Compact Framework Directional Interface

By

Matt Young

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

June 2006

Compact Framework Directional Interface

By

Matt Young

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

Thomas Matthew Young	Date
Robert E. Schlemmer, Faculty Advisor	Date
Patrick C. Kumpf, Ed.D. Interim Department Head	Date

Acknowledgements/Dedication

The Senior Design process has proven itself to be a fulfilling learning experience. I feel that my project extends past the scope of the current Information Technology curriculum and explores some unique and cutting edge technologies. For allowing this unique type of project, I would like to acknowledge and thank Professor Robert Schlemmer for his support and continued encouragement.

As my project progressed, I had to spend many hours researching and learning about developer tools and methodologies pertaining to the project. The project has in many ways consumed my life. Many of my friends and family deserve a special thank you as they have been forced into knowing way more about this project than they could have ever wished. I appreciate their time spent listening to my various rants about what I needed to accomplish and how I was going to do it.

I have had to make many changes in my life to get on a path towards completion of my degree. Of all challenges faced, this capstone experience has been the most important and overwhelming of my academic career. I am dedicating the CFDI project to my son, Kyle. He has inspired me to be a role model and provided me with the drive and determination to overcome obstacles and reach my goals. It is my hope that he will someday look at my accomplishments and realize he is capable of his own success if he works hard and keeps his eyes open to the opportunities around him.

Table of Contents

List of Illustrations

Abstract

Compact Framework Directional Interface (CFDI)
By
Matt Young

Mobile devices are becoming more useful in their abilities to integrate into a user's mobile lifestyle. With a little imagination, customized mobile applications can improve productivity or simplify everyday tasks. The Compact Framework Directional Interface (CFDI) project was conceived based on a goal to demonstrate this idea and the convergence of some unique and cutting edge technologies. The Windows Mobile 2003 enabled Smartphone packs a lot of powerful tools for mobile applications. Utilizing the .NET Compact Framework and Visual Studio 2003 C#, this application makes use of Bluetooth wireless connectivity, Pocket Outlook Object Model for accessing the built-in contact management database, and Microsoft MapPoint Web Services for Location-based Services and Map Rendering.

The CFDI allows a Smartphone user to wirelessly connect to a Bluetooth enabled Global Positioning System (GPS). The user is able to stream GPS location data from the Bluetooth device and display this information on the mobile device. Using the GPS coordinates and another user interface to reference personal contact addresses, the application can then retrieve current map images from the Microsoft MapPoint Web Service, showing the current location and preferred destinations.

Compact Framework Directional Interface

1. Project Description and Intended Use

1.1 Project Description

Mobile devices are becoming more useful in their abilities to integrate into a user's mobile lifestyle. Specifically, mobile phones are now more affordable and accessible for all demographics. These devices are indispensable tools for business as well as personal use. Having the knowledge and tools for programming mobile devices exposes many valuable functions for the mobile user. With a little imagination, developers can create unique customized mobile applications to improve productivity or simplify everyday tasks.

1.2. Implementation

The Compact Framework Directional Interface (CFDI) project was conceived based on a goal to demonstrate some unique and cutting edge technologies. The first phase of this project consists of creating a wireless data connection between a Windows Mobile 2003 based Smartphone and Bluetooth-capable Global Positioning System (GPS) receiver. The data transmitted by the GPS constantly updates itself and repeatedly sends data strings via the Bluetooth radio. Because there is not a basic Bluetooth data capturing tool available for the Smartphone device, I have created my own application functionality to establish the data connection between devices and also to parse the data. The National Marine Electronics Association (NMEA) has established a widely used standard for GPS device data strings (Appendix A). This standard provides a blueprint for interpreting data from the GPS, allowing the Smartphone application to display a familiar GPS interface

(2). A graphical representation of GPS data is necessary so the user can make use of information provided by the GPS receiver. The directional bearing data is displayed as a 360 degree moving compass, based on physical movement of the GPS receiver in relation to the satellites. The rate of travel, latitude, longitude, current time, and status of the satellite(s) signal is also displayed and updated as changes arise.

To create a useful application for this location-specific data, the project integrates the Smartphone and GPS hardware as an organizational tool designed for mobile professionals. The user is able to reference customer addresses with the application and more efficiently find and route daily tasks. Custom mapping is generated using internet access provided by the cellular network and Microsoft MapPoint Web Services technology. This functionality is unique in the fact that the mapping features can be customized extensively. Customer contacts referenced are displayed on the map as push-pins. Contact information is typically synchronized with a PDA-type device using Outlook and ActiveSync software. Capitalizing on the pre-existing database structure, the application is designed to store customer address information on the device using Pocket Outlook, a built-in feature of Windows Mobile 2003. An address location can easily be picked from existing contacts or added via a friendly user interface, allowing integration with MapPoint's RenderMap Web Service and generated map image. When using my application, the user will be more efficient, not having to fumble with paper maps or address books when dispatched to a scheduled visit.

1.3 User Profile

This customized application is designed with only one user profile in mind. This application is useable by any mobile device user who needs to manage daily destinations or errands. Map rendering and hardware communication are internal features of the Smartphone application. The user is responsible for input of the daily "task" destinations either through a user interface or by choosing from stored contacts in Pocket Outlook. The menu structure conforms to best practices for Smartphone applications, allowing easy one-handed operation by the user (4).

2. Design Protocols

A Use Case Diagram representing the following Use Cases is shown in Figure 1 on the following page. The first Use Case, "View GPS Info", consists of a graphical GPS display, created using Visual Studio 2003. The Bluetooth data connection is established using built-in functionality provided by the .NET Compact Framework on the device and a third-party Bluetooth API to manipulate existing Sockets functionality. The Windows Mobile OS allows a Smartphone user to pair or bond with Bluetooth devices directly using built-in functionality. The OS maintains this list of bonded devices which is accessed by the CFDI application. This allows the user to select the GPS device from a list of devices known to the Smartphone. When the connection is established, a data stream is created in code and I have an internal class designed to read all incoming data and parse it into usable information. Appendix A shows the NMEA-0183 Standard data strings received from the GPS data connection. My application processes the comma

delimited information and displays it in a readable format of latitude, longitude, speed, bearing, time, and satellite status.

