# UNDERGRADUATE RESEARCH: NOVEL INTEGRATION OF PDAS, GPS AND BAR CODE SCANNER VIA AN EMBEDDED VISUAL BASIC PROGRAM FOR A UTILITY ASSET MANAGEMENT SYSTEM

**Peter Mark Jansson, Jeffrey Tisa, Gregory Garwood**
**Rowan University**

## Abstract

Undergraduates in the Electrical and Computer Engineering Department of Rowan University have undertaken cutting edge software research and system integration for the electric utility clients of its Engineering Clinic Program. In this innovative classroom / laboratory course the problem students set out to solve was the development of an integrated software / hardware system to enable the cost-effective asset management of utility equipment in the field (poles, streetlights, switchgear, etc.). The choice of technology included a handheld PDA, a Bluetooth-based global positioning system (GPS) device and a generic barcode scanner. The challenge was to develop customizable Embedded Visual Basic (eVB) code that would enable any utility personnel to establish equipment location in the field or be directed to any piece of equipment that had been previously located by employing the device in their utility vehicle. Portability, low-cost and customizable graphical user interfaces were criteria for adoption by the companies and the project has resulted in a very affordable smart PDA, that can simultaneously scan a barcode, establish the GPS coordinates in a database and record other important utility information. The device has the capability to also serve as a locator system to lead any utility personnel unfamiliar with parts of the service area to any specific piece of field equipment that has been placed into the system database. This becomes more important to the maintenance of operational efficiency and rapid service restoration as utilities merge and take-over previously unfamiliar regions of service. The integration of the three devices via a single eVB program is an innovation that significantly reduces the cost, complexity and affordability of performing asset management for remote and dispersed equipment. It is also a curricular innovation to provide students with such a cutting edge research opportunity that has immediate commercial application for its clients. The undergraduate students work with their professor to provide the deliverable to the clients in this course and regularly provide status reports and demonstrations of progress to the client to assure that the work will meet specification when completed.

## Project Background

Rowan University College of Engineering has a unique engineering clinic sequence that involves students in real-world projects and problem solving from the freshman through the senior year[1-2]. This interdisciplinary experience enables students to work on exciting and cutting edge applications of the very technologies they learn about in the rest of the engineering curriculum and see how those technologies improve life for business and people. In this project a group of utility companies and their hardware suppliers requested that Rowan engineers develop a low cost, hand-held device that would assist them in the management of their field assets. As utilities merge and the electrical infrastructure changes to meet ever-growing customer electrical demand keeping track of the location and maintenance records of literally thousands upon thousands of pieces of hardware in the field becomes challenging. This problem was a perfect challenge to our students ability to integrate complex systems, develop software applications that could track inventory and its location, interface multiple systems with many software modules, and ultimately develop a working prototype of the final system which could be demonstrated to the project funding companies as satisfying their needs. It was a clear satisfaction of ABET requirements that

Universities interface with companies to assure that what they are teaching is responsive to the current needs of the marketplace.

The handheld device chosen for this project had to be capable of communicating with several devices simultaneously while presenting an intuitive interface to the user. The iPAQ H3970[3] (Figure 1) PDA fit this criterion. This PDA's core processor is an Intel PXA250 400MHz X-Scale processor. It runs on the Pocket PC 2002 operating system. Pocket PC is very similar to Windows therefore, anyone familiar with Microsoft Windows would begin to feel comfortable using the iPAQ rather quickly. The secure digital (SD) slot provides a means to expand the iPAQ’s internal 64MB of RAM with additional non-volatile memory. This is especially useful in our application because in the event that the internal PDA battery completely discharges, the device reverts to factory settings and all personal data and programs not on the SD card are lost. Additionally, there is an expansion slot that can be used for any of several expansion packs available for the PDA and integrated Bluetooth for connecting to wireless peripherals. Finally, the base of the iPAQ contains a serial port that handles RS-232 data communication as well as access to charge the internal lithium polymer battery. This is the port used to synchronize with a PC using Microsoft’s Activesync program.

