

Winter 2007

Project54 handheld system development and pilot deployment

Christopher Gaudreau
University of New Hampshire, Durham

Follow this and additional works at: <https://scholars.unh.edu/thesis>

Recommended Citation

Gaudreau, Christopher, "Project54 handheld system development and pilot deployment" (2007). *Master's Theses and Capstones*. 327.
<https://scholars.unh.edu/thesis/327>

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact nicole.hentz@unh.edu.

PROJECT54 HANDHELD SYSTEM DEVELOPMENT AND PILOT DEPLOYMENT

BY

CHRISTOPHER GAUDREAU

B.S. University of New Hampshire, 2005

THESIS

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Science

In

Electrical Engineering

December, 2007

UMI Number: 1449584

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI®

UMI Microform 1449584

Copyright 2008 by ProQuest LLC.

All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 E. Eisenhower Parkway
PO Box 1346
Ann Arbor, MI 48106-1346

This thesis has been examined and approved.



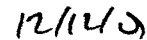
Thesis Director, Andrew L. Kun
Assistant Professor of Electrical Engineering



W. Thomas Miller, III
Professor of Electrical Engineering



William H. Lenharth
Research Associate Professor of Electrical
Engineering



Date

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my thesis advisor Dr. Andrew Kun for his support and guidance throughout my research. Dr. Kun, thank you for the opportunity to work on this exciting and innovative project, Project54.

I would also like to thank Dr. William Lenharth and Dr. Thomas Miller for serving on my thesis committee. I thank Dr. William Lenharth for his support and for the opportunity to be part of Project54. I thank Dr. Thomas Miller for his guidance throughout my years at the University of New Hampshire.

I thank Kriste Krstovski and Andras Fekete for their direct support and patience.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF ACRONYMS	x
ABSTRACT	xi
CHAPTER	PAGE
1. INTRODUCTION	1
1.1 Problem Definition	1
1.2 Goals	2
1.3 Approach	3
1.4 Thesis Organization	6
2. BACKGROUND	7
2.1 PDAs in Ubiquitous Computing	7
2.2 Wireless Security	10
2.3 Location Awareness	11
2.4 Project54 PDA Background	13
3. HANDHELD SOFTWARE PACKAGING	14
3.1 Handheld Install Packages	14
3.1.1 <i>Information Setup File</i>	17

3.1.2	<i>Configuration DLL</i>	18
3.1.3	<i>Creating the Final Installation Package</i>	20
4.	NEW HANDHELD APPLICATIONS	24
4.1	Proxy Application Overview	28
4.1.1	<i>Proxy Improvements</i>	28
4.2	WhelenMPCO Application	30
4.2.1	<i>WhelenMPCO Messaging</i>	30
4.2.2	<i>WhelenMPCO GUI and SUI</i>	33
4.3	PatrolScreen Application	34
4.3.1	<i>PatrolScreen Functionality</i>	34
4.3.2	<i>PatrolScreen Messaging</i>	36
4.3.3	<i>PatrolScreen GUI</i>	38
4.4	SymbolScanner Application	39
4.4.1	<i>SymbolScanner Messaging</i>	41
4.4.2	<i>SymbolScanner GUI</i>	43
4.5	P54H Configuration Utilities	44
4.5.1	<i>Application Selection</i>	44
4.5.2	<i>Proxy Configuration</i>	47
4.5.3	<i>Encryption Configuration</i>	48
5.	INTERNAL TESTING	51
5.1	Laboratory Setup Testing	52
5.2	Automated Testing	54
5.2.1	<i>Testbot Button Press Simulation</i>	57

5.2.2 Testbot Speech Simulation	58
5.2.3 Window Size Configuration Utility	59
5.3 Automated Testing Results	59
6. DEPLOYMENT RESULTS	69
6.1 Handheld Log File Results	70
6.2 Handheld Survey Results	79
7. CONCLUSION	83
8. SUGGESTIONS FOR FUTURE WORK	86
LIST OF REFERENCES	88
APPENDICES	91
APPENDIX A INTERNAL TESTING TEST #1	92
APPENDIX B INTERNAL TESTING TEST #2	93
APPENDIX C HANDHELD SYSTEM SURVEY	94
APPENDIX D INSTITUTIONAL REVIEW BOARD APPROVAL	95

LIST OF TABLES

Table 3.1 Descriptions of Information file fields	18
Table 3.2 Description of Setupdll.dll Functions	19
Table 4.1 AES encryption functions used in the Proxy Application	29
Table 4.2 List of messages that can be sent from the WhelenMPCO Application	32
Table 4.3 List of messages that can be received by the P54H WhelenMPCO application	32
Table 4.4 List of messages that can be sent from the PatrolScreen application	37
Table 4.5 List of messages that can be received by the PatrolScreen application	37
Table 4.6 List of Messages that can be received by the SymbolScanner application	41
Table 4.7 List of messages that can be sent from the SymbolScanner application	43
Table 5.1 Durability Test Results for number of button pushes and voice commands used	60
Table 5.2 Results for the first round of grammar bug testing	62
Table 6.1 Data from Officer One indicating a traffic stop conducted using the handheld	75
Table 6.2 Likert scale survey results for Officer One, Officer Two and Officer Three	81

LIST OF FIGURES

Figure 1.1 Block Diagram of Project54, the Handheld Prototype and Proposed Changes	5
Figure 3.1 Installation Diagram for the Prototype P54H System	14
Figure 3.2 New Proposed Installation Configuration for the P54H System	16
Figure 3.3 Example of code used to run an application after installation	19
Figure 3.4 Installation Process for the P54H system	21
Figure 3.5 Windows CE Application Manager Add/Remove Programs configuration	23
Figure 4.1 Inter-application messaging scenario between the In-Car and Handheld Systems	25
Figure 4.2 Block diagram of the P54H Prototype System with the proposed new applications and configuration utilities	26
Figure 4.3 WhelenMPCO GUI for the Handheld	31
Figure 4.4 PatrolScreen GUI for the handheld in the NHSP configuration	35
Figure 4.5 PatrolScreen GUI for the handheld with a Traffic Advisor	35
Figure 4.6 WhelenMPCO and PatrolScreen Application parameters for the NHSP configuration in the PDAWhelenMPCO.txt file	39
Figure 4.7 SymbolScanner License data GUI	40
Figure 4.8 Symbol MC50 PDA hardware button usage by the P54H system	42
Figure 4.9 Application Selection GUI	45
Figure 4.10 Application Parameters that must be specified in the allapps.txt file	46

Figure 4.11 Proxy Configuration GUI	47
Figure 4.12 Encryption Configuration GUI	49
Figure 5.1 Steps taken during the Internal Testing of the P54H system	51
Figure 5.2 Laboratory setup for testing the P54H system	53
Figure 5.3 Text file configuration for Testbot simulation	55
Figure 5.4 Testbot GUI	57
Figure 5.5 Grammar Bug Testbot test file	61
Figure 5.6 Grammar Bug Testing Results for PDA1	64
Figure 5.7 Grammar Bug Testing Results for PDA2	64
Figure 5.8 Grammar Bug Testing Results for PDA3	66
Figure 5.9 Grammar Bug Testing Results for PDA4	66
Figure 5.10 Grammar Bug Testing Results for PDA5	67
Figure 6.1 GUI usage and SUI usage per application for Officer One	71
Figure 6.2 GUI usage and SUI usage by date for Officer One	72
Figure 6.3 Voice Command recognition accuracy and percentage of errors by type for Officer One	73
Figure 6.4 Voice command recognition from April 3 – April 4 for Officer One	74
Figure 6.5 Voice command recognition from April 5 – May 18 for Officer One	74
Figure 6.6 GUI usage and SUI usage per application for Officer Two	77
Figure 6.7 GUI usage and SUI usage by date for Officer Two	78
Figure 6.8 Voice Command recognition accuracy and percentage of errors by type for Officer Two	79

LIST OF ACRONYMS

2D	Two Dimensional
EULA	End User License Agreement
GUI	Graphical User Interface
MB	Megabyte
P54H	Project54 Handheld
PDA	Personal Digital Assistant
SD	Secure Digital
SR	Speech Recognition
SUI	Speech User Interface
TTS	Text-to-Speech

ABSTRACT

PROJECT54 HANDHELD SYSTEM DEVELOPMENT AND PILOT DEPLOYMENT

by

Christopher Gaudreau
University of New Hampshire, December, 2007

Police officers have been using the Project54 system in their vehicles for years. However once they step out of the vehicle they lose access to in-car services. Our goal was to develop and test a new Project54 Handheld (P54H) software system which would grant access to in-car services outside the vehicle and to evaluate the potential usefulness of those services. For this purpose applications were developed to interface with the in-car Project54 System on a handheld device. We then comprehensively tested the P54H system and confirmed that it operates correctly. We first performed functionality tests on a laboratory mock-vehicle setup. Next, we performed stability tests on the P54H system. After deployment data was retrieved from three deployed handheld systems and quantitatively evaluated to identify the use of the P54H graphical user interface (GUI) and the P54H speech user interface (SUI). Feedback from three officers, in conjunction with retrieved data analysis gave a preliminary indication which services were most beneficial to officers.

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Despite the success of the Project54 in-vehicle system, once officers step out of the vehicle they lose access to all in-car services. Performing records queries from outside the cruiser can only be accomplished by calling the dispatcher using a portable radio. This solution is not good enough because most New Hampshire officers do not use portable radios and portable radios do not have the power output of the vehicle's radio which causes a smaller range of coverage. Officers may also like to change the light configuration on the vehicle during a traffic stop which can only be accomplished by returning to the vehicle. The lack of access to in cruiser services from outside of the cruiser is the main problem addressed in this thesis.

The second problem addressed in this thesis is the question of how officers would use services outside the vehicle. We would like to provide only those services that would be useful to officers. Although there have been discussions concerning the type of support officers would like from a PDA no research has been conducted which evaluating the use of vehicle services outside the vehicle.

1.2 Goals

The primary goal of this research was to develop and test a new Project54 handheld (P54H) system. The new P54H system had to meet certain conditions. Any application developed had to mimic its counterpart from the embedded system, both in the visual sense and in its functionality. We wanted applications to have the same buttons, text fields, status lights, names, locations, and voice commands. A push-to-talk (PTT) button should also be used to alert the system of an incoming voice command. We anticipate the capabilities of PDA devices in the future will far exceed what is now possible allowing the system to be expanded to a standalone program, however the P54H software for this research should be developed as a periphery to the in-vehicle system. The system should log data essential to the evaluation of use and reliability. In addition the P54H system should be simple to install and configure on the host handheld, taking no more than 30 minutes to successfully synchronize to a vehicle's Project54 system.

The second goal was to evaluate how an officer used the system and determine which services should/should not be provided. In essence we wanted to answer the following questions:

- Was the handheld software reliable?
- How would an officer use the touch screen?
- Was it difficult to control the handheld using voice commands?
- Was the handheld software frustrating to use?
- Was the handheld comfortable to use with one hand?

- Did the handheld make daily tasks easier to perform?
- How often was the handheld used during the course of an officer's shift?
- What was the overall level of satisfaction with the handheld system?

1.3 Approach

The method of attaining a complete Project54 handheld system can be described in five proposed steps. We proposed to elaborate upon the Project54 handheld prototype [1][2] as shown in Figure 1.1. The first proposed step was to create a method of easily packaging, distributing, and configuring the software. The second proposed step was to develop additional needed applications. The third proposed step was to internally test the software to ensure everything was working correctly. The fourth proposed step was to deploy the handheld system to a few police cruisers to collect usage data. Finally, in the fifth proposed step, data would be collected and analyzed.

The first proposed step was to make installation and configuration of the handheld system simple and quick for installers. This included creating a package that contained all of the necessary files that could be installed onto a handheld device connected to a computer through Microsoft ActiveSync [3]. We also proposed that all configuration utilities should run automatically and the P54H software should be registered with the host handheld during installation.

The second proposed step was to create new applications to offer additional functionality to the user. Proposed applications are highlighted in red in Figure 1.1. We proposed to create an application capable of controlling the

built-in scanner on the Symbol MC50 PDA. To meet our goal of consistency between the car and handheld systems we proposed to create a PatrolScreen Application as well as a new lights application, WhelenMPCO.

The third proposed step was to conduct internal testing of the completed system to ensure the system was working correctly and was robust. This required testing the handheld system connected to mock vehicle setup in our laboratory. Testing for robustness could be accomplished by physically using the system or by configuring an application to run continuous tests. To test the system for robustness by personally using the system would require many weeks. We proposed creating an application enabling us to simulate GUI button pushes and SUI voice commands for testing purposes.

The fourth proposed step was to deploy the handhelds to several operating cruisers. We proposed to collect data about handheld usage over a two week deployment period. To conduct an evaluation of the system we proposed to record all button pushes, voice commands, any system errors, inter-application messages and any issued voice commands.

The fifth and final proposed step was to conduct a comprehensive evaluation of the handheld system based upon the data recorded during usage and feedback from officers. This includes, but is not limited to, answering the question put forth in our goals.

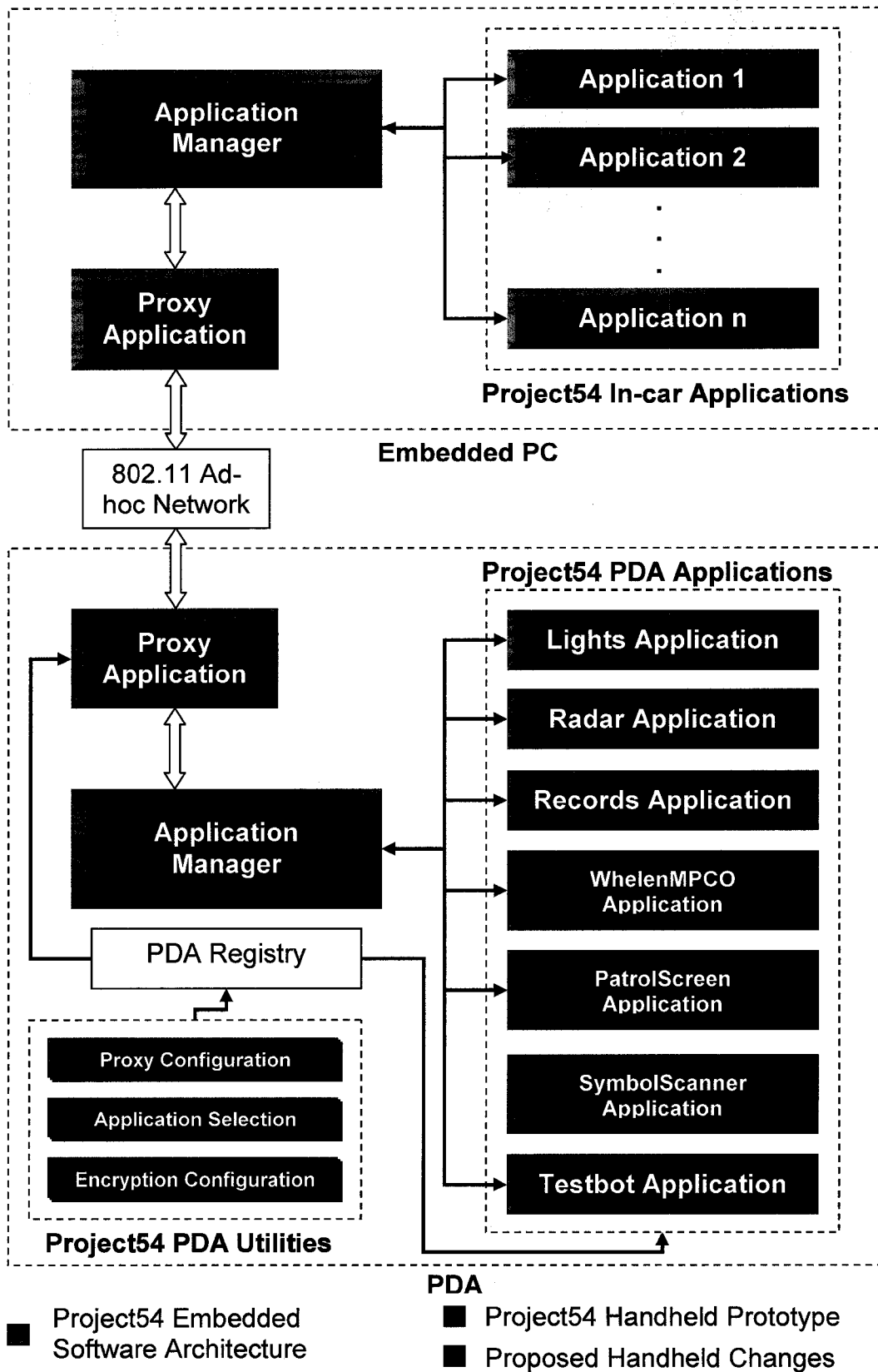


Figure 1.1 Block Diagram of Project54, the Handheld Prototype and Proposed Changes

1.4 Thesis Organization

This thesis is organized in the following order. In chapter 1 we define the problems, goals, and give a quick overview of the proposed approach. Chapter 2 contains background on other projects conducted using handheld computers in a ubiquitous setting. Related material on wireless security and location awareness are also included. Chapter 3 explains some code management techniques and how the software was packaged for installation on handheld devices. In Chapter 4 we then discuss any new Project54 handheld applications created for this research. Chapter 5 explains the internal testing conducted as well as results obtained from this testing. Results of the deployment of the software are discussed in Chapter 6. The conclusion is in Chapter 7 followed by suggestions for future work in Chapter 8.

CHAPTER 2

BACKGROUND

2.1 PDAs in Ubiquitous Computing

The use of personal digital assistants (PDAs) in everyday life has greatly increased in the past few years. Increasing processor power and memory size has attracted many researchers to explore mobile projects on PDA devices. As a result, a multitude of applications has emerged in the police enforcement and other fields. Since PDAs are small and light weight they are often used in the study and implementation of ubiquitous computing. Ubiquitous computing is the idea of integrating computers into everyday life in a pervasive manner. Projects in PDA ubiquitous computing research may deal with PDAs as a teaching aid or a communication device for example [4,5]. The personal data assistant can be a very powerful tool in implementing a variety of ubiquitous computing solutions.

Ubiquitous computing is still in its infancy. It is important for researchers to define a framework of rules that any ubiquitous project should adhere to. Aura, a project developed at Carnegie Mellon University, defines an architectural framework for ubiquitous computing applications [6]. This framework includes three key features. First, user tasks are defined explicitly and autonomously from a specific environment. The surrounding environment is not directly affected by each user which provides variability and flexibility to any ubiquitous system.

Second, user tasks are represented as combinations of independent services which allow the environment to assess if a task can or can not be completed. Lastly, the environments are equipped to self-monitor and renegotiate task support in the presence of the variation of resources. They propose that any system developed to be utilized in a ubiquitous setting should follow these basic guidelines.

Also developed at Carnegie Mellon University is the Pebbles project which explores how personal handheld computers inter-operate with desktop and built-in computers seamlessly in real time [7]. Applications were developed to enable a PDA to become an extension of another computer. Examples include sending keyboard keys, scrolling, sending menu and toolbar commands, and controlling power-point presentations to name a few. Myers et al. have designed Pebbles to help integrate the growing infrastructure that includes PDAs and Palm systems. Project54 follows this trend, attempting to extend the usability and functionality found in a police cruiser in a mobile fashion, onto a PDA.

Myers et al. have also investigated the use of multiple PDAs connected to a PC [8]. This includes applications which allow multiple users to draw simultaneously on a single PC. Applications also allow for users to modify PowerPoint slides, each with their own pen. As PDA use becomes commonplace there will be times at which multiple PDAs will be present in the same location, such as the scene of an accident. In such a case the use of multiple PDAs to share vital information among officers, both on the scene and arriving, could facilitate coordination and efficient use of man-power.

The University of California, San Diego, developed a project that uses PDAs in a community-oriented ubiquitous computing setting. There are location-aware applications that are supported such as digital graffiti and dynamic maps of the UCSD campus, as discussed by Griswold et al. [4]. Classroom activities such as asking questions and student feedback were explored as well.

The ActiveCampus project attempted to answer some fundamental questions about the use of mobile applications in a campus setting [4]. Applications such as ActiveClass show the use of mobile applications in a classroom setting as students can post anonymous questions, vote for which question they would like answered, and provide feedback to the professor about class speed and basic comprehension. A PDA can be quite awkward to use in a classroom setting though. Different strategies for use of the physical device in conjunction with the pen and paper of the classroom were also discussed.

Another setting where PDAs could be of benefit would be that of a busy hospital. Aziz et al. compared the usefulness of replacing an existing pager communication system with a system using multiple PDAs. The aim of the pilot study was to show that a PDA with a built-in mobile telephone would be more efficient in facilitating communication between healthcare providers than a hospital pager [5]. Efficient communication in such a setting is important because it can directly impact human life.