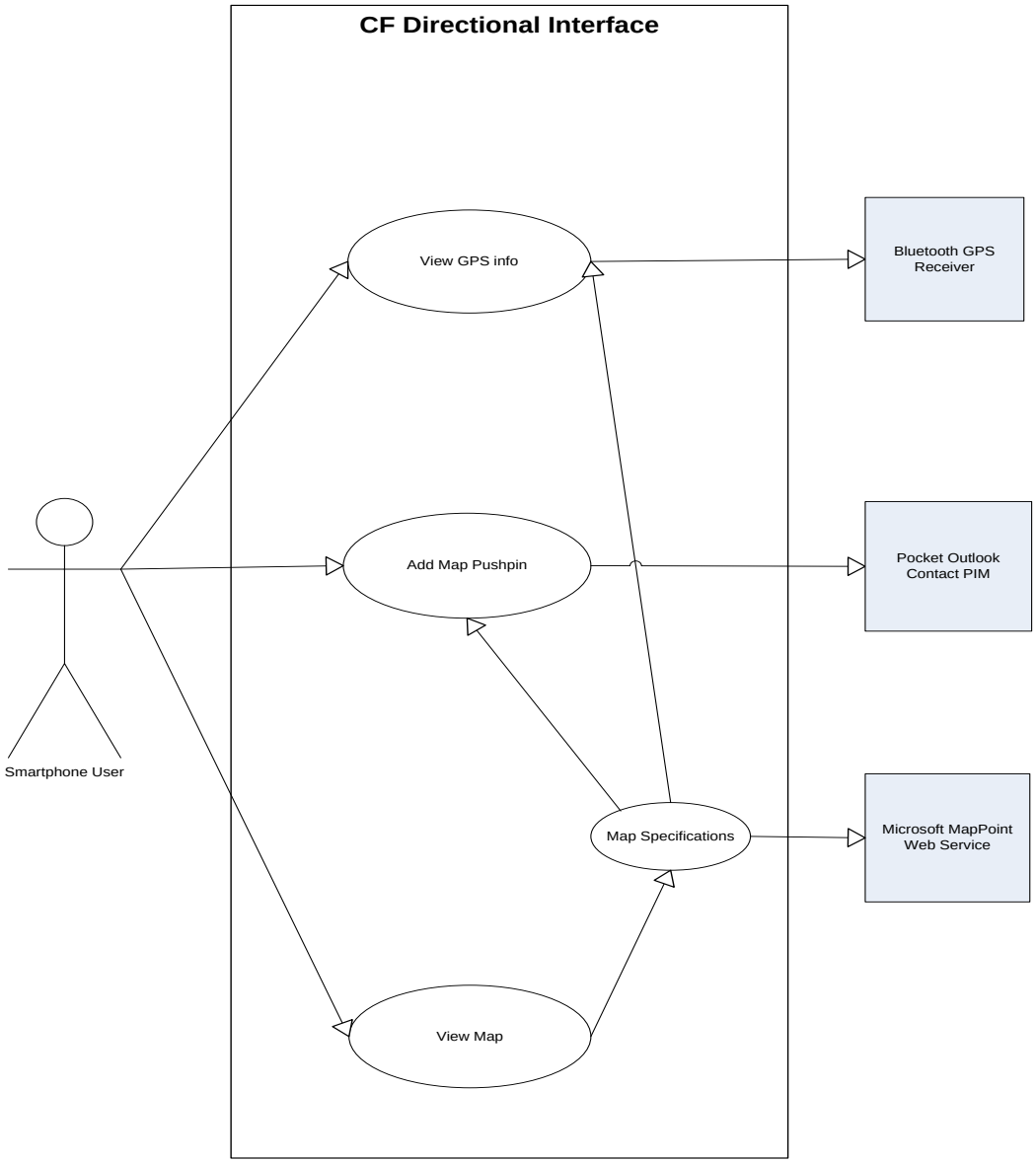

Figure 1: Use Case Diagram – CF Directional Interface

The next Use Case, “Add Map Pushpin”, creates an internal list of daily destinations which can be input manually or by referencing the Pocket Outlook database.

Windows Mobile devices have native code for accessing the Pocket Outlook Object Model. The unmanaged code is most easily manipulated by using a third-party wrapper library. I have chosen to use API's provided by "InTheHand.WindowsMobile.Pocket Outlook.dll (4)." With this I was able to easily gain access to the internal contact list stored in Pocket Outlook. Pocket Outlook is the best solution to storing commonly used addresses, and now can be referenced to add pushpin locations to newly generated map images.

The final Use Case, "View Map," requests information from one or both of the other Use Cases. Using MapPoint technology, the application use data from the GPS unit and contact list to create a Map query. To create a map query, the application must send Map Specifications to the web service. Although there are many different options a developer can pass to the MapPoint Web Service, the Map Specifications for the CFDI primarily consists of an array of pushpin locations, map image size, and latitude/ longitude of the map center. By using MapPoint Web Services, the application footprint is reduced to a size more suitable for a mobile device and information is always current, as it is maintained by Microsoft remotely.

