The Remote Hawkeye

Final Report

Version 1.0
April 16, 2002

Integrated Product and Process Design
University of Florida

United States Special Operations Command

Dr. Sumi Helal

Eric Grobelny
Doug Holubek
Fenghai Lan
Gregg Mize
Ryan Palacheck
Hady Perez
Matt Potok

The Remote Hawkeye

Volume I
Final Report

Version 1.0
April 16, 2002

Integrated Product and Process Design
University of Florida

United States Special Operations Command

Dr. Sumi Helal

Eric Grobelny
Doug Holubek
Fenghai Lan
Gregg Mize
Ryan Palacheck
Hady Perez
Matt Potok

Table of Contents

Volume I Final Report......2

1.1 EXECUTIVE SUMMARY......4
1.2 BUSINESS CASE......4
1.3 WIRELESS CAMERA UNIT......6
1.4 PROXY SERVER......9
1.5 REMOTE VIEWING SYSTEM......10
1.6 SYSTEM INEGRATION......12
1.7 CONCLUSION......14

Volume II Product and Process Documentation......16

Volume III Acceptance Test Results and Report......23

Volume IV User-Guide......30

Volume V Deliverables......36

Section 1.1 Executive Summary

United States Special Operations Command (SOCOM), based out of Tampa, Florida has sponsored a project for this year's Integrated Product and Process Design program. SOCOM’s goal was to develop a surveillance system in which an officer can view images from virtually anywhere in the world. Through this vision an officer can enter an area in which surveillance is desired, plant a Wireless Camera Unit (WCU), and retreat to safety.

The WCU consists of four main components which include the Axis 2120 Network camera, TINI microprocessor, Motorola i95 SmartPhone, and a 12 volt battery. Once the WCU is established, its objective is to capture images and successfully relay them to a proxy server.

The main goals of the proxy server are to capture the images from the WCU, store an archive of recent images, and create a Graphical User Interface (GUI) to allow viewing in remote locations. Once the image is stored on the proxy server it is available for view. The images can now be viewed by an Internet connection.

SOCOM wanted a light-weight, portable device which an officer could use to obtain these images. The Remote Hawkeye design team chose the Ipaq 3670 handheld device for viewing them. Once the Ipaq establishes a connection via General Packet Radio Service (GPRS), it can now view images day or night. The GUI also contains features such as album or streaming modes. When streaming mode is selected the officer can receive the most recent images, approximately every 10 seconds. Album mode allows the officer to scroll through images taken previously. Another option allows the user to shut down the system to conserve battery power. The last option contained on the GUI allows the officer to e-mail an image with an attached message. The integration and implementation of this cutting edge technology has proven successful in accomplishing SOCOM’s major goals for this project.

Section 1.2 Business Case

Below is a list of the components used for completion of the working prototype.

Component Costs

Ipaq 3670 Pocket PC	$1,100
Axis 2120 Network Camera	$1,000
Motorola i85 Smartphone	$ 500
GPRS Modem	$ 350
Wireless Network Cards	$ 150
Miscellaneous	$ 100
TINI Microprocessor	$ 50
12 Volt Battery	$ 30
Battery Charger	$ 27

The prototype can be broken down into 3 subcategories. Each subcategory contains a sub-total.

Wireless Camera Unit

Axis 2120 Network Camera	$1,000
Motorola i85 Smartphone	$ 500
TINI Microprocessor	$ 50
12 Volt Battery	$ 30
Battery Charger	$ 27
Sub-Total	$1,607

Remote Viewing System

Ipaq 3670 Pocket PC	$1,100
GPRS Modem	$ 350
Sub-Total	$1,450

Proxy Server

Wireless Network Cards	$ 150
Miscellaneous	$ 100
Sub-Total	$ 250

Grand Total **$3,307**

Comparison Case

To build a complete working prototype it cost approximately $3,307. This product was built as a proof of concept for USSOC. The main objective is to get this working prototype in the field to better survey a potential harm. Therefore, this product will be used internally and has the potential to save lives. This product is not intended to be re-sold therefore a comparison study has been substituted for a typical business case.

The Remote Hawkeye can be compared with Unmanned Aerial Vehicles (UAV's). Both UAV's and the Remote Hawkeye provide surveillance images, relay images to a remote location, and do not require officers to be present to operate. One benefit of a UAV is the ability to travel over land. However, the cost is significantly higher than the Remote Hawkeye, as seen below. Thus, many working systems could be set up to cover a vast region for only a fraction of the cost.

Unit Costs

Unmanned Aerial Vehicle	$10,000,000
Remote Hawkeye	$ 3,307

(Approximately 3,000 Remote Hawkeye's can be produced for every 1 Unmanned Aerial Vehicle.)

Section 1.3 Wireless Camera Unit

The Wireless Camera Unit, or WCU, is the surveillance system of the Remote Hawkeye. It consists of a network camera (Axis2120), a microcontroller (Dallas Semiconductor's TINI), and a SmartPhone (Motorola's i95). The Axis2120 camera serves as the image generator and compressor. It is fully configurable to take JPEG images in either a streaming mode (approximately 25 frames per second) or a periodic mode (the period is customizable by the user). The camera can also utilize the File Transfer Protocol (FTP) to transmit its JPEGs to another device on its network via Fast Ethernet (100 Mbps). The TINI microcontroller acts as the system controller. It receives the images from the Axis2120 camera via Ethernet and sends this image through its serial port to the SmartPhone at a data rate of 115 Kbps. It also receives control commands from the Proxy Server (relayed by the SmartPhone) in order to support functions such as zoom, pan, tilt, and sleep-mode. The i95 SmartPhone is the transmission device of the WCU. It takes images from the TINI microcontroller, packetizes them into User Datagram Protocol (UDP) packets, and then sends these packets via Integrated Dispatch Enhanced Network (iDEN) to the proxy server at a data rate of 40 Kbps. The phone also receives commands from the proxy server and relays them to TINI through their serial connection.

