Wireless Telemetry and Command (T&C) Program

By

Hui Jiang

Dr. Stephen Horan

A technical report in partial fulfillment of the requirement for the Degree

Master of Science in Electrical and Computer Engineering

Manuel Lujan Jr. Center for

Space Telemetering and Telecommunications

New Mexico State University

Klipsch School of Electrical and Computer Engineering

Box 30001, Dept. 3-0

Las Cruces, NM 88003-0001

(505) 646-3012

Sponsored by NASA Research Grant NAG5-7520

April 5, 2000

TABLE OF CONTENTS

ACKNOWLEDGEMENT

This work was supported by the National Aeronautics and Space Administration through research grant NAG5-7520.

CHAPTER 1: INTRODUCTION

The Wireless Telemetry and Command (T&C) program is to investigate methods of using commercial telecommunications service providers to support command and telemetry services between a remote user and a base station. While the initial development is based on ground networks, the development is being done with an eye towards future space communications needs. Both NASA and the Air Force have indicated a plan to consider the use of commercial telecommunications providers to support their space missions. To do this, there will need to be an understanding of the requirements and limitations of interfacing with the commercial providers. The eventual payoff will be the reduced operations cost and the ability to tap into commercial services being developed by the commercial networks. This should enable easier realization of IP services to the end points, commercial routing of data, and quicker integration of new services into the space mission operations. Therefore, the ultimate goal of this program is not just to provide wireless radio communications for T&C services but to enhance those services through wireless networking and provider enhancements that come with the networks.

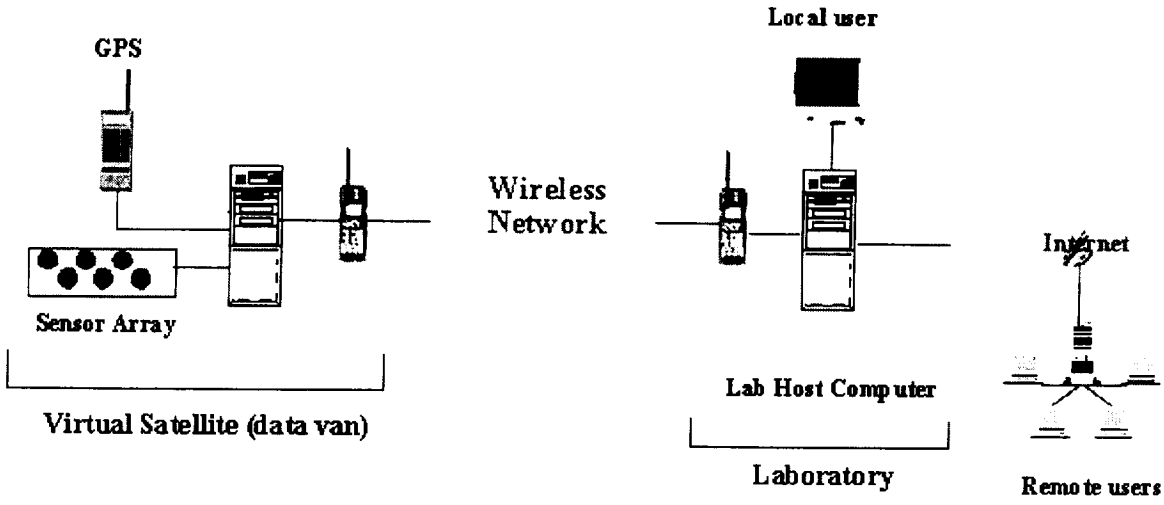
In the following chapters, the detailed technical procedure will be showed step by step. Chapter 2 will talk about the general idea of simulation as well as the implementation of data acquisition including sensor array data and GPS data. Chapter 3 will talk about how to use LabVIEW and Component Works to do wireless communication simulation and how to distribute the real-time information over the Internet by using Visual Basic and ActiveX controls. Also talk about the test configuration and validation. Chapter 4 will show the test results both from In-Lab test and Networking Test. Chapter 5 will summarize the whole procedure and give the perspective for the future consideration.

CHAPTER 2: DATA ACQUISITION

2.1 General Idea of Simulation

2.1.1 System Configuration

The system configuring the Wireless T&C testbed is shown in the following diagram. Sensors will be the data source. These will be located in a data van that can be driven to remote locations for remote operations or data transfer while driving. The data van will have a computer to manage data transfers with the world outside the van. The laboratory in Thomas & Brown will host a corresponding computer to act as an interface between the testbed and the rest of the Internet. The selection of the protocols to manage the data flow across the wireless network will be part of this program. Whenever possible, commercial software will be utilized rather than specifically-written code.



Wireless T&C Configuration

2.1.2 System Data Flow

The data flow will involve the following steps:

Step 1. Sensors

These will acquire the data to form the experimental data source and that the user will eventually monitor and control. The sensors were chosen for easy availability and familiarity for a variety of users.

1. Acceleration - two axes to sense movement (can be correlated with GPS data to provide an interesting display for the user)

2. Temperature

3. Barometric pressure (can be correlated with GPS data to provide an interesting display for the user)

4. GPS position and velocity data

Step 2. Interface computer to gather the data from the sensors and packages it for transmission. Also reacts to operator commands

Step 3. Wireless network to provide bi-directional data transport.

Step 4. Host computer to be the interface between the testbed and the rest of the Internet community. Will also allow local user input and data display.

2.1.3 Development Steps

The following incremental development plan is based upon gradually expanding the capabilities of the testbed and allowing the data end points to become separated at greater distances.

Step 1. Build circuitry to acquire sensor data. This includes the two axis of acceleration data, one temperature sensor, one pressure sensor, and GPS data using one of the LabVIEW interface boards to host the sensors and supply power. Test and verify correct operation of the sensors and the LabVIEW boards.

Step 2. Interface the sensors with the computer for data acquisition using LabVIEW.

1. Acquire data from sensors at a specific rate (one sample per sensor once per 5 to 10 seconds should be sufficient) and package the data into a defined format (synch word, sensor values) for transmission; these can be either in binary or ASCII format as determined easier for transmission.

2. Acquire GPS data at the same rate and supply current position and time measurement as an ASCII string appended to the sensor readings to make a single telemetry frame.

3. Prepare to host configuration in the data van and transmit the data.

Step 3. Connect the laboratory host computer to the sensor data interface computer and enable bi-directional data flow.

1. Use dedicated RS-232 connections (null modems) for example (USB connections in place of RS-232 is also acceptable if the wireless network will have USB connections or a convenient translator is found)

2. Enable data display and data entry at the host computer at the host computer terminal and receive periodic updates from the sensor computer to the laboratory host computer

3. Enable data display and data entry for a remote user station across the Internet and receive periodic updates from the sensor computer to the user computer.

Step 4. Insert the wireless technology (currently, Nokia cell phone on the Voicestream network)

1. replace the null modem connection with the cell telephone service

2. demonstrate remote operations with the van in mobile mode

Step 5. Perform end-to-end T&C operations

1. Demonstrate remote user accessing across the Internet can receive real-time telemetry from the data van

2. Demonstrate remote user accessing across the Internet can execute real-time commands to change some aspect of the data collection

3. Send data files to/from the end point computers

2.1.4 Questions to be Answered by this Program

This development program is being to designed to answer the following minimum set of questions. Naturally, others will grow from these and the basic questions will evolve in time.

A. What are the best ways to send the remote data and interface with the Internet?

B. What are the limitations that wireless providers place on data transfer operations? Do we need additional service options to efficiently transfer the data?

C. What are the best ways to package file, real-time, and streaming data to allow them to co-exist on the wireless channel?

D. How reliable is the data transfer (end-to-end channel BER)? Do any type of underlying protocol options need to be modified for this type of channel error environment?

E. Can the data van computer be given an IP address and be treated as another Internet node and make the wireless network transparent to the user?

F. Can the wireless service provider additional service options, e.g. Short Message Service, be exploited for quick-look telemetry or quick-look commanding?

G. What are the commercial spinoff/public outreach potentials of this program?

H. What will it take to extend this concept to space-based data sources and LEO telecommunications satellite constellation service providers?

2.1.5 Desired Outcomes

The Wireless T&C program should enable NMSU to have the following capabilities:

1. Develop a Virtual Satellite (VS) in the communications data van that can be operated remotely as the van is used as a mobile platform.

2. Provide IP services to the VS as it operates remotely (make the VS a node on the Web).

3. Gain an understanding of the limitations and troubles, as well as the successes, that actual satellites will have in running Internet services over commercial networks and LEO telecommunications satellite constellations.

4. Show commercial spinoff possibilities for NASA-sponsored technology and leverage this technology into other programs.

2.2 Sensor Array Circuit Design

2.2.1 General Design Diagram

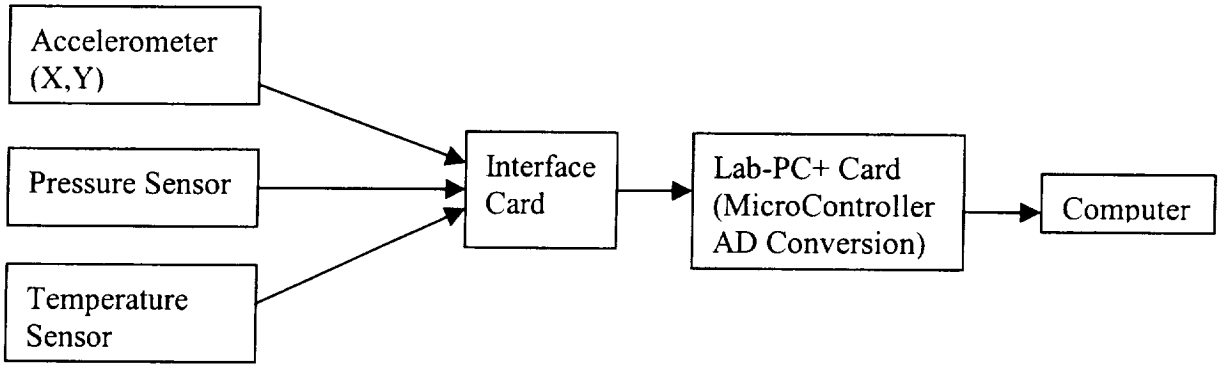


Figure 2.1 General design diagram of sensor array

2.2.2 Acceleration

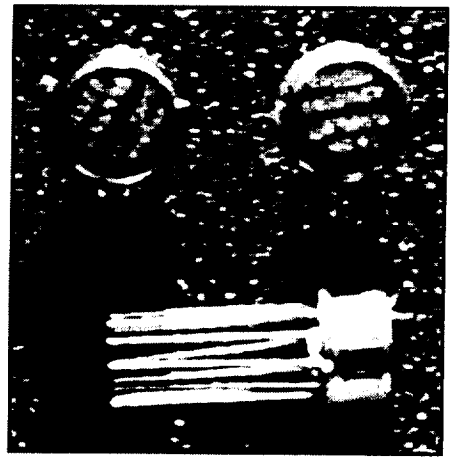
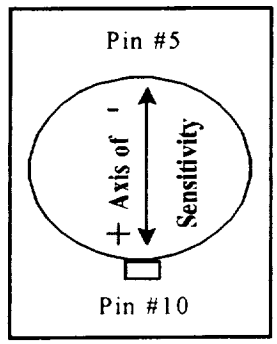