3. Deliverables

The following goals were set for the final deployment of the CFDI application, to be submitted for Senior Design III:

- The user will be able to interact more with the Bluetooth GPS connection.
    - starting and stopping data stream to minimize battery drain
    - choosing GPS device from list of found Bluetooth Devices
- The user will have improved GUI displaying GPS information.
    - Latitude, Longitude coordinates
    - Speed and Directional Bearing
    - Moving compass display of corresponding Bearing data
- The user will be able to input at least two destinations to add to the map rendering specifications sent to the MapRender Web service.
- The device will interact with Microsoft's MapPoint Web Service to retrieve current map graphics for display.
    - Allow pan/zoom functionality for rendered maps
    - Demonstrate use of customized symbols for place markers

4. Project Planning

The following sections show how my time and budget were planned to enable the efficient completion of this project. The project plan consists of a scheduled timeline, resources, and budget.

4.1 Project Schedule

4.1.1 Design Freeze Schedule

Figure 2 shows a timeline of progress made on my prototype application throughout the second quarter of Senior Design. Most tasks were right on target. One change from the baseline plan was implementing map functionality before adding a list of destinations. This switch was logical because the contact location is only useful if map functionality is available.

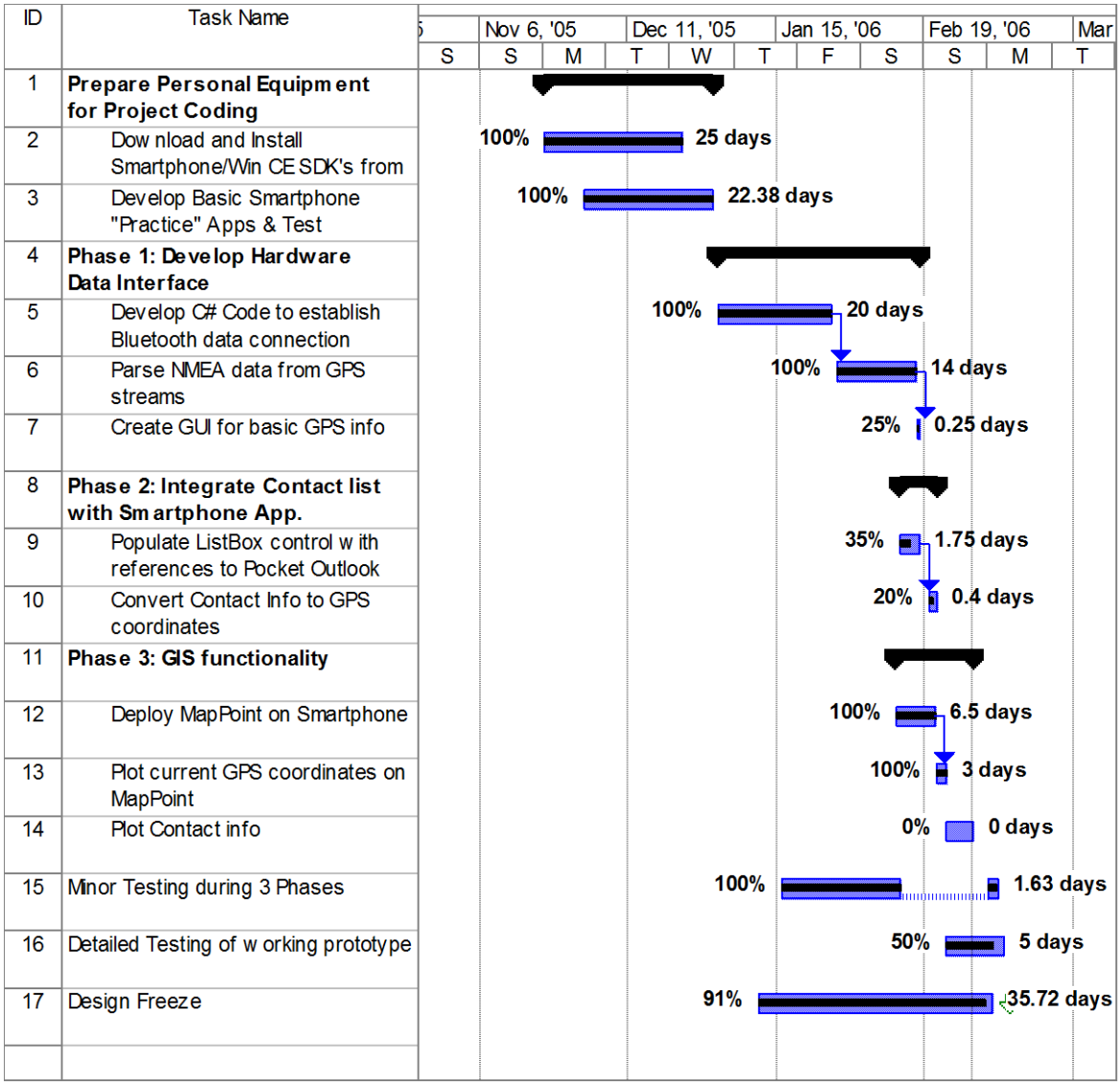

Figure 2: Design Freeze Gantt timeline

4.1.2 Final Deployment Schedule

Figure 3 shows a timeline of progress made while adding final changes to my application throughout the third quarter of Senior Design. All tasks have been completed on target, and the functionality provided by the working prototype of the Design Freeze process has really evolved as more features have been added and refined.