Technical Details

The Axis2120 camera is a sophisticated COTS network camera with a built-in web server running a very reliable Linux-based operating system. The camera is highly customizable in the aspects of image viewing, image size, and compression level. It also supports remote data transfers through FTP and motion detection algorithms. It has 16 MB of DRAM, 4 MB of Flash, an Artpec-1 compression chip, and an Etrax 100, 32-bit RISC processor. The Axis2120 weighs 0.25 Kg, has a height of 57 mm, a width of 86 mm, and a length of 183 mm, and can capture 24-bit color images at illumination levels from 1 to 5,000 lux.

For the Remote Hawkeye, the camera was configured to capture an image and FTP it to TINI in two-second intervals. The image size was chosen to be 352 x 288 with a compression level of "medium." This resulted in high-detailed, relatively compact JPEGs of less than 8 KB each. No other customizations were done on the camera.

The TINI microcontroller is a compact, web-server-capable system that runs a reliable Unix-style operating system. It is supports Java on start-up and FTP and Telnet access. TINI is equipped with two serial ports, one Ethernet port, 512 KB of nonvolatile SRAM, 512 KB of Flash, and a DS80C390 processor running at 36.864 MHz.

TINI serves as the intermediate step between the phone and camera. The Axis camera can only store data remotely via FTP or email. Since the SmartPhone does not support either, TINI must be used. TINI receives the image via FTP and stores it into a user-created directory called *hawkeye*. The camera stores the image as *image.jpg* and uses an over-writing scheme for all proceeding transfers. As a result, only one file is actually stored on TINI and the *FileToSerialVer0.java* program running on TINI can use a simple, new-file detection scheme, such as the deletion of the file and then testing for its existence. One inherent problem with this scheme is the camera overwriting the image file on TINI while the image is being prepared for transmission to the SmartPhone. Luckily, even though the camera is setup to overwrite the image file every two seconds, it is not able to achieve this until the image file is deleted. The other inherent problem is sending the image to the SmartPhone before it is written in its entirety by the camera. This problem occurs since the image.jpg file is created when the very first byte is FTPed to TINI, and the program is just checking for the file's existence, not the entire file. In order to avoid this, the JPEG protocol is utilized. Every .jpg file follows the JPEG protocol in which the end of the file is defined by two consecutive bytes, FF and D9. Therefore, before TINI sends a file to the SmartPhone, it buffers the image while checking for the "End-of-the-file" bytes. Once they are detected, transmission of the file can begin. The Hawkeye program running on TINI utilizes a Thread that listens for commands sent on the serial line. These commands are one byte in length and allow TINI to initiate a reaction to the command accordingly. Only two commands (Send-File-Size and Send-Image-Data) are presently supported by TINI, however further commands are easily implemented by adding to a switch statement in the Java code and a global Boolean variable which allows the main program to react to the command.

The Motorola i95 SmartPhone is a sophisticated cellular phone that runs on a KVM platform and supports Java2 Micro Edition (J2ME) programs. The phone is equipped with 256 KB of data memory, 384 KB of program memory, and 256 KB of heap memory. It also has an 8-bit color display capable of displaying Portable Network Graphics (PNG) files. A serial connection is provided that can support transfers up to 115200 Kbps. The network interface is iDEN, which has a data rate of 40 Kbps. Protocols, such as UDP and TCP/IP, are supported over its network connection.

The SmartPhone is the transmitter of the system. It is the device that actually sends the images to the proxy server so that the RVS can view them. Since the phone is the slowest device in the WCU, it initiates all data transfers to the proxy server. For example, if the SmartPhone is ready to send an image, it sends the Send-File-Size opcode to TINI and then waits for a reply from TINI. TINI's reply will be the file size and the SmartPhone instantiates a byte buffer of that size. Once the buffer is ready to hold the data, the SmartPhone sends the Send-Image-Data opcode to TINI and the data transfer commences. Like TINI, the SmartPhone utilizes the JPEG protocol's "End-of-the-file" bytes in order to ensure the entire file has been sent. Once the image file is buffered, it is packetized into 1000 byte packets. Each packet is then sent, one-by-one, to the proxy server via UDP. After the last packet is transferred, the process begins anew to send the next image to the server. Below is a diagram that illustrates the entire process taken by the WCU to send one image to the proxy server. One limitation encountered during the development of the MIDlet code on the SmartPhone was the use of a bi-directional UDP socket. Due to this, the control aspect of the project could not be completed, however, the code to support this is incorporated in the final, working version.

