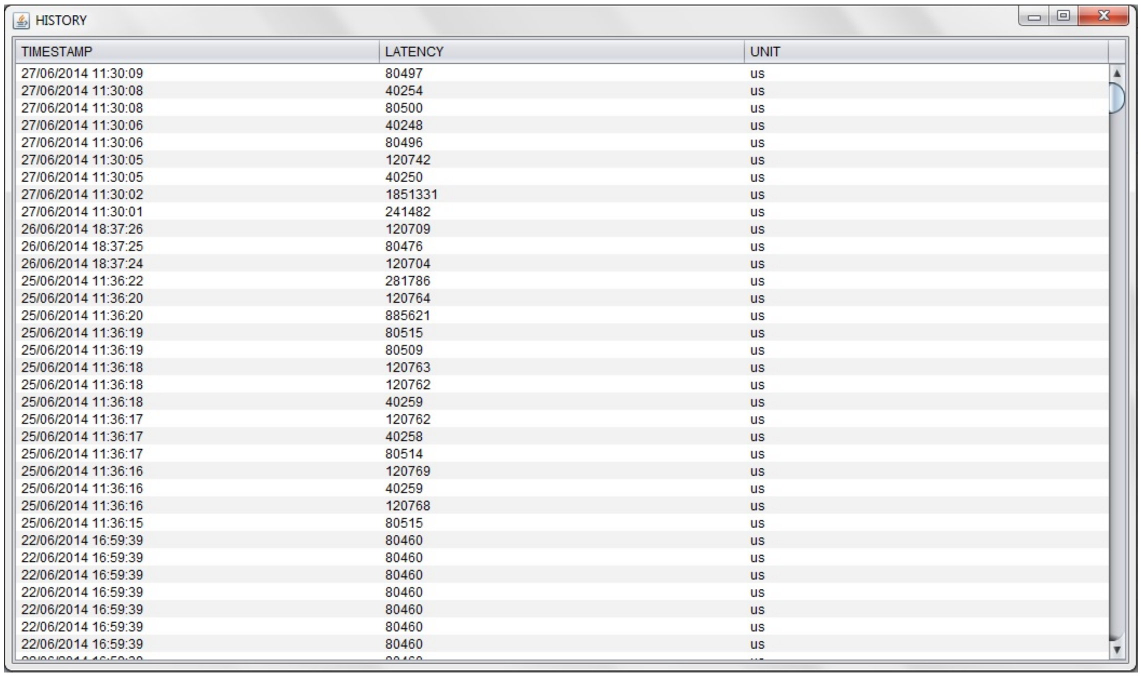

Message

Invalid Start Latency

Figure 13: *Exception for Invalid Latency Format*

VII. Results

- Data was successfully received and stored in the database.
- A GUI is designed for storing and retrieving of the data to and from the data base.



TIMESTAMP	LATENCY	UNIT
27/06/2014 11:30:09	80497	us
27/06/2014 11:30:08	40254	us
27/06/2014 11:30:08	80500	us
27/06/2014 11:30:06	40248	us
27/06/2014 11:30:06	80496	us
27/06/2014 11:30:05	120742	us
27/06/2014 11:30:05	40250	us
27/06/2014 11:30:02	1851331	us
27/06/2014 11:30:01	241482	us
26/06/2014 18:37:26	120709	us
26/06/2014 18:37:25	80476	us
26/06/2014 18:37:24	120704	us
25/06/2014 11:36:22	281786	us
25/06/2014 11:36:20	120764	us
25/06/2014 11:36:20	885621	us
25/06/2014 11:36:19	80515	us
25/06/2014 11:36:19	80509	us
25/06/2014 11:36:18	120763	us
25/06/2014 11:36:18	120762	us
25/06/2014 11:36:18	40259	us
25/06/2014 11:36:17	120762	us
25/06/2014 11:36:17	40258	us
25/06/2014 11:36:17	80514	us
25/06/2014 11:36:16	120769	us
25/06/2014 11:36:16	40259	us
25/06/2014 11:36:16	120768	us
25/06/2014 11:36:15	80515	us
22/06/2014 16:59:39	80460	us
22/06/2014 16:59:39	80460	us
22/06/2014 16:59:39	80460	us
22/06/2014 16:59:39	80460	us
22/06/2014 16:59:39	80460	us
22/06/2014 16:59:39	80460	us
22/06/2014 16:59:39	80460	us

Figure 14: *Final Result*

VIII. Future Tasks

- **Flexible messages and timestamps:**
  - **Flexible messages:**
    Now, we receive just fixed 34 bytes of data. It can be also possible for receiving variable string.
  - **Timestamps:**
    The timestamp detail is sent every time. This can be avoided by synchronizing the time of sender (ARM9 Controller) and receiver (Computer) at start-up of the connection. This helps in reducing the number of bytes as well as reduces the transmission time.
- **Search by time Criteria:**
  Now, we retrieve the values from database based on Date and Latency. A new criteria of time can also be developed.
- **A web Application for retrieving of values:**
  Now, we just work on the local computer and creation of a web application will help in retrieving the values from elsewhere.

IX. List of Figures

References

[1] http://docs.oracle.com/javase/tutorial/uiswing/lookandfeel/nimbus.html

[2] https://code.google.com/p/java-simple-serial-connector/wiki/jSSC_Start_Working

[3] http://plugable.com/drivers/prolific/

[4] https://code.google.com/p/java-simple-serial-connector/wiki/jSSC_examples

[5] http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html

[6] http://www.java2s.com/Code/Jar/o/Downloadojdbc14102040jar.htm

[7] http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html