It is important to use handheld technology to its fullest potential. However although advancements have been made in technology to increase the storage space and processor speed of most personal data assistants to nearly desktop

computer specifications, these capabilities are grossly underutilized [9]. Eagle et al. at MIT use the wearable computing infrastructure to stream high quality audio over an 802.11b network or store access point ID and signal strength readings from PDAs as they access data. Today's handhelds are also capable of collecting and storing mass amounts of data, such as collecting data about ground wells near a construction site, or mapping wells with GPS software, effectively improving worker efficiency and reducing paperwork [10].

2.2 Wireless Security

Whenever a system is dealing with sending sensitive data over a wireless connection the law requires a certain level of security. In the case of a computing system inside a police cruiser there will likely be records and personal data. There are many methods for ensuring security including only accepting messages from known and trusted sources and encoding all wireless messages. However, the complete protection of sensitive data is a very difficult and complex issue even with the use of encryption [11]. One proposed solution to this problem is the Ephemerizer project developed at Sun Microsystems. Methods of improving security will not affect reliability though. Reliability can only be certain as long as the wireless device is within working distance of the access point (AP) being used. For example as police officers move away from their vehicles it is possible they will move out of their area of wireless service. This would render the handheld system useless. Denial of service attacks are also a prevalent problem with wireless security.

Basic encryption can be utilized by any device using an 802.11 wireless link. The most common encryption schemes are wired equivalency privacy (WEP) and wifi protected access (WPA). Any WEP key however can be cracked with readily available software within minutes. WEP is simply meant to deter casual snooping. As neither of these encryptions could provide the required security for police applications a different standard was used. Advanced encryption standard (AES) is a relatively new standard that supports 128, 192, and 256 bit length keys. This standard is currently being used for both unclassified and top secret government information and is considered to be very secure to brute force attacks with current technology.

Employing a system relying on a wireless network for data and voice communication is not a trivial task. There are projects however that have successfully used wireless networks in a police setting. The Renton, Washington Police Department, in collaboration with Cisco Systems, setup and used a city-wide 802.11b wireless network. The network needed to be as secure as their older network and still work with Novell's eDirectory services [12]. Police officers could access this network with wireless devices such as PDAs to get essential information quickly, efficiently, and securely. The network included access to local and national databases and helped to keep patrol cars on the road.

2.3 Location Awareness

The idea of location awareness is a new and rising idea. Location awareness could be achieved for example by using signal strength readings from

nearby access points. Figure 1 shows the accuracy of the different technologies that can be used in location sensing. As shown the achievable accuracy of location based upon measurements from an 802.11 access point is very low, only within 10 to 100 meters [13]. Therefore algorithms have been created to use multiple access points and pre-recorded measurements to obtain much higher accuracy. One such algorithm is the Nibble system developed at UCLA [14]. Krumm also discusses methods for converting raw signal strength data into location information [15].

One project that uses multiple access points to ascertain location information is the PlaceLab project. PlaceLab uses pre-recorded signal strength readings from multiple access points stored in a central database to calculate current location information [16]. This information is then sent to the user. Location aware technology has currently made little impact on the mobile computing world. For this technology to make a significant impact a location standard would have to be created [17]. This would allow for easy interoperability and would help generate general applications to be used in basic solutions.

In the case of outdoor mobile location awareness it is necessary to develop a relative positioning system. Relative positioning in this context means that objects determine their spatial relation not using an underlying infrastructure, their location is determined by other objects instead of fixed known points [18]. Necessary sensing and measurements such as time of flight of signals or intensity of emitted fields are taken only between enabled objects not including a

central access point or any external support. This would be the case in a police cruiser setting. There are design issues with such a system including scalability of devices and efficiency in measurements, allowing multiple objects to determine spatial relationships.

Although the location of the officer may not be particularly important, the use of signal strength readings could be very important in informing an officer that the handheld is out of the required proximity to the access point to send and receive data. Many times new systems, especially concerning new technology, are uncomfortable and hard to use causing users to discard the system completely. Having the ability to warn the user of connection issues because of poor signal strength could help alleviate some of those concerns.

2.4 Project54 PDA Background

Initial Project54 PDA research involved the development of a remote access system utilizing a PDA running the Palm Operating System, an 802.11b wireless card, and a 2D barcode scanner [19]. The Palm prototype system used an external card for scanning and an external radio connection for communication. In later research the system was converted to a PDA running the Windows CE OS [1]. This system contained applications which allowed access to in-car services such as a Radar Application [20] and a Scanner Application [21]. The Windows CE prototype supported the use of speech commands. The prototype also used a 2D barcode scanner to capture license data and an 802.11b wireless connection to communicate with the in-car system.

CHAPTER 3

HANDHELD SOFTWARE PACKAGING

Staying consistent with the first proposed step we created a means of packaging and distributing the software to a handheld. In this chapter we are going to explain the methods of packaging the software and how it is distributed to a handheld.

3.1 Handheld Installation Packages

To install the P54H prototype system onto a handheld device there were two steps involved. The method of installation is shown in Figure 3.1. The installation began by connecting the handheld device to our computer through Microsoft ActiveSync. Each individual file for the project was then copied over to the handheld device.

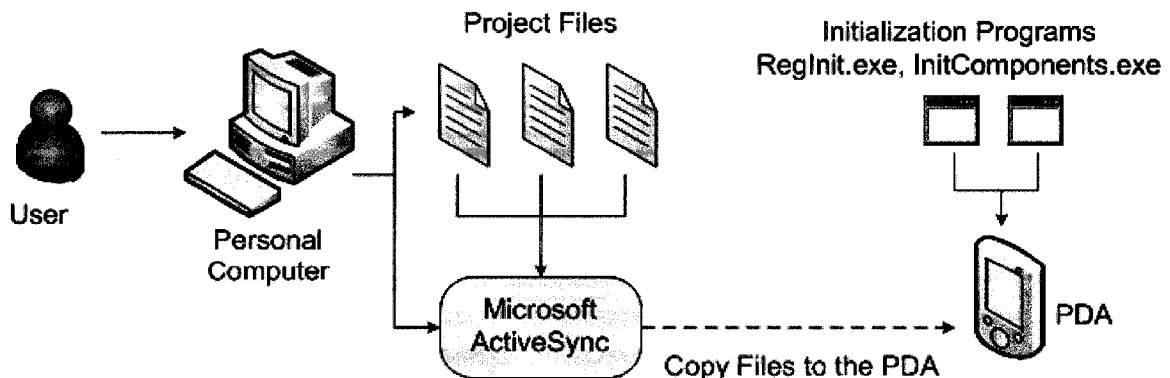


Figure 3.1 Installation Diagram for the Prototype P54H System

Once the files are copied the RegInit.exe and InitComponents.exe programs had to be manually executed which initialized the system. At this point the system was capable of running, however it would not be synchronized to an in-vehicle system.

To help facilitate quick and simple installation the process of packaging and configuration of the P54H software was automated beyond the original procedure of installation for the prototype. The new method of installation is shown in Figure 3.2. The first step was to create a cabinet file containing all of the files transferred to the handheld during installation. Cabinet files are used to organize installation files that are copied to a user's system [22]. The cabinet file is then combined with an end license user agreement (EULA) and readme file into a package using EZSetup [23] that can be executed from a personal computer. The next step was to connect a handheld through ActiveSync to our computer. Executing the package created by EZSetup installs the P54H software on the handheld through ActiveSync.

Contained in the Windows CE software development kit (SDK) is a tool capable of creating cabinet files. This tool is called CabWiz.exe. To create a cabinet file using CabWiz we must provide the tool an information (.inf File in Figure 3.2) file. An information file is a basic text file which contains the necessary information to set up an installation package. This information includes the path where the installation files are found on the personal computer, what the file names are and where they are to be placed on the handheld device. The fields contained in an information file are explained in section 3.1.1.

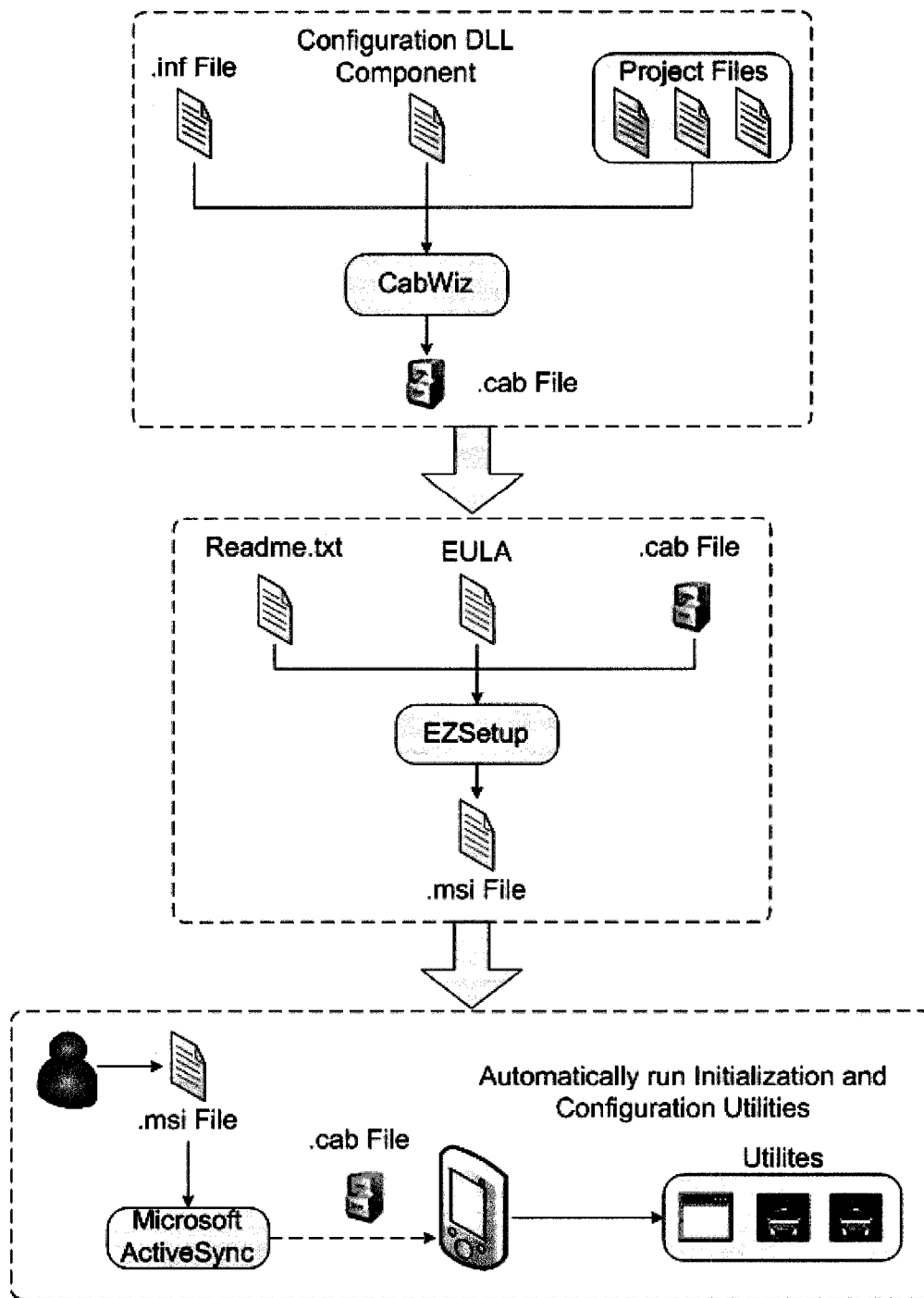


Figure 3.2 New Proposed Installation Configuration for the P54H System

Once all of the files of an installation are copied to the handheld the P54H system must register its various dynamic link library (DLL) files for them to work properly inside the Windows operating system. This is accomplished by

executing the InitComponents.exe file found in the installation root directory. There are also various configuration applications which help to customize the handheld code. For a handheld to function correctly with an in-vehicle system most of these applications will be needed. The proposed solution was to include the configuration and customization of the P54H system in the installation process by running these scripts and configuration utilities once all files were copied to the handheld. This was accomplished by including a configuration DLL in the cabinet file which is discussed in section 3.1.2.

3.1.1 Information Setup file

The information file is essentially a configuration file used by the operating system. There are many fields contained within the file that cover all the aspects of creating an installation package and where files should be placed when installed. Table 3.1 shows the various fields and their descriptions as they are used for the P54H system. For more information refer to [24]. Handheld devices running Windows CE do not have a conventional C drive as most personal computer do. Therefore when specifying an installation directory we simple use “\Project54PDA”. Once the information file is complete we can then build a cabinet file. If any new files are added to the system the information file must be updated to reflect those changes or they will not be included when a new package is created.

Parameter	Description
[Version]	Specifies operating system
[CEStrings]	Specifies application name and install directory
[SourceDisksNames]	Specifies directories where files are stored on your personal computer a unique ID
[SourceDisksFiles]	List of all the files included in the package identified by their ID in [SourceDisksNames]
[Files.Common]	List of all files being placed directly into the install directory
[Files.x.x]	List of files being placed in a directory other than the install directory. Example: [Files.Grammars.Radar]
[DefaultInstall]	Specifies any setup files, all of the file headings, and a shortcut heading if there is one
[DestinationDirs]	List of install directories for various files
[Shortcuts.All]	Specifies the name of the shortcut and which executable to run

Table 3.1 Descriptions of Information file fields

3.1.2 Configuration DLL

A configuration DLL named setupdll.dll was created to allow custom actions to be taken during installation [25]. We proposed to automate the configuration of the P54H system which included running utilities during installation after the application was installed. The configuration DLL allowed us to customize the installation process. There are four specific functions that are contained in the setup file, which are listed in Table 3.2. The primary function used during installation of the P54H system is the Install_Exit function which is called after installation of the application is complete.

Function	Description
Install_Init	Runs code just before installation
Install_Exit	Runs code just after installation
Uninstall_Init	Runs code just before uninstallation
Uninstall_Exit	Runs code just after uninstallation

Table 3.2 Description of Setupdll.dll Functions

An example of the code used to run our application selection program can be seen in Figure 3.3. Once all files are copied to the handheld device the Install_Exit function, highlighted in red, is executed.

```

////////////////////////////////////
//PURPOSE : HANDLES TASKS DONE AT END OF INSTALLATION
////////////////////////////////////
codeINSTALL_EXIT Install_Exit(
    HWND hwndparent, LPCTSTR pszinstalldir,
    WORD cfaileddirs, WORD cfailedfiles, WORD cfailedregkeys,
    WORD cfailedregvals,
    WORD cfailedshortcuts)
{
    SHELLEXECUTEINFO appwiz = {0};
    appwiz.cbSize = sizeof(SHELLEXECUTEINFO);
    appwiz.fMask = SEE_MASK_NOCLOSEPROCESS;
    appwiz.hwnd = NULL;
    appwiz.lpVerb = NULL;
    appwiz.lpFile = L"\\Project54PDA\\Initialization\\PDA Application Selection.exe";
    appwiz.lpParameters = L"";
    appwiz.lpDirectory = NULL;
    appwiz.nShow = SW_SHOW;
    appwiz.hInstApp = NULL;
    ShellExecuteEx(&appwiz);
    WaitForSingleObject(appwiz.hProcess, INFINITE);
    //return value
    return codeINSTALL_EXIT_DONE;
}

```

Figure 3.3 Example of code used to run an application after installation

In this case a shell execute command is given to run an application named “PDA Application Selection.exe” contained in the “Initialization” folder in the P54H root directory on the handheld. The WaitForSingleObject(appwiz.hProcess, INFINITE) command specifies to wait until the application selection program is exited before running the next line of code in the Install_Exit function. This way multiple configuration utilities can be executed sequentially. If a new utility is

added to configure the handheld system and we would like it to automatically run during installation, code similar to what is shown in Figure 3.3 must be added to the Install_Exit function contained in the setupdll.dll component.

The Uninstall_Exit function also proved useful for deleting any files that were created after installation. When an application is uninstalled from a handheld device all of the files that were copied to the device during installation are deleted. However any files that may have been created in the application directory after installation will not be removed by the un-installation process. Using the Uninstall_Exit function we can search for and delete any new files and directories within the installation directory. These files are deleted to completely remove the P54H system from a handheld.

3.1.3 Creating the Final Installation Package

To create an MSI file, or Windows installer, we combined a cabinet file, a readme.txt file about the project and an EULA using EZSetup. This was accomplished by providing the language of use, English, the three files mentioned above and the name of the new MSI file to EZSetup through the computer's command line.

To ensure the installation package would perform as anticipated it was tested. This was accomplished by running through the installation and verifying that all files were installed in their appropriate directories and each configuration utility ran when it was executed. One of our goals was to simplify the installation process for both engineers and installers. Figure 3.4 maps the entire process

from building a Windows installer package to complete installation of the software as performed by an engineer. It is important to note that an installer would be provided with an MSI file to begin installation whereas an engineer would create their own.

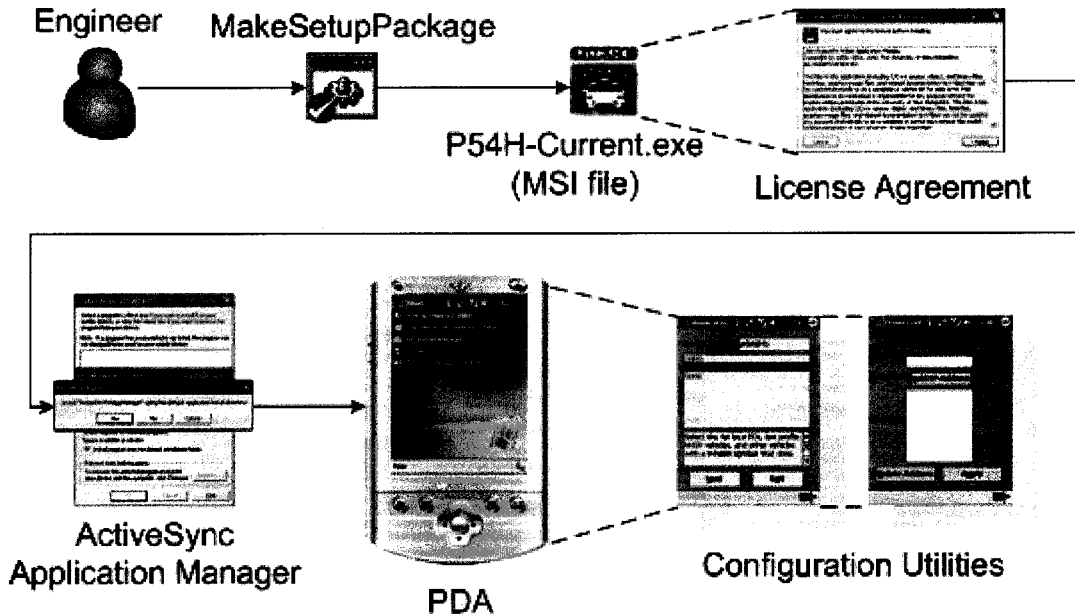


Figure 3.4 Installation Process for the P54H system

From the perspective of an engineer working on code changes the first step toward installing the P54H software onto a handheld is to build an MSI file. To do this an engineer must download the handheld code from the Project54 server onto their local computer. Contained in the directory into which the software was downloaded is a Windows NT Command Script called MakeSetupPackage. When MakeSetupPackage is executed all of the applications contained in the directory are compiled and combined into a cabinet file. The cabinet file is then sent to EZSetup and a new MSI file is created.

To begin installation the engineer would execute the MSI file by “double-clicking” it. Next a license agreement will appear on the screen. Once agreed to

the handheld software will be installed on the local computer for later use. If a handheld is not connected to the local computer through ActiveSync installation will stop until a connection is made. Once a handheld is connected the Windows CE Application Manager will ask if we would like to install to the default directory. Although it is not necessary to install to the default directory it is recommended. Next the project files will be copied to the handheld. Lastly, the initialization and configuration utilities are executed sequentially.

The reasoning behind installing the handheld software on the local machine is one of efficiency. Installation of the software onto handhelds after the first one is accomplished by connecting a new handheld to the local computer through ActiveSync and using the Windows CE Application Manager to install the software using the copy stored on the local computer. When the Application Manager is accessed in ActiveSync the engineer is given a list of Windows CE applications stored on the local computer that they can install. To begin the new installation, an engineer would select Project54 P54AppManager as shown in Figure 3.5 and click OK. He/She would then complete the installation by following the steps outlined above.

This installation process has been completed a minimum of 30 times in the laboratory. In all cases the installation was successful. The process of installation from the execution of the MSI file to installation completion took less than 5 minutes in all cases.