Command	**Opcode (one byte in size)**
Send-File-Size	0x41
Send-Image-Data	0x42

Table 1.1

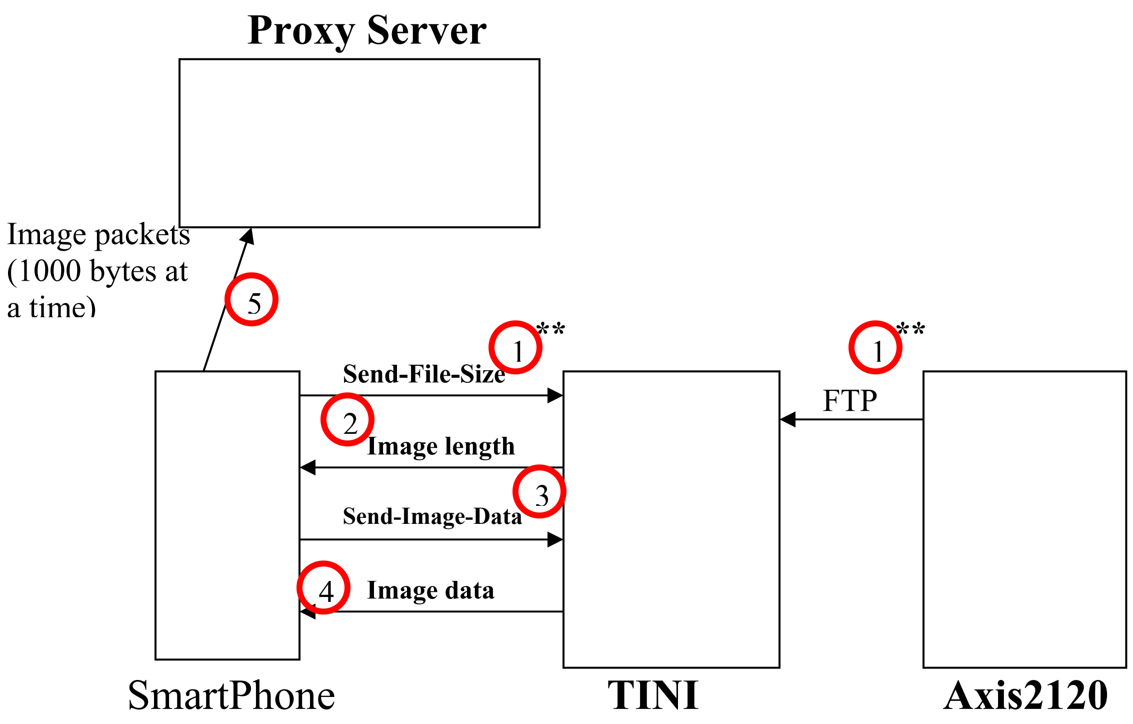


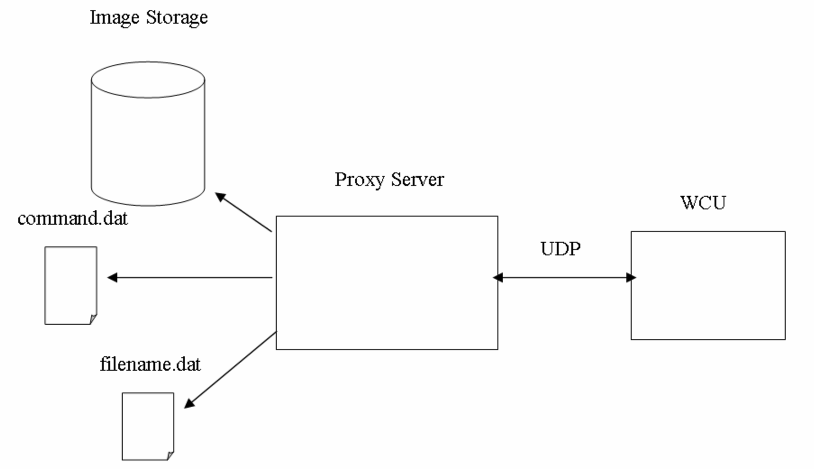
Figure 1.2
**These transfers can be performed in any order or concurrently.

Section 1.4 Proxy Server

The Proxy Server, or PS, serves as the intermediary between the Wireless Camera Unit (WCU) and the Remote Viewing System (RVS). It is a software program written in java and it is able to run on any computer that has internet connection. The Proxy Server communicates with The Wireless Camera Unit using User Datagram Protocol (UDP). It listens for the data packets sent from the Wireless Camera Unit, then manipulates the received data and store it as image files in the hard drive of the machine where the Proxy Server is located. The image files are stored in JPEG format and they are named by the order they have been received. Furthermore, the Proxy Server records the name of the latest received image to a file, and this file is used by the Remote Viewing System to locate the latest image and display it to the user. The last function of the Proxy Server is to relay control commands that were made by the Remote Viewing System to the Wireless Camera Unit. The Proxy server constantly checks commands that are written in a file by the Remote Viewing System, reads in the commands and sends them to the Wireless Camera unit. However, this functionality is not implemented in the current build of the project due to complications on the Wireless Camera Unit.

Technical Details

The Proxy Server consists of two modules; the main module is responsible for image reception and storage, and the second module handles commands forwarding. Both modules use a same socket to communicate with the Wireless Camera Unit.
The main module listens for data on port 8136, it received 1k bytes of data at a time until the whole image is received. Then, the server stores the image and writes the name of the image to a file named "filename.dat". The second module checks the existence of a file named "command.dat" , if the file exists, the second module will read in the command from the file and send it to port 8136 of the Smart Phone (a component of the Wireless Camera Unit).



Section 1.5 Remote Viewing System

The Remote Viewing System or RVS is the front end system of the Remote Hawkeye. It consists of an IPAQ handheld pc, which the officer is going to use on the field to view images taken by the Wireless Camera Unit or WCU. The IPAQ is equipped with a GPRS modem to access the Proxy Server. Once a connection is established between the IPAQ and the Proxy Server, it accesses jsp files on the server that are used to generate the html files that make the User Interface.

The user interface on the RVS has two modes of operation. Once you log in to the system you can either stream current images, view past images stored on the image archive, or send an email alert with a relevant image to an officer higher up in the chain of command. When using the streaming mode, a file named imageFeed.jsp is called. This file gets the last image stored on the Proxy Server and generates an html file to display this image. When using the Album Mode, a file named imageFeedAlbum.jsp is called. This application allows the user to browse archived images. Another function of the interface is its ability to send emails with the attached images requested by the user. When the user clicks on the icon on the interface to send an email it calls sendmail.html. This file consists of an email form where you specify the email address of the recipient, the email subject and a short message. When the user clicks on the send button, it calls sendmail.jsp. This file processes the email using the Javamail library and generates an html page that notifies the user that the message was sent successfully.

There are some functions of the Remote Hawkeye that are considered Phase 2 including pan, tilt, and zoom controls. Although these functions have not yet been implemented, user controls for these features have been built into the user interface. This was done in order to ease the addition of these new functions.

Technical Details

The IPAQ is a handheld pocket pc. It has a windows operating system and supports programs such as Internet Explorer, email programs, etc. The IPAQ has a 206 MHz Intel StrongARM 32-bit RISC Processor, TFT LCD display, and 64MB RAM. The maximum image resolution is 240 X 320. With the use of the optional expansion slot, it supports GPRS modems and 802.11b wireless cards.

Three Java Server Pages (jsp) scripts run on the server and control the user interface. The jsp file that handles the email processing is sendmail,jsp. This file is called when a user clicks on the submit button on the email form. The function this file has is the following: 1) it sets the smtp server your going to use. 2)it sets all the email parts such as recipient, subject and message using the information gathered from the email form. It also gets the image from the file and attaches it to the message. It then gets all the parts of the message and compiles it in one big structure and then sends the message. This is all done using the javamail library.