ID	Task Name
1	**Improved functionality for bluetooth connection**
2	Improve Start/Stop Bluetooth Connection Functionality
3	Add UI to choose GPS device from list of known devices
4	Create code for device list functionality
5	**Fix issues with prototype**
6	Change method for running background data processing
7	**Improve GPS data display**
8	Improve Layout and Readability of GPS data
9	Add user-friendly representation of Directional Bearing
10	**Include specific locations in rendered map image**
11	Create UI for addresses to be included in maps rendered
12	**Tech Expo Preparations**
13	Prepare Visual Display for Expo
14	Tech Expo 2006
15	**Improve Map usability**
16	Add ability to zoom and move Map image
17	Upload custom pushpin icon to MapPoint Web Service account
18	Include custom pushpin on map image
19	**Finish Project**
20	Prepare Final Presentation
21	Prepare Final Documentation

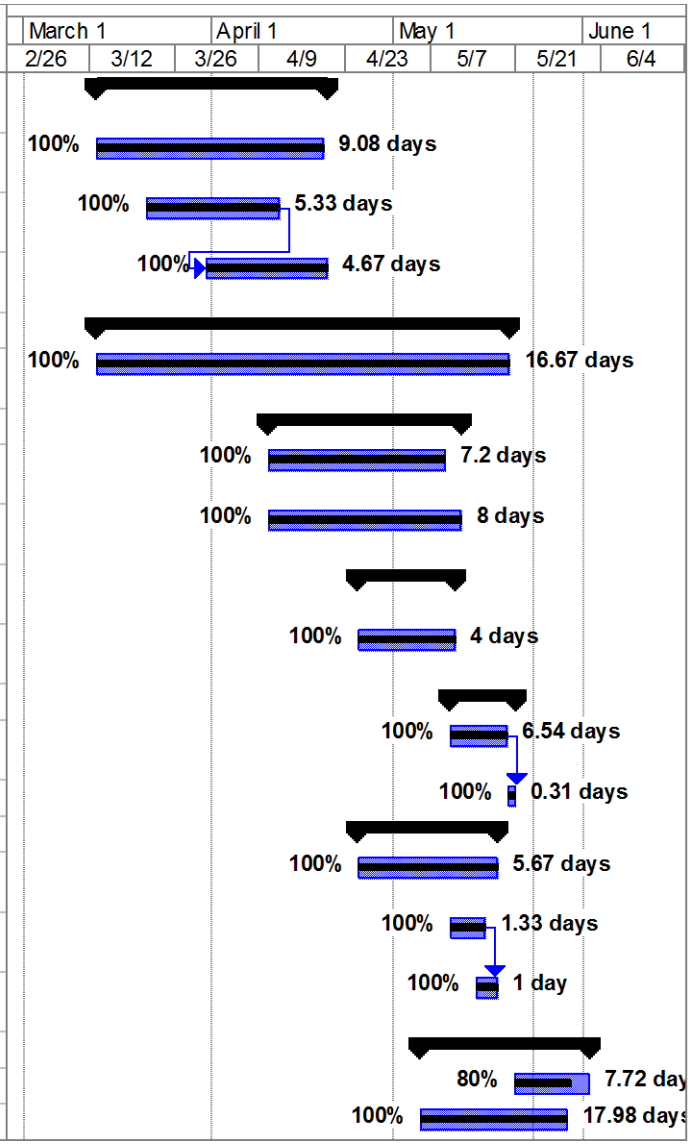

Figure 3: Final deployment Gantt timeline

4.2 Resources and Budget

Table 1 lists different resources used for this project, including hardware equipment, software development tools, persons involved in the progress of this project. Table 2 shows each task and how an estimated budget is applied for the Design Freeze.

ID	Resource Name	Type	Material Label	Standard Rate
1	Microsoft Visual Studio .NET 2003	Material	Software	$1,199.00
2	Matt Young	Work		$14.00/hr
3	Bob Schlemmer (robert.schlemmer@uc.edu)	Work		$80.00/hr
4	Microsoft MapPoint 2004	Material	Software	$299.00
5	Motorola MPx220 Smartphone	Material	Device/Hardware	$199.00
6	Pharos iGPS-BT Bluetooth GPS receiver	Material	Device/Hardware	$171.99
7	Kensington USB Bluetooth Adapter	Material	Device/Hardware	$24.88
8	C# SDK for Smartphone (MSDN Download)	Material	Software	$0.00
9	Smartphone Emulator Images (MSDN Download)	Material	Software	$0.00
10	OpenNetCF.Bluetooth class libraries (OpenNetCF.org)	Material	Software	$0.00
11	InTheHand.WindowsMobile.PocketOutlook class libraries	Material	Software	$0.00

Table 1: Resources

ID	Task Name	Total Cost	Variance	Actual	Remaining
1	**Prepare Personal Equipment for Project Coding**	**$2,002.00**	**$2,002.00**	**$2,002.00**	**$0.00**
2	Download and Install Smartphone/Win CE SDK's from	$336.00	$336.00	$336.00	$0.00
3	Develop Basic Smartphone "Practice" Apps & Test	$168.00	$168.00	$168.00	$0.00
4	**Phase 1: Develop Hardware Data Interface**	**$2,663.01**	**$2,663.01**	**$2,282.11**	**$380.90**
5	Develop C# Code to establish Bluetooth data connection	$1,756.28	$1,756.28	$1,756.28	$0.00
6	Parse NMEA data from GPS streams	$224.00	$224.00	$224.00	$0.00
7	Create GUI for basic GPS info	$507.87	$507.87	$126.97	$380.90
8	**Phase 2: Integrate Contact list with Smartphone App.**	**$1,063.00**	**$1,063.00**	**$335.02**	**$727.98**
9	Populate ListBox control with references to Pocket Outlook	$759.00	$759.00	$265.65	$493.35
10	Convert Contact Info to GPS coordinates	$224.00	$224.00	$44.80	$179.20
11	**Phase 3: GIS functionality**	**$2,312.34**	**$2,312.34**	**$1,536.76**	**$775.58**
12	Deploy MapPoint on Smartphone	$927.00	$927.00	$927.00	$0.00
13	Plot current GPS coordinates on MapPoint	$535.00	$535.00	$535.00	$0.00
14	Plot Contact info	$737.72	$737.72	$0.00	$737.72
15	Minor Testing during 3 Phases	$488.00	$488.00	$488.00	$0.00
16	Detailed Testing of working prototype	$451.87	$451.87	$225.94	$225.94