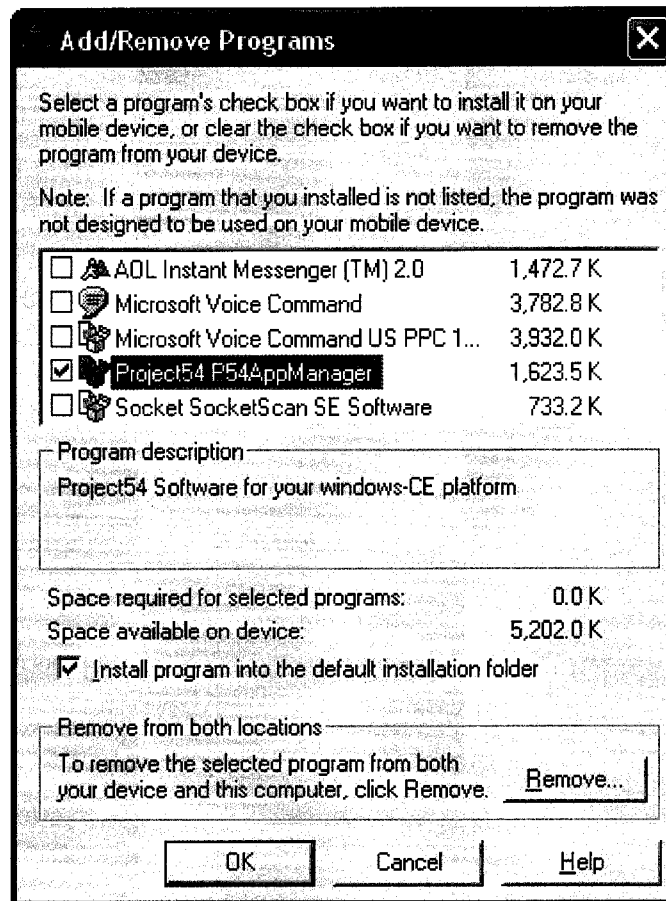


Figure 3.5 Windows CE Application Manager Add/Remove Programs configuration

CHAPTER 4

NEW HANDHELD APPLICATIONS

The second proposed step of this research was to create new applications. The motivation to create new applications was two-fold. The first was to meet our goal of consistent functionality between the in-vehicle system and handheld system. Second, there were certain applications inherited from the prototype system that offered little synchronization to an in-vehicle system. This is best explained by looking at how the two systems interact.

Communication between the handheld and in-vehicle systems was accomplished through Proxy Applications running on each. The scope of this research does not cover the Proxy Application, however it is vital to the success of the handheld system and therefore is discussed in section 4.1. Any inter-application messages [26] that need to be sent to an application on the opposing system are sent from the Proxy Application. Inter-application messages in this case are used to update the other system about any status changes. Figure 4.1 highlights an example of the structure used to update the status of the vehicle on both systems. In this scenario the user activates the front radar antenna on the police vehicle from the handheld system.

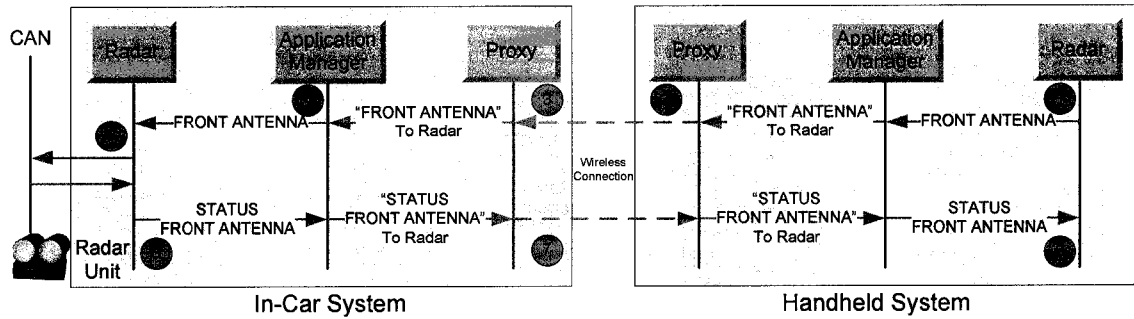


Figure 4.1 Inter-application messaging scenario between the In-Car and Handheld Systems

The user has two options to invoke a FRONT ANTENNA command (event 1 in Figure 4.1). The user can either press the button labeled Front Antenna on the Radar graphical user interface (GUI) or can speak the Front Antenna voice command while the push-to-talk (PTT) button is depressed. Once the command is invoked the handheld Radar Application sends out the FRONT ANTENNA command which is first handled by the Application Manager on the handheld. The command is passed to the Proxy Application where it is sent out over the wireless link on the handheld (event 2 in Figure 4.1). The Proxy Application on the in-car system receives the message (event 3 in Figure 4.1) and, after verifying the message source, sends it to the Application Manager on the in-car system. The Application Manager forwards the message to the Radar Application (event 4 in Figure 4.1) where it is processed. The Radar Application processes the FRONT ANTENNA command and sends the appropriate control area network (CAN) message to the radar hardware unit (event 5 in Figure 4.1).

Once the radar unit processes the command it sends a message to the in-car Radar Application confirming that the front antenna has been activated. The Radar Application then sends a status update to the handheld Radar Application (event 6 in Figure 4.1). This message travels back to the handheld through the

Proxy (event 7 in Figure 4.1) where it is received and forwarded to the handheld Radar Application (event 8 in Figure 4.1). The handheld Radar Application then updates its status concerning the front antenna to reflect that it has been turned on. The handheld Radar Application then updates its GUI to show that change.

The Lights Application from the prototype system did not have a feedback structure to handle status updates from the in-car system. Therefore, taking into account the inter-application messaging structure new applications were created. Figure 4.2 diagrams the P54H prototype system and the proposed new applications and configuration utilities added to create the new system. It should be noted that other applications and utilities were created for testing purposes which are explained in chapter 5.

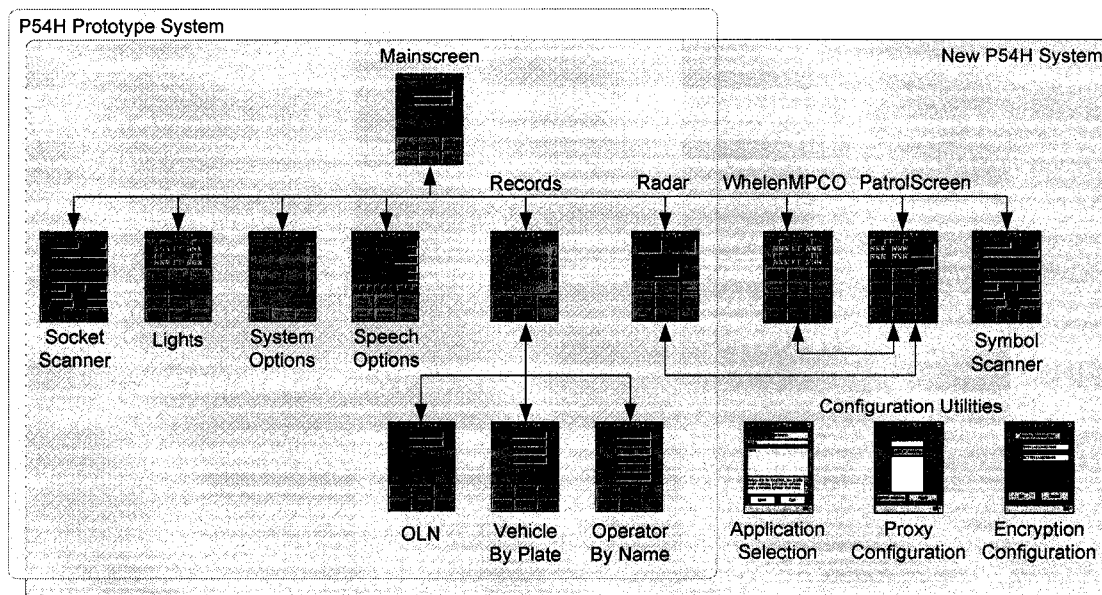


Figure 4.2 Block diagram of the P54H Prototype System with the proposed new applications and configuration utilities

A good base of applications existed from the prototype that could perform basic interaction with the in-vehicle system such as Radar and Records

Applications. These particular applications could remain fairly generic and interact successfully with many different vehicle setups. This however was not possible with the basic Lights Application. Therefore a new lights application, WhelenMPCO, was created to synchronize to the lights configuration found in New Hampshire State Police vehicles. The WhelenMPCO Application is discussed in section 4.2.

The application most commonly used on the in-vehicle system is the patrol screen. This application combines the essential functions from various other applications. For example there are lights, radar and radio controls on the PatrolScreen Application on the in-vehicle system. There are also links to all of the parent applications. In the attempt to keep the P54H system consistent with the in-vehicle system a patrol screen application was designed to interact with the New Hampshire State Police lights and radar configurations. There are certain limitations that were taken into account when creating the PatrolScreen Application which are discussed in section 4.3.

The scanner application created with the prototype system was used to control a scanner that connected to the CompactFlash (CF) socket of a PDA. The problem with using a CF scan card was its non-rugged nature. Similar to USB devices, the card could be broken. The possibility of such a malfunction is not acceptable for law enforcement officers and therefore a new PDA was selected for this research. The PDA selected needed to run the WindowsCE operating system, should be ruggedized and finally would ideally have a built-in scanner. Based on these criteria we selected the Symbol MC50 PDA for our

testing. The MC50 was a rugged PDA which was tested to withstand a three foot drop onto a hard surface. The MC50 model also came with a built-in scanner capable of scanning two dimensional (2D) barcodes. We proposed to design an application to control the scanner to facilitate easy records checking, allowing us to receive barcode data found on most drivers' licenses. The SymbolScanner application is discussed in section 4.4.

4.1 Proxy Application Overview

The Proxy Application carries out the wireless communication portion of the in-vehicle/handheld Project54 system. Any messages that need to be routed to an application found on the opposite system are sent from one Proxy Application to another. The entire message, in the format of (source, destination, message_id, message_text) is placed in a UDP packet and sent to the Proxy Application with the IP address specified during initialization. Inter-application messaging between the handheld and in-vehicle systems requires an IP based network link [19].

4.1.1 Proxy Improvements

The nature of the information passed between the handheld system and the in-vehicle system can be very sensitive. The Windows operating system provides basic protection for information sent out of a wireless link in the form of WEP or WPA encryption. The WEP and WPA encryption standards however do

not provide the level of security required by this research. Instead the Proxy Application encrypts the data messages using 128-bit AES encryption with functions found in the application interface (API) written by Dr. Brian Gladman. The encryption functions used in the Proxy Application are listed in Table 4.1. The method of creating the encryption keys are discussed in chapter 5.

Encryption Function	Description
init_aes_encryption_key(AesKeyLen, AesKey)	Initialize the encryption key with the key stored in the string AesKey
init_aes_decryption_key(AesKeyLen, AesKey)	Initialize the decryption key with the key stored in the string AesKey
aes_encryption(&buffer[sbuflen], &ebuffer[sbuflen])	Encrypt the message stored in the string buffer and copy the result into the string ebuffer
aes_decryption(&rbuffer[buflen], &plaintextbuffer[buflen])	Decrypt the message stored in the string rbuffer and copy the result into the string plaintextbuffer

Table 4.1 AES encryption functions used in the Proxy Application

The AES keys are initialized when the Proxy Application loads. When a message is received by the Proxy to be sent out over the wireless link it is first passed to the aes_encryption function to be encrypted. When a message is received from the wireless card and is from a secure source it is decrypted with the aes_decryption function before being sent to the Application Manager.

The second improvement to further provide security was an IP based solution. The proposed solution allows the Proxy Application to only receive messages from IP addresses in a list designated by the installer/officer. Any messages received by either Proxy Application with an IP address not matching an address on the list are discarded. When a message is received by the Proxy Application the IP address is extracted and compared against the list designated earlier. If a match is found the message is decrypted and sent to the Application

Manager, otherwise it is deleted. Project54 is still susceptible to denial of service attacks.

4.2 WhelenMPCO Application

The WhelenMPCO lights application was designed around the WhelenMPCO Application for the in-vehicle system. The major difference between the two applications is their function, although this is not seen by the user. The use of voice commands or graphical user interface (GUI) button pushes resulting in lights activating on the police cruiser could be accomplished from either application, however the handheld application could not control the Whelen light bar without the in-vehicle application. The in-vehicle application is the only application that will send messages directly to the light bar, activating or deactivating lights. The WhelenMPCO Application designed for the handheld simply relays messages to the in-vehicle application. These messages are interpreted by the in-vehicle application and lights are activated or deactivated.

4.2.1 WhelenMPCO Messaging

The WhelenMPCO handheld Application communicates with the in-vehicle system using the Project54 inter-application messaging system. Messages that can be sent to any other Project54 application are shown in Table 4.2. These messages can be invoked by pressing GUI buttons seen in Figure 4.3 or by using SUI voice commands. The PDA display is a touch screen monitor which allows users to press buttons with either their fingers or a probe.

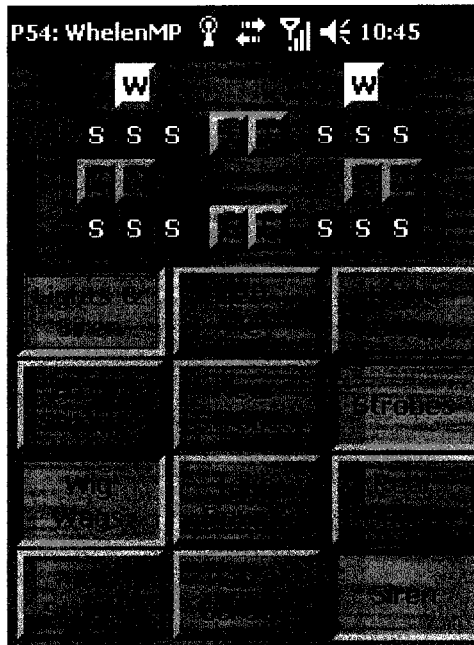


Figure 4.3 WhelenMPCO GUI for the Handheld

For example pressing the “Left Alley” button will send the LEFT ALLEY message to the in-vehicle system to activate the left alley lights. Pressing the button again invokes a LEFT ALLEY OFF message which is sent to the in-vehicle system which results in deactivation of the left alley lights.

The handheld WhelenMPCO Application will send all of the messages except SHOW WINDOW to either turn lights on or off. In the special case of a LIGHTS AND SIREN message the light bar strobes and wigwags are activated as well as the vehicle’s siren in wail mode. The display of both the handheld and in-vehicle system will reflect these functions. Figure 4.3 shows an example of the WhelenMPCO GUI with the lights and siren activated.

The SHOW WINDOW command is common to all P54H applications that contain a GUI. This message will be sent to an application when its GUI should

be shown. The WhelenMPCO Application can send this message to the Mainscreen and PatrolScreen Applications.

Messagebody	Description
LIGHTS AND SIREN (OFF)	Activate/Deactivate the lights and siren. (strobes, wigwags, and siren)
LEFT ALLEY (OFF)	Activate/Deactivate the left alley
RIGHT ALLEY (OFF)	Activate/Deactivate the right alley
FRONT STROBES (OFF)	Activate/Deactivate the front strobes
REAR STROBES (OFF)	Activate/Deactivate the rear strobes
STROBES (OFF)	Activate/Deactivate the front and rear strobes
WIGWAGS (OFF)	Activate/Deactivate the wigwags
TAKE DOWNS (OFF)	Activate/Deactivate the take downs
REAR FLOODS (OFF)	Activate/Deactivate the rear floods
WAIL (OFF)	Activate/Deactivate the siren (wail)
SHOW WINDOW	Change to a different P54H application GUI

Table 4.2 List of messages that can be sent from the WhelenMPCO application

The WhelenMPCO Application can accept any of the lights control messages from Table 4.2 preceded by STATUS. A description of these messages can be seen in Table 4.3.

messagebody	Description
STATUS lights_control_message	Updates the application that lights have been activated/deactivated. E.g. STATUS STROBES OFF
SHOW WINDOW	Show the P54H WhelenMPCO GUI

Table 4.3 List of messages that can be received by the P54H WhelenMPCO application

Status messages are used to update any application that receives feedback from another Project54 application. When a status message is received by the WhelenMPCO Application the current state of the software is updated. After the message has been processed the GUI is refreshed to show any changes in status. This assures that an officer will have up-to-date information whether using the handheld or in-vehicle system. STATUS messages are used only to update the software, they do not have an effect on the car's hardware.

4.2.2 WhelenMPCO GUI and SUI

There are two ways for a user to interact with the WhelenMPCO Application, through its GUI or SUI. Both the GUI and SUI enable the user to control the lights utilized by the in-vehicle system. For this research the P54H WhelenMPCO Application was configured with the New Hampshire State Police settings. There are twelve buttons provided to the user that carry out various tasks. The majority of buttons control lights functions and are labeled in this way. A button labeled "Right Alley" will turn on the right alley lights when depressed. This will be reflected in the GUI by painting the RA status lights white (Figure 4.1). A user can also activate the right alley lights using the "RIGHT ALLEY" voice command [1]. The voice command for each button is defined as the button label itself. The voice command to deactivate a button is the button label followed by the word OFF. Therefore to turn the right alley lights off a user would speak "RIGHT ALLEY OFF" while holding the PTT button.

4.3 PatrolScreen Application

The creation of a PatrolScreen Application was proposed to provide the same functionality found on the in-vehicle system. Similar to the WhelenMPCO lights application, the PatrolScreen was designed to work best with the New Hampshire State Police cruiser configuration.

The P54H PatrolScreen Application combines functionality from Lights and Radar Applications. No radio controls are present, a major difference from the in-vehicle application. As with all of the P54H applications there is only a very small area for the GUI, at best only one quarter of the size of the in-vehicle GUI. The area constraint results in fewer available button positions and fewer status lights and display boxes. It was therefore necessary to evaluate which functions to include in the PatrolScreen.

4.3.1 PatrolScreen Functionality

As mentioned there are no radio controls available through the P54H software. Determining the importance of these controls for the P54H PatrolScreen Application therefore was not necessary. The basic functionality that the PatrolScreen Application needed was access to the Mainscreen, Radar and Lights Applications. Access to the Records Application was not included because of the limited space. Figure 4.4 shows the PatrolScreen GUI with the NHSP configuration and Figure 4.5 shows the PatrolScreen configuration with a traffic advisor included. A traffic advisor is a light bar usually found in the rear

window of a vehicle which helps to direct traffic depending on the pattern of the blinking lights.

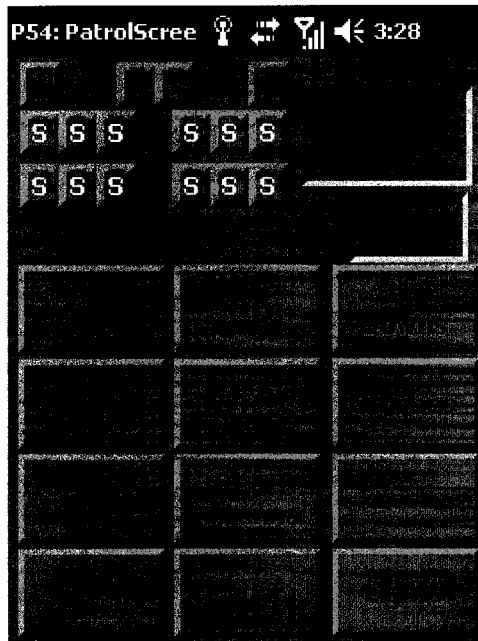


Figure 4.4 PatrolScreen GUI for the handheld in the NHSP configuration

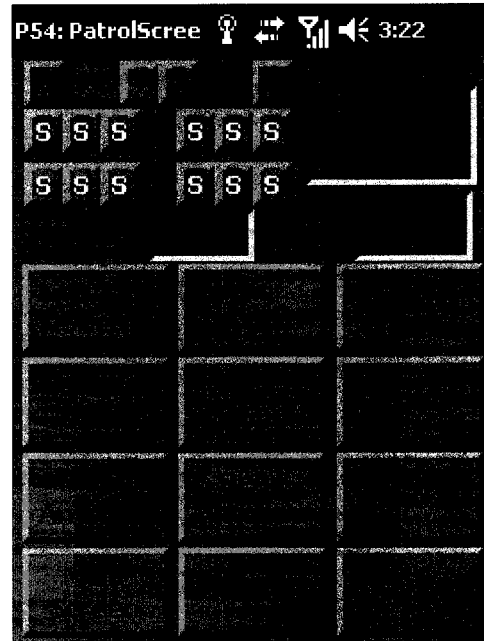


Figure 4.5 PatrolScreen GUI for the handheld with a Traffic Advisor

There are only three radar controls found on the in-vehicle PatrolScreen Application. These are the ability to switch between the front and rear antennas, turn an antenna off and lock the current speed. The buttons to access these functions are the “Front Antenna”, “Rear Antenna” and “Lock” buttons shown in Figures 4.4 and 4.5. These three functions cover the main focus of the Radar Application. Along with the three buttons for radar functions, two text areas are also displayed. One area displays the current measured speed from the radar unit and a slightly smaller text area displays the speed recorded when the lock command is given. At this time there is no need to display the police vehicle speed as the vehicle should not be moving during handheld use.