Two other scripts, imageFeed.jsp and imageFeedAlbum.jsp, handle loading pictures from the image archive. ImageFeed.jsp reads the most recent image name from filename.dat and displays it every 3 seconds. ImageFeedAlbum.jsp initially loads the most recent picture. It then allows the user to browse through the image archive. This is done by manipulating a variable called currentImage in imageFeedAlbum.jsp. This variable is passed in the querystring each time the picture is changed.

Section 1.6 System Integration

Research and COTS Decision Making Stage:

In order to integrate Commercial Off The Shelf (COTS) hardware with custom software research of available products and technologies was necessary. The camera had to comply with the customer's requirements as much as possible. The Axis2120 camera was chosen because it is able to output JPEG files instead of a proprietary format. Also, the camera we chose needed to output images through ethernet or serial. Using ethernet, which is standard for most network cameras, was possible because we had the TINI microcontroller to convert ethernet to serial and handle the image buffering. The Axis2120 lacked pan and tilt capability. While other models of Axis cameras offered these features, they were determined to be too expensive for a proof of concept project.

The decision to use the TINI board was made while researching the cameras, since cameras that output serially (i.e. directly to the phone) were not available. Since the images needed to be first sent to the phone serially and in a discernable format, deciding the role of TINI and the capabilities of the camera were decisions that relied on each other.

The Motorola i95 SmartPhone was an obvious choice to transmit images wirelessly. Higher throughput technologies such as 802.11b wireless Ethernet could not cover the range that was necessary. The i95, which uses Java technology, was the driving force behind using Java technology as our middleware. TINI, which also uses a subset of Java, simplified integration of the components on the WCU.

The decision to use Java technology led us to explore similar options on the Remote Viewing System and Proxy Server. A Servlet or Java program seemed the most logical choice to accept the posting of the image. Similarly, using Java Server Pages and Servlets were chosen to drive the user interface.

Once the details of the System Architecture of the Remote Hawkeye were decided (hardware and software), the next stage of development was to acquire the necessary hardware and middleware and begin prototyping.

Component Usage:

This stage's primary goal was to get the group acquainted with the software and hardware and to start prototyping. First level prototyping involved simulating one task at a time. Using telnet and FTP, we were able to store images on TINI and set the camera up to FTP an image to TINI.

The code for TINI included two major components: receiving an image and sending an image. To perform each of these prototypes, TINI received of an image from a computer via ethernet, and TINI sent an image to a computer via serial interface.

The first phase of the phone software development was to implement one way transmission of a Portable Network Graphic (PNG, which unlike JPEG is viewable on the phone). This was done using a midlet that fetched the image from a website. A separate prototype included having the phone serially send text to and receive text from a computer.

The Proxy Server's first task was to grab an image from a website and store it on disk. The Remote Viewing System's (dynamic web content on the same server viewed by the iPaq) first prototype was to configure a user interface and to fit that interface on the display of the iPaq.

Component Testing – Single Pipeline:

At this point the prototyping was split into two autonomous divisions, WCU to PS transmission and the RVS. This was because of the independence of these two pieces of the system.

Sending one image from the WCU to PS was essential to the Remote Hawkeye's design. Therefore, it involved integrating each of the WCU components with the PS. First, TINI's code was integrated into a multithreaded serial to ethernet conversion program. The camera was integrated as the FTP host to TINI and the camera-TINI-computer section of pipeline was complete.

The next integration prototype for the smartphone was to fetch an image from an Internet site and send it to our server. We developed a test program to download a file from a specified web server and send it to the proxy server. This was successful leaving serially reading an image from TINI to the smartphone as the only missing portion of the pipeline. In order to execute the transmission properly, we created a simple protocol between TINI and the smartphone, which allows the phone to receive an image on demand and for TINI to tell the smartphone the size of the image before sending it.

In parallel to the above development, a user interface was created for the iPAQ. This interface was generated by jsp scripts on the server. Alert email functionality was also added.

Once each segment of the transmission worked properly we began Alpha testing (first working product). In the Alpha testing phase, we encountered problems in the SmartPhone to TINI prototype, problems with maintaining SmartPhone service, and port conflict problems.

Component Integration – Continuous Pipeline:

Once the Alpha Testing was completed, the software was modified to support multiple image transmissions. Further development allowed continuous transmission of images from the WCU to the PS.

The initial Proxy Server model was replaced by a server application that listens to a specific port and receives data. During this stage, the PS application was integrated with the RVS. The server program read images continuously and wrote the name of the image files to "filename.dat." This integration enabled the RVS to refresh the most current images and to scroll through old images in album mode.

Psuedo-streaming and album mode were then implemented on the RVS. Once each component of the system was able to shoulder the load of multiple images, Beta testing began.

Beta testing has included integrating TINI and the camera with a battery and placing the WCU in a protective container. Testing cellular service, battery life, data throughput, and the user interface verified the system's functionality.

The Remote Hawkeye, now in Beta testing, can be upgraded and refined in several ways. Some potential improvements include a refined file storage system for a large number of files to simplify album mode, and system start and stop capability.

Section 1.7 Conclusion

The Remote Hawkeye represents the culmination of two semesters of planning, research and development. Its goal was to provide SOCOM with a wireless surveillance system using cellular technologies. The team members utilized effective planning, research, design principles, communication, teamwork, and cost analysis to accomplish this objective. Although some features of the system were left for future expansion, the Remote Hawkeye accomplished its core objectives and is therefore a success.

Cellular Internet technologies are expanding rapidly and their role in future defense projects will only expand. Cellular communication will offer defense agencies an alternative to satellites and wireless LANs. The Remote Hawkeye represents a step in this direction.