Table 2: Design Freeze Schedule Budget

ID	Task Name	Total Cost	Variance	Actual	Remaining
1	**Improved functionality for bluetooth connection**	**$648.96**	**$648.96**	**$648.96**	**$0.00**
2	Improve Start/Stop Bluetooth Connection Functionality	$217.92	$217.92	$217.92	$0.00
3	Add UI to choose GPS device from list of known devices	$165.60	$165.60	$165.60	$0.00
4	Create code for device list functionality	$265.44	$265.44	$265.44	$0.00
5	**Fix issues with prototype**	**$1,116.96**	**$1,116.96**	**$1,116.96**	**$0.00**
6	Change method for running background data processing	$1,116.96	$1,116.96	$1,116.96	$0.00
7	**Improve GPS data display**	**$607.92**	**$607.92**	**$607.92**	**$0.00**
8	Improve Layout and Readability of GPS data	$172.80	$172.80	$172.80	$0.00
9	Add user-friendly representation of Directional	$435.12	$435.12	$435.12	$0.00
10	**Include specific locations in rendered map image**	**$330.96**	**$330.96**	**$330.96**	**$0.00**
11	Create UI for addresses to be included in maps rendered	$330.96	$330.96	$330.96	$0.00
12	**Tech Expo Preparations**	**$0.00**	**$0.00**	**$0.00**	**$0.00**
15	**Improve Map usability**	**$471.36**	**$471.36**	**$350.88**	**$120.48**
16	Add ability to zoom and move Map image	$314.40	$314.40	$193.92	$120.48
17	Upload custom pushpin icon to MapPoint Web Service account	$57.12	$57.12	$57.12	$0.00
18	Include custom pushpin on map image	$99.84	$99.84	$99.84	$0.00
19	**Finish Project**	**$0.00**	**$0.00**	**$0.00**	**$0.00**

Table 3: Final Deployment Schedule Budget

5. Proof of Design

I have compiled some images showing the working functionality of my application. The images that follow demonstrate critical tasks and use cases of the CFDI application. These images also convey how the application is navigated to create a complete user experience.

- Figure 4 shows images of the process to establish a Bluetooth connection with the Pharos GPS device. The Item list is directly linked to the built-in Bonded Bluetooth Devices list in the Smartphone OS.

Figure 4: Bluetooth Connectivity

- Figure 5 shows how data has been successfully parsed into usable information and the GUI is updating with GPS information upon corresponding events.

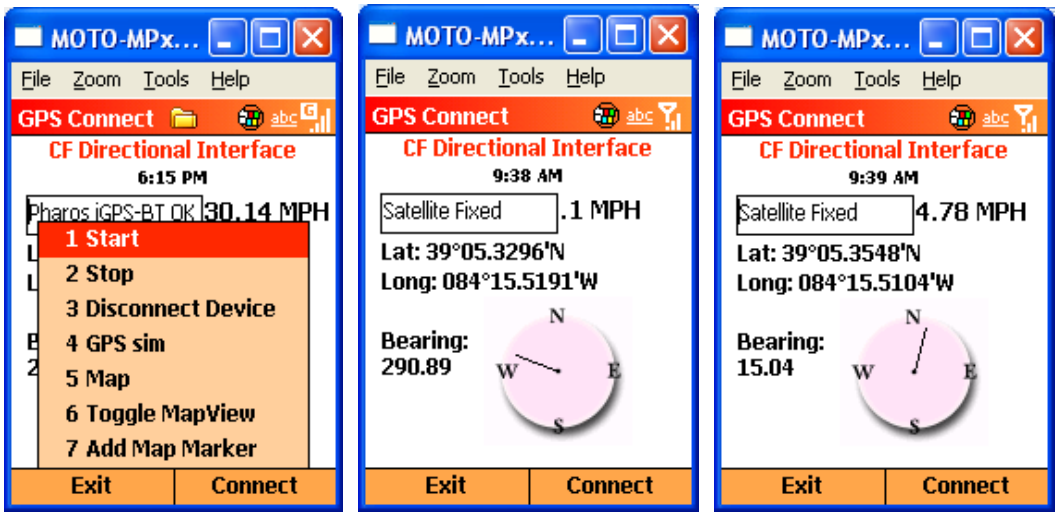

Figure 5: Readable GPS data

- Figure 6 shows MapPoint Web Service successfully responding to Evaluation credentials and location specifications. Also menu selections controlling Map View Use Case are shown. Map image shown also demonstrates usage of custom location icons on map images.