The objective of the asset management system we needed to design is to assure the GPS has the capability to function simultaneously with a barcode scanner, the PDA and the SD card. Many GPS kits currently on the market either use expansion sleds, similar to Figure 1, or the SD card slot to communicate to the PDA. These approaches would not be adequate since both of these ports on the iPAQ are already performing other required functions. However, the integrated Bluetooth communication feature of the iPAQ PDA was ideal in that it provided an alternative means of communicating with another peripheral device anywhere within a 10 meter radius. The GPS is the perfect peripheral for this application since it is not necessary for the user to have it in his/her possession to work properly. The Bluetooth GPS receiver offered by Socket Communications[4] (Figure 1, center) has received great reviews and is the premier portable GPS integration solution at the present time. It contains a lithium ion battery that is specified to run for a minimum of 6 hours with a full charge. The kit from Socket includes a cigarette lighter adapter for use in automobiles, mapping software (if needed), and a leather case to store the receiver. The accuracy is 10 meters from the actual position, 0.1 m/s from actual velocity, and is synchronized to satellite Greenwich Mean Time within 1 microsecond. An external antenna port is available to potentially increase GPS performance if the receiver does not have access to the open sky within a vehicle.

The barcode scanner for the iPAQ had to be easy to use and integrated with the body of the PDA. Symbol offers the SPS 3000[5] (Figure 1,) in a “sled” that slides directly onto the device. It comes with a software development kit that provides embedded Visual Basic (eVB) drivers that can be customized and incorporated into any eVB program. The barcode scanner makes data collection effortless and helps to eliminate the use of paper forms which can often be returned incomplete or unclear. The equipment described above must be integrated via software to function seamlessly with a single easy to use interface that will ensure detailed and accurate data collection in the field. Microsoft embedded Visual Basic (eVB) was chosen for completing this task because of its ease of use and prominence in the PDA programming industry. The approach taken was to create a “wizard” that guides the user through every step of the data entry process. Additionally, the program takes advantage of Microsoft’s built-in help library included in the Pocket PC 2002 operating system. Drop down menus can be activated by the user to provide help messages to assist with the data entry process. Various automatic alerts have also

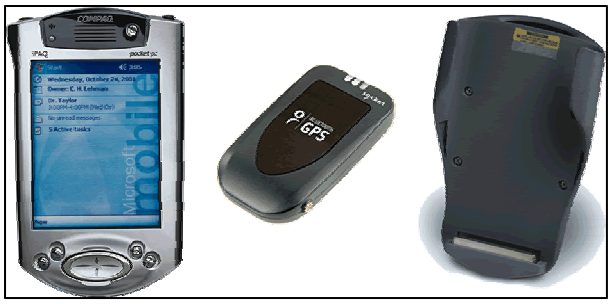


**Figure 1:** iPAQ H3970 (Left), Socket Bluetooth GPS Receiver (Center), and Symbol SPS3000 Barcode Scanner Sled ( Right).

been added using the help system to let the user know if, for example, the PDA is no longer able to communicate with the GPS receiver. It then goes a step further to suggest ways that the user may correct the problem. This will ensure that both experienced and inexperienced PDA users will be able to quickly feel comfortable with the new technology. The program includes options to install utility pole information as well as other utility equipment information that may be involved in a particular installation (i.e. transformers, streetlights, conductors, switchgear, etc.) The data is then stored in a database that can be transferred to a company's central computer system at the end of each workday. Specific details with respect to the format of the database are revealed in the following section.

## System Software Development

The interface between the barcode scanner, GPS and the user is maintained through the PDA. The user interface (UI) was designed to include both the ergonomics of current PDA navigation applications and the advanced functionality required to reliably and effortlessly collect utility asset data in the field. The tasks associated with this often include acquiring utility pole numbers or geospatial information system (GIS) coordinates, date/time information, equipment serial numbers, GPS location information, etc. The focus of this particular program is to have the capability to capture this data and save it to a database to be downloaded at the local utility operations center. In order to implement each aspect of the requested functionality, the tasks were broken down into individual parts, coded in eVB, and then joined together into a single graphical user interface (GUI).