The Remote Hawkeye

Volume II
Product and Process Documentation

Version 1.0
April 16, 2002

Integrated Product and Process Design
University of Florida

United States Special Operations Command

Dr. Sumi Helal

Eric Grobelny
Doug Holubek
Fenghai Lan
Gregg Mize
Ryan Palacheck
Hady Perez
Matt Potok

Table of Contents

Volume I Final Report **2**

Volume II Product and Process Documentation **16**

2.1 PRODUCT SUBSYSTEM AND COMPONENT SUBSYSTEM 18
2.2 HARDWARE COMPONENT SPECIFICATIONS 18
2.3 ARCHITECTURE COMPONENT SPECIFICATIONS 18
2.4 SOFTWARE COMPONENT SPECIFICATIONS 19
2.5 BILL OF MATERIALS 19
2.6 PROCESS SPECIFICATIONS 20
2.7 IMPACT ASSESSMENT 21

Volume III Acceptance Test Results and Report **23**
Volume IV User-Guide **30**
Volume V Deliverables **36**

Section 2.1 Product Subsystem and Component Subsystems

The Remote Hawkeye's design consisted of three major subsystems: The Proxy Server, Remote Viewing System and the Wireless Camera Unit. Each of the subsystems can be broken down into their hardware and software components. This hierarchical level of design translated into well defined product development stages. Consequently, a close relationship between the product and process was achieved. The product development stages include: Hardware Component Specifications, Architecture Component Specifications and Software Component Specifications.

Section 2.2 Hardware Component Specifications

One of the system requirements for the Remote Hawkeye was to develop the system using Commercial Off The Shelf technology. This meant that software design decisions were dependent on the hardware chosen to build the system. Therefore, the functionality of the hardware components had to be determined and planned before purchasing them. This meant analyzing the advantages and disadvantages of the hardware and the kind of software functionality that could be designed for them.

Section 2.3 Architecture Component Specifications

When designing the overall system architecture two different models were used. The first was a peer to peer transfer between the Wireless Camera Unit and the Remote Viewing System. This model fit the basic customer requirements of having an officer receive images from a surveillance system wirelessly, but it had some limitations. A handheld device has limited storage capability, therefore limiting the amount of images that could be viewed. Further design analysis was needed.

The Proxy Server model eliminated the problem of persistent storage of a large number of images. It also allowed for a more intuitive user interface. The image capacity is now limited by the available disk space on the Proxy Server. The user interface now consists of dynamic web content that can be viewed by the Remote Viewing System as well as any other device with a graphical web browser and an Internet connection.

Section 2.4 Software Component Specifications

Most of the Remote Hawkeye's development phase consisted of writing the custom software. Java technology was used since it the dominant language for the SmartPhone and it could be integrated seamlessly with TINI and the Proxy Server, because both are originally designed to be web servers. There are a total of six software programs that make up the custom software on the Remote Hawkeye: two JSP programs, one Java application, one MIDlet and one TINI Java program.

Section 2.5 Bill of Materials

The following is a list of components of the system. Several components are listed that are simply development tools or that are not yet available. These include the following:

- GPRS modem (not readily available)
- The wireless network cards (aided process development and communication)
- An estimation of miscellaneous expenses.

Component	Cost
Ipaq 3760	$1,100
Axis 2120 Network Camera	$1,000
Motorola i95 SmartPhone	$500
GPRS Modem	$350
Wireless Network Cards	$150
Miscellaneous	$100
TINI Microprocessor	$50
12 Volt Battery	$30
Battery Charger	$27
TOTAL	$3307

Section 2.6 Process Specifications

The process of designing the Remote Hawkeye was accomplished with a Waterfall Model of development. This model explains how we treated each phase in the development as separate processes. Below is a diagram, which describes the development process of the project:

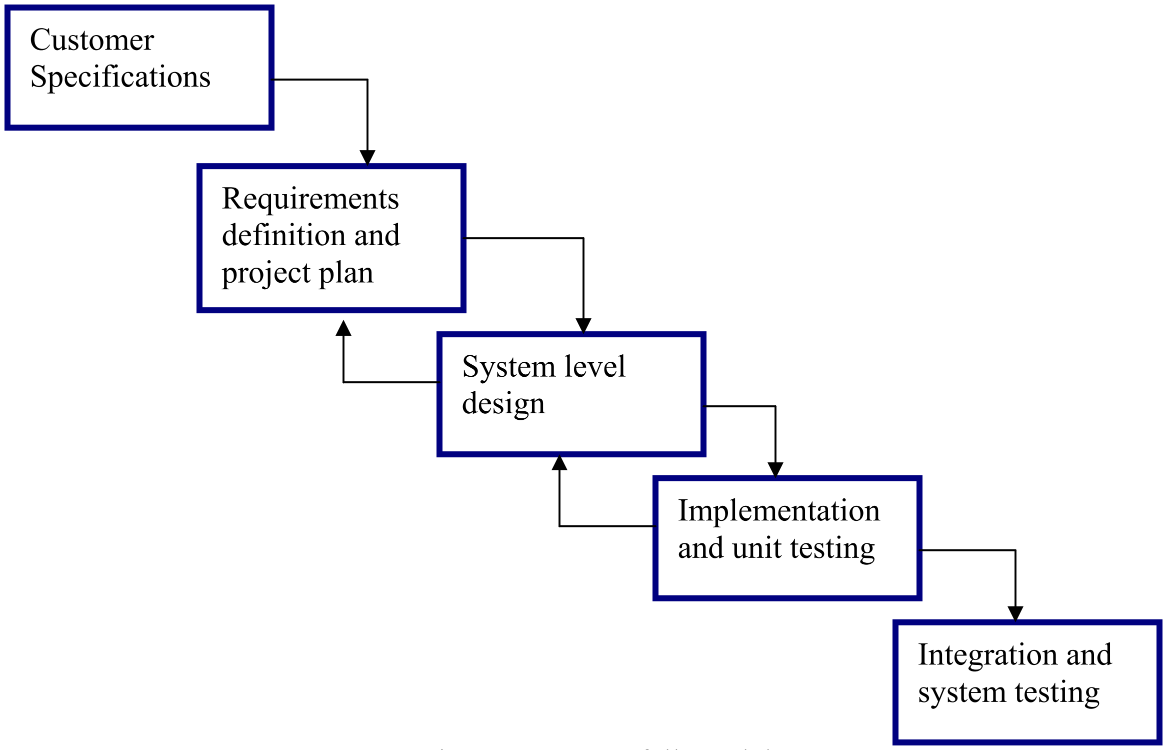

Customer Specifications
Requirements definition and project plan
System level design
Implementation and unit testing
Integration and system testing

Figure 2.1 Waterfall Model

A more detailed specification of each process is as follows:

1) <u>Customer specifications</u>. The first task that was accomplished was to travel to Tampa and meet with SOCOM. At the meeting, the customer specifications were defined. The scope and goals of the project were also defined.