Figure 2.2 - Accelerometers used in the project

The sensor used to measure acceleration is called an **accelerometer**. The accelerometers used in constructing the payload are illustrated in Figure 2.2. There are two ± 5g accelerometers and one ± 50g accelerometer. The particular ones being used here are the same type used in automobile air

bag systems to detect when a sudden deceleration (crash) is encountered. They are smaller than a dime, have 10 wires sticking out of their bottom, and an orientation tab at their base to indicate the direction of measurement. This is illustrated in Figure 2.3 where the accelerometer only measures accelerations along the line between the orientation tab and pin 5. If the acceleration is directed from pin 10 towards pin 5, then the acceleration is negative while if the acceleration is directed from pin 5 towards pin 10, the acceleration is positive. There will be two accelerometers used in this project to give a two-axis measurement system to measure the forces exerted on the data van. The two ± 5g accelerometers will be set to measure the full ± 5 g. Given a positive or negative sign to the acceleration measurement to indicate the direction of the acceleration [4].

Figure 2.3 - Accelerometer orientation direction



For proper measurement, the two 5-g accelerometers should be mounted orthogonally in the circuit. When these measurements are made, the 0-g acceleration should produce a 2.5-V measurement on the output. If the accelerometer is experiencing a + 1-g acceleration, then the output voltage should be larger than 2.5 V. If the accelerometer is experiencing a - 1-g acceleration, then the output voltage should be less than 2.5 V.

The acceleration sensors produce a voltage that is proportional to the acceleration. The mathematical mapping between acceleration, A, in g and output voltage, V, in *Volts* is

$$V = A * const + 2.5$$

where *const* is a numerical constant that depends on the model of the accelerometer used. For the 5-g accelerometers, *const* = 0.400 *V/g*. This equation can inverted to get back the acceleration value once the voltage measurement in the project is made. The equation for the acceleration, A, in g given the measured voltage, V, in *Volts* is

$$A = \frac{(V - 2.5)}{const}$$

The acceleration, A, in g can be converted to acceleration, a, in *meters per second*^{2} by $a = 9.807A$. The acceleration, a, in *meters per second*^{2} is the time rate of change in velocity, v, in *meters per second* or $a = dV/dt$. This can be used to find the velocity by integrating the acceleration, a, using

$$v = \int_0^t a(\tau)d\tau$$

Notice that in the integral, we have $a(\tau)$. This is to indicate that acceleration measurement is time-varying

In a similar way, the velocity, v, in *meters per second* is the time rate of change of the position, z, in *meters* or $v = dz/dt$. Assuming that the rocket flies in a perfectly straight line, all of the motion will be along the z-accelerometer direction. In that case, the distance, z, can be gotten by

$$z = \int_0^t v(\tau)d\tau$$

To compute the integrals, the integral equations are converted into algebraic equations and use numerical integration techniques.

2.2.3 Pressure

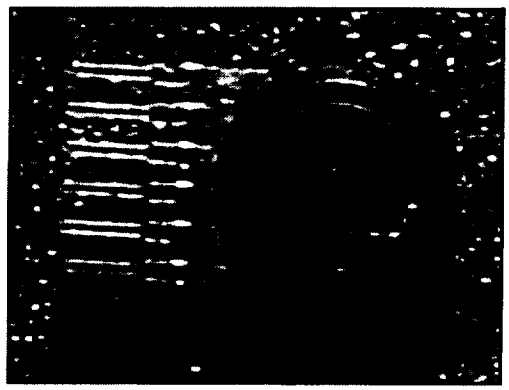


Figure 2.4 Pressure sensor used in project

The earth's atmospheric pressure can be measured with a pressure sensor like the one shown in Figure 2.4. A self-contained sensor was chosen to produces a voltage

proportional to the external pressure. The atmospheric pressure will change as a function of weather and height above the earth as illustrated in Figure 2.5 where data from the U.S. Standard Model Atmosphere have been plotted. There are two curves on the plot: one for summer conditions (squares) and one for winter conditions (crosses). The Model Atmosphere is described by an equation to give pressure, P, in millibars from the height above sea level, h, in kilometers. The summer-months equation is

$$P = 1034.827e^{-0.12807h}$$

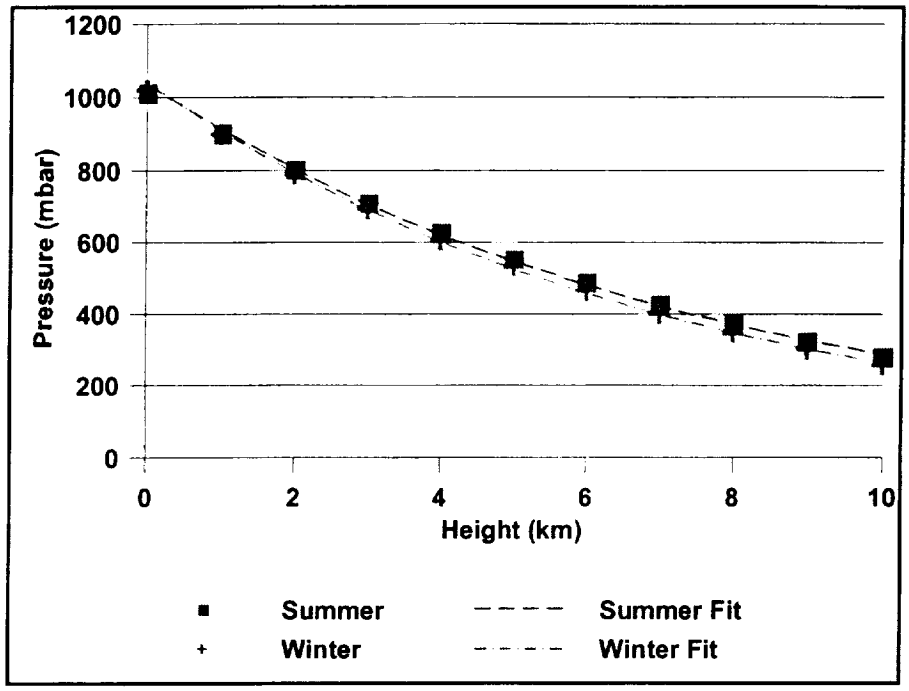


Figure 2.5 - Atmospheric pressure as a function of height for the summer and the winter months (US Standard Model Atmosphere data and the equation modeling the data).

while the winter-months equation is

$$P = 1039.55e^{-0.13737h}$$

The atmospheric pressure can be used as an approximate height indicator. To make a really accurate height indicator, the exact surface atmospheric pressure will be known and then the pressure as a function of height as exactly measured at the test location. That is beyond the

scope of this program but government and scientific research laboratories do those measurements as part of their programs. The model atmosphere equation used here will give a close approximation to the true height measurement[4].

The pressure sensor can be tested by making a measurement of the atmospheric pressure prior to test. Because NMSU are approximately 1 km above sea level in Las Cruces, the atmospheric pressure at ground level is typically 875 millibar so a measurement should be able to make while testing circuit and determine if the pressure sensor is correctly.

The manufacturer of the pressure sensor gives the following mathematical mapping between barometric pressure, P, in millibar and the output voltage, V, from the sensor:

$$V = 5\left(0.0009 * P - 0.10\right)$$

Using this equation, the barometric pressure P can obtained from the voltage, V, with the equation

$$P = \frac{\left(\frac{V}{5}\right) + 0.10}{0.0009}$$

By using the Model Atmosphere for summer conditions, barometric pressure can be converted to an approximate height measurement. The mathematical mapping between pressure, P, in millibar and the height, H, in meters is

$$H = -7808.23 \ln\left(\frac{P}{1034.827}\right)$$

The ln() function is the natural log function (logarithm base e).

2.3 GPS Data Acquisition

The Global Positioning System (GPS) is used in many applications. Part of reason it is so useful is that there are industry-standard methods for representing the data. In this project several

of the standard messages will be examined and use them to determine a position and time setting.

The message format is a structured text string where each filed in the message is separated by commas[5]. The format is

$GPidentifier, field1, field 2, field 3, ..., field N, *hh

where

a. identifier is the name of the message type

b. fields1 through field N are message-specific parameters

c. *hh where is the character "*" and hh a two digit checksum.

The field order and contents are specific to the message type. message types listed in Table will be considered in this project. The exact contents for each message are listed in Table 2.

The data in the fields has specific formats as follows:

a. Longitudes are in degree, minute, and fractional minute format; e.g. 100°43.1234′ would be encoded as 10043.1234. A second field for east or west is encoded as E or W.

b. Latitudes are in degree, minute, and fractional minute format; e.g. 15°25.5678′ would be encoded as 1525.5678. A second field for north or south is encoded as N or S.

c. UTC time is encoded as hours, minutes, seconds, and fractional seconds as hhmmss.ss.

d. Message status is encoded with an "A" to indicate a valid navigational message while a "V" is used to indicate an invalid navigational solution.

e. The date is encoded as day, month, and last two digits of the year; e.g. January 15, 1999 would be encoded as 150199.