After the radar controls were placed there were six button positions remaining for light controls. The eight light controls considered were: Lights and Siren, Front Strobes, Rear Strobes, Strobes, Wig Wags, Take Downs, Rear Floods and the Siren. The first obvious choice was to include the Lights and Siren because it incorporates many different lights and the siren. We chose next to include all three of the strobe light functions. Lastly the wig wag and take down functions were added to the GUI. However these two buttons are dynamic. They can be changed, through registry settings, to control other functions. For example, if the vehicle being used has a traffic advisor installed in the light bar the take downs button can be changed to control the traffic advisor lights (Figure 4.3). Status lights for the strobes, wigwags and take downs are also included in the GUI. In the special case where a traffic advisor is included a textbox indicating the mode of the advisor is also included in the GUI.

4.3.2 PatrolScreen Messaging

The messaging system for the PatrolScreen uses the same methods as the WhelenMPCO Application. Messages that the PatrolScreen can send using the NHSP configuration are listed in Table 4.4. The possible lights control messages are listed in Table 4.2. The radar control messages that can be sent from the PatrolScreen are: FRONT ANTENNA, REAR ANTENNA, LOCK, STATIONARY CLOSING, STATIONARY AWAY, STATIONARY BIDIRECTIONAL. The stationary commands control whether the radar will

measure the speed of vehicles traveling toward the unit, away from the unit, or in either direction.

The PatrolScreen sends and receives messages to the in-vehicle system using the Proxy Application, as the Lights, Radar and Records Applications do. To receive information from two applications, the in-vehicle Lights and Radar Applications, the P54H PatrolScreen requests feedback from each of these applications once it becomes active.

Messagebody	Description
Lights_control_message	Lights command that activates/deactivates light bar lights or the siren
Radar_control_message	Radar command that controls the radar units
SHOW WINDOW	Change to a different P54H application GUI

Table 4.4 List of messages that can be sent from the PatrolScreen application

The messages that can be sent to the P54H PatrolScreen Application are status messages from the in-vehicle Lights and Radar Applications. These messages are explained in Table 4.5. The light and radar control messages are the same ones used in Table 4.4.

messagebody	Description
STATUS Lights_control_message	Status updates about lights activated/deactivated
STATUS Radar_control_message	Status updates about radar unit changes
STATUS Radar_speed	Current speed measured by the radar unit

Table 4.5 List of messages that can be received by the PatrolScreen application

When a radar speed message is received it is displayed in the text area labeled target as seen in Figures 4.4 and 4.5. At any time the current speed can

be locked, by using the LOCK command. This will copy the current radar speed into the lock text area.

4.3.3 PatrolScreen GUI

The button labels and status light display was designed to be flexible. This way the application could be configured to control more than one set of light and radar controls. The configuration for the NHSP is just one possible way for the application to be customized. To change the functionality of any of the buttons found on the GUI we would need to change registry settings.

On the in-vehicle system there is a configuration utility that allows the user to set custom label names and voice commands for both the Lights and PatrolScreen Applications. When the user exits this program a text file is created which contains all of the settings he/she just specified. To configure the handheld with the same settings we would copy this file to the handheld Project54PDA directory and run a script named UpdateRegistry. The registry parameters for the NHSP are shown in Figure 4.6.

As mentioned earlier if a custom configuration includes a traffic advisor the GUI will automatically create a text area that will display the current status of the advisor.

```

PDAWhelenMPCO - Notepad
File Edit Format View Help
{PatrolScreen}
  <ButtonLink>
    [ButtonTwo]      b2
    [ButtonThree]    b3
{WhelenMPCO}
  <ButtonLabel>
    [SwitchOne]      Front|Strobes
    [SwitchTwo]      Rear|Strobes
    [SwitchThree]    Strobes
    [ButtonTwo]      wig|wags
    [ButtonThree]    Take|Downs
    [ButtonFour]     Rear|Floods
    [ButtonFive]     Left|Alley
    [ButtonSix]      Right|Alley
    [ComboButton]    Lights &|siren
  <ComboButton>
    [Combo]          s3,b2,wa,
  <Parameters>
    [TrafficAdvisor] false
  <Vocabulary>
    [SwitchOneOn]    FRONT STROBES
    [SwitchOneoff]  FRONT STROBES OFF
    [SwitchTwoOn]    REAR STROBES
    [SwitchTwooff]  REAR STROBES OFF
    [SwitchThreeOn] STROBES
    [SwitchThreeoff] STROBES OFF
    [ButtonTwoOn]    WIG WAGS
    [ButtonTwooff]  WIG WAGS OFF
    [ButtonThreeOn]  TAKE DOWNS
    [ButtonThreeoff] TAKE DOWNS OFF
    [ButtonFourOn]  REAR FLOODS
    [ButtonFouroff] REAR FLOODS OFF
    [ButtonFiveOn]  LEFT ALLEY
    [ButtonFiveoff] LEFT ALLEY OFF
    [ButtonSixOn]   RIGHT ALLEY
    [ButtonSixoff]  RIGHT ALLEY OFF
    [ComboButtonOn] LIGHTS AND SIREN
    [ComboButtonoff] LIGHTS AND SIREN OFF
  <Animation>
    [SwitchOne]      fs,
    [SwitchTwo]      rs,
    [SwitchThree]    ss,
    [ButtonTwo]      ww,
    [ButtonThree]    td,
    [ButtonFour]     rf,
    [ButtonFive]     la,
    [ButtonSix]      ra,
  <Lightsoff>
    [SwitchOne]      true
    [SwitchTwo]      true
    [SwitchThree]    true
    [ButtonTwo]      true
    [ButtonThree]    true
    [ButtonFour]     true
    [ButtonFive]     true
    [ButtonSix]      true
    [ComboButton]    true

```

Figure 4.6 WhelenMPCO and PatrolScreen Application parameters for the NHSP configuration in the PDAWhelenMPCO.txt file

4.4 SymbolScanner Application

The handheld device chosen for this research was the Symbol MC50 PDA. This device was chosen for its rugged design which included a built-in 2D

image scanner. We proposed to develop an application capable of controlling this scanner.

The SymbolScanner application is capable of activating the image scanner on the handheld device and evaluating the data received from the scanner. Different types of license data include the operator's full name, date of birth, license number and endorsements to name a few.

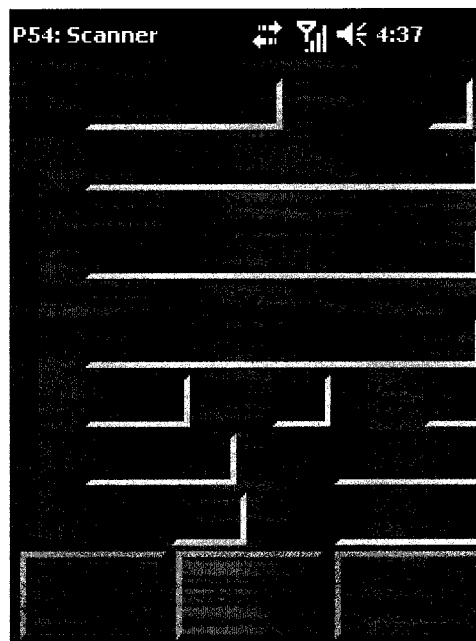


Figure 4.7 SymbolScanner License data GUI

The GUI is capable of displaying the current status of the scanner as well as the information obtained from the successful scan of a license. This data is parsed and displayed in a manner easy to read. Figure 4.7 shows the fields into which data from a successful scan would be displayed. However it was not possible to run a records query from the scanner application itself therefore making the GUI useful for testing only purposes.

The ability to use the scanner in the Records Application was included in the functionality of the prototype P54H system. Using the scanner in the

Records Application allowed the user to perform records queries while also using the scanner. Using the same idea, the scanner on the Symbol PDA can be used by the Records Application using inter-application messaging.

4.4.1 SymbolScanner Messaging

There are two types of messaging established within the SymbolScanner application. P54H inter-application messaging is used to control the scanner from applications other than the SymbolScanner. Standard Windows messaging is used to communicate between the physical scanner and the SymbolScanner application. A list of P54H inter-application messages that can be sent to the SymbolScanner application and their description can be seen in Table 4.6.

messagebody	Description
TRIGGER	If the scanner is enabled this will tell the scanner to begin scanning
ENABLESCAN	Enables the scanner
DISABLESCAN	Disables the scanner

Table 4.6 List of Messages that can be received by the SymbolScanner application

The ENABLESCAN and DISABLESCAN messages were implemented as a method of controlling when the scanner could be used. Most applications do not have the capability of triggering the scanner, however the Symbol MC50 PDA contains two built-in hardware trigger buttons, shown in Figure 4.8.. These buttons trigger the scanner if it is properly initialized and enabled. This could lead to officers triggering the scanner when it was not needed.

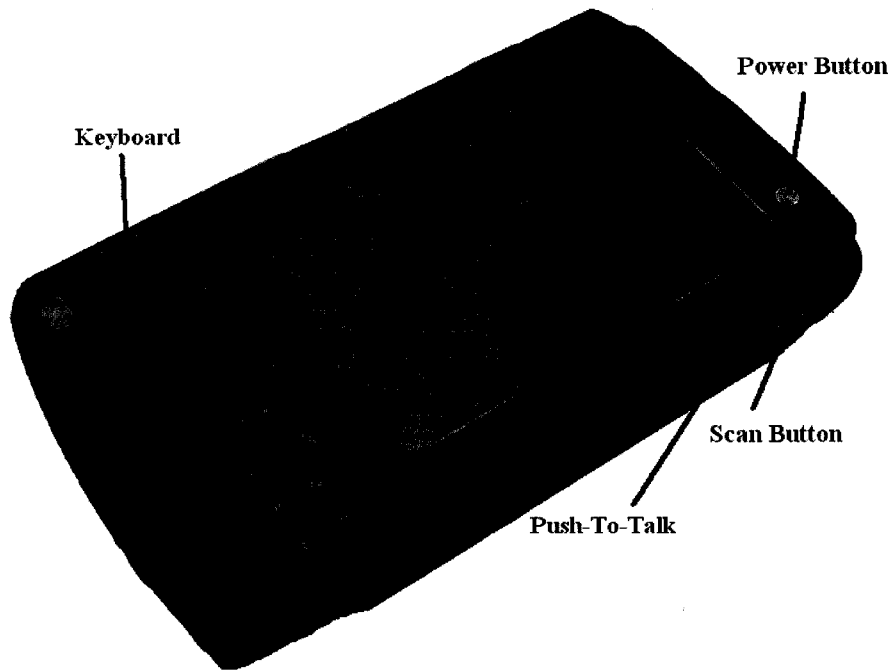


Figure 4.8 Symbol MC50 PDA hardware button usage by the P54H system

The TRIGGER command is interpreted by the SymbolScanner Application and the UM_SCAN message is posted to activate the scanner.

The SymbolScanner Application sends messages to other P54H applications based upon the handle of the current window. For example if a user is in the Operator License Number screen in the Records Application messages from the SymbolScanner Application will be sent to the Records Application. The messages contain an identification field and data for that field. The identification field, message and an explanation of the message are shown in Table 4.7.

A child window that remains invisible at all times handles communication between the physical scanner and the P54H interface. This window initializes the scanner by preparing a buffer for data storage and opening a handle to control the scanner. The UM_SCAN, UM_STARTSCANNING, UM_STOPSCANNING, and UM_CLOSESCANNER messages are sent from the

SymbolScanner Application to the child window to modify the scanners status. The functions used to interact with the physical scanner are contained in the SCNAPI32.lib and ScanCApi.h files. ScanCApi is included in the Symbol Mobility Developer Kit v1.2 for C.

Message_id	messagebody	Description
OLNDAQ	licence_number	Specifies the licence number of the operator
OLNDAJ	licence_state	Specifies the license state of the operator
OLN	UPDATEGUI	Updates the GUI of the Operator by Licence Number window
NAMEDAA	operator_name	Specifies the operator's full name
NAMEDBB	operator_dob	Specifies the operator's date of birth
NAMEDBC	operator_gender	Specifires the operator's gender
NAMEDAJ	operator_state	Specifies the license state of the operator
NAME	UPDATEGUI	Updates the GUI of the Operator by Name and Date of Birth window

Table 4.7 List of messages that can be sent from the SymbolScanner application

4.4.2 SymbolScanner GUI

Although the SymbolScanner Application GUI is not useful in the normal use of the P54H software, it was essential for debugging purposes. In the SymbolScanner GUI the current status of the scanner was updated via a large text area. This would let the user know when the scanner was properly initialized, enabled and when data had been received. To test the parsing a second window was added which showed the parsed information from a license (Figure 4.7). The functionality to read back the results of a successful scan once

the data was parsed was also added. This feature could be useful for future work as it allows an officer to scan a license and have the information read back, allowing the officer to keep his/her eyes on the environment.

4.5 P54H Configuration Utilities

To alter the configuration of the P54H system it is necessary to edit parameters stored in the handheld's registry. To do so requires either an additional program installed on the PDA or access to a personal computer with Microsoft ActiveSync installed. We proposed to circumvent manual registry changes by creating configuration utilities that could accomplish the same task. This ultimately cut down on installation time, allowing an installer to quickly enter the required settings into an easy to use program.

There were three configuration utilities created to help ease the install process. One program aided in selecting the proper applications for the configuration the installer desires. The second program of note allows the installer to easily enter the parameters relevant to wireless communication over the Proxy. Lastly, an encryption program was created to generate pseudo-random 128-bit encryption keys. These three programs are shown in Figure 4.2 in the Configuration Utilities section.

4.5.1 Application Selection

Similar to the in-vehicle system, the P54H software can be configured to interface with different hardware configurations. To allow installers to manually

change the applications loaded on startup without needing to edit the registry by hand greatly cuts down on installation time. This also allowed easy access to any P54H applications that are used specifically for testing. An example of the Application Selection GUI is shown in Figure 4.9. Users are asked to select which lights application they would like to use. Once they highlight a selection they can exit the program or click “Next” which allows them to select the Radar Application, etc.

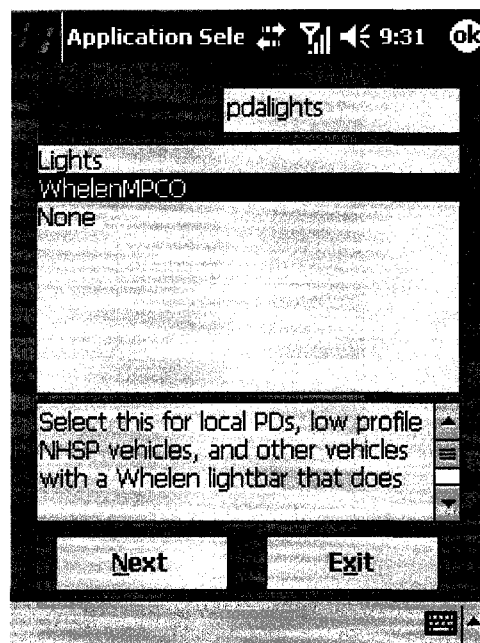


Figure 4.9 Application Selection GUI

To populate the list of available applications for each device type the application selection utility reads from a text file named allapps.txt found in the root handheld directory Project54PDA. Each application is separated into a device type so that only one “Lights” application can be selected for example. Each line in the allapps.txt file designates a different application. Contained on each line are specific parameters pertaining to the specified application. Figure

4.10 displays the allapps.txt file and highlights different parameters specified there. The “Application Name” field shown in Figure 4.10 denotes the name of the application as is specified in the application code. The “Application GUID” field lists the globally unique identifier (GUID) for the application which is used by the Application Manager to reference that application. The “Device Type” field is used to separate applications into specific lists. For example, any application that controls a scanner would be listed in the pdascanner device type. The “Mainscreen Button Position” field specifies where a button for the application will be placed on the Mainscreen GUI. Lastly, the “Mainscreen Button Label” field specifies the text label for the button placed on the Mainscreen GUI.

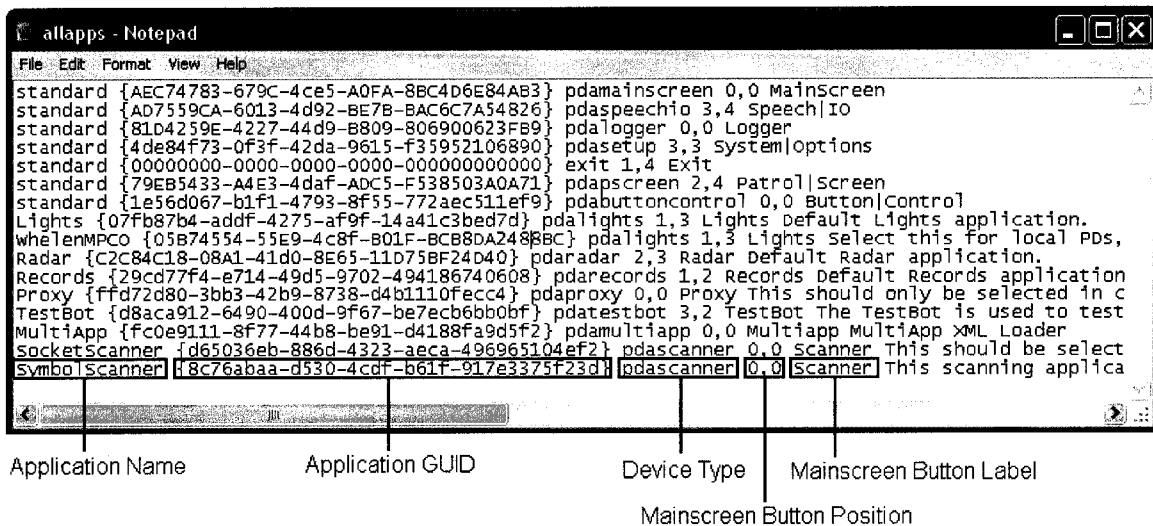


Figure 4.10 Application Parameters that must be specified in the allapps.txt file

Any text found after the “Mainscreen Button Label” (from Figure 4.10) field is considered part of a description of the application. While the application selection utility is closing the selected P54H applications and their parameters are stored in registry. The Application Manager will then load the selected applications during startup.

4.5.2 Proxy Configuration

For communication to succeed between the handheld and in-vehicle systems the Proxy must first have knowledge of the opposite system's internet protocol (IP) address. In addition an application list needs to be supplied to the Proxy Application which indicates the applications that will be sending and/or receiving messages across the Proxy. The Proxy configuration tool creates this list and stores it in registry for the installer. The IP address is the only required information by the tool.

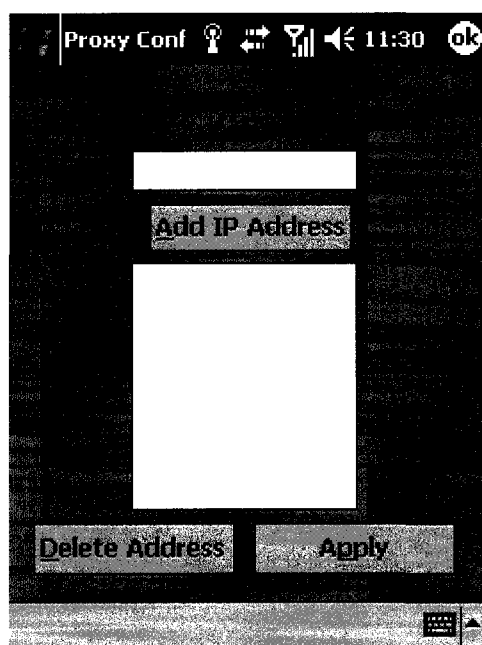


Figure 4.11 Proxy Configuration GUI

To add an IP address to the list of friendly sources a user would enter the address on the single line in Figure 4.11 then click the “Add IP Address” button. Any number of addresses can be added. To save any changes to the list of addresses click the “Apply” button. When the utility saves the IP addresses

entered, they are written into registry in both an IPList and an AppList. The AppList not only contains the names of applications communicating through the Proxy but also the address at which they can be found. In the case where more than one address is found in the IPList only the IP address at the top of the IP list shown in the configuration tool will be associated with applications in the AppList. This is because a PDA can only be configured to control one system at a time. Any other IP addresses found in the list are still considered secure sources and therefore messages will be accepted from them. A secure source could be another PDA for example.

4.5.3 Encryption Configuration

The Proxy Application encrypts all messages before they are sent out over the wireless link. The identical key is used to encrypt and decrypt each message on both the handheld and in-vehicle systems. The encryption requires a 32-hexidecimal character key. The key is stored in registry and is accessed by the Proxy Application during startup. If a key is not found in registry no encryption is used. The encryption configuration tool creates a random key and stores it in the appropriate place.

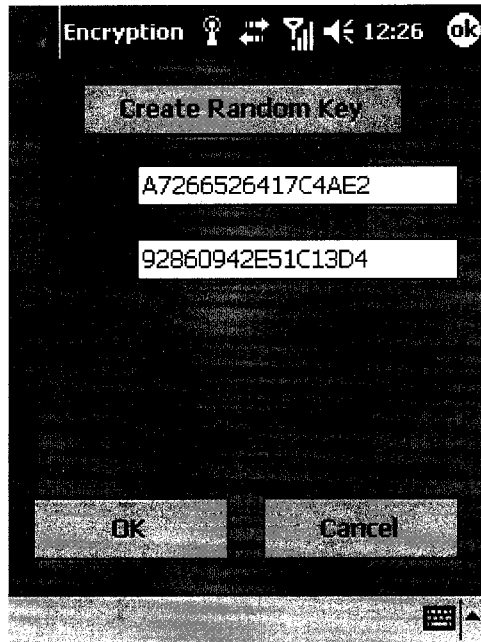


Figure 4.12 Encryption Configuration GUI

To generate a key a user would press the “Create Random Key” button on the encryption configuration GUI shown in Figure 4.12. A key can also be manually entered, the first 16 digits on the first line and the second 16 digits on the second line as labeled in the GUI. When the key is written to registry the two lines are combined into one key. The key that is currently stored in registry can also be viewed at the bottom of the GUI which facilitates quick entry of the key into the registry of the in-vehicle system. The in-car system synchronized with the handheld uses the same key and must be copied manually into the registry on the in-car system.