The first, and most arduous task, was designing a navigational program to communicate with the GPS receiver. In order to obtain the GPS information via Bluetooth, an MSCOMM component was used as well as a communications module named "modWatchDog." MSCOMM handles serial communication for the application by sending and receiving data through the serial port, in this case that port is Bluetooth. The "modWatchDog" module handles COMM timeouts, bad data, lost ports, and status messages. The eVB code for this module is shown in Table 1 below.

**Table 1 "modWatchDog" Module**

```
Option Explicit

Public m_intNoComms As Integer

Public Sub ToggleComms()


  If m_intNoComms = 10 Then
    Call ShowStatus(HLP_TIMEOUT, ICN_WARNING, 7000)
    frmMain.tmrWatchDog.Enabled = False
    Call CloseComms
    Exit Sub
  ElseIf m_intNoComms = 2 Then
    Call CloseComms
  ElseIf m_intNoComms > 4 Then
    Call OpenComms
  End If


  m_intNoComms = m_intNoComms + 1

End Sub

Public Sub OpenComms()

  If frmMain.comMain.PortOpen = False Then
    frmMain.comMain.PortOpen = True
  End If

End Sub

Public Sub CloseComms()

  If frmMain.comMain.PortOpen = True Then
    frmMain.comMain.PortOpen = False
  End If

End Sub
```

After communication had been established, a raw data buffer was needed to personally handle the input stream rather then allowing the inherent buffer in MSCOMM to deal with it. This buffer is called "modBuffer" and works hand in hand with MSCOMM saving, resizing, and managing memory. Table 2 contains the code associated with this important module. A second module called "modNMEAParser" asynchronously uses the buffer's data to sort, create, filter, and convert the encoded NMEA input message into usable GPS sentences. These sentences (GPRMC, GPGGA, GPGSA, and GPGSV) are stored in arrays and updated whenever fresh data is available. These two modules are both a fundamental part of parsing GPS data streams.

**Table 2 "modBuffer" Module**

```
Option Explicit

'************************************************
***********
'*  Created By: Greg Garwood           Date:11/05/03

'*
'*  Description:  This module receives the raw data from
'*              the gps module and stores it until its
'*              ready to be parsed.  This is where all
'*              "Garbage" is filtered out.
'*
'************************************************
***********

Public m_strRawBuffer As String
Public m_str4SentenceBuffer As String


'stores the raw data from the com port
Public Sub BufferRawData(strData As String)

   m_strRawBuffer = m_strRawBuffer & strData
   Call ManageBufferSize

End Sub

'manages the size of the buffer incase of an over run
Public Sub ManageBufferSize()

   If Len(m_strRawBuffer) > 1000 Then
      m_strRawBuffer = Right(m_strRawBuffer, 500)
   End If
End Sub

'waits for 4 complete sentences to gather then filters
'the garbage and if successful, sends it to the Parser
'Buffer to be organized

Public Function Create4SentenceBuffer() As String

   Dim i As Integer
   Dim intSentenceCounter As Integer
   Dim intBegPos As Integer
   Dim intEndPos As Integer
   Dim strTemp As String
   Dim strSentenceBuffer As String
   Dim blnOutOfSentences As Boolean


   blnOutOfSentences = False
   'frmMain.lblRawData.Caption = m_strRawBuffer
   Do

      intBegPos = InStr(m_strRawBuffer, SET_BEG)
      If (Not intBegPos = 0) Then
         m_strRawBuffer = Right(m_strRawBuffer,
Len(m_strRawBuffer) - (intBegPos - 1))
         intEndPos = InStr(m_strRawBuffer, SET_END)
         If (Not intEndPos = 0) Then
            intSentenceCounter = intSentenceCounter + 1
            strSentenceBuffer = strSentenceBuffer &
Mid(m_strRawBuffer, 1, intEndPos) & SET_DEL
            m_strRawBuffer = Right(m_strRawBuffer,
Len(m_strRawBuffer) - (intEndPos + 1))
         Else
            blnOutOfSentences = True
         End If
      Else
         blnOutOfSentences = True
      End If

   Loop Until (intSentenceCounter = 4) Or
(blnOutOfSentences = True)

   If (Not blnOutOfSentences = True) Then
      m_str4SentenceBuffer = strSentenceBuffer
      Call SendRawDataToParser
   End If

End Function

'sends the "garbageless" data to the Parser Buffer to be
organized iff the
'parser buffer is ready for more.  if its not then we buffer our
data some more

Public Function SendRawDataToParser() As String

   If m_strNMEA = Empty Then
      m_strNMEA = m_str4SentenceBuffer
      m_str4SentenceBuffer = Empty
      Call SeperateIntoSentences
   End If

End Function
```