2) Requirements definition and project plan. The requirements enabled the software requirements to be developed. This phase involved learning and researching available technologies within the scope of project that would satisfy the software requirements. These requirements allowed for the creation of a project plan, which would serve as a guide for the next three phases of development.

3) System level design. In this phase, the design document, prototype plans, and quality assurance and test plans were documented. An analysis and evaluation of the requirements definition and project plan was made. The System Level Design Report served as the model for the next two phases to follow.

4) Implementation and unit testing. This consisted of designing and building the software components of the system. The software components of the system were further broken down into incremental development phases.

5) Integration and system testing. In this phase of development, the Remote Hawkeye's components were refined and tested before they were integrated together to form a working product. Once the final product was developed, verification and quality tests were performed on the system.

Manufacturing Plans

The system consists of Commercial Off The Shelf hardware and custom software. This design model eliminates any manufacturing.

Section 2.7 Impact Assessment

The benefits from the Remote Hawkeye system include: large scalability, low price, a wider range of use, and potentially saved lives. The system can be scaled in several ways. First, it can be integrated with other moving devices such as tactical mobile robots. Second, it could be refined for any number of phase 2 features. Lastly, it can be easily integrated with rapidly advance cellular Internet technologies, like G3. The price of the system, approximately $3,300, is much lower than advance surveillance systems such as Unmanned Aerial Vehicle ($10,000,000). Currently, tactical mobile robots use wireless Ethernet, which has a high throughput and short range. The Remote Hawkeye's range extends to anywhere where there is cellular service for the Remote Viewing System and the Wireless Camera Unit. This range allows for the officer to be much farther away from a potentially hazardous situation and therefore could potentially save lives.

Volume III
Acceptance Test Results

Version 1.0
April 16, 2002

Integrated Product and Process Design
University of Florida

United States Special Operations Command

Dr. Sumi Helal

Eric Grobelny
Doug Holubek
Fenghai Lan
Gregg Mize
Ryan Palacheck
Hady Perez
Matt Potok

Table of Contents

Volume I Final Report **2**

Volume II Product and Process Documentation **15**

Volume III Acceptance Test Results and Report **23**

3.1 INTRODUCTIOIN 24
3.2 ACCEPTANCE TESTS 25
3.3 TEST RESULTS 27
3.4 SUMMARY 29

Volume IV User-Guide **20**

Volume V Deliverables **36**

Section 3.1 Introduction

This report describes the acceptance testing of the Remote Hawkeye system. Acceptance testing was used to determine the extent to which the final Remote Hawkeye design met the basic Customer Requirements and System Specifications gathered at the beginning of the design process. These requirements and specifications are listed below.

This report is organized into three major sections. The first section summarizes the completed acceptance tests. The second section provides more detail on the results of the tests as related to each of the customer requirements. Finally, the third section presents a summary of the project's progress.

Requirement ID	Customer Requirement
<R1>	Capture image
<R2>	Transfer image wirelessly to remote viewing system
<R3>	Transfer image to the Internet
<R4>	Provide user interface on remote viewing system
<R5>	Allow camera to be controlled by RVS
<R6>	Night vision
<R7>	Durable
<R8>	Long battery life
<R9>	Sleep mode/Power management
<R10>	Motion detection

Table 3.1

Specification ID	Specification Description	Requirement(s) Fulfilled
<S1>	Capture image 640x480 JPEG	<R1>
<S2>	Transfer image to proxy < 10 sec.*	<R2>, <R3>
<S3>	Image from proxy to RVS < 5 sec.*	<R2>
<S4>	GUI – Image Viewing (8bit color, 80x60)	<R4>
<S5>	GUI – Camera Control	<R4>, <R5>
<S6>	GUI – Image Request	<R4>
<S7>	GUI – Sleep Mode Control	<R4>, <R9>
<S8>	Camera – Zoom (x4) < 3sec. *	<R5>
<S9>	Camera – Pan (90°) <3sec. *	<R5>
<S10>	Camera – Tilt (30°) < 3sec. *	<R5>
<S11>	Capture image - Illumination = 3lux	<R6>
<S12>	Withstand 1ft. drop	<R7>
<S13>	Weatherproof	<R7>
<S14>	Power for 5 hours of full operation	<R8>
<S15>	Power for 36 hours in sleep mode	<R9>
<S16>	Detect moving object < 45mph	<R10>
<S17>	Complete system wake-up in < 5 sec	<R9>
<S18>	Images Stored on PS >= 1 GB	<R3>

Table 3.2

Section 3.2 Acceptance Tests

Overview

Four tests were designed to efficiently and completely test the extent to which the final Remote Hawkeye design met the basic Customer Requirements and System Specifications gathered at the beginning of the design process. Unfortunately, some of the project's specifications remain untested. The following table displays the untested specifications and the reasons that they were not tested.

Specification ID	Specification Description	Reason Not Tested
<S3>	Image from proxy to RVS < 5 sec.	GPRS modem not yet available. A test using the 802.11b card used for development would not be indicative of actual performance.
<S8>	Camera – Zoom (x4) < 3sec.	Phase 2
<S9>	Camera – Pan (90°) <3sec.	Phase 2
<S10>	Camera – Tilt (30°) < 3sec.	Phase 2
<S12>	Withstand 1ft. drop	Due to the high cost of components, we did not test the durability of the system.
<S13>	Weatherproof	Due to the high cost of components, we did not test the systems ability to function in various weather conditions.
<S15>	Power for 36 hours in sleep mode	Phase 2
<S16>	Detect moving object < 45mph	This specification was provided to the team to give a rough guideline as to the expected refresh rate of the system. This specification was not tested because the success of a test would depend on how far from the moving object that the WCU was placed.
<S17>	Complete system wake-up in < 5 sec	Phase 2

Table 3.3

Description of Tests

The tests described below were conducted to demonstrate the extent to which the system meets basic requirements. The final Remote Hawkeye design was used for all tests.