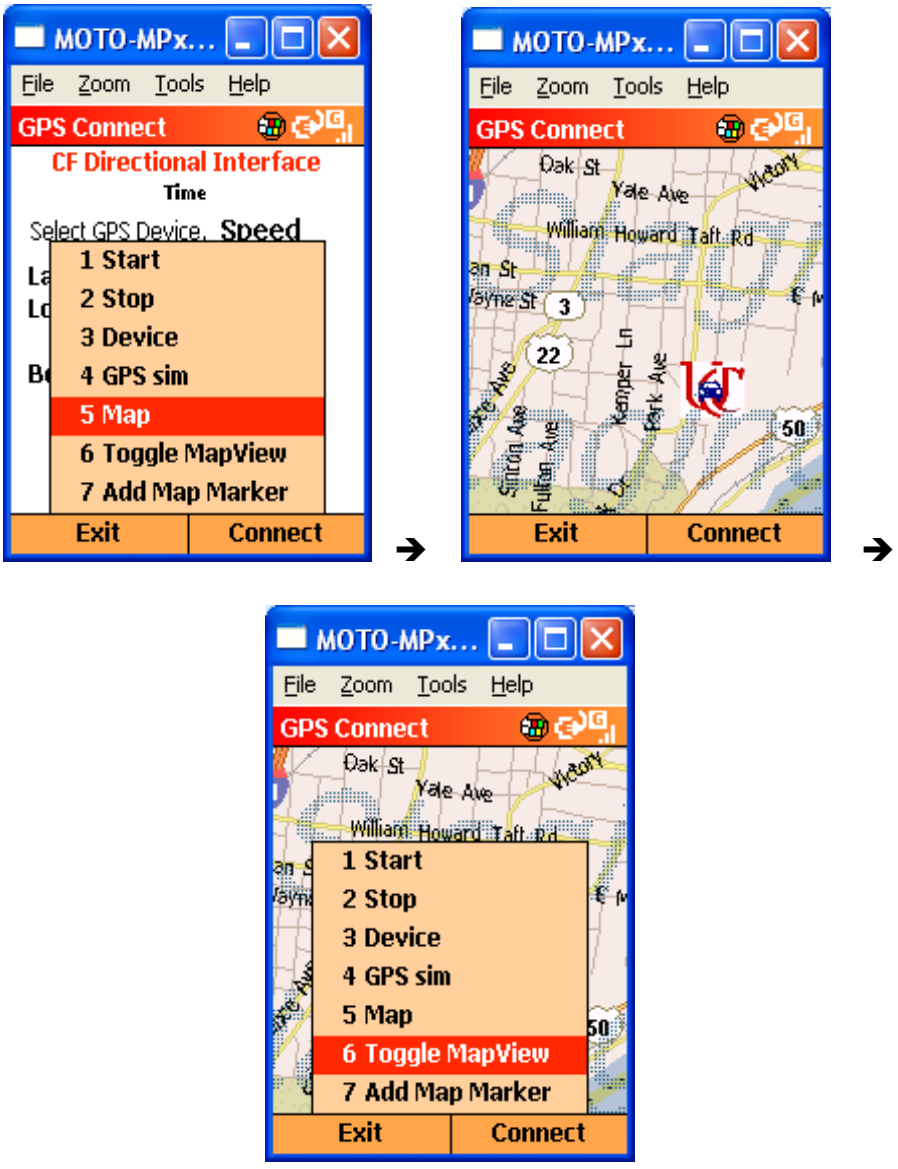

Figure 6:
Menu Selections and Bitmap image from MapPoint Web Service

- Figure 7 shows the functionality for the Add Map Pushpin Use Case. The menu item is selected and the item list linked to the Smartphone's internal Pocket Outlook contact list is displayed for choosing a contact. After a contact is chosen, if an address exists in the Pocket Outlook database, the UI input fields are automatically populated. If the user chooses, he/she may manually enter address information via the phones keypad. The MapPoint *findservice* Web Service returns latitude and longitude coordinates for the address and the location for the pushpin is added to the map specifications for the next map image request.

Figure 7: Add Map Pushpin Use Case functionality

- Figure 8 shows Pan/Zoom functionality of the map image. The Smartphone user can move the map image by pressing the directional pad on the device.