The key is generated using a pseudo random number generator supplied by the Windows operating system. The generator returns a number between zero and RAND_MAX, a value specified by the operating system. Once the generator returns a number it is then interpreted as one of the 16 possible

hexadecimal values. This is done by defining an integer equal to $1/16^{\text{th}}$ of the range between the zero and the value of `RAND_MAX`. Next a random number is generated and divided by the integer value defined earlier. The division returns an integer value between 0 and 16 which is then converted to hexadecimal value. Once 32-hexidecimal characters have been generated they are displayed in the configuration GUI. The key is saved to the registry when the application is exited. It is also important to note that the random number generator is seeded with a time value based upon milliseconds since the system has started so that the keys generated appear random.

CHAPTER 5

INTERNAL TESTING

The third proposed step was to conduct internal testing on the P54H system. We proposed to conduct internal testing on the P54H system using both a mock in-vehicle laboratory setup and by automating GUI button pushes and SUI commands before deployment. The process taken to test the P54H system is shown in Figure 5.1.

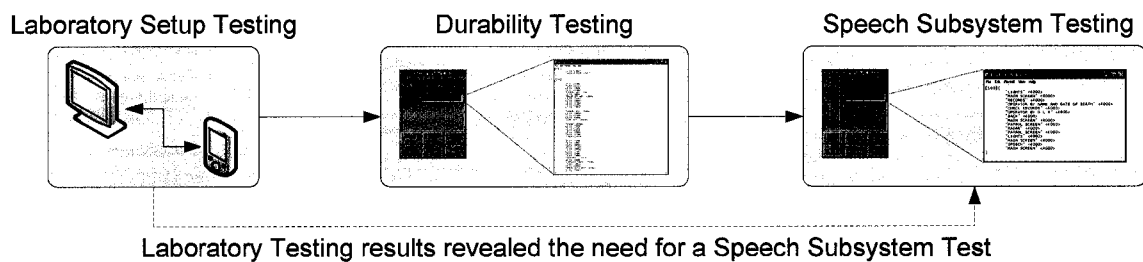


Figure 5.1 Steps taken during the Internal Testing of the P54H system

The preliminary phase of testing consisted of a mock in-vehicle setup located in the Project54 laboratory and a Symbol MC50 PDA running the P54H system. The laboratory configuration with a handheld being tested is shown in Figure 5.2. Shown in the Figure is the Project54 “Labcar” which is the two front seats of a police cruiser with a complete car setup.

When testing of the P54H system with the “Labcar” concluded a durability test of the system was performed next. To accomplish this testing a new application, Testbot, was created. The Testbot Application is discussed in

section 5.2. Durability tests consisted of repetitive use of the GUI and SUI over roughly 12 hour tests. These tests and test results are discussed in section 5.3.

Reviewing the log files obtained from the tests conducted on the laboratory setup revealed a possible problem contained in the speech subsystem. We found that grammar files were not loading into the speech recognizer when the command to do so was given. Each grammar file is created for a specific application or GUI. Therefore having an incorrect grammar file loaded would result in the voice commands for the current window being unrecognizable by the speech recognizer. To test the validity of this problem we proposed to employ the Testbot Application in simulating voice commands which resulted in grammar file changes. The results of the speech subsystem tests are discussed in section 5.3.

5.1 Laboratory Setup Testing

We began by troubleshooting the actual installation of the handheld into the vehicle. Included in the installation process was setting up an ad-hoc network between the vehicle's 802.11 wireless card and the handheld device, configuring the handheld software to match the in-vehicle software and setting up both the encryption and IP addresses for the Proxy Application.

The ad-hoc network connection is, in most cases, already configured in the vehicle for use with the wireless update software. Therefore to make a wireless connection the handheld searched for the wireless network "qwerty" and connected using a static IP address designated by the installer. The IP address

scheme used during the first deployment stage consisted of taking the IP address from the in-car access point and changing one octet to a different number. Once a connection was established between the wireless cards the Proxy was configured. This was accomplished by entering the IP address of the in-car access point into the configuration utility. It is important for users to note that the wireless access point in the vehicle is not activated by default. When Project54 starts the access point is off and must be turned on using the button labeled "Wireless" on the Mainscreen GUI.

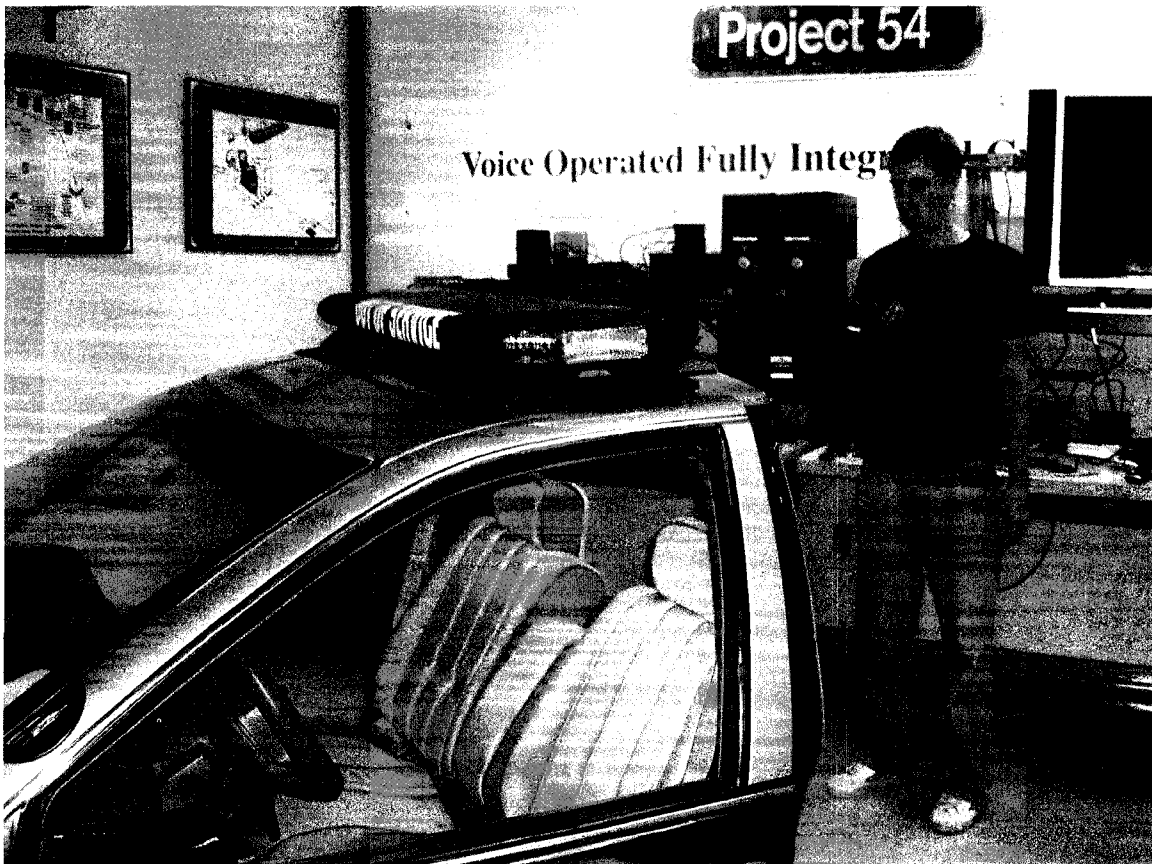


Figure 5.2 Laboratory setup for testing the P54H system

The most prevalent problem during initial testing was achieving a connection between the two devices. It seemed reasonable to deploy the system with the problem unresolved as the problem was rare.

Configuring the Proxy Application using the utilities described in Chapter 4 was successful during testing. The parameters supplied to the Proxy Application are a list of applications from the in-vehicle systems to communicate with, the IP address of the in-vehicle access point and a 32-hexidecimal encryption key. Once this was accomplished we began testing the functionality of the complete system involving the vehicle and handheld. All GUI buttons and all voice commands were tested to ensure proper functionality. All of the testing activity including recording voice commands was logged. Any bugs found were fixed until the product was ready for a small scale deployment.

The voice commands recorded during testing uncovered a possible problem with the speech system. Recorded speech commands were analyzed for four people that helped test the handheld system. Out of the 342 voice commands recorded 6 of the commands that should have invoked a grammar change did not. We examined the possible speech subsystem problem through automated testing.

5.2 Automated Testing

The automated testing proposed for this research involved simulating button presses and speech commands. This allowed for extended rigorous testing of both the GUI and SUI by an application instead of by an engineer. It

was also possible to run simultaneous tests across multiple handheld devices. We proposed to use an application to accomplish this task. The in-car system contained an application that simulated both button presses and voice commands. Therefore the Testbot Application was created based on the in-car application to handle automated testing [27].

The Testbot Application uses a user defined text file to simulate specific button pushes or voice commands. Before examining how the Testbot causes physical button pushes or simulates voice commands we will examine the text file specified by the tester to begin automated testing. The Testbot Application requires specific lines of text to simulate either button pushes or voice commands. Figure 5.3 shows an example of a text file used for simulating through the Testbot.

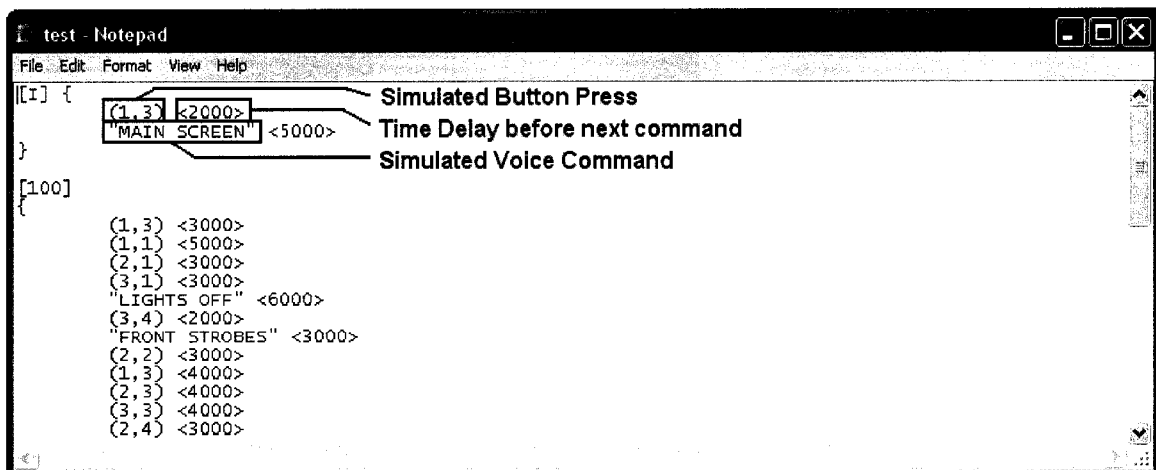


Figure 5.3 Text file configuration for Testbot simulation

To indicate in the text file that we would like to simulate a button press we would indicate the button position inside parentheses, (1,3) for example. Following this is a time value in milliseconds contained inside carrots. If no value is specified the system will use a two second delay by default. For the simulation

of voice commands we place the voice command typed in capital letters inside quotation marks. It is important to note that this command should match the output of a grammar command which may differ from the words spoken.

There are two types of tests that can be carried out by the Testbot Application, single tests and loop tests. The user will choose which test to complete in the Testbot GUI, shown in Figure 5.4. During a loop test it is possible to repeat only certain portions of our text file. Any line beginning with [!] will only be executed on the first pass through the file only. This can be specified for a group of commands by delimiting them with curly brackets as shown in Figure 5.3. Any lines beginning with a number in brackets however will be executed on subsequent passes through the loop test. The number between the brackets represents a percent value of this line or group of lines being executed through this pass of the loop test. For example any command preceded by [100] will be executed 100% of the time. A loop test will continue until it is terminated by the user. Any test running will be paused if the Testbot GUI is entered.

Now that we have created our text file for testing the system we need to copy it to our handheld device. This file must be placed in the \Project54PDA\Testfiles directory on the handheld. If the file is not placed in this directory the Testbot Application will not have access to it. Also contained in the testfiles directory is a sample Testbot file that explains the methods of creating a new test file.

We can now run our test by first entering the Testbot GUI, shown in Figure 5.3, and specifying our file in the File Name field.

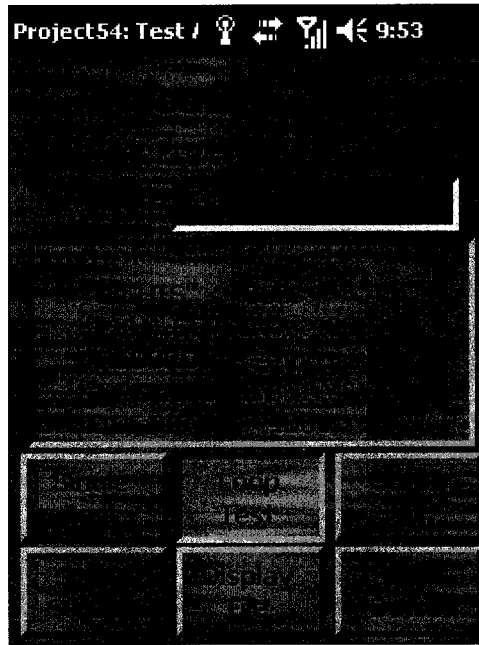


Figure 5.4 Testbot GUI

We then have the option of viewing the contents of this file by pressing the “Display File” button. A Testbot script is loaded into the Testbot Application and commands are executed when the “Single Test” or “Loop Test” buttons are pressed and the user returns to the Mainscreen. The test will begin two seconds after returning to the Mainscreen.

5.2.1 Testbot Button Press Simulation

The Testbot Application uses Windows messages to simulate button presses. The button coordinates specified in the text file are converted to absolute coordinates based on the resolution of the handheld device’s touch screen. Essentially the x and y positions of the button are scaled to fit within the boundaries of the touch screen. Therefore the resolution of the touch screen plays an integral role in the functionality of the Testbot Application. By default we

assume the resolution is 320 pixels long and 240 pixels wide. If the native resolution is not the default size the Testbot will simulate button presses at the wrong position on the touch screen. To solve this problem the resolution dimensions can be stored in registry and read in by the Testbot Application on its first pass through testing. A configuration utility was created to change the screen resolution used by the Testbot Application.

Once the coordinates are scaled to an absolute position on the touch screen a mouse down event is sent to the operating system followed by a mouse up event X milliseconds later. X is 100 by default but can be specified in the test file. Once the button press is complete the test will pause for the time specified in the test file before moving to the next simulated command.

5.2.2 Testbot Speech Simulation

Voice command simulation is used to test the portion of the speech subsystem excluding the speech recognizer. The voice commands simulated by the Testbot are equivalent to the output of the speech recognizer had the corresponding voice command been recognized. This proved useful for testing the grammar change mechanism of the speech subsystem.

When a voice command is recognized by the Testbot Application it is first appended to a wide character string containing the phrase SIMSPEECH. This completed command is then sent to the Speech Input Output Application where it is forwarded to the application that is currently shown. Whenever the speech input output application receives a message beginning with SIMSPEECH the

remaining phrase is forwarded to the current application as if the speech command had been recognized by the speech recognizer. Once the message is received by the current application it will be carried out if the command is valid for that application. In the instance where the command is not recognized by the current application nothing will happen.

5.2.3 Window Size Configuration Utility

The Testbot has been used to test the durability of the P54H system on multiple handhelds. The resolutions of the handhelds used were either 640x480 or 320x240. The resolution did not affect the look of the GUI, however the Testbot Application did not work correctly for both resolutions. The WindowSize utility was created to solve this problem.

The WindowSize utility records the resolution of the handheld into registry. As the only resolutions used in this research were as stated earlier these are the only two options given in the WindowSize utility. The two options listed can be selected with a radio button and the resolution is recorded when the utility exits. This does not change the resolution of the actual handheld.

5.3 Automated Testing Results

The first tests conducted using the Testbot Application were done to determine the durability of the handheld software. Long term effects of use through both button pushes and voice commands had not been evaluated with the prototype system. It was unlikely but possible that system performance could

deteriorate as usage life progressed both over the course of a day and over the course of weeks. These tests showed that system performance remained constant over a 15 to 20 hour period during rigorous use throughout all of the 8 tests conducted.

There were two test scripts used during the durability tests which are shown in appendixes A and B. The scripts contained various patterns of both voice commands and button pushes. The tests were conducted on the Symbol MC50 handheld device over the course of approximately two weeks. Eight durability tests were conducted and the results are shown in Table 5.1.

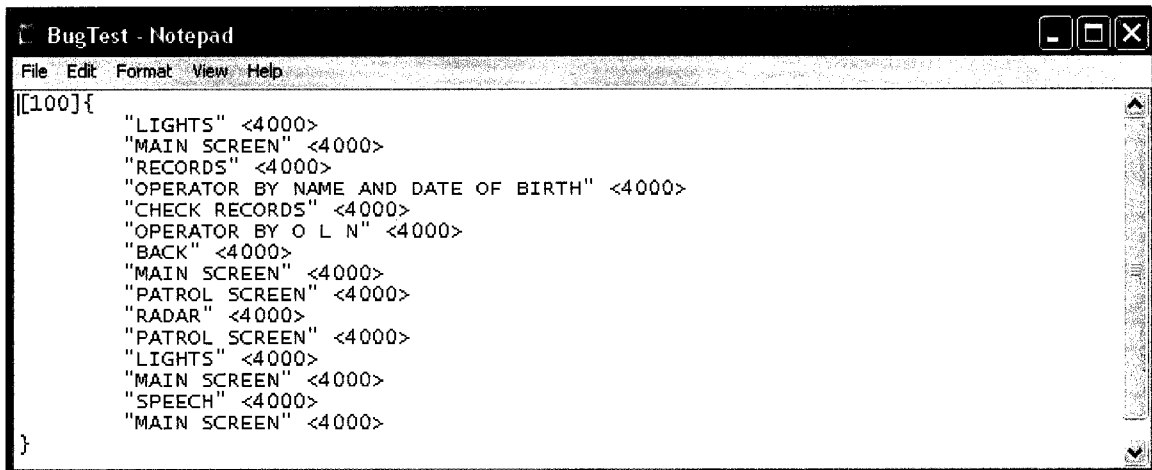
Test #	Start Date and Time	Finish Date and Time	Test Script	Buttons Pressed	Voice Commands
1	7/10/06 4:05 PM	7/11/06 9:04 AM	test1.txt	10726	4214
2	7/11/06 2:17 PM	7/12/06 8:18 AM	test1.txt	11369	4467
3	7/13/06 12:33 PM	7/14/06 8:56 AM	test1.txt	12997	5106
4	7/17/06 3:45 PM	7/18/06 8:55 AM	test1.txt	10847	4261
5	7/24/06 4:01 PM	7/25/06 8:29 AM	test1.txt	10384	4078
6	7/25/06 3:08 PM	7/26/06 9:25 AM	test2.txt	7106	6011
7	7/31/06 4:11 PM	8/1/06 9:37 AM	test2.txt	6743	5703
8	8/1/06 3:31 PM	8/2/06 1:27 PM	test2.txt	8530	7217

Table 5.1 Durability Test Results for number of button pushes and voice commands used

There were 78702 total simulated button pushes and 41057 total simulated voice commands. Voice commands and button pushes were performed before and after testing and no noticeable delays for each action to be processed by the system were found in either case. The handheld upon which

the tests were conducted never crashed or froze throughout the tests. We concluded that the handhelds were reliable enough to be utilized in a small scale deployment. The length of the deployment was determined by other variables which will be discussed later.

One last set of tests were conducted before the handhelds were deemed ready for deployment. This set of tests involved trying to duplicate and eliminate the potential grammar change condition within the speech subsystem. The tests consisted of a set of voice commands which resulted in changing applications every few seconds. The file used for these tests are shown in Figure 5.5. Each grammar change was logged along with every simulated voice command.



```
[[100]{
    "LIGHTS" <4000>
    "MAIN SCREEN" <4000>
    "RECORDS" <4000>
    "OPERATOR BY NAME AND DATE OF BIRTH" <4000>
    "CHECK RECORDS" <4000>
    "OPERATOR BY O L N" <4000>
    "BACK" <4000>
    "MAIN SCREEN" <4000>
    "PATROL SCREEN" <4000>
    "RADAR" <4000>
    "PATROL SCREEN" <4000>
    "LIGHTS" <4000>
    "MAIN SCREEN" <4000>
    "SPEECH" <4000>
    "MAIN SCREEN" <4000>
}
```

Figure 5.5 Grammar Bug Testbot test file

The first grammar bug tests were carried out on one PDA to evaluate if there was a clearly identifiable problem in the speech subsystem. The results of the first 6 tests were inconclusive, however errors were encountered which supported the existence of a problem. Results of these 6 tests are shown in Table 5.2.