From the listed sample messages, determine when and where the samples were taken:

a. longitude, latitude, and altitude

b. date and time

c. number of satellites

Table 1. Message Types

Identifier	Name
GGA	Global Positioning System Fix Data
GLL	Geographic Position Longitude/Latitude
RMC	Transit Specific

Table 2. NMEA GPS Message Formats

Field	Description	Field	Description	Field	Description
GGA	message identifier	GLL	message identifier	RMC	message identifier
hhmmss.ss	UTC	ddmm.mmm	Latitude	hhmmss.ss	UTC
ddmm.mmmm	Latitude	N/S	north or south	A/V	status (valid/ not valid)
N/S	north or south	dddmm.mmmm	Longitude	ddmm.mmmm	Latitude

Table 2. NMEA GPS Message Formats					
dddmm.mmmm	Longitude	E/W	east or west	N/S	north or south
E/W	east or west	hhmmss.ss	UTC	dddmm.mmmm	Longitude
I	Quality	A/V	status (valid/ not valid)	E/W	east or west
Ii	number of satellites	*		speed	knots
d.d	HDOP	hh	checksum	course	degrees
Iiiii	altitude			ddmmyy	Date
M	meters			dd.	magnetic variation
	not used			E/W	variation sense
	not used			*	
	not used			hh	checksum
	not used				
*					
Hh	checksum				

Navigation Messages Example:

$GPRMC,212013.56,A,3215.7824,N,10644.8316,W,00.0,000.0,100499,10.,E*79

$GPGSA,A,3,24,08,05,30,09,10,06,04,,,,,2.0,1.1,1.7*34

$GPGSV,3,1,09,24,73,355,47,05,69,287,48,10,47,149,42,04,34,041,39*7F

$GPGSV,3,2,09,30,33,315,42,08,26,257,42,09,19,218,36,06,12,293,39*71

$GPGSV,3,3,09,07,07,095,,,,,,,,,,,,*4C

$GPGLL,3215.7823,N,10644.8316,W,212014.750,A*20

$GPGGA,212014.75,3215.7823,N,10644.8316,W,1,08,1.1,01186,M,,,,*33

$GPRMC,212014.75,A,3215.7823,N,10644.8316,W,00.0,000.0,100499,10.,E*78

$GPGSA,A,3,24,08,05,30,09,10,06,04,,,,,2.0,1.1,1.7*34

$GPGSV,3,1,09,24,73,355,45,05,69,287,48,10,47,149,44,04,34,041,39*7B

$GPGSV,3,2,09,30,33,315,42,08,26,257,42,09,19,218,36,06,12,293,39*71

$GPGSV,3,3,09,07,07,095,,,,,,,,,,,,*4C

$GPGLL,3215.7823,N,10644.8315,W,212017.000,A*22

$GPGGA,212017.00,3215.7823,N,10644.8315,W,1,08,1.1,01187,M,,,,*30

$GPRMC,212017.00,A,3215.7823,N,10644.8315,W,00.0,000.0,100499,10.,E*7A

$GPGSA,A,3,24,08,05,30,09,10,06,04,,,,,2.0,1.1,1.7*34

2.4 Acquiring Data with LabVIEW

The first design concept was to use BlueEarthResearch Inc.'s TB-Qcom MicroController

to do the AD conversion. The problem is one needs to deal with the signal format (ensure that is ascii or not), and this will occur some problem when using LabVIEW to read the data in. Later, NI's DAQ board was being decided to use, a DAQ type data acquisition method which is composed of two cards, a PCI card which is installed inside in the computer and a SC-207X Series board, a termination breadboards which input the signal from the sensor array circuit built on and connects to the Lab-PC+ card by a data cable. LabVIEW provides a measurement & automation directory in Window's Explore. Since Lab-PC+ card is not Plug&Play card, after installing PCI card, one needs to manually configure the card according to the following steps[6].

Step 1. Click measurement & automation directory. Under that, click devices & interfaces subdirectory.

Step 2. Under that, you will see Lab-PC+ shows up. Press the right button of the mouse, choose property from the popup menu.

Step 3. In " **System**" panel. This panel is used to assign the device number and check or assign system resource to the card. In this case, set device number to be "1", set interrupt request to be "3", direct memory access to be "3", input/output range to be "ox120-ox13f".

Step 4. In "**AI**" panel. This panel is used to select default input settings for the device. In this case, set polarity/range to be "-5v~5v", set mode to be" Referenced Single Ended".

Step 5. In "**AO**" panel. This panel is used to select default output settings for the device. In this case, set polarity to be "Bipolar".

Step 6. In "**Accessory**" panel. This panel is used to select accessory that is attached to the device. In this case, since there is nothing there, just set accessory to be "None".

Step 7. In "**OPC**" panel. This panel is used to select the analog input recalibration period in seconds. This is only need if you are using NI-DAQ OPC OPC (Object Linking and Embedding (OLE) for Process Control) server.

After configuring Lab-PC+, it is time to use LabVIEW to write program to read the signal

from the device. Choose "AI Sample Channel" VI, set device number to be "1", set channel "0", which is interface card's temperature sensor channel. Repeat this step, set the channel number to be the one you connect accelerometer(X/Y), pressure sensor. Then, choose "Format Into String" VI because it is simple and straight-forward, in this VI, set the corresponding parameter into "format string". After that, combine the input with GPS's part and write the whole string to "Serial Port Write" VI.

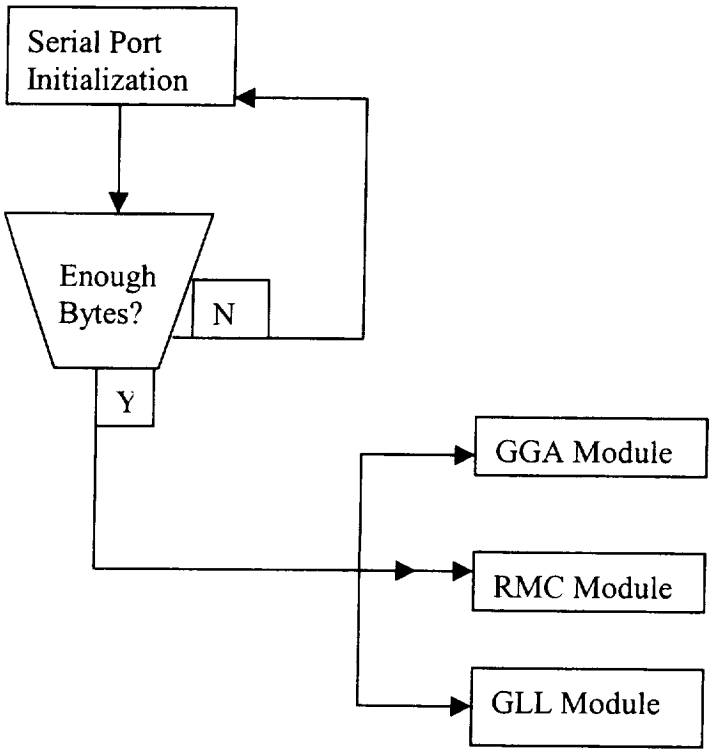


Figure 2.6 GPS data acquisition diagram

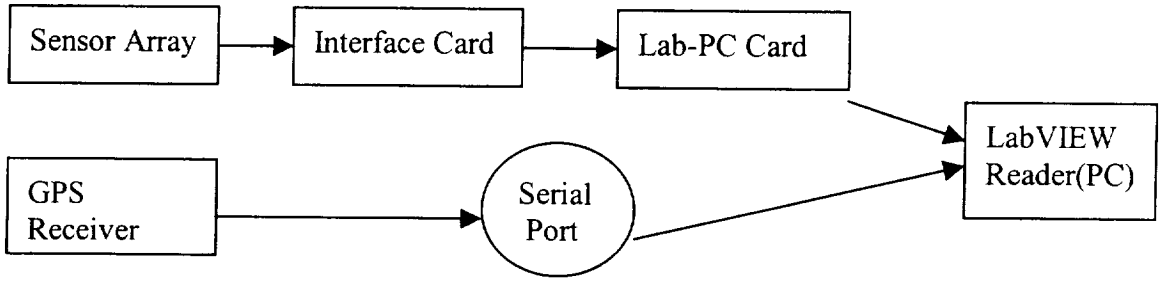
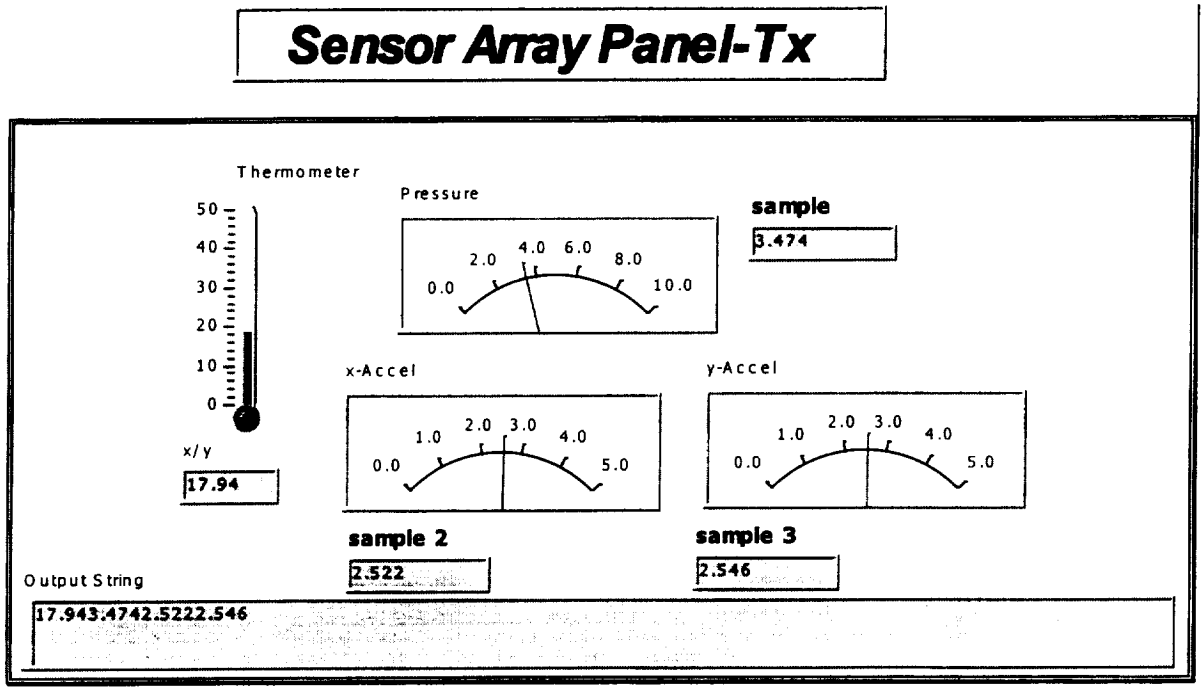