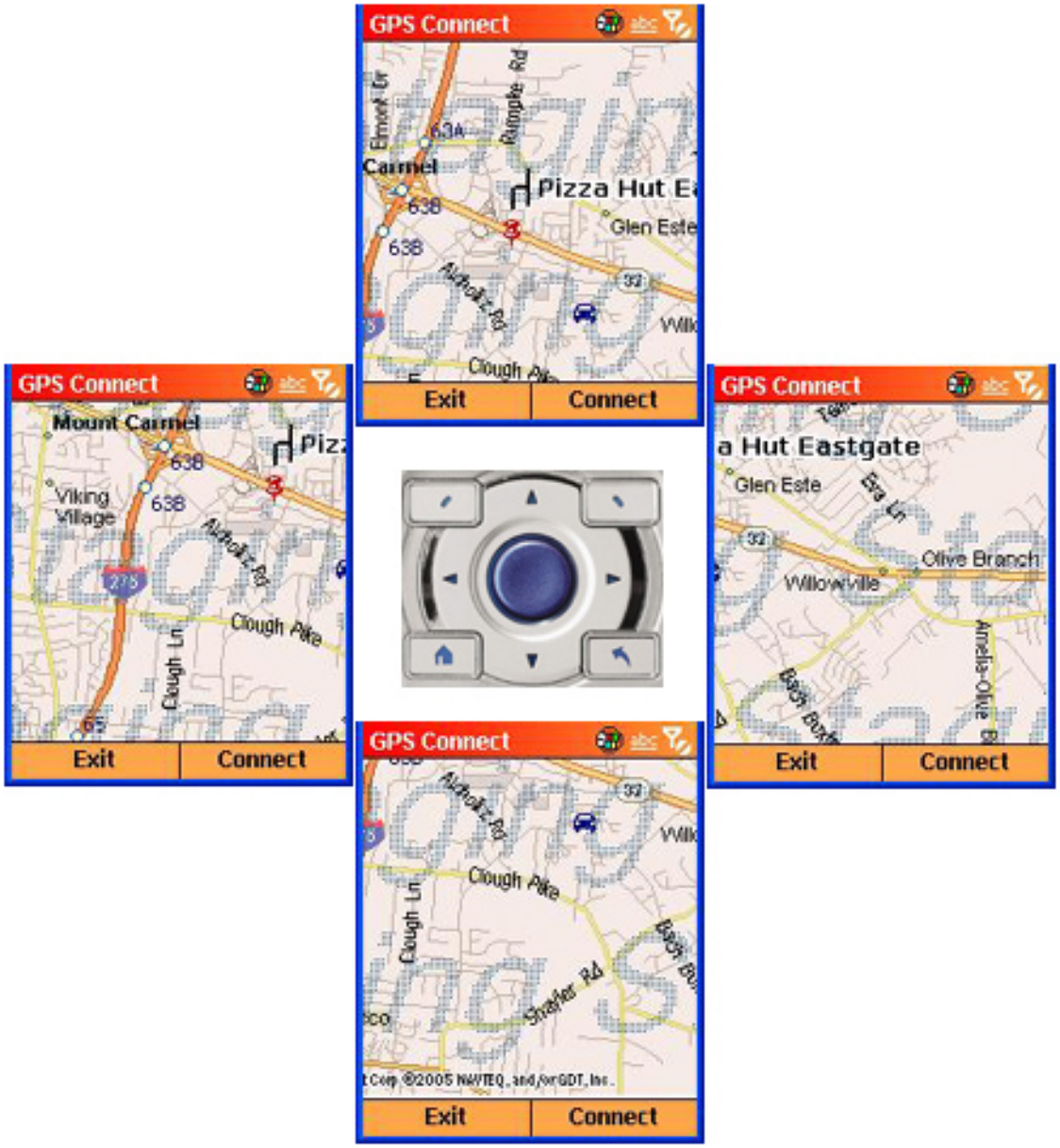

Figure 8: Directional Pad response for map image panning

- Figure 9 shows the zoom functionality of the Map View. The application responds to the user pressing the action button of the device located in the center of the directional pad. The button press event of the action button requests a new map of the viewed area but creates a zoom-in effect by reducing the coverage area of the map by half (1 mile x 1mile map becomes .5 mile x .5 mile) until the covered area shows .2 miles x .2 miles.

Figure 9: Action Button response for map image zoom effect

6. Testing

Test procedures were implemented during the progress of the project. The design is such that the individual modules are dependant upon previous phases of the project. To implement the first phase of this project, a Bluetooth data connection between the Pharos GPS receiver and the MPx-220 Smartphone device was necessary. To test how this connection would be established, I began by obtaining GPS data streams to my PC using a Kensington USB Bluetooth Dongle. Using HyperTerminal, a successful connection was made by setting up communication with the USB Bluetooth serial port. With this connection established, I was able to observe the actual data sentences

transmitted by the GPS receiver and the rate at which the data was updated (see Appendix A, figure 1-A.).

The next step was to capture data on the mobile device. My Smartphone device is currently running the .NET Compact Framework v1.1, which does not provide managed code to manipulate Bluetooth data connections already established by the device's operating system. Within my application, I implemented a third-party open source library. OpenNETCF.org's Bluetooth v1.3 simplifies the code needed to establish the Bluetooth device discovery and connection (3). I was able to instantiate a new Bluetooth connection and receive the raw GPS data sentences to a textbox control, successfully testing this function of my application on the device.

To make usable sense of the data received from the GPS, I referred to the NMEA-0183 GPS standard. For this project, only the $GPRMC data sentence is needed, as it contains all the pertinent information for Latitude, Longitude, Bearing, Speed, and Satellite Fix status. Appendix A shows how the data sentence is interpreted. In my application, the data is parsed into the appropriate data formats, then the information is made available by raising events when information changes. At first, each data item had its own corresponding event. But when testing the GUI, I found this was too many events to be processed and still maintain other functionality in my application. To improve performance, I reduced the number of events occurring to three per data sentence. Also I have changed the processes from running on threads to running on a timer, in 1.5 second intervals. This early version of .NET compact framework does not allow the developer to manually abort threads once created, which occupies valuable resources necessary for running applications on the Smartphone form factor.

As I studied more about the MapPoint Web Service, I decided to stray from the baseline schedule. To allow for a more logical workflow in this project, I began development on the map rendering phase and then integrated the contacts into the map rendering specifications afterwards. The Web Service allows the developer to request map data based on an extended list of map specifications. The capabilities of this Web Service are well beyond the scope of this project. The first step to implement this map functionality into my application was to call a method from the MapPoint Web Service using the credentials provided to me when I signed up for a MapPoint developer evaluation account (6). The Web Service authenticated successfully and further tests showed successful map rendering based on latitude and longitude data and a specified area of coverage (for example, 2 miles x 2 miles). As the project progressed, I was able to manipulate the various features made available by the MapPoint Web Service such as converting address information to map coordinates, custom location icons, and pushpins.

7. Conclusions and Recommendations

Through the development of the CFDI project, I have come to many roadblocks. Thanks to the structure of the Senior Design program and the newly added Project Management course, I have been able to break down the various aspects of completing this project and surpass obstacles gracefully. By effectively testing the various parts of my project, I was able to educate myself and create new ways to manipulate the built in .NET compact framework. I feel as though I have gained much valuable knowledge and ultimately reached my original goal to demonstrate the enormous capabilities of mobile devices and the convergence of some unique and cutting edge technologies.

As I began development and continued to progress throughout this process, one key rule in technology was made very clear to me. This is the fact that technology is constant advancing and changing. I began this project with a proposed set of development tools and equipment. Although Windows Mobile 2003 Smartphones are indeed powerful devices, these devices cannot always be upgraded to the newer Windows Mobile 5.0, which itself has tremendous advantages over the device used in the CFDI project. Also Visual Studio 2005 also has numerous tools available specifically for allowing easier development of mobile device applications. Due to time constraints I was not able to explore these newer technologies. But as a general recommendation for developing a similar application for mobile devices, I highly suggest using the latest Windows Mobile 5.0 Technology. This OS consists of the .NET compact framework v2.0 which contains all the managed code necessary for developing applications that can manipulate Bluetooth connections or Pocket Outlook Object Model, and has extended control of threading and graphics classes.

Appendix A

NMEA-0183 Standard, $GPRMC Sentence (Position and Time Data)
"Recommended minimum specific GPS/Transit data"

The following refers to the NMEA-0183 GPS standard used by the CF Directional Interface (1). It shows examples of the raw data transmitted by the GPS device and Table 1-A describes how the information is parsed into usable formatting. Figure 1-A on the next page is a screen shot of this actual data stream from the GPS device.