The second task involved integrating the barcode scanning function into the program. Symbol provides an easy to use component that may be used to perform specific scanning tasks. However, as with any eVB component, it is based upon the MFC (Microsoft Foundation Classes), and the COM (Component Object Model) is used to give embedded visual C functionality to eVB. This COM did not expose every available function and thus an additional module was needed in order to expose the lower-level functions. Therefore, a function named "modScanner" was written via API (Application Programming Interface) to present the functions previously unavailable through COM. Thus, using the COM and "modScanner" components together provides powerful functionality.

The third aspect in the prototype system is the maintenance of a database containing the data collected in the field between synchronizations at the local office. In order to maintain such a potentially large amount of information in a PDA, XML (extensible markup language) was chosen as the database language. However, XML is not directly supported by eVB, therefore, a module was needed to provide the functionality of XML. This module is called "modXMLWrapper" and again uses the power of API as well eVB object handling to perform quick and easy methods of maintaining a database.

With this powerful foundation established as the underlying code for the UI, an equally powerful method of displaying information was needed. Displaying information on a PDA requires a significant amount of processor time. An interesting, yet obvious, way of display mass amounts of information without overloading the processor is to create an event based display module. The module interfaces with "modNMEAParser" to display only information that needs to be updated. This is done asynchronously when "modDisplayEngine" detects information on the present screen that has changed. Upon detection, an event is raised with the value, sentence, and tag associated with its label. That label, and only that label, is then updated. This means that only one label is being updated when values change rather then every label on the screen. Processor slice time drops significantly and there is no lag time. While all modules are operating asynchronously, a seamless marriage between functionality and ease of use is realized.

Table 3 summarizes the PDA software program as a whole. There are a total of 9 different modules joined together to provide functionality to "frmMain," which is what the user sees when using the program. The first step in parsing the GPS data takes place in "modBuffer." This module receives the data through the Bluetooth serial port and stores it until it is ready to be parsed. When it is time to refresh the screen with new data the buffer filters out the extra information in the message and prepares it to be parsed into usable data. When data is ready to be parsed it leaves the location from which it was stored by "modBuffer" and enters the "modNMEAParser" module. This module is based on the NMEA standard for satellite GPS transmission. It finds and separates the raw GPS data and converts it into units that can be used by the navigation engine and display engine. This involves several unit conversions along with trigonometric calculations to achieve the final effect of a line on a compass pointing in the direction the GPS receiver is traveling. The graphical user interface uses screen images familiar to any iPAQ user but outfitted with data in support of the utility asset management system being developed for our industrial clients. Sample screen shots from the PDA user interface are shown in Figures 2 and 3. The first screen image (Figure 2) represents the data that is being continuously parsed from the datastream passing through the GPS system. This is actually captured by the eVB program and displayed on the PDA screen for the user to observe various parameters such as their current location, traveling speed, total satellites available and current number of satellites in view, altitude, Greenwich Mean Time, etc. The second screen image (Figure 3) shows the user interface where asset information can be entered into the database either by manual key entry using the keypad entry of the iPAQ or via one or two barcode scans that will be automatically initiated if selected. The device was presented to the sponsors during a demonstration of the project status in December 2003 and the customer reaction was quite favorable. Users who had little experience with PDAs found the new equipment installation wizard extremely user friendly on data entry. This was true for use of the barcode scanner or for the manual data entry process. The help engine, as explained above, relays help messages to the user via the main form. A few examples of the functions included in this are:

**Table 3:** *Summary of PDA/Bluetooth GPS Asset Management Program*

| Module/Object Name | Function | Lines of Code |
|---|---|---|
| "frmMain.ebf" | Intefaces directly to the user | 414 |
| "modAddNewWizard.bas" | Controls "add new equipment wizard" through frmMain | 239 |
| "modBuffer.bas" | Recieves raw data from gps module and stores until it is ready to be parsed. Filters out the "Garbage." | 87 |
| "modDeclares.bas" | Declares various variables | 60 |
| "modDisplayEngine.bas" | Controls compass and heading marker on the PDA display through frmMain when GPS screen is active. | 572 |
| "modFunctions.bas" | Custom functions created to perform various tasks not included in eVB | 47 |
| "modHelpEngine.bas" | Controls slidebars that alert user of an error. If the slidebars are clicked a separate form is triggered that attempts to help the user solve the problem. | 54 |
| "modNMEAParser.bas" | Separates filtered sentence from modBuffer into separate NMEA sentences. Scales and calculates all data for use by NavEngine and DisplayEngine | 309 |
| "modScanner.bas" | Uses Symbol SPS3000 drivers to control scanner based on user input. | 69 |
| "modWatchdog.bas" | Keeps track of port status (open/closed), timers, and communication status | 38 |
| "modXMLWrapper.bas" | Creates XML data base and stores information entered by user | 225 |
| | **Size of Program (lines of code)** | |
| | 2114 | |

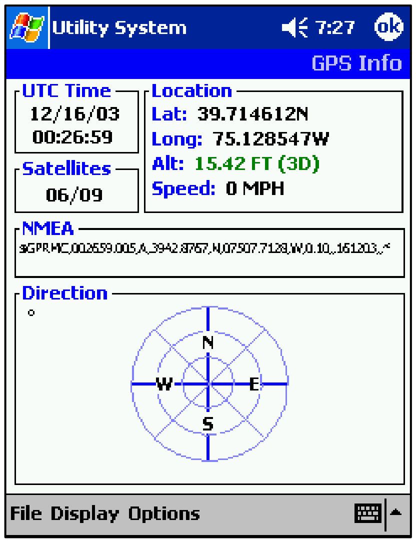
out of Bluetooth communication range (i.e., the GPS data is currently not available to the iPAQ), barcode scanner not attached, and battery low. When clicking on the warning, the help engine is programmed to take the user to a screen that will explain the problem and attempt to suggest ways that it could be resolved. Errors are usually found via the modWatchdog module of the eVB software. This module keeps track of timers, port status, and communication status. Therefore, if the watchdog suspects there is problems the help module will be triggered to alert the user.

The barcode scanner is controlled by one module, "modScanner." This is built upon the Symbol SPS3000 driver library that is included with the barcode scanner. As explained earlier, some additional functions had to be created to integrate the scanners full capability into the customized GPS program. Data that is entered to be stored, whether manually or by barcode, is stored using the "XMLWrapper" module. This module creates a database file written the XML language and stores the data in "tags" that are very similar to the tags seen in HTML language.