Acceptance Test ID	**Name**	**Description**
<AT1>	Full System Test	The system was started and its behavior was observed. Observations included: ▪ Average interval between images arriving on the server = 4 sec ▪ GUI setup ▪ Successful streaming of images ▪ Successful browsing of archived images
<AT2>	E-mail Alert Test	The design team was not able to configure the main development server to support two Java APIs necessary for the e-mail alerts to function. However, this functionality was tested on a local version of the tomcat server. The alert button was pressed and the e-mail form was completed. The e-mail arrived at the specified e-mail address with the current image attached.
<AT3>	Low Light Test	The system was run at night. The camera unit was placed outside and resulting image quality was observed.
<AT4>	Power Test	All WCU batteries were charged to full capacity and the system was started. The system stayed at full operation for ______. It was determined that the phone's battery dies first. Unfortunately, no other batteries are available since the i95 is not yet on the market.

Table 3.4

Section 3.3 Test Results

Overview

This section describes the results of the acceptance tests as related to each of the customer requirements.

<R1> Capture Image

It was determined that the system met this requirement based on the results of <AT1>.

However, the image resolution specification, <S1>, was not met. The specifications call for 640 x 480 JPEGs while the system output is 352 x 288 JPEGs.

<R2> Transfer image wirelessly to remote viewing system

It was determined that the system met this requirement based on the results of <AT1>.

Additionally, the system meets the specification for image transfer rate from the camera unit to the server, <S2>. The specification calls for transfer in under 10 seconds and the system transfers images every 4 seconds on average.

As previously mentioned, the rate of transfer from the server to the viewing system was not tested.

<R3> Transfer image to the internet

It was determined that the system met this requirement based on the results of <AT1> and <AT2>. All images are immediately streamed to a website and the user has the ability to send images via e-mail if he or she chooses.

<R4> Provide user interface on Remote Viewing System

It was determined that the system met this requirement based on the results of <AT1>.

The user interface includes an image viewing area, mode select buttons, zoom controls, an alert button, pan controls, tilt controls, a start system button, and a stop system button. This interface meets specifications <S4>, <S5>, <S6>, and <S7>.

<R5> Allow camera to be controlled by RVS

This functionality is now considered Phase 2. There has been progress made in this area, however. The user interface includes RVS controls and control channel design has begun on the server and camera unit.

<R6> Night Vision

This functionality is now considered Phase 2.

However, the system was able to capture images in relatively low light conditions as demonstrated by <AT3>.

<R7> Durable

This requirement was not tested as described above.

<R8> Weatherproof

This requirement was not tested as described above.

<R9> Long battery life

This requirement was tested by <AT4>. Although the system did not meet this requirement, the only component that dies in less than 9 hours is the phone. Unfortunately, the phone is not yet on the market and no power alternatives were found.

Therefore, the system fails to meet <S14> although all areas where the design team had control do meet this specification.

<R10> Motion Detection

This requirement is now considered Phase 2.

Section 3.4 Summary

The following tables summarize the extent to which the final Remote Hawkeye design meets basic requirements and specifications:

Requirement ID	Customer Requirement	Status
<R1>	Capture image	PASS
<R2>	Transfer image wirelessly to remote viewing system	PASS
<R3>	Transfer image to the Internet	PASS
<R4>	Provide user interface on remote viewing system	PASS
<R5>	Allow camera to be controlled by RVS	Phase 2
<R6>	Night vision	Phase 2
<R7>	Durable	Phase 2
<R8>	Long battery life	FAIL
<R9>	Sleep mode/Power management	Phase 2
<R10>	Motion detection	Phase 2

Table 3.5

Specification ID	Specification Description	Requirement(s) Fulfilled	
<S1>	Capture image 640x480 JPEG	<R1>	FAILED
<S2>	Transfer image to proxy < 10 sec.*	<R2>, <R3>	PASS
<S3>	Image from proxy to RVS < 5 sec.*	<R2>	NOT TESTED
<S4>	GUI – Image Viewing (8bit color, 80x60)	<R4>	PASS
<S5>	GUI – Camera Control	<R4>, <R5>	PASS
<S6>	GUI – Image Request	<R4>	PASS
<S7>	GUI – Sleep Mode Control	<R4>, <R9>	PASS
<S8>	Camera – Zoom (x4) < 3sec. *	<R5>	NOT TESTED
<S9>	Camera – Pan (90°) <3sec. *	<R5>	NOT TESTED
<S10>	Camera – Tilt (30°) < 3sec. *	<R5>	NOT TESTED
<S11>	Capture image - Illumination = 3lux	<R6>	NOT TESTED / PASS
<S12>	Withstand 1ft. drop	<R7>	NOT TESTED
<S13>	Weatherproof	<R7>	NOT TESTED
<S14>	Power for 5 hours of full operation	<R8>	FAILED
<S15>	Power for 36 hours in sleep mode	<R9>	NOT TESTED
<S16>	Detect moving object < 45mph	<R10>	NOT TESTED
<S17>	Complete system wake-up in < 5 sec	<R9>	NOT TESTED
<S18>	Images Stored on PS >= 1 GB	<R3>	^*

^*Depends on server used. Development server did not have 1 GB available, but the image archive size will be determined by the customer.

Table 3.6

Volume IV
User-Guide

Version 1.0
April 16, 2002

Integrated Product and Process Design
University of Florida

United States Special Operations Command

Dr. Sumi Helal

Eric Grobelny
Doug Holubek
Fenghai Lan
Gregg Mize
Ryan Palacheck
Hady Perez
Matt Potok

Table of Contents

Volume I Final Report..2

Volume II Product and Process Documentation..15

Volume III Acceptance Test Results and Report..21

Volume IV User-Guide..29

4.1 INTRODUCTIOIN..32
4.2 WIRELESS CAMERA UNIT..33
4.3 PROXY SERVER..34
4.3 REMOTE VIEWWING SYSTEM..35

Volume V Deliverables..36

Section 4.1 Introduction

This document serves as a user's manual for the Remote Hawkeye surveillance system. The system is broken down into three individual components, the Wireless Camera Unit (WCU), the Proxy Server (PS), and the Remote Viewing System (RVS). The remainder of this document will explain how to set up and configure each component so that the system becomes usable.

Section 4.2 Wireless Camera Unit

The WCU is made up of four individual parts, an Axis 2120 camera, a Dallas Semiconductor TINI microprocessor, Motorola's i95 SmartPhone, and a 12 volt battery. Shown below is a block diagram of how the WCU should be connected.