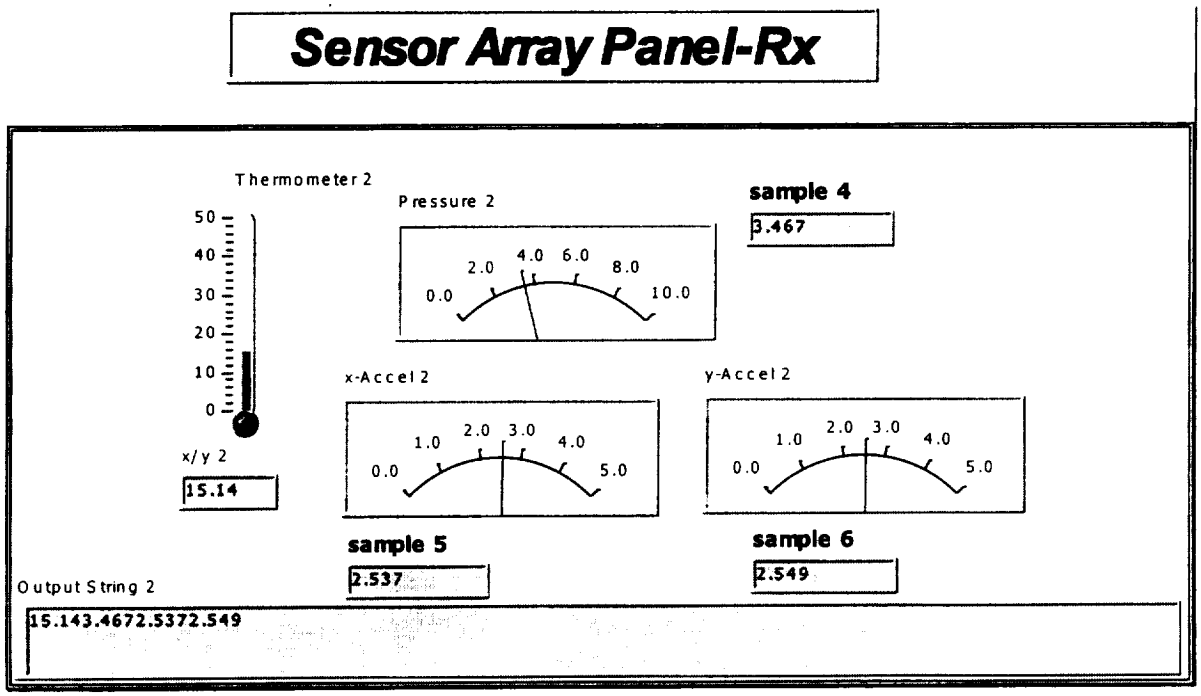
Figure 2.7 Data acquisition diagram used in LabVIEW program

2.5 Testing and Validating LabVIEW Modules

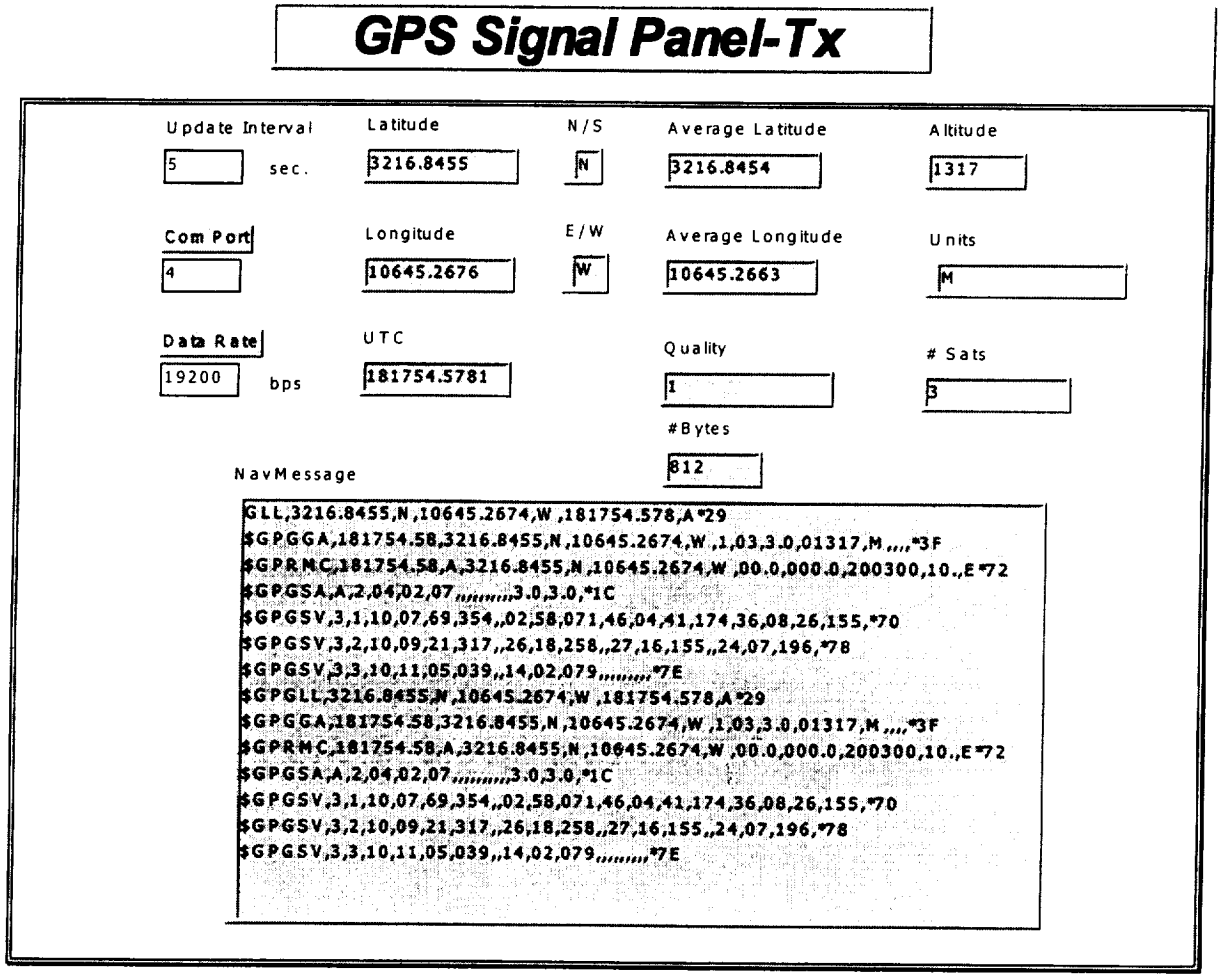
In the sensor array's sender side, one of the test looks like the following graph[1]:



In the receiver side, the result will same at same time if there is no delay. Unfortunately there is about 1 second delay there. In fact, we already achieve the goal of the project since we got the real-time data in just a small time difference. Here is the result of one second delay:

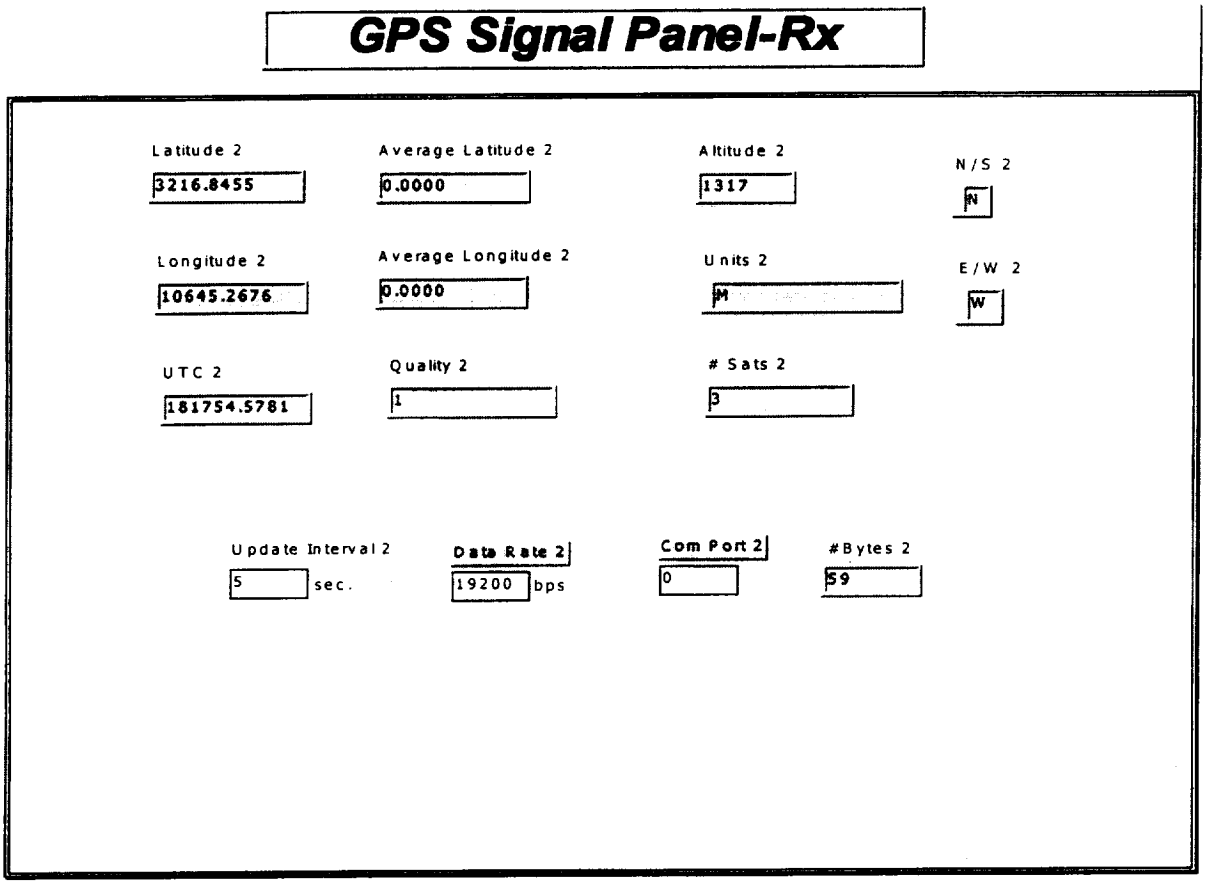


In the GPS message sender part, after setting the proper parameters such as synchronize the data rate between the GPS receiver and LabVIEW reader, here we choose 19200bps since it is the recommended data rate of GPS receiver. Same as updata level. Also make sure the right com port being input in the program according to its actual connection. The result is looks like the following graph:



Note: Generally, it will take about 10-15 minutes to get the initial GPS information upon power up since GPS receiver will need at least 3 satellites to computer exact location data. Sometimes it may take a longer time depending on the experiment environment such as polarity of circumstance, etc.