Example (signal not acquired):

$GPRMC,235947.000,V,0000.0000,N,00000.0000,E,,,041299,,*1D

Example (signal acquired):

$GPRMC,092204.999,A,4250.5589,S,14718.5084,E,0.00,89.68,211200,,*25

Field	Example	Comments
Sentence ID	$GPRMC	
UTC Time	092204.999	hhmmss.sss
Status	A	A = Valid, V = Invalid
Latitude	4250.5589	ddmm.mmmm
N/S Indicator	S	N = North, S = South
Longitude	14718.5084	dddmm.mmmm
E/W Indicator	E	E = East, W = West
Speed over ground	0.00	Knots
Course over ground	0.00	Degrees
UTC Date	211200	DDMMYY
Magnetic variation		Degrees
Magnetic variation		E = East, W = West
Checksum	*25	
Terminator	CR/LF	

Table 1-A: $GPRMC data usage

Appendix A

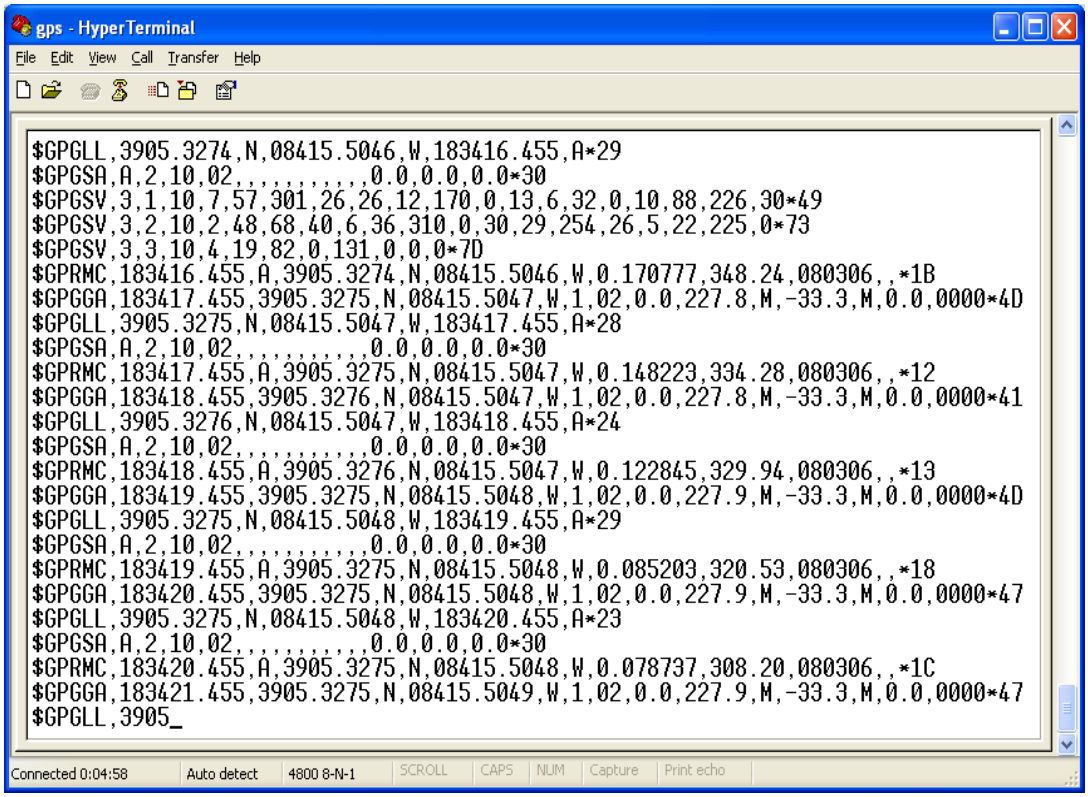

Figure 1-A: Actual (Raw) data from GPS device as captured in HyperTerminal

Resources

1. “Common GPS NMEA Sentences”. http://www.commlinx.com.au/NMEA_sentences.htm, Accessed Jan. 25, 2006.

2. Egbert, Robert I. and Joseph E. King. *The GPS Handbook.* Burford Books, 2003.

3. Foot, Peter. “Bluetooth v1.3” OpenNetCF.Net.Bluetooth.dll. http://www.opennetcf.org. Accessed January 2006.

4. Foot, Peter. “PocketOutlook” InTheHand.WindowsMobile.PocketOutlook.dll. http://www.inthehand.net. Accessed April 2006.

5. Muench, Chris. “Developing an Effective Smartphone User Interface” Microsoft, MSDN. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnppcgen/html/devSPUI .asp. October, 2002.

6. Microsoft. “Microsoft MapPoint Web Service”. http://www.microsoft.com/mappoint/products/webservice/default.mspx. 2005.

7. Person, Jon. “Writing Your Own GPS Applications”. http://www.codeguru.com/Csharp/Csharp/cs_misc/designtechniques/article.php/c8079. December 29, 2004.

Works Consulted

1. Boy Scouts of America. *Orienteering.* Boy Scouts of America, Irving, Texas 2005.

2. Deckmyn, Dominique. “Bluetooth: a quick study”. http://www.computerworld.com/news/2000/story/0,11280,45794,00.html. June 2000.

3. Ferguson, Jeff and Brian Patterson, Jason Beres, Pierre Boutquin, Meeta Gupta. *C# Bible.* Wiley Publishing, 2002.

4. Sjöström, Andreas. “Combining Web Services and Pocket PC Phone Edition to Create Value”. http://msdn.microsoft.com/mobility/understanding/articles/default.aspx?pull=/library/en-us/dnnetcomp/html/ppcphoneweb.asp. April, 2003.

5. Krell, Bruce. *Pocket PC Developer’s Guide*. Berkeley, California: McGraw-Hill/Osborne, 2002.

6. Wigley, Andy and Peter Roxburgh. *Building .NET Applications for Mobile Devices.* Microsoft Press, March 6, 2002.

7. Yao, Paul and David Durant. *.NET Compact Framework Programming with C#.* Addison Wesley Professional, May 24, 2004.