Through analysis of the files logged during testing we were unable to draw any concrete conclusions about the existence of a bug in the speech subsystem. There were points at which the grammar file was not successfully loaded even though the message to load the file was received by the system. These occurrences were rare however, comprising approximately a tenth of a percent of the total grammar changes. To check if an error had occurred the log comprising of a list of every grammar change was checked against another list known to be correct.

Test Date	Total Number of Grammar Changes	Total Number of Errors
10/17/06	22426	25
11/08/06	19697	0
11/13/06	41168	6
11/22/06	18625	3
11/27/06	30172	21
11/29/06	7438	0

Table 5.2 Results for the first round of grammar bug testing

During the first round of testing, consisting of the 6 tests mentioned, only two of the tests yielded greater than 20 grammar change errors. The error rates in those two instances were 0.111 percent and 0.0696 percent. We did not find a pattern that could explain the existence of the errors we found which led us to a second round of testing.

The second round of testing conducted involved the use of multiple handhelds running the exact same test simultaneously. This way we could rule

out the possibility of the bug occurring on one particular handheld and collect much more information in a shorter time span. We used the same Testbot text file to test for an error bug in the speech subsystem with the second round of testing.

The log files of these tests were analyzed and an emerging pattern was noticed in the results. The majority of errors were occurring just before the handheld device was running out of memory. The log files were growing large enough to fill the 128 megabytes (MB) of available memory on the handheld. The results of these tests can be seen in Figures 5.6 through 5.10.

Included in the results is the number of errors that occurred within the last 50 change grammar commands given for that test. This helped to show the pattern mentioned earlier. An application was created to check for errors in the log files. The application checked a known list of grammar changes against the list of grammar changes recorded during testing. If an error was discovered the application would skip ahead in its list of grammar changes until there was a match found with the next recorded grammar file change.

The results for PDA1 over the 6 tests shows a range of error from 0 to 0.08 percent and nearly all errors in changing grammar files occurred within the last 50 commands given. The results shown in Figure 5.7 for PDA2 are similar to the results for PDA1 except with an even smaller range of error values. In all of the figures showing the error rate for grammar changes a number appears above all bars in the graph. The numbers represent (number of errors) / (total number of grammar changes).

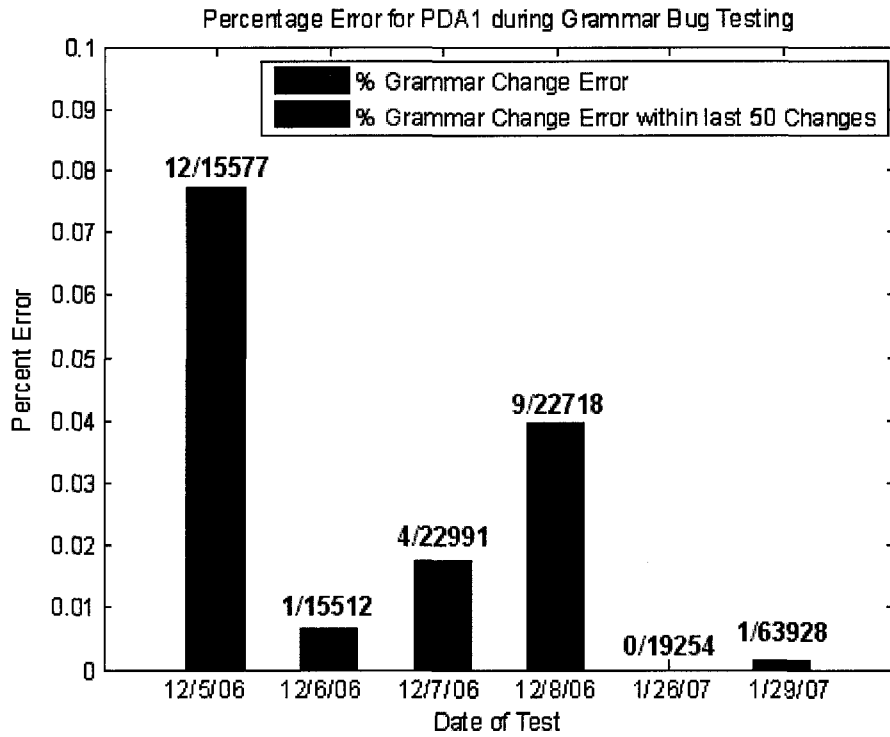


Figure 5.6 Grammar Bug Testing Results for PDA1

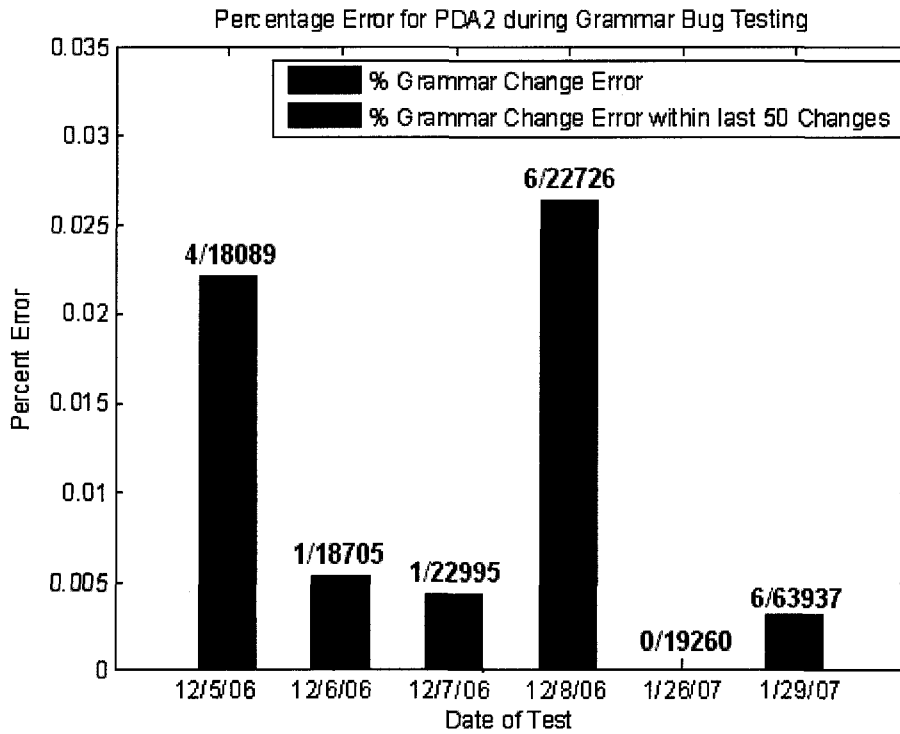


Figure 5.7 Grammar Bug Testing Results for PDA2

The results from the 5 PDAs tested are all nearly identical. Their percentage of error were all in the range from 0 to 0.08 and the majority of all errors occurred during the last 50 change grammar commands given for the test being evaluated. There are a couple possible reasons for errors occurring during the final few minutes of each test. Errors can be introduced by the user when exiting the P54H system. Grammar changes occur whenever a new application is called to show its GUI. To exit the P54H system the user must first navigate to the Mainscreen Application. This is usually the cause of the error in the case when the entire test results in only one grammar change error. The second possible cause of errors with grammar changes occurs just before the PDA runs out of usable memory. Memory on the Symbol MC50 handhelds is shared between storage space and memory for running programs. Once the 128MB of available memory is full the handheld will freeze, requiring a reset of the system.

The total number of grammar change commands given on all PDAs including all tests during the second phase was 731620. There were 98 errors found in these tests which resulted in a final error of 0.0134 percent. 74 of the 98 total errors occurred either as the program was exiting, which is not truly a grammar change error but rather an error introduced by the method of checking for errors, or as the amount of available memory became very small, less than 5MB, due to large log files. We therefore hypothesized that there was no bug in the speech subsystem but rather the PDAs performance was degrading due to insufficient amounts of memory. There were still 24 errors found in the log files which are unexplained and should be addressed in future work.

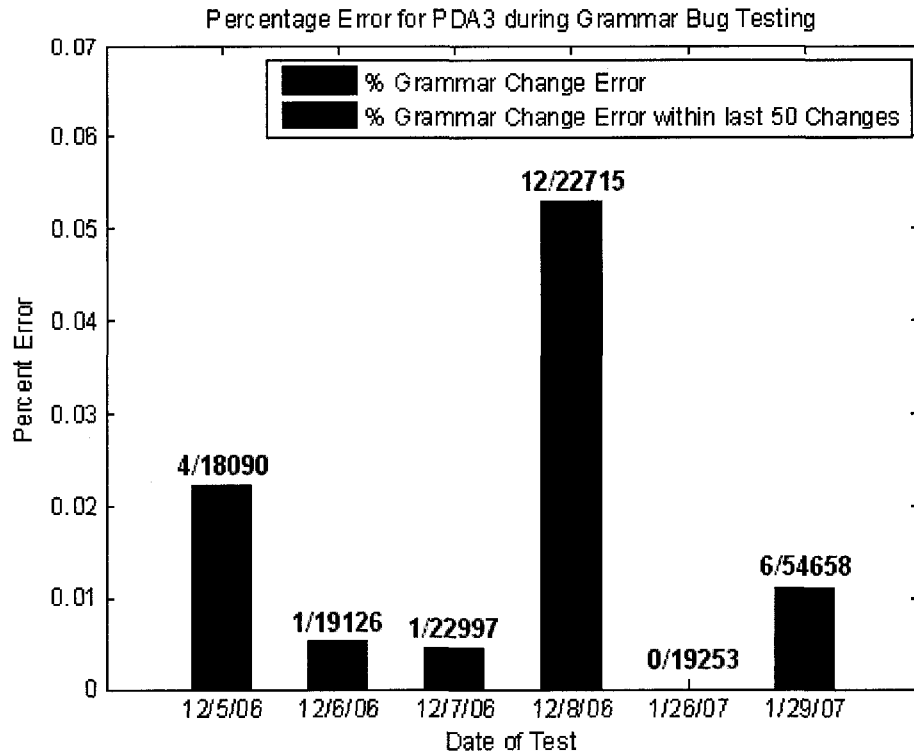


Figure 5.8 Grammar Bug Testing Results for PDA3

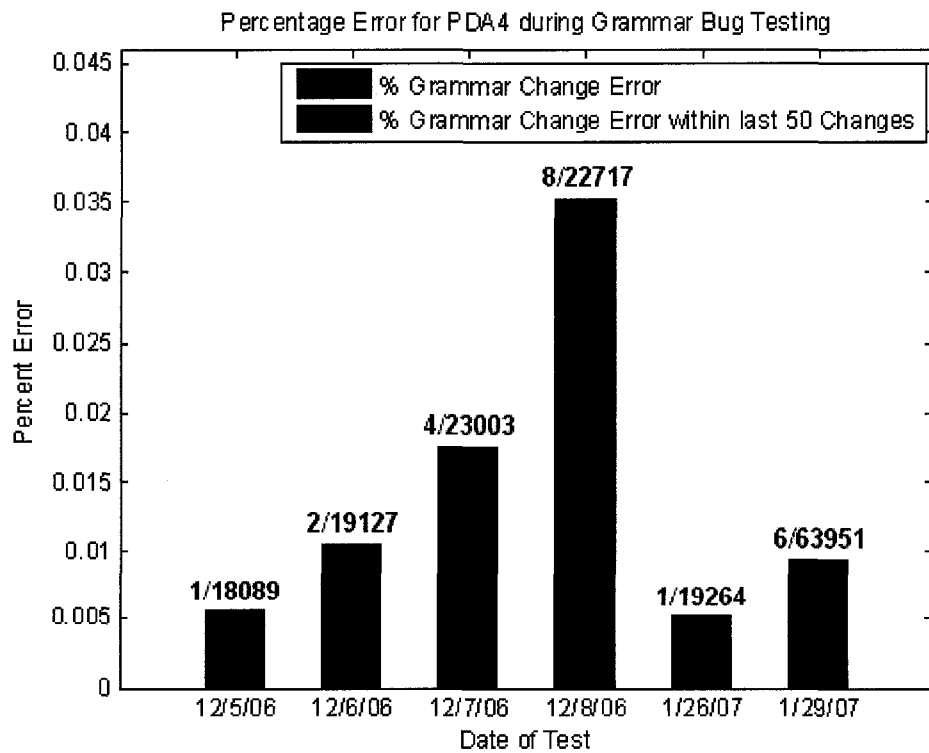


Figure 5.9 Grammar Bug Testing Results for PDA4

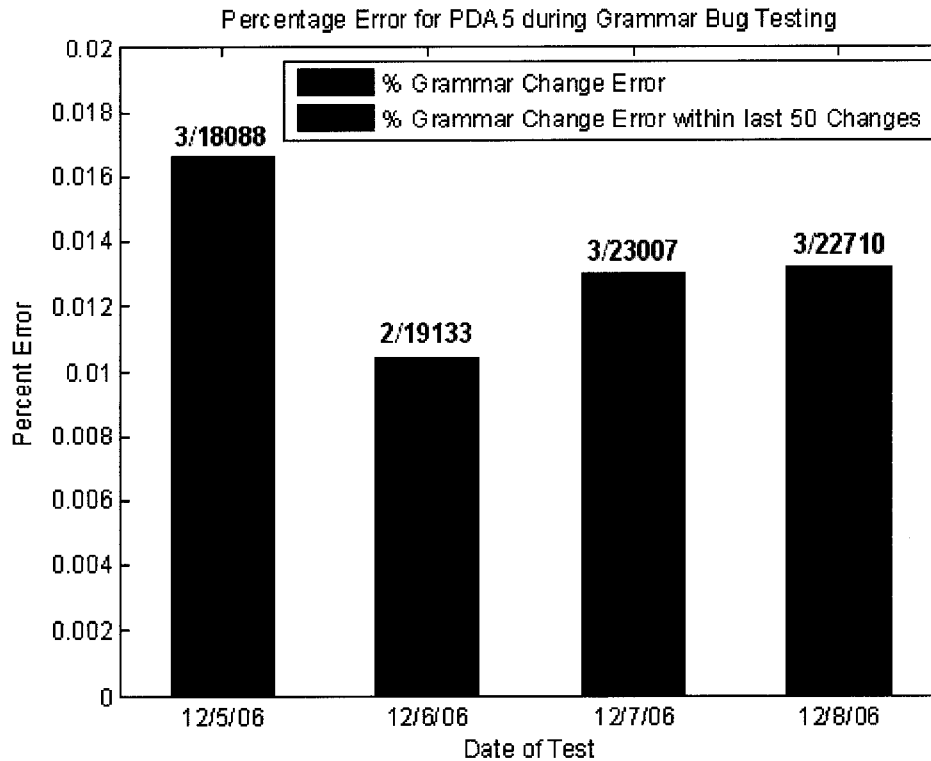


Figure 5.10 Grammar Bug Testing Results for PDA5

The last step in preparing the system for deployment was to make sure the system never reached a state where memory was not available. Looking at the log files created during testing we could approximate how long the deployment could be before the handheld reached this state. We also had to take into consideration the size of the sound files recorded every time a voice command was issued. Ideally we would have liked to deploy the handhelds with all data being recorded on a Secure Digital (SD) expansion card. This would have solved all problems of data storage and memory shortage. However, we encountered a data corruption problem which prevented the deployment of the handheld with an SD card. Therefore to help eliminate memory problems the size of all log files was capped to 10 MB. After the maximum file size was reached data is overwritten. This resulted in the loss of data over a large length

of time but prevented the handheld software from freezing or crashing due to memory shortage. Using an average sound file size of 20KB and assuming an officer would use 25 speech commands per day on average 5MB of storage would be needed for sound files over the two week deployment. The size of log files was also calculated for an average of 2000 total button pushes and voice commands per day. Using an average of 2000 per day the log files would use approximately 25MB of storage over a two week period. Taking into account memory constraints and the size of log files and recorded sound files we proposed to deploy the handheld system for a period of two weeks.

The 10MB file size limit solution was tested with the Testbot Application to investigate if errors would still occur with a set amount of memory available for storage. Two tests were conducted after the PDAs had been deployed. The first automated test resulted in 21373 grammar changes with zero errors. The second automated test resulted in 21397 grammar changes with zero errors. Although the existence of bug was possible we deployed the handhelds because of the small percentage of errors encountered. With a speech recognition rate of approximately 85% [28] this bug would introduce a fraction of a percent worse performance which we deemed acceptable for this deployment.

CHAPTER 6

DEPLOYMENT RESULTS

The fourth proposed step was to deploy multiple handhelds to officers in the New Hampshire State Police. Because of the limited time frame for this research and the number of available handheld units the pilot deployment consisted of three handhelds deployed for a 2-3 week time period. The fifth proposed step was to use information recorded on the handhelds while in the field to evaluate how they were used. First though, the handheld system was configured to log data. The data logged included four logs, the P54 error log, the P54 event log, the P54 GUI button log and the P54 sniffer log. The error log contained any system errors encountered, the event log contained speech subsystem messages, the GUI button log developed by Edward Bourbeau [29] contained data recorded whenever a button was used on the touch screen and the sniffer log contained all inter-application messages. We also configured the handheld system to record a sound file every time the PTT button was used. Information about each sound file was recorded in a P54 audio list file.

We conducted two evaluations of the handheld system. In the first evaluation we examined the log files obtained from the handhelds. The logs contained information about how the officer used the GUI and SUI. These results can be found in section 6.1. In the second evaluation a survey about the

handheld system was completed by the officers that used the system. The survey, shown in Appendix C, applied a Likert scale evaluation to statements about the system. The survey results are explained in section 6.2.

6.1 Handheld Log File Results

The log files recorded during deployment were examined to get an idea of how the officers interacted with the handheld system. Specifically we examined how the officers used the GUI and SUI as well as how well the SUI performed. The results were obtained by parsing the log files into usable data. Any PTT button pushes were not included in the button push totals.

It is also important to note the difference between task level activity and functional level activity. For example a particular task may take 15 button pushes to accomplish through the GUI but only one voice command to accomplish through the SUI. If a user accomplished the task once using each method he/she would appear to favor the use of the GUI. In actuality the user only accomplished the task twice, once through the GUI and once through the SUI. The only case where the task level activity was not the same for both the GUI and SUI was when a user scrolled through records data. Therefore to show task level results all button pushes resulting in records data scrolling are shown separate from the remaining data.

Figure 6.1 shows the number of buttons pushed compared to voice commands used for each of the 5 major applications for officer one. The 5 major applications are Mainscreen, Patrol Screen, WhelenMPCO, Radar, and Records.

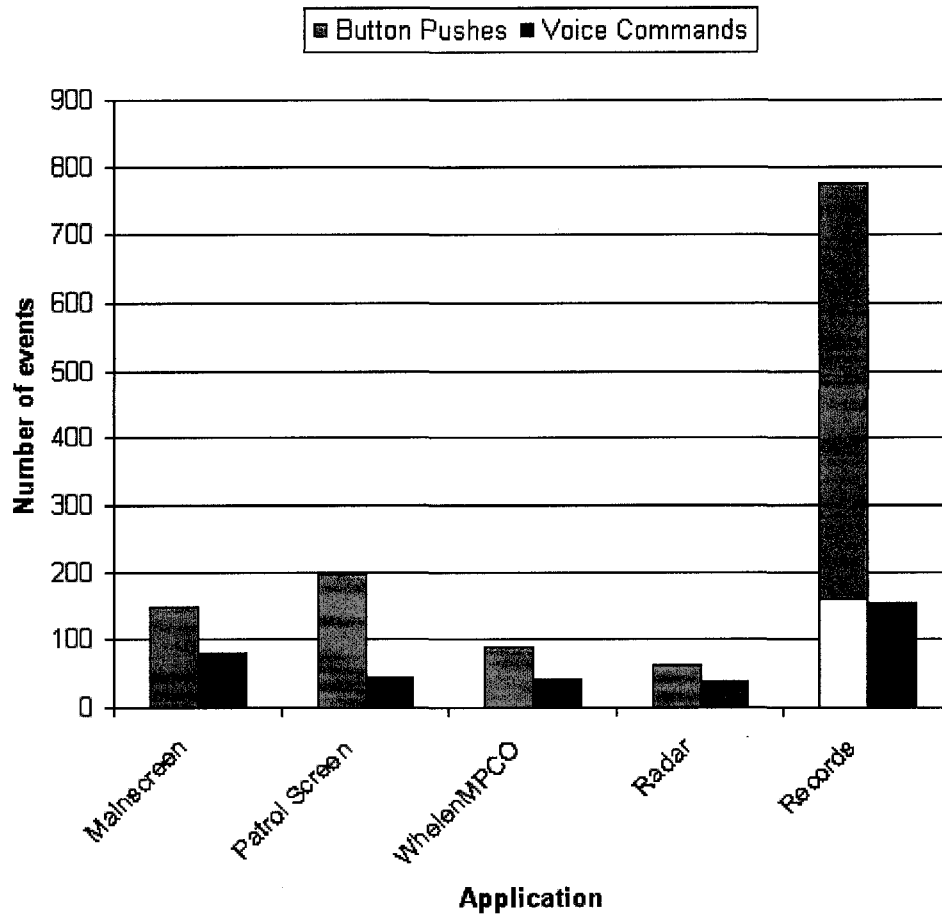


Figure 6.1 GUI usage and SUI usage per application for Officer One

Figure 6.1 shows that the GUI used more than the SUI for most applications. This is especially apparent in the case of the Records Application. However if we remove the button pushes used to scroll through records data the number of button pushes drops from 775 to 159 as highlighted in yellow. The results show that the officer was twice as likely to use the touch screen instead of voice commands.