In the receiver side, the result is same. Here is one second delay graph.



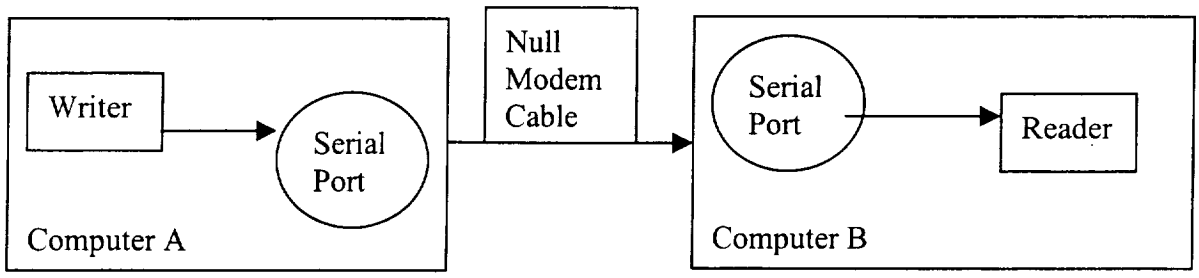
The data on both sender side and receiver side is exactly same. There is no delay effect here since GPS data refresh rate is lower compare to the sensor array data.

CHAPTER 3: SIMULATION

3.1 Wireless Communication Simulation By Using LabVIEW&CW

Wireless communication simulation was done by two steps.

Step1. Use null-modem cable connect one serial port to another serial port in one computer or between two computers. The following figure shows how it works.



In the reader part, use same way as GPS reader does. After initializing the select serial port to the specified settings by using "Serial Port Init" VI, the data being readed from com port is sent to "Scan From String" VI which will scan Scans the input string and converts the string according to format string, so put the exactly same string as writer's "Format Into String" does. Increase the number of parameters by popping up on the node and selecting Add Parameter or by placing the Positioning tool over the lower left or right corner of the node and then stretching it until you reach the desired number of parameters. Then, output the data to the corresponding representation.

Step 2. Use commercial wireless phone service. Order the data transmission service from VoiceStream Inc. After they set access entry parameter of their database, you can use mobile phone to send/receive data stream across the wireless network. The mechanism of reader and writer works the same way as previous one.

Figure 3.1 Basic Diagram of Simulation

3.2 Distribution Over the Internet By Using VB&ActiveX Controls

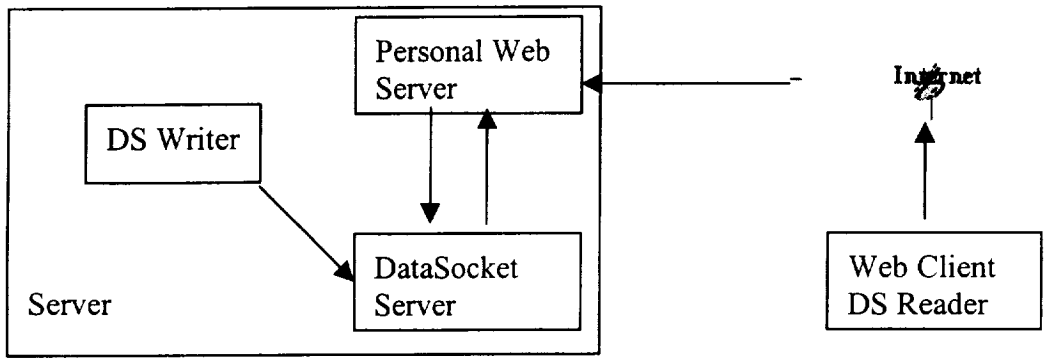


Figure 3.2 General Implementation of Simulation

3.2.1 What is DataSocket

DataSocket is a programming tool that enables user to read, write, and share data between applications and/or different data sources and targets. DataSocket can access data in local files and data on HTTP and FTP servers. If a user uses the general purpose file I/O functions, TCP/IP functions, and FTP/HTTP requests to transfer data, user must write separate code for each protocol. DataSocket provides a unified API for these low-level communication protocols. The User can connect to different data sources without writing different code to support different data formats and protocols because the DataSocket control converts data for transfer and passes the actual values to your application. To connect to a data source location, you need to specify a URL. Like URLs one uses in a Web browser, the data source locator can point to different types of data sources depending on the prefix. The prefix is called the *URL scheme*. DataSocket recognizes several existing schemes, including `http` (hypertext transfer protocol), `ftp` (file

transfer pro-tocol), `file` (local files), and `OPC` (OLE for Process Control). The DataSocket also has a new scheme, `dstp` (DataSocket transfer protocol), for sharing live data through DataSocket Servers. The ComponentWorks DataSocket package includes three tools[2]:

• **DataSocket ActiveX Control** – Component that connects applications to data sources or targets and shares data between them. Because DataSocket is an ActiveX control, you can use it to develop data-sharing applications in ActiveX containers such as Visual Basic, Visual C++, and Borland Delphi.

• **DataSocket Server** – Executable that communicates and exchanges data between two applications using the dstp protocol. For example, in a student lab, the professor runs DataSocket Server on the lab server. When one runs the DataSocket Server on the server, one makes data accessible to other DataSocket applications on the same computer or other computers connected through a TCP network, such as the Internet.

• **DataSocket Server Manager** – Configuration utility for the DataSocket Server through which you can specify the machines that can create items, read items, and write items.

3.2.2 Building an Interactive DataSocket Reader Web Page

To develop the DataSocket client Web page, a DataSocket reader component is created with which users connect to the server, automatically read and display the data, and disconnect from the server. In this project, DSReader Component is created by using ActiveX controls and Visual Basic code to manipulate those controls and this component will be saved as an ActiveX control. After making the ActiveX control, it will be inserted it into a Web page. An ActiveX control is a software component that is used to build applications. Its file extension is `.ocx`.

This part of project consists of four major steps which as following [3]:

Step 1. Creating the DSReader Component Using ActiveX Controls in Visual Basic – Create a component (ActiveX control) using ComponentWorks DataSocket and Visual Basic 6.0. Later, this component will be inserted on Web page. The DSReader component reads and

displays the data sent from the acquisition application.

Step 2. Creating the HTML File – Use the Visual Basic Application Setup Wizard to build the DSReader Web page.

Step 3. Displaying Live Data in the DSReader Web Page – View the DSReader Web page with Internet Explorer. Connect the DSReader Web page to the DataSocket Server and read and display wave data published by the DSWriter application.

Step 4. Interacting with the Data on the DSReader Web Page – Modify the DSReader component to interact with the data. Reload the component and Web page, and reconnect to the DataSocket Server to interact with wave data.

National Instruments recommends that developer can download and use the following tools to build an ActiveX control and insert it on a Web page. All of these tools free over the Web and one must install ComponentWorks 2.0 or later, Visual Basic 5.0 Control Creation Edition or higher version, and Internet Explorer 4.0 or higher version to complete the project.

• **ComponentWorks 2.0 DataSocket and User Interface (UI) controls and DSWriter Executable.**

• **Visual Basic 5.0 Control Creation Edition** – which can be download from the Microsoft Developer Network. Visit `msdn.microsoft.com` and search for **Visual Basic Control Creation Edition** in Additional MSDN Online Areas only.

• **Internet Explorer 4.0 or later and FrontPage Express** – Download Internet Explorer from the Microsoft Web site (`www.microsoft.com`). As installing Internet Explorer, select **Full Installation** from the installation options to install FrontPage Express.

The detailed procedures are as following [3]:

Step 1. Creating the DSReader Component Using ActiveX Controls in Visual Basic

Design a component that connects to the DataSocket Server, reads live data from the

server, and displays that data on a graph or textbox.

Opening a New Project and Loading ActiveX Controls

1. Launch Visual Basic 6.0 from the Windows **Start** menu.

2. In the New Project dialog, open a new ActiveX Control.

3. Right click on the toolbox and select **Components**. If the toolbox is not visible, select **Views»Toolbox**.

4. Select **National Instruments CW DataSocket** and **National Instruments CW UI**. If the ComponentWorks controls are not in the Controls list, press the **Browse** button and select `cwds.ocx` and `cwui.ocx` from the Windows System directory.

5. Click **OK**.

Step 2. Designing the Component

A form is the gray window or area in which controls are placed and indicators to create the user interface for application. The following steps are used to develop the DSReader interface.

1. Place a CommandButton on the form and change the **Name** caption to `ConnectButton` and the **Caption** property to `Connect`.

2. Place a CommandButton on the form and change the **Name** caption to `DisconnectButton` and the **Caption** property to `Disconnect`.

3. Place a TextBox on the form for the source URL. Keep its default properties. Add a label to identify the data source.

4. Place a Label on the form to display the connection status. Change its **Name** caption to `StatusLbl` and **Border Style** property to **Fixed Single**. Delete the default text for the **Caption**

property. Add a label to identify the Status display.

5. Place some textboxes for data representation on the form. Keep its default properties.

6. Place a DataSocket control on the form. The DataSocket control will not be visible at runtime.

Step 3. Developing the Code

After placing controls on the form, write Visual Basic code to respond to events. An event might be an action, such as a mouse movement, or a change in state, such as a completed acquisition. To develop the event procedure for an ActiveX control in Visual Basic, double click the control to open the code editor, which automatically generates a default event procedure for the control. The event procedure skeleton includes the control name, the default event, and any parameters that are passed to the event procedure.