The 2100+ lines of eVB code operate together and are quite robust during our initial trial tests. The seamless integration of modules provides a one interface simple to understand tool from the user perspective. In our testing the clients were pleased not only with the speed of the system operation, its completeness but also the ability to store all the collected asset data in a simple to use database that could easily be downloaded to a PC for further processing and analysis at the home office

## Future Plans

The demonstration of the prototype to our utility and utility manufacturer clients has provided us with clear feedback that the project is on the right track. In the coming semesters this clinic


Utility System 7:27 ok

GPS Info

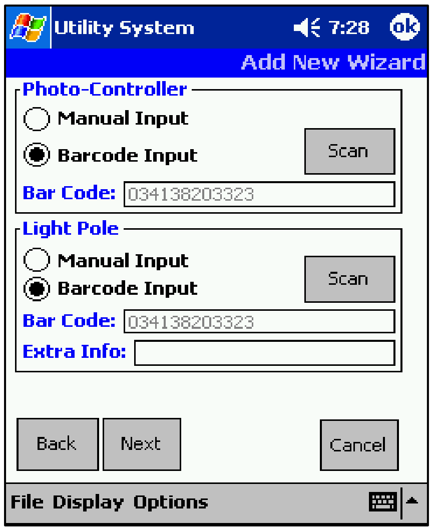
UTC Time
12/16/03
00:26:59

Location
Lat: 39.714612N
Long: 75.128547W
Alt: 15.42 FT (3D)
Speed: 0 MPH

Satellites
06/09

NMEA
$GPRMC,002659.005,A,3942.8767,N,07507.7128,W,0.10,,161203,,*

**Figure 2:** GPS Navigator Screenshot


Utility System 7:28 ok

Add New Wizard

Photo-Controller
( ) Manual Input
(●) Barcode Input
Bar Code: 034138203323

Light Pole
( ) Manual Input
(●) Barcode Input
Bar Code: 034138203323
Extra Info:

**Figure 3:** Barcode/manual Input Screenshot

project will focus on integrating this package with other important utility data acquisition priorities and further demonstrate the robustness of this technology in field trials via a prototype system for Autumn 2004 delivery. The success of this technology demonstrates the value of research performed at the undergraduate level to industry as well as the value of this research experience to engineering students who may be interested in seeking a career in product development, research and development or academia. In future semesters we will also consider alternative means to transfer this data to the homebase at the utility (i.e; new versions with wireless transceiver to communicate data from PDA to home base computer, etc.)

## Bibliography

1. J. L Schmalzel, A. J. Marchese, J. Mariappan and S. A. Mandayam, "The Engineering Clinic: A four-year design sequence," presented at the 2nd An. Conf. of Nat. Collegiate Inventors and Innovators Alliance, Washington, D.C., 1998.

2. J. L Schmalzel, A. J. Marchese and R. P. Hesketh, "What's brewing in the Clinic?," *HP Engineering Educator*,2:1, Winter 1998, pp. 6-7.

3. Hewlitt Packard – iPAQ H3970 specifications: http://h41102.www4.hp.com/Products/ipaq/pocketpc/h3970/specs.html

4. Socket Communications Bluetooth GPS receiver specifications: http://www.socketcom.com/product/GP0804-405.asp

5. SPS 3000 specifications: http://www.Symbol.com/products/mobil_computers/mobile_pocket_sps3000_data.html

## Biographies

Peter Mark Jansson is Associate Professor of Electrical and Computer Engineering at Rowan University and leads numerous Junior Senior Engineering Clinic Teams in solving real world engineering problems each semester. He teaches Networks, Sustainable Design, Power Systems and research includes renewable power systems. He received a PhD from the University of Cambridge, MSE from Rowan University and a BSCE from M.I.T.

Jeffery Tisa, BSECE, is a Masters student in electrical and computer engineering at Rowan University and is completing his research and dissertation on "Transient Analysis of the High Pressure Sodium Lamp/Ballast System Leading to the Projection of Remaining Lamp Life".

Gregory Garwood is a senior Electrical and Computer Engineering student at Rowan University and is pursuing Automation and Control Systems as a concentration as well as Software Engineering. In his spare time he is employed as a Junior Software Engineer at Automation & Control Inc in West Berlin, NJ.