Looking at the data on a per day basis shows the same pattern. Officer one used button pushes twice as much as voice commands to control the handheld. This data is shown in Figure 6.2.

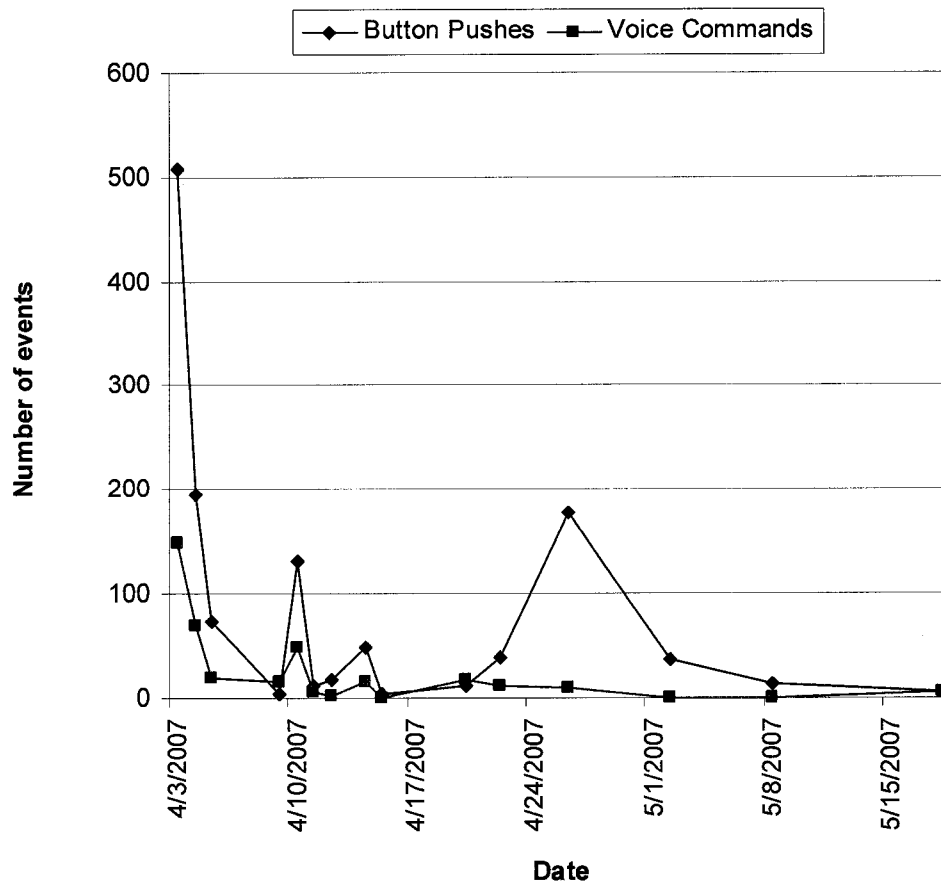


Figure 6.2 GUI usage and SUI usage by date for Officer One

From the perspective of a safety standpoint it would be safer for officers to use the handheld with speech input, allowing them to keep one hand free. However if speech commands do not work correctly, frustration could result in officers using the touch screen more frequently. The speaker independent voice engine used by the handheld system has shown accuracy results above 95% [2]. However these results were obtained in a laboratory setting where noise was not a factor. We therefore evaluated the effectiveness of the SUI in a real world setting.

Figure 6.3 shows the recognition percentage of voice commands for officer one. Also shown are the type of errors encountered quantified by a

percentage value. The results from officer one show only a 75% recognition accuracy with 79% of the recognition errors being system errors. To look for errors we play back the recorded audio files and compare the recorded sound to the results from the speech recognizer. If we hear one particular command while playing the recorded audio and the speech recognizer returned a different result we mark it as a system error. User errors occur when an officer presses the PTT button too late or releases the PTT button too soon. The miscellaneous error category in Figure 6.3 refers to misrecognized commands due to excessive noise.

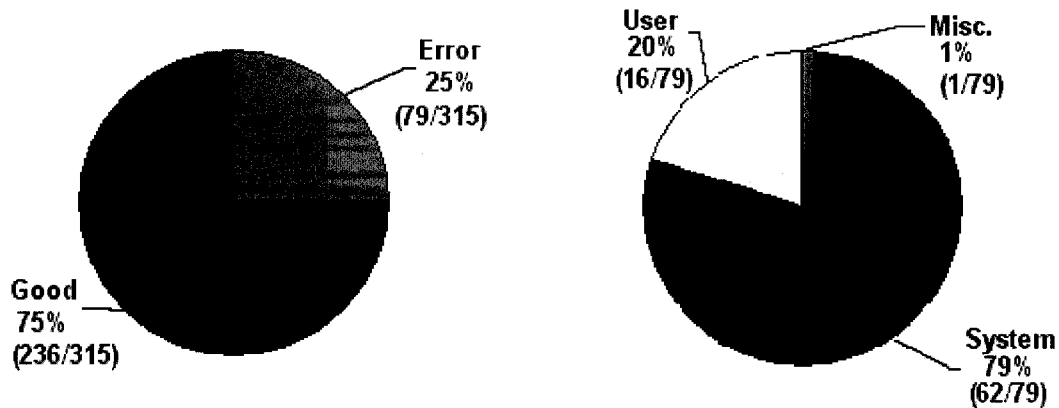


Figure 6.3 Voice Command recognition accuracy and percentage of errors by type for Officer One

There were a large number of system errors however 39 out of the 62 encountered were from the voice commands “Records” and “Check Records”. This was because there are two pronunciations of the word “Records” and the voice recognizer on the handheld only understands one. As with any voice activated system there is a certain learning curve for speech interaction. Once the officer started using the other pronunciation of the word “Records” the accuracy of the SR improved as shown in Figure 6.4 and Figure 6.5.

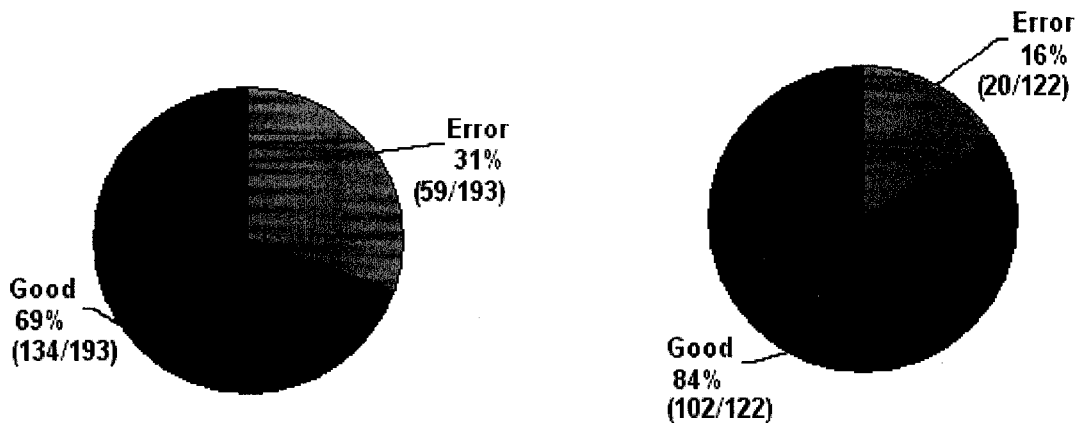


Figure 6.4 Voice command recognition from April 3 – April 4 for Officer One

Figure 6.5 Voice command recognition from April 5 – May 18 for Officer One

Figure 6.4 shows the recognition accuracy for officer one over the first two days he used the handheld system. Figure 6.5 shows the recognition accuracy for officer one after the first two days. From April 5 through May 18 his recognition accuracy improved by 15% from 69% to 84%. This can be attributed to a learning curve found with any new tool. These results are similar to those found with the in-car system when it was first used.

Looking at the data for officer one we can discern information about how the system was used. When the officer first received the handheld he tested the GUI and SUI separately. He navigated through the system using button presses before attempting to use the speech interface. Once he was comfortable with the GUI he began using the SUI to control the handheld.

After a few days of use the officer began to interchange his use of the GUI and SUI. Table 6.1 shows data recorded on the handheld of officer one performing a traffic stop with the handheld.

Current Application	Time	Command
PatrolScreen	9:27:47 AM	Rear Antenna Button Pressed
PatrolScreen	9:27:52 AM	Rear Strobes Button Pressed
PatrolScreen	9:27:54 AM	Rear Strobes Button Released
PatrolScreen	9:28:10 AM	REAR STROBES voice command
PatrolScreen	9:28:24 AM	MAIN SCREEN voice command
Mainscreen	9:28:34 AM	RECORDS voice command
Records	9:28:40 AM	OPERATOR BY NAME AND DATE OF BIRTH voice command
Records	9:28:41 AM	Hardware scan activated
Records	9:29:19 AM	CHECK RECORDS voice command
Records	9:29:30 AM	U P Button Pressed
Records	9:30:03 AM	D O W N Button Pressed

Table 6.1 Data from Officer One indicating a traffic stop conducted using the handheld

One way officer one used the handheld to perform traffic stops was to park his car on the side of the road, stand approximately 100 feet away, view radar information through the handheld and activate his lights as a speeding car approached. Using the data from Table 6.1 we can construct the following scenario. With the PatrolScreen Application active he turned on the rear radar antenna. Five seconds later the officer activated the strobe lights for two seconds using button pushes. Fifteen seconds later he activated the strobe lights with a voice command to signal a vehicle to pull to the side of the road. He then navigated to the Records Application using voice commands. The officer entered the operator by name and date of birth GUI through a voice command

and proceeded to scan a license using the MC50 hardware scan button (Figure 4.8). The officer issued the CHECK RECORDS voice command to query the suspect's records and scrolls through the data returned.

The above scenario was the most common use of the handheld for officer one. The data showed that he primarily used button pushes to control the lights and radar unit. Once the officer wanted to access the Records Application he switched to using voice commands. This was apparent in the data because there were large sections of only button pushes followed by a section of only voice commands. The only exception for the preference of voice commands over button pushes in the Records Application was for scrolling through records data.

The PatrolScreen and Records Applications were the two applications predominantly used by officer one because they are both used in traffic stops. 70.2% of button pushes recorded, excluding those used to scroll through records data, were done in either the PatrolScreen or Records Application. 70.8% of voice commands issued by officer one occurred in either the PatrolScreen or Records Application.

There are three methods of activating the built-in 2D scanner through Project54. A user can use a voice command, use the touch screen, or use a hardware button on the PDA. Out of the 90 times the scanner was activated, officer one used 0 voice commands, 1 touch screen button push, and 89 hardware button pushes. Officer one almost exclusively used the PDA hardware buttons to activate the scanner.

There were zero instances where a task was exclusively completed using either the GUI or SUI. With the exception of the Records Application the GUI was used more than the SUI for officer one.

Figure 6.6 shows the number of buttons pushed compared to voice commands used for each of the 5 major applications for officer two. It should be noted that this particular officer had an unsuccessful experience with the handheld system. He experienced connection problems between the wireless card in his cruiser and the wireless card in the PDA. The connection problem was later resolved by Andras Fekete¹.

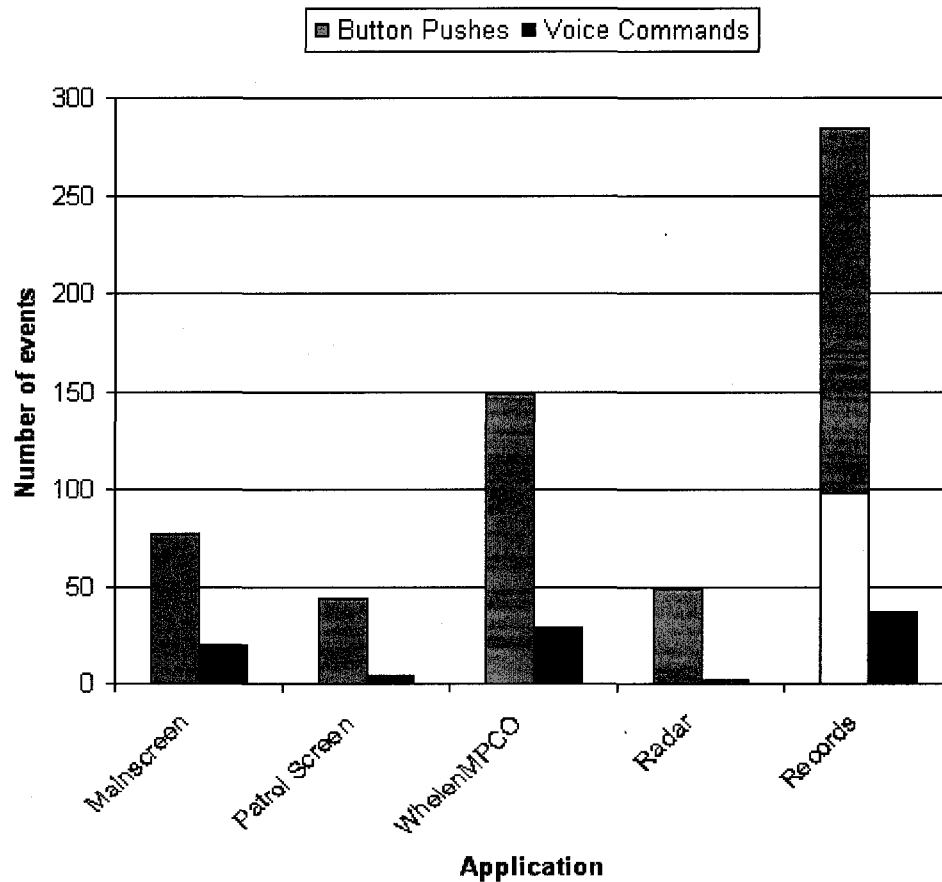


Figure 6.6 GUI usage and SUI usage per application for Officer Two

¹ Learned of the Connection Bug fix from Andrew L. Kun through personal communication on August 3, 2007.

We can see from Figure 6.6 that officer two used the GUI more than the SUI. Again there were a large number of button pushes in the Records Application attributed to scrolling through records data. With the button pushes due to scrolling removed the button presses drop to 96 for the Records Application which is highlighted in yellow. Looking at the same data per day we can see that after the first day of use the officer used the GUI approximately twice as much as the SUI, the same as officer one. Figure 6.7 shows the amount of button presses and voice commands used by officer two per day.

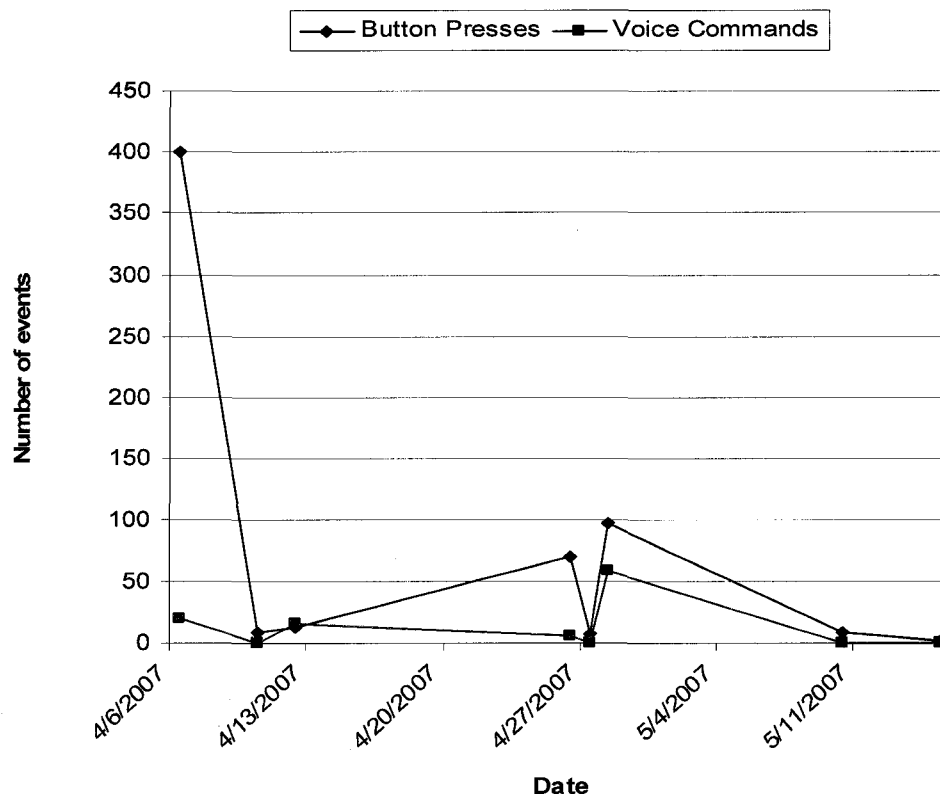


Figure 6.7 GUI usage and SUI usage by date for Officer Two

The voice recognition results for officer two are show in Figure 6.8. The results are with a few percent of the results for officer one. There were far fewer

voice commands and button pushes for officer two which was a result of the connection problems mentioned earlier.

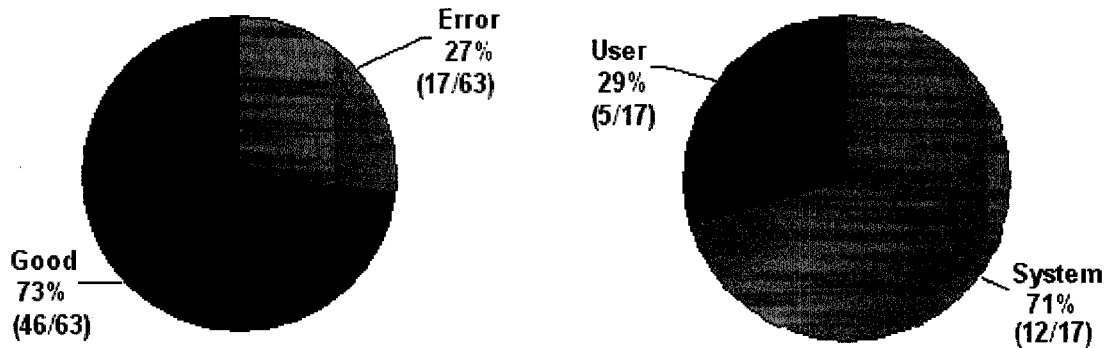


Figure 6.8 Voice Command recognition accuracy and percentage of errors by type for Officer Two

Data was also collected for officer three, however the amount of data collected was too small to offer any valuable information pertaining to the use of system or individual services.

6.2 Handheld Survey Results

The results from the handheld surveys can be placed in two categories. The results from the Likert scale questions and the additional comments given by the officers. There were 16 statements given to the officers that they evaluated using the following Likert scale: Strongly Agree, Agree, Neither Agree nor Disagree, Disagree, Strongly Disagree. The results from the Likert scale statements are shown in Table 6.2.

Survey's were completed and returned by the three officers mentioned earlier. No data was presented for officer three because of the small amount of data collected, only 60 button pushes and 27 voice commands. Reviewing the results from the three officers we can clearly see two very different user

experiences. The three officers agreed on a few of the statements. They all agreed the handheld software was reliable, the scanner returned accurate information and the handheld was easy to use when using either the Lights or Radar. Aside from these statements the officers' opinions differed. For instance officer one had a successful experience using speech to control the handheld while officer two did not. As a result officer one had an easier time using the handheld with only one hand. In the case of officer two it is hard to discern how much impact the connection problems experience during deployment had on the survey results.

Reviewing the results in Table 6.2 we can conclude that we had two successful and one unsuccessful deployment of the handheld system. While we would like officers to have a positive experience with the handheld system, the negative experience in this case resulted in excellent user feedback. There were seven comments given by the officers which included problems encountered and suggestions for future expansion.

The one comment given by all of the officers dealt with the volume level of the handheld speakers. The speaker volume was adequate for office use but was low for use in an outdoor environment. This was especially apparent when an officer stood outside the vehicle in the break-down lane on the interstate.