1. In the DSReader component, the DataSocket control is needed to connect to the DataSocket Server and automatically read and update the data when the **Connect** button was clicked. Double click the **Connect** button, and add the following code:

```
Private Sub connectButton_Click()

CWDataSocket1.ConnectTo Text1.Text, cwdsReadAutoUpdate

End Sub
```

2. When the DataSocket control is updated with new data, the textbox shows it. Double click the DataSocket control, and add the following code:

```
Private Sub CWDataSocket1_OnDataUpdated(ByVal Data As CWDSLib.CWData)

Dim Total As String

    Total = Data.Value

    Total1.Text = Total

    Bytes1.Text = Len(Total)
```

```
    Thermometer.Text = Mid(Total, 1, 5)
    Pressure1.Text = Mid(Total, 6, 5)
    XAccel1.Text = Mid(Total, 11, 5)
    YAccel1.Text = Mid(Total, 16, 5)
    UTC1.Text = Mid(Total, 21, 11)
    Latitude1.Text = Mid(Total, 32, 9)
    Longitude1.Text = Mid(Total, 41, 10)
    Altitude1.Text = Mid(Total, 51, 4)
    Sats1.Text = Mid(Total, 55, 1)
    Quality1.Text = Mid(Total, 56, 1)
    NS1.Text = Mid(Total, 57, 1)
    EW1.Text = Mid(Total, 58, 1)
    Units1.Text = Mid(Total, 59, 1)
End Sub
```

3. To display the connection status (whether DataSocket is connected or unconnected) in the Status field, double click the DataSocket control, select `OnStatusUpdated` from the event list in the code window, and add the following code to the `OnStatusUpdated` event:

```
Private Sub CWDataSocket1_OnStatusUpdated(ByVal Status As Long, ByVal Error As
Long, ByVal Message As String)
StatusLbl.Caption = Message
End Sub
```

4. Finally, while DataSocket is retrieving data from the data source, the user can press the **Disconnect** button to disconnect from the server and release system resources used by the DataSocket Server. Once disconnected, the DataSocket control retains the last data loaded so the

user can continue to access it after the connection terminates. Double click the **Disconnect** button, and add the following code:

```
Private Sub DisconnectButton_Click()

CWDataSocket1.Disconnect

End Sub
```

Step 4. Saving the Application and Making the ActiveX Control

I. Naming the Project and Control

The file name is not the same as the control name, so a name is need to assign to the project and control through the Properties Window.

1. Open the Project Explorer, which can be accessed with the **View»Project Explorer** command.

2. Select Project1 in the Project Explorer and change its **Name** property in the Properties Window to **Reader**.

3. Click the form to select it and change its **Name** property in the Properties Window to **DSReader**.

II. Saving the Project

1. Select the **File»Save Project As** command.

2. Save the control as `DSReader.ctl`.

3. Save the project as `DSReader.vbp`.

III. Making the ActiveX Control

1. Select **File»Make DSReader.ocx**. Visual Basic makes the ActiveX control and saves it in the location that the project resides.

2. Select **Project»Reader Properties»Component**, and change the Version

Compatibility to **Binary Compatibility**.

3. Save the project (**File»Save**).

Step 5. Creating the HTML File

The Application Setup Wizard (Visual Basic 5) and the Package and Deployment Wizard (Visual Basic 6) will generate an Internet distribution package, which contains a cabinet (`.cab`) file and the HTML file with the DataSocket Reader component. The cabinet file contains information about the DSReader component, and it is automatically downloaded, expanded, and installed by Internet Explorer so that users on other computers can view the DSReader component with their Web browser. One thing should be noted, if the DSReader component in Visual Basic is change, the OCX and Internet distribution package (`.cab` and `.htm` files) have to be remade.

1. Launch either the Application Setup Wizard or the Package and Deployment Wizard to create an Internet distributable package.

2. Complete the Wizard. As the Wizard prompts you for information, keep the following things in mind:

• Creating an Internet package containing an HTML file and a cabinet file, which specifies the Internet download setup.

• No need to include the property page DLL because property pages wasn't created for the DSReader component.

• Use discretion when marking your ActiveX components safe for scripting or initialization, Accept the default values if not sure about which options to use,

3. Copy the HTML and cabinet files to the same directory on an HTTP server to view the Web page from other net-worked computers or over the Internet. Step 7 will give information about accessing the HTML file with a Web browser using the HTTP protocol. There are three

files associated with the Visual Basic project: `.vbp` (project file), `.ctl` (DSReader component), and `.vbw` (workspace file). These files will be needed to modify the DSReader component in Visual Basic. After modify the Visual Basic project, the `.ocx` file should be remade, which saves the DSReader component as an ActiveX control that can be embedded in an HTML (`.htm`) file. After making the `.ocx` file, the previous `.cab` and `.htm` files can deleted and use either the Application Setup Wizard or the Package and Deployment Wizard to create a new Internet distribution package.

Step 6. Displaying Live Data in the DSReader Web Page

With the following procedure, wireless T&C program can be simulated.

1. Launch the DataSocket Server from the Windows **Start** menu (**Programs»National Instruments ComponentWorks»DataSocket Server**).

2. Open `DSReader.htm` in Internet Explorer.

3. Press the **Connect** button on the DSReader Web page. Notice that DataSocket Server shows that one process is connected and the status on the DSReader Web page confirms that DataSocket is connected. However, no data is being plotted on the DSReader graph or textbox because there is no data being published.

4. Launch the DSWriter executable (`\ComponentWorks\samples\Visual Basic\DataSocket\DSWriter.vi`).

5. Connect circuit and GPS receiver to get the real data. Because DSWriter has not connected to the DataSocket Server, DSReader is not affected.

6. Enter the URL `dstp://localhost/wave` in DSWriter and press **Connect (AutoUpdate)**. DSReader receives the wave data from the DSWriter application.

To use the DSReader Web page on other computers, enter the actual host name of the server, not `localhost`, which connects only to a DataSocket server on the same machine.

Step 7. Accessing the DSReader Web Page with an HTTP Protocol

To view the DSReader Web page from networked computers or over the Internet, post the HTML and cabinet files on an HTTP server. The Application Setup Wizard generates the HTML file and a cabinet file. The cabinet (`.cab`) file contains information about all of the components included in the DSReader component, and it is automatically downloaded, expanded, and installed by Internet Explorer when referenced from an HTML page using the <CODEBASE> tag. The generated HTML includes the <CODEBASE> tag. With its default security setting, Internet Explorer downloads only digitally signed ActiveX controls, which means that the origin of the control can be traced to a registered, commercial software developer. The DSReader component I built is unsigned. To load the DSReader Web page, the security settings in Internet Explorer should be adjusted.

1. View the security options by selecting **View»Internet Options** in Internet Explorer.

2. On the Security tab, select **Trusted site zones** in the Zone listbox and verify that the security level is **low**.

3. Click on the **Add Sites** button.

4. Type in the name of the Web site, including the http://.

5. Uncheck the **Require server verification (https://) for all sites in this zone** option.

6. Click **Add**.

7. Click **OK**.

When loading the DSReader page, a dialog box warning that the ActiveX control is unsigned will get.

3.3 Test Configuration and Validation

In order to get extra serial ports, a PCI card with two extra serial ports being installed and

configured. Also as interface card for National Instruments' Lab-PC++ card being installed in PCI slot inside the computer. Connect Lab-PC+ card which has sensor array built in to the interface card through parallel data cable. Connect GPS receiver to serial port 4.

Running DSWriter program from LabVIEW, setting up "Update Level" as 5 seconds and "Com Port" to be 4, " Data Rate" as 19200 bps. Also, set the output port as serial port 5. In the reader part set "Update Level" and "Data Rate" as same as Writer part except set reading "Com port" as serial port 0. Use a null modem cable (Shielded RS-232C Cable) connect serial port 5 and 0.

Running DSTP server and Personal Web Server. In the web browser, input "http://128.123.9.50/DSReader.htm" in the URL box, input " dstp://128.123.9.50/wave" in the "Source" textbox when ActiveX control being downloaded and installed. Use DataSocket Server Manager to make sure everything else is right.

CHAPTER 4: RESULTS

4.1 In-Lab Test Results

In In-Lab Test, use null modem cable to connect data acquisition machine with lab host. The result is good except there is one second delay in the lab host due to the data transmission delay.

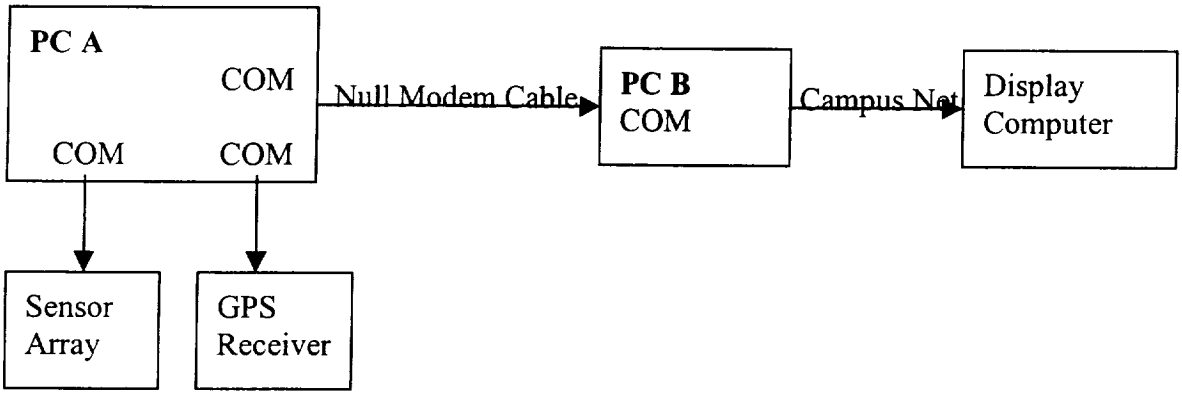


Figure 4.1 In-Lab Test Configuration

Since the result of In-Lab test is same as the networking test. So the results are only given in the following networking test. Here is the figure of sensors array board and GPS receiver that used in the test:

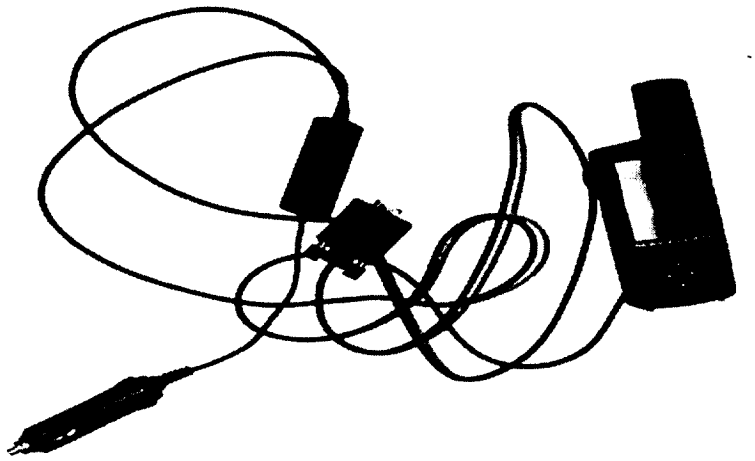


Figure 4.2 GPS receiver used in the test

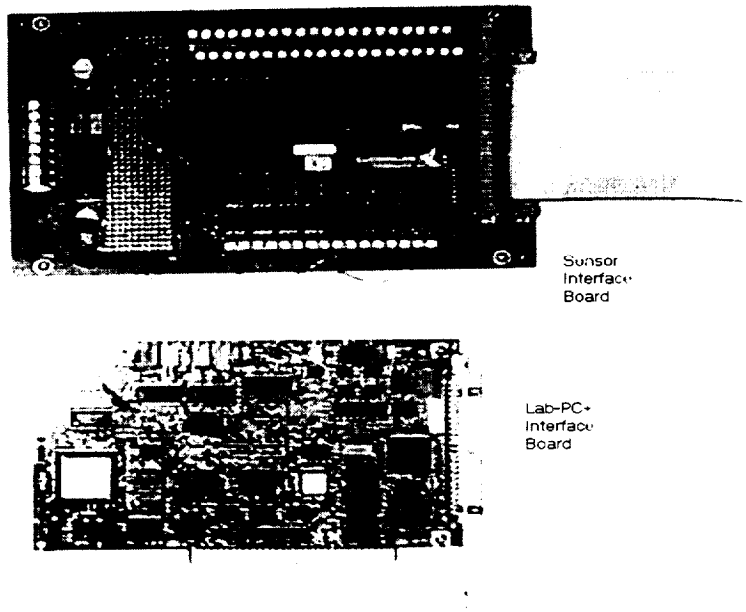


Figure 4.3 Sensors Array Board and Interface Card

4.2 Networking Test Results

In the networking test, the steps are as following:

Step 1. Load cell phone data driver onto PC with data acquisition hardware. Cell phone will use one com port. Or use modem dial up to the networking center to simulate it.

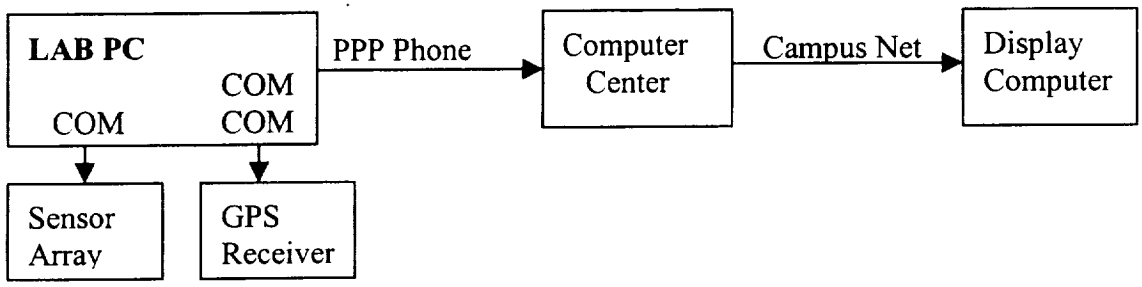


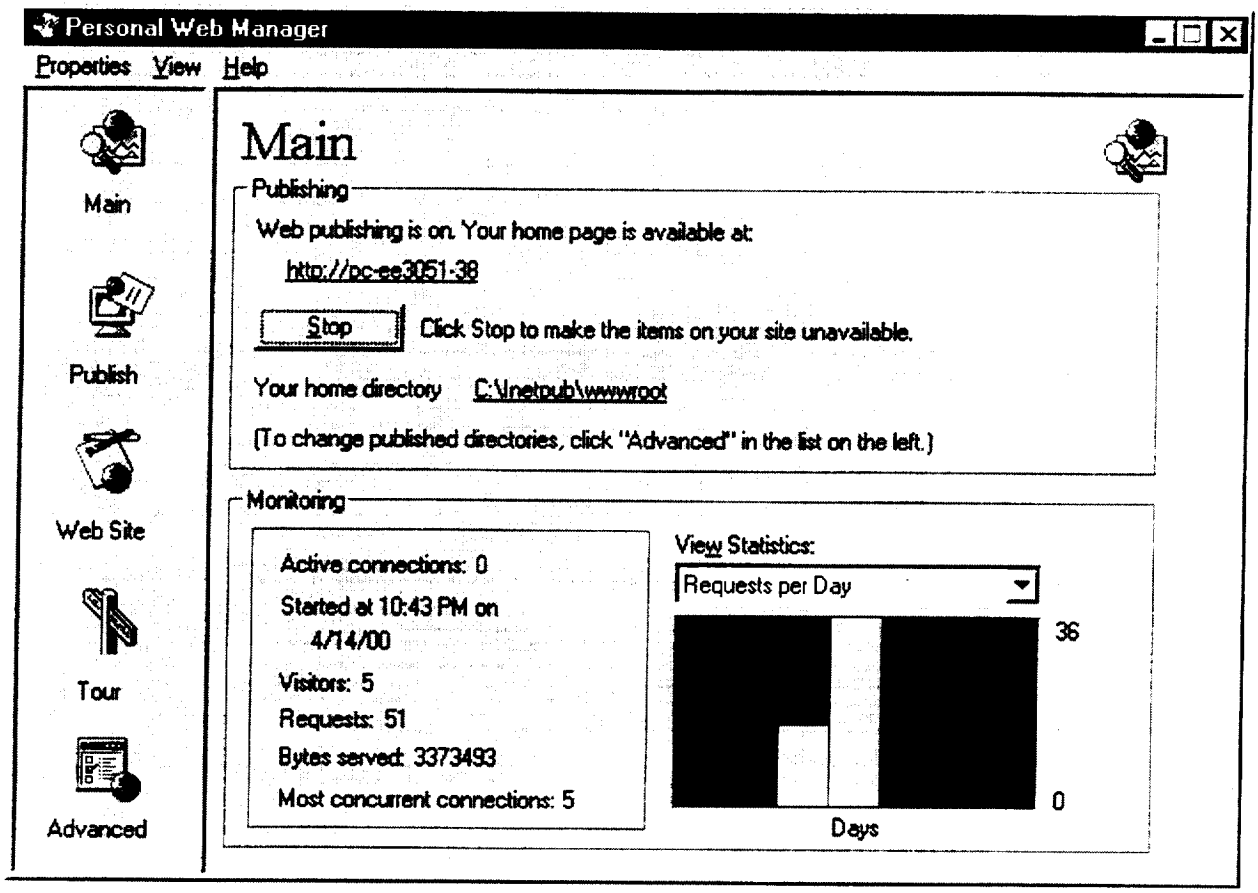
Figure 4.4 Networking Test Configuration

When doing network test, make sure the following procedures being done:

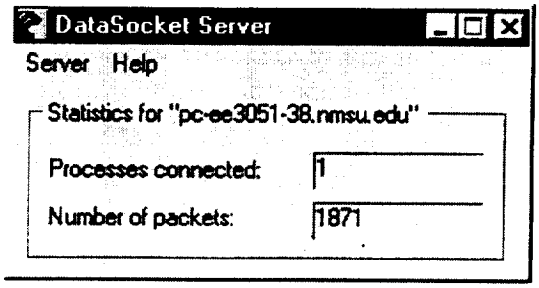
1. Use WinIpCfg.exe which available under C:\Windows directory to find the exact IP address which just allocated by RAS(Remote Access Server)/DHCP(Dynamic Host Configuration Protocol) server.

2. Make sure you type in the correct IP address in both web browser's URL and DSTP's URL.

Step 2 Setup and configure the Personal Web Server and start running it.



Step 3. Make sure the datasocket server is being start and running normal.



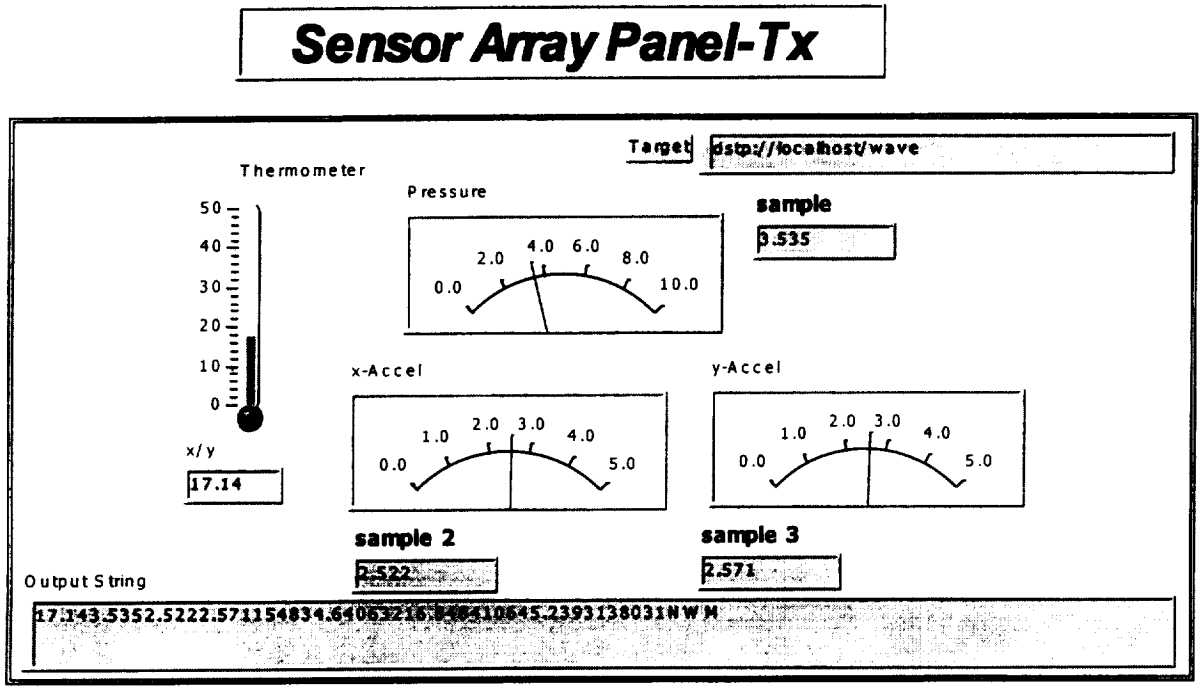
Step 4. To use PC /Cell phone/Phone:

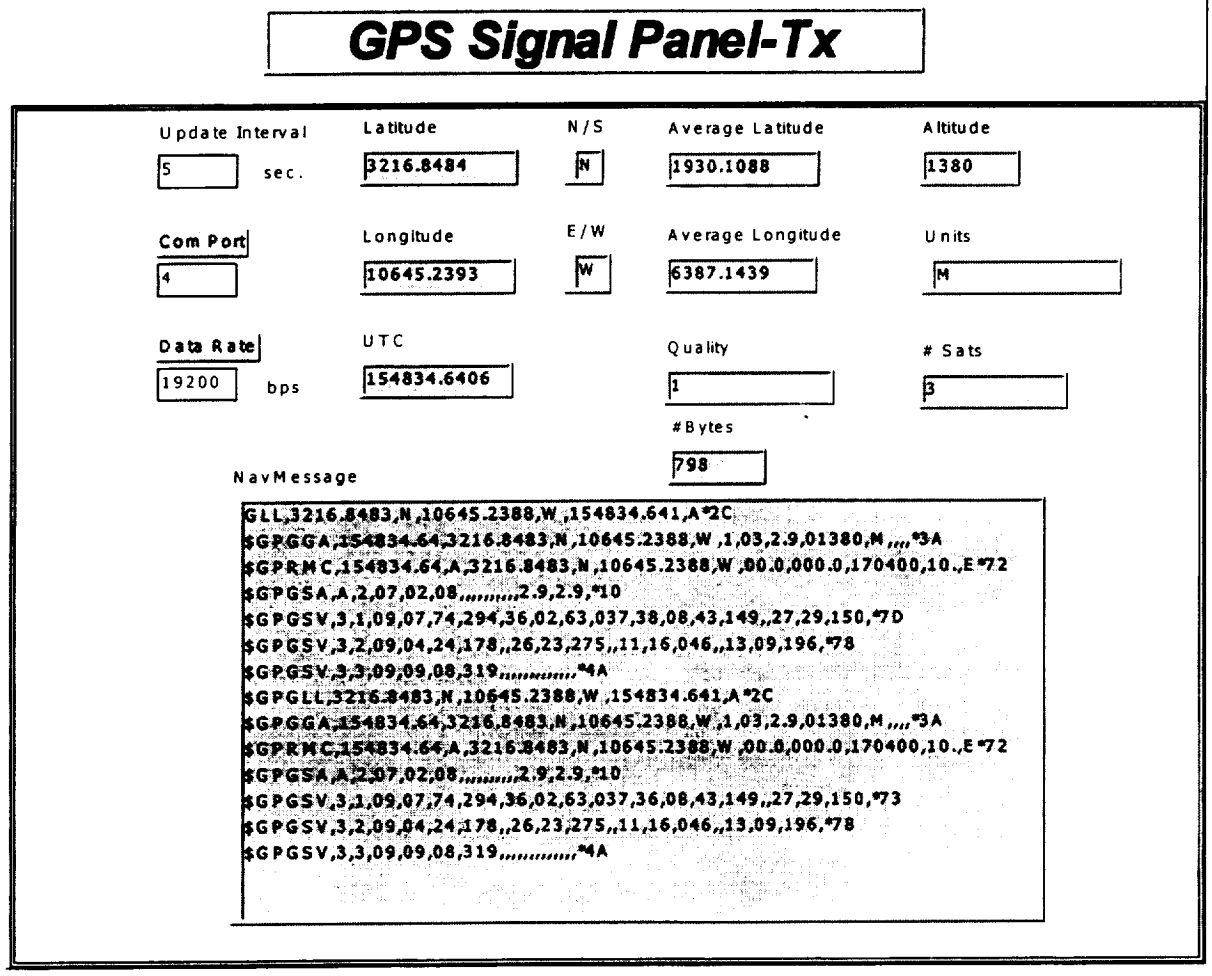
a. Start dial-up networking and call computer center. Check for the settings.

b. Establish IP link between PC with data acquisition hardware and display PC.

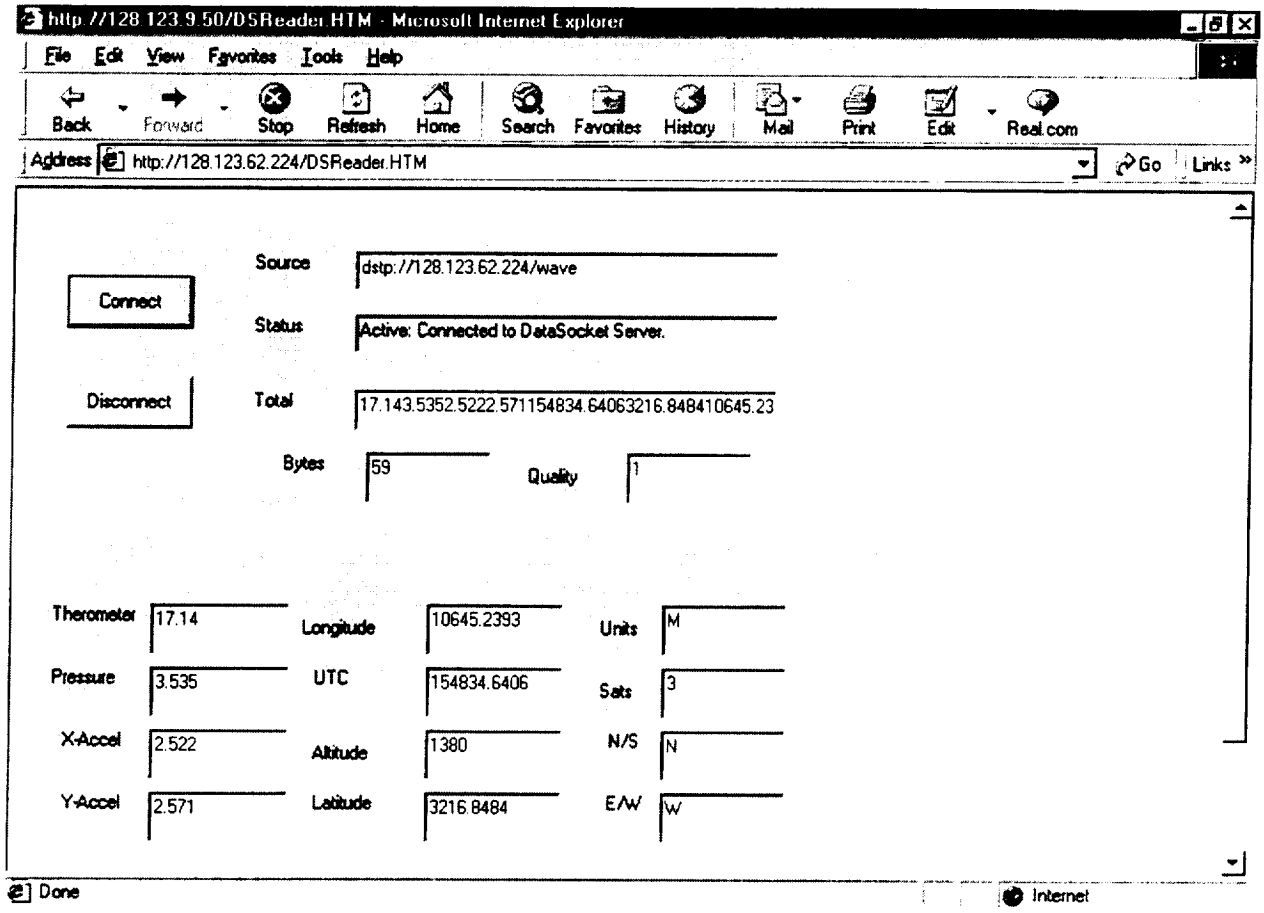
Step 5. Transferring data.

Here is the result shows up in the LabVIEW program in Lab PC.





In the show PC, the result is exactly same as Lab PC.



The test has been repeated both in-lab and networking for more than 20 times, the results are the same. Although the interface is simple text representation, it proves that we can utilize the exist commercial way to do wireless telemetry and command very conveniently. Think about it, people can sit in the office and just by watching the web-browser, they can know what is going on thousands miles far away or even in the real space by using satellite.

CHAPTER 5: CONCLUSION

In this project, an economical and efficient way to do wireless telemetry and command being explored. Implement simple circuit to fulfill data acquisition task, writing simple LabVIEW program to contact with equipment, acknowledge the specific format of GPS messages, researching on how to make data available over the Internet by using Component Works and ActiveX control and write simple code to realize it, integration and cooperation all the stuff together and make them running smoothly , that is all what I learned from this project. During the process, the plan being optimized several times, we try to use several ways to do the project such as choose different MicroController to do the data acquisition, finally we choose Lab-PC+ card etc. Also we meet some difficulties on doing this project such as what is the best way to distribute the data over the Internet so people can use web browser to easily look at it without doing too much work, etc.

This project is just the first step to fulfill wireless T&C program. In order to transfer real-time multimedia data by using commercial way such as possibility of transfer bulk video data by using IP packet data flow and 3G (Third Generation) services of CDMA system. There are still a lot of works to do for dealing with these kinds of topics. There are many issues need to be done in the future.

REFERENCES

1. LabVIEW manuals, National Instruments, 1999.

2. Component Works manuals, National Instruments, 1999.

3. Building an Interactive Web Page with DataSocket, Application Note 127, National Instruments, June 1999

4. Payload Design, Stephen Horan, 1999.

5. GPS Lab, Stephen Horan, 1999.

6. Lab-PC+ Plus manual, 1999.