As mentioned earlier officer two had problems with the wireless connection between his handheld and the wireless card in his vehicle. The connection problems caused the handheld and in-car systems to synchronize in a sporadic manner. The two devices would connect for a short period of time

before disconnecting for some unknown reason. This happened repeatedly resulting in an unfriendly user experience.

	Officer One	Officer Two	Officer Three
The handheld software was reliable and did not crash.	Agree	Agree	Strongly Agree
The touch screen was frustrating to use.	Disagree	Neither Agree nor Disagree	Disagree
It was easy to control the handheld using speech commands.	Agree	Disagree	Neither Agree nor Disagree
Speech output from the handheld was easy to understand.	Neither Agree nor Disagree	Disagree	Disagree
The handheld often did not understand my speech input.	Disagree	Agree	Disagree
There was a noticeable delay for status updates sent between the handheld and the car.	Disagree	Neither Agree nor Disagree	Disagree
The scanner worked quickly.	Agree	Disagree	Neither Agree nor Disagree
The scanner gave accurate information.	Agree	Agree	Agree
The handheld software was frustrating to use while performing records checks.	Disagree	Neither Agree nor Disagree	Neither Agree nor Disagree
The handheld software was easy to use while viewing radar information.	Agree	Agree	Agree
The handheld software was frustrating to use while activating/deactivating lights.	Strongly Disagree	Disagree	Neither Agree nor Disagree
The handheld device was comfortable to use with one hand.	Agree	Disagree	Agree
The handheld made my daily tasks easier to perform.	Agree	Neither Agree nor Disagree	Neither Agree nor Disagree
I often needed two hands to control the handheld device.	Disagree	Strongly Agree	Disagree
I used the handheld very often during the day.	Agree	Disagree	Disagree
My overall satisfaction with the handheld system is high.	Agree	Disagree	Agree

Table 6.2 Likert scale survey results for Officer One, Officer Two and Officer Three

The third comment given addressed the design of individual applications. For example the Radar Application contains text field which displays the “Patrol

Speed". The officer commented that the handheld unit would most likely never be used in a "moving mode" and therefore a display field for the patrol speed is unnecessary. Instead the officer would like to make the remaining two text fields which display current target speed and highest/lock speed larger. Also he would like to see a button to activate the strobes added to the Radar Application. This way an officer could activate the strobe lights without switching to the PatrolScreen Application.

The officers would also like to see a change in the scanner application. Currently if a license is scanned on the handheld the information fills the fields on the handheld only. While this is fine when running a records check from outside the vehicle it would be useful to have the fields on the in-car system populate on a successful scan the same way a tethered scanner does. This would eliminate the need for the tethered scanners in a vehicle.

Lastly some rearranging of the button positions in handheld applications to match the format of the in-car system was suggested. This would be a very simple operation that would allow a user familiar with the in-car system to intuitively use the GUI of the handheld system.

CHAPTER 7

CONCLUSION

Our primary research goal was to develop and test a new Project54 handheld system, thus providing officers access to in-car services from outside the vehicle. The secondary goal was to evaluate how the system was used by officers. The development of the new handheld system had to meet certain constraints. First, any applications developed had to mimic the functionality and look of any in-car applications, using the same buttons, voice commands, text fields and status lights. Second, a PTT button had to be used to capture voice commands. Third, the handheld system had to be developed as a periphery to the in-car system, not capable of controlling hardware devices alone. Fourth, data had to be recorded that could be later used to evaluate the system. Lastly, the handheld software had to be simple to install and configure with an in-car system.

The first proposed step of this research was to make installation and configuration of the handheld system simple and quick for installers. We consolidated the handheld software into a software package which is installed through Microsoft Activesync. During the installation process all necessary configuration utilities run automatically. The utilities are stored on the handheld device for later configuration.

The second proposed step was to create new applications. We created three applications for the new handheld system. The SymbolScanner application was developed to interface with the Symbol MC50's built-in hardware scanner. We also developed PatrolScreen and WhelenMPCO Applications to better match the functionality found on the in-car system.

The third proposed step was to conduct internal testing of the completed system. We conducted two types of testing, laboratory and automated. We tested the handheld system on our laboratory setup which contains a fully functional in-car system and found that each button and voice command worked properly. We also conducted automated robustness testing of the handheld system. We accomplished this by developing the Testbot Application which allowed button presses and voice commands to be simulated. We found that the software was stable over at least a twelve hour period. We also tested the speech subsystem to confirm grammar files were changing when they were supposed to and found no problems with the subsystem.

The fourth proposed step was to deploy the handhelds to several operating cruisers and collect data about handheld usage. The handhelds were deployed to two officers in the NH State Police and data was collected for each officer over a two week period.

The fifth proposed step was to conduct a comprehensive evaluation of the handheld system using data recorded on the handhelds and feedback from officers. We evaluated the system in two ways. The log files and recorded voice commands were examined to evaluate GUI versus SUI usage and speech

recognition. We then evaluated the handheld system using feedback from a handheld survey given to the officers that used the system.

By completing the above steps we were able to accomplish our goal of developing, testing, and deploying a new Project54 handheld system. Using the handheld system, officers are able to access in-car services from outside the vehicle.

Our second goal of evaluating the P54H System was not quite accomplished. The ideas obtained concerning the usability of the handheld system and which services proved most useful were preliminary only. The limited deployment provided only a brief insight into how the system was integrated into an officer's daily tasks.

CHAPTER 8

SUGGESTIONS FOR FUTURE WORK

The officers provided some useful feedback toward future development of the Project54 handheld system. The problem addressed by all officers involved the volume of the speakers found on the handheld device. The environment outside a police vehicle contains a high level of background noise which overpowered the volume of the handheld speakers, especially in the case where large amounts of traffic were present. One solution to this problem would be to develop a wireless microphone/speaker combination device. This device could use a Bluetooth wireless interface and could be worn by an officer.

Buttons found on each GUI could be rearranged to better match the configurations found on the in-car system. It may also be advantageous to remove the use of status lights on the handheld GUIs. The already limited space may be better used for buttons. This would allow button sizes to grow, making them easier to use.

The Records Application needs most of its touch screen real estate to display records data. Making the viewing window as large as possible would help officers to review records data quickly. The scrolling functions could also be bound to hardware buttons instead of software buttons to allow quick scrolling with one hand. This would also allow the viewing window to expand. Lastly,

voice feedback of the records data similar to how it is presented in the car could alleviate the need to scroll through the data entirely.

To address the problem of limited memory on the handheld the possibility of transferring data to the in-car system during downtime could be explored. The data would be sent via the wireless link between the systems and would help alleviate the growing issue of memory shortage on the handheld.

Lastly, the speech bug explored in this research could be looked into further. As an offshoot of this research it would also be interesting to explore the differences in speech recognition between the in-car and handheld systems. Noisy and clean speech command utterances could be run through each system which would allow a direct comparison of each recognition engine.

LIST OF REFERENCES

- [1] Krstovski, K., "Handheld Project54 Speech User Interface," Masters Thesis, University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratories (CATLAB)-Project54, September, 2005.
- [2] Krstovski, K., Kun, A., Miller, W.T., III, "Speech interaction with handheld computers in the Project54 system," 6th International Conference on Ubiquitous Computing (UbiComp'04), Electronic Adjunct Proceedings, Nottingham, England, September 7-10, 2004
- [3] Microsoft Inc., "Microsoft ActiveSync",
<http://www.microsoft.com/windowsmobile/activesync/activesync45.msp>
[Accessed: April 23, 2007]
- [4] Griswold, W., Shanahan, P., Brown, S., Boyer, R., "ActiveCampus – Experiments in Community-Oriented Ubiquitous Computing." In IEEE Computer, Vol. 37, No. 10, pages 73-81, 2004.
- [5] Aziz, O., Panesar, S., Netuveli, G., Paraskeva, P., Sheikh, A., Darzi, A., "Handheld computers and the 21st century surgical team: a pilot study", BMC Medical Informatics and Decision Making, 5:28, August 18, 2005.
- [6] Sousa, J. and Garlan, D., "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments." Software Architecture: System Design, Development, and Maintenance (Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture), Kluwer Academic Publishers, August 25-31, 2002. pp. 29-43.
- [7] Myers, B., "Using Handhelds and PCs Together", Communications of the ACM, Volume 44, No. 11, November 2001. pp. 34-41.
- [8] Myers, B., Stiel, H., Gargiulo, R., "Collaboration Using Multiple PDAs Connected to a PC", Presented at: The 1998 ACM Conference on Computer Supported Cooperative Work, Seattle, WA, November 14-18, 1998.
- [9] Eagle, N. and Pentland, A., "Wearables in the Workplace: Sensing Interactions at the Office." In 7th IEEE International Symposium on Wearable Computers, October 2003.

- [10] Couture, J., "PDA Application Development at the Department of Transportation", *Maine IS Technology*, Volume 7, Issue 2, February, 2004. pp. 5-6.
- [11] Perlman, A., "Ephemerizer: Making the Data Disapper, The", Sun Microsystems Laboratories, Technical Report Number: TR-2005-140, Feb 1, 2005
- [12] Cisco Systems. "City of Renton Police Department: Wireless Community Network Protects Lives, Enhances Work", 2005.
- [13] Hazas, M., Scott, J., Krumm, J., "Location-Aware Computing Comes of Age", *IEEE Computer Magazine*, Vol. 37, No. 2, February, 2004
- [14] Castro, P., Chiu, P., Kremenek, T., Muntz, R., "A Probabilistic Room Location Service for Wireless Networked Environments", *Proceedings of IEEE UbiComp 2001*, pp. 18-34, Atlanta, Georgia, 2001
- [15] Krumm, J., "Probabilistic Inferencing for Location." In *Proc. Workshop on Location-aware Computing*, part of UBIComp Conf., Seattle, WA, October 2003.
- [16] Schilit, B., LaMarca, A., McDonald, D., Tabert, J., Cadag, E., Borriello, G., Griswold, W., "Bootstrapping the Location-enhanced World Wide Web", *UbiComp 2003, Location-aware computing workshop, Proceedings of the 2003 Workshop on Location-Aware Computing*, pp. 1-3, October 12, 2003, Seattle, WA.
- [17] Cadman, J., "Deploying Commercial Location-Aware Systems", *UbiComp 2003, Location-aware computing workshop, Proceedings of the 2003 Workshop on Location-Aware Computing*, pp. 4-6, October 12, 2003, Seattle, WA.
- [18] Krohn, A., "Relative Positioning", *UbiComp 2003, Location-aware computing workshop, Proceedings of the 2003 Workshop on Location-Aware Computing*, pp. 22-24, October 12, 2003, Seattle, WA.
- [19] Kun, A. L., Dogan, K., "A prototype remote access and mobile data transaction system for police cruisers", *IEEE Spring VTC2002*, Birmingham, AL, May 6-9, 2002
- [20] Krstovski, K., "Handheld Project54 Radar Application", Technical Report ECE.P54.2005.5, University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratories (CATLAB)-Project54, July 2, 2005.

- [21] Krstovski, K., "The Scanner Application for the Mobile Project54 Software System", Technical Report ECE.P54.2004.8, University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratories (CATLAB)-Project54, July 20, 2004.
- [22] Microsoft Inc., "Cabinet File Description", <http://www.microsoft.com/technet/prodtechnol/ie/ieak/techinfo/deploy/60/en/cab.msp?mfr=true> [Accessed: May 9, 2007]
- [23] Ludwig, S., "EZSetup", <http://www.pocketpcdn.com/articles/ezsetup.html>, [Accessed: 2005]
- [24] Microsoft Inc., "Creating an .inf File", <http://msdn2.microsoft.com/en-us/library/ms924764.aspx> [Accessed: 2005]
- [25] Macrovision, "HOWTO: Creating a Windows CE Setup DLL", <http://support.installshield.com/kb/view.asp?articleid=Q104092> [Accessed: 2005]
- [26] Miller, W. T., "Overview of Project54 inter-application messaging: Local and distributed messaging architectures", Technical Report ECE.P54.2004.11, University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratories (CATLAB)-Project54, September 30, 2004.
- [27] Lynch, R. L., "The SpeechBot", Technical Report ECE.P54.2003.17, University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratories (CATLAB)-Project54, 2003.
- [28] Turner, L., Kun, A. L., "Evaluating the Project54 speech user interface," Third International Conference on Pervasive Computing (Adjunct Proceedings), Munich, Germany, May 8-13, 2005.
- [29] Bourbeau, E., "A Prototype System for Human-Computer Interaction Logging, Post-Processing, and Data Visualization for the Project54 System", Masters Thesis, University of New Hampshire, Consolidated Advanced Technologies for Law Enforcement Laboratories (CATLAB)-Project54, September, 2007.

APPENDICES

APPENDIX A

INTERNAL TESTING TEST #1

```
test1 - Notepad
File Edit Format View Help
[[I] {
    (1,3) <2000>
    "MAIN SCREEN" <5000>
}
[100]
{
    (1,3) <3000>
    (1,1) <5000>
    (2,1) <3000>
    (3,1) <3000>
    "LIGHTS OFF" <6000>
    (3,4) <2000>
    "FRONT STROBES" <3000>
    (2,2) <3000>
    (1,3) <4000>
    (2,3) <4000>
    (3,3) <4000>
    (2,4) <3000>

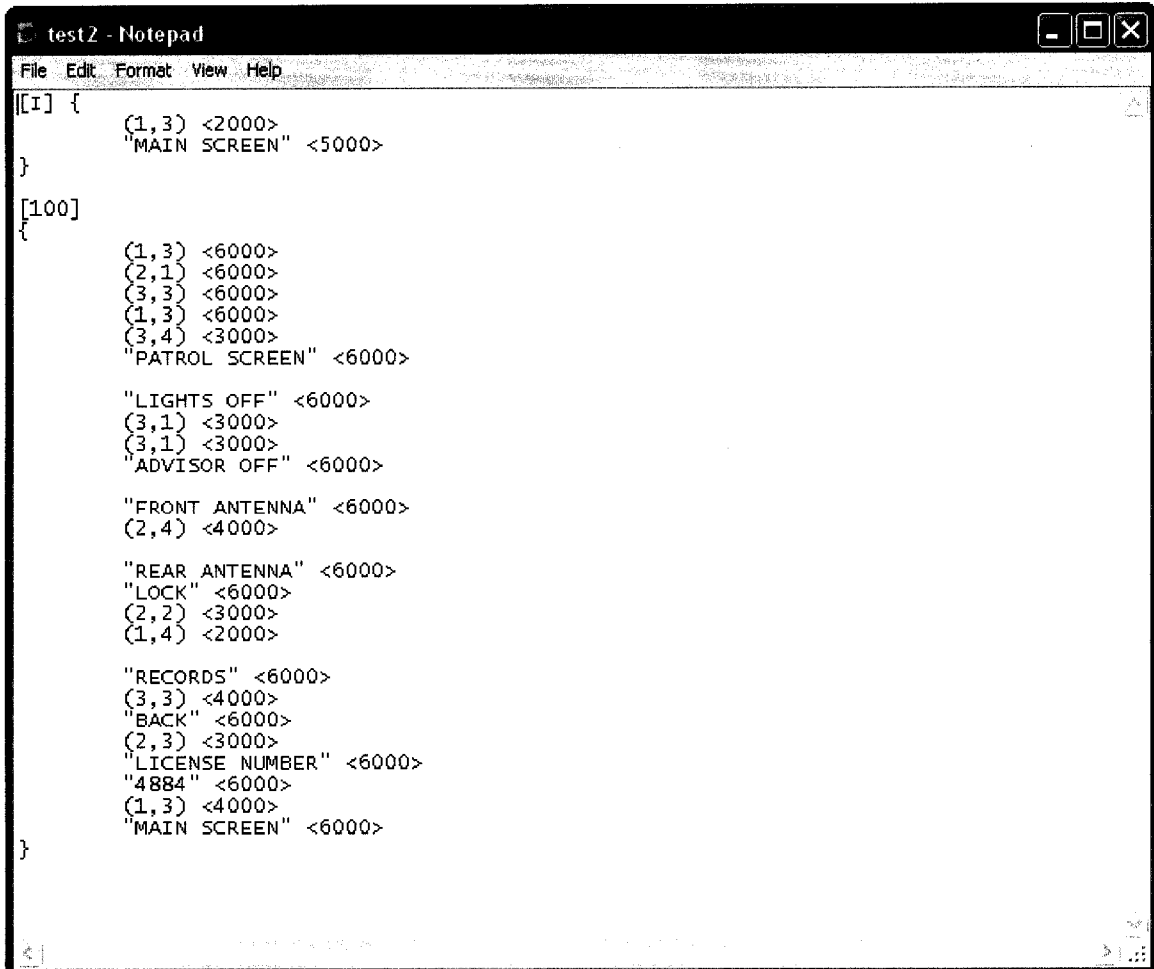
    "LIGHTS OFF" <5000>
    (1,3) <4000>
    (2,3) <4000>
    (3,3) <4000>
    "LOCK OFF" <4000>

    "RADAR" <5000>
    (1,2) <5000>
    (1,2) <5000>
    (1,3) <4000>
    (2,3) <4000>
    (3,3) <4000>
    "MAIN SCREEN"

    (1,2) <5000>
    (1,3) <3000>
    (2,4) <8000>
    "BACK" <2000>
    (2,3) <5000>
    (2,3) <5000>
    (3,3) <4000>
    (1,4) <5000>
    "SCANNER" <3000>
    (1,3) <5000>
    "LICENSE CHECK" <2000>
    "SCAN" <8000>
    (3,4) <2000>
    "SPEAK DATA OFF" <2000>
    (1,4) <5000>
}
}
```

APPENDIX B

INTERAL TESTING TEST #2



```
test2 - Notepad
File Edit Format View Help
[[I] {
    (1,3) <2000>
    "MAIN SCREEN" <5000>
}
[100]
{
    (1,3) <6000>
    (2,1) <6000>
    (3,3) <6000>
    (1,3) <6000>
    (3,4) <3000>
    "PATROL SCREEN" <6000>

    "LIGHTS OFF" <6000>
    (3,1) <3000>
    (3,1) <3000>
    "ADVISOR OFF" <6000>

    "FRONT ANTENNA" <6000>
    (2,4) <4000>

    "REAR ANTENNA" <6000>
    "LOCK" <6000>
    (2,2) <3000>
    (1,4) <2000>

    "RECORDS" <6000>
    (3,3) <4000>
    "BACK" <6000>
    (2,3) <3000>
    "LICENSE NUMBER" <6000>
    "4884" <6000>
    (1,3) <4000>
    "MAIN SCREEN" <6000>
}
```

APPENDIX C

HANDHELD SYSTEM SURVEY

Handheld System Survey

Project54™
332 Morse Hall
Durham, NH 03824

You are receiving this survey because the Project54 Handheld system has been installed in your vehicle. We are looking for feedback on various aspects of the handheld system. Please fill out the survey as accurately as possible and feel free to include any additional comments about the system.

This survey can be returned via email to cmg@unh.edu or sent via fax to 603.862.1761.

Name					
Vehicle Number					
Email					
Phone					
	Strongly Agree	Agree	Neither Agree nor Disagree	Disagree	Strongly Disagree
The handheld software was reliable and did not crash.					
The touch screen was frustrating to use.					
It was easy to control the handheld using speech commands.					
Speech output from the handheld was easy to understand.					
The handheld often did not understand my speech input.					
There was a noticeable delay for status updates sent between the handheld and the car.					
The scanner worked quickly.					
The scanner gave accurate information.					
The handheld software was frustrating to use while performing records checks.					
The handheld software was easy to use while viewing radar information.					
The handheld software was frustrating to use while activating/deactivating lights.					
The handheld device was comfortable to use with one hand.					
The handheld made my daily tasks easier to perform.					
I often needed two hands to control the handheld device.					
I used the handheld very often during the day.					
My overall satisfaction with the handheld system is high.					

APPENDIX D

INSTITUTIONAL REVIEW BOARD APPROVAL

University of New Hampshire

Research Conduct and Compliance Services, Office of Sponsored Research
Service Building, 51 College Road, Durham, NH 03824-3585
Fax: 603-862-3564

05-Jul-2007

Kun, Andrew
Electrical & Computer Eng Dept
Kingsbury Hall
Durham, NH 03824

IRB #: 2980

Study: Speech Sample Collection for Speech Recognition Engine Comparison and Development

Review Level: Expedited


Approval Expiration Date: 24-Jun-2008

The Institutional Review Board for the Protection of Human Subjects in Research (IRB) has reviewed and approved your request for time extension for this study. Approval for this study expires on the date indicated above. At the end of the approval period you will be asked to submit a report with regard to the involvement of human subjects. If your study is still active, you may apply for extension of IRB approval through this office.

Researchers who conduct studies involving human subjects have responsibilities as outlined in the document, *Responsibilities of Directors of Research Studies Involving Human Subjects*. This document is available at <http://www.unh.edu/osr/compliance/irb.html> or from me.

If you have questions or concerns about your study or this approval, please feel free to contact me at 603-862-2003 or Julie.simpson@unh.edu. Please refer to the IRB # above in all correspondence related to this study. The IRB wishes you success with your research.

For the IRB


Julie F. Simpson
Manager

cc: File