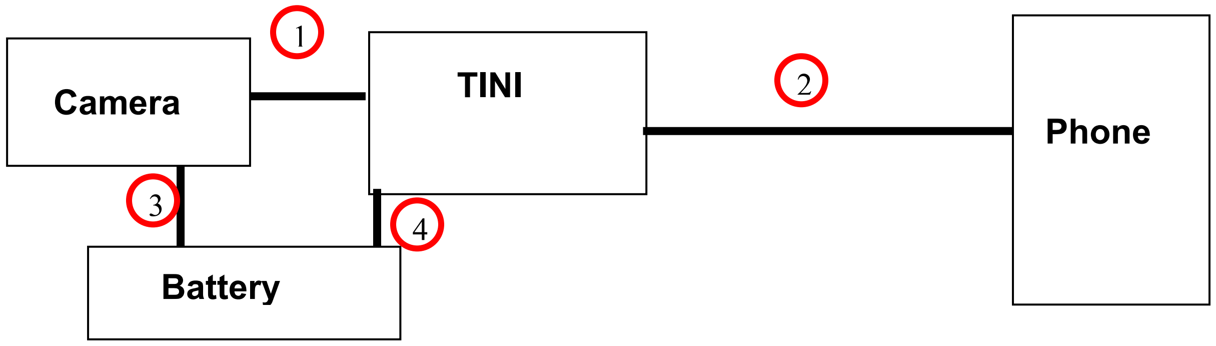


Figure 4.1

The following numbers correspond to the numbers above circles in red:

1. Connect the camera to TINI using a cross-over Ethernet cable.
2. Connect TINI to the SmartPhone through the serial port, using the cable provided by Motorola.
3. Connect the camera to the battery by connecting the black wire to the black wire, and the white wire to the white wire
4. Connect TINI to the battery by connecting the wire with writing to the white wire, and the wire with no writing to the black wire.

The camera and TINI are configured to begin running once they are powered on. Transmission of images will begin when the MIDlet on the SmartPhone is run. To run the MIDlet on the SmartPhone, follow these steps:

1. Power on the phone
2. Press Menu
3. Scroll down to Java Apps and press Select
4. Scroll down to hawkeye3 and press Run
5. Press the Call button to run the applet

Section 4.3 Proxy Server

The proxy server will serve as the main storage facility of this system. Below you will find the necessary steps that must be taken to get a server up and running:

1. Copy the source code of “HawkeyeServer.java” from the Hawkeye CD to the home directory of the machine.
2. Compile the “IPPDServer.java” using the following command

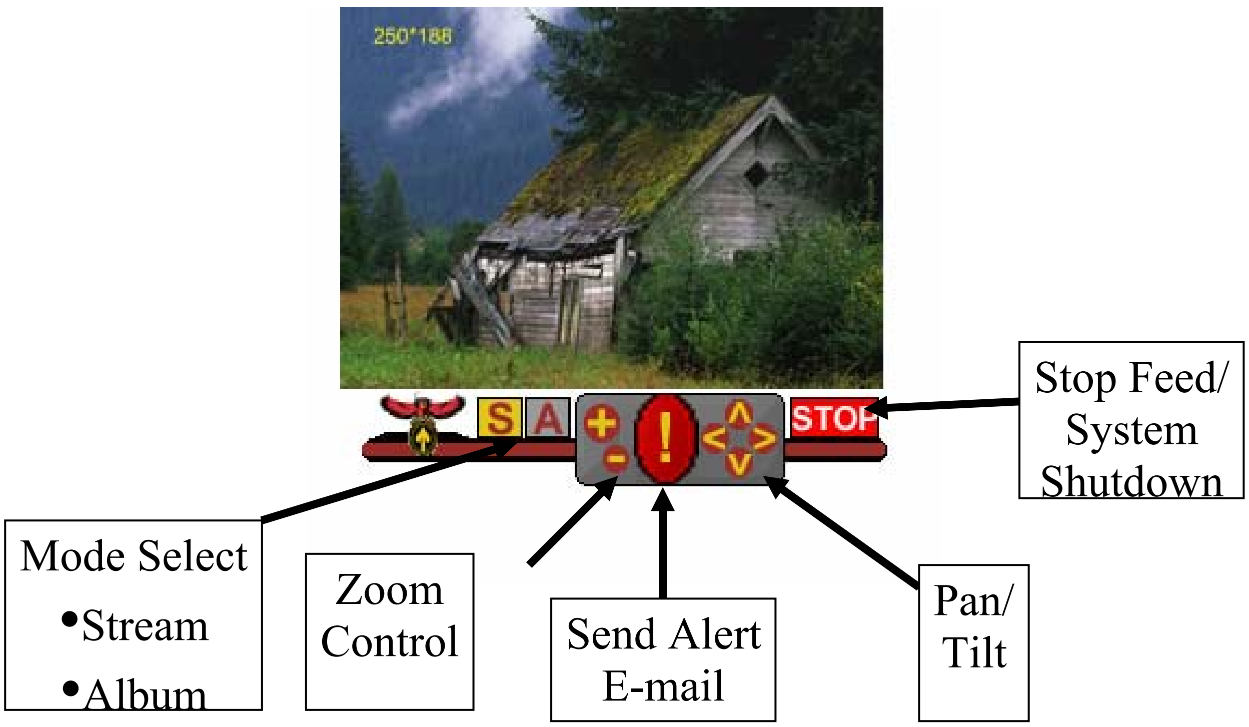
```
H:/javac IPPDServer.java
```

3. Copy the file “filename.dat” from the Hawkeye CD to the home directory where the “HawkeyeServer.java” locates.
4. Start the server by using the following command.

```
H:/java HawkeyeServer
```

Section 4.4 Remote Viewing System

The RVS serves as the user interface of the Remote Hawkeye surveillance system. To install the JSP files, copy the Hawkeye folder from the CD to the JSP directory of the Tomcat Server (Please consult your Tomcat Server installation manual for details). The interface is shown in the figure below:



The controls for the interface are described as follows:

1. **Mode Select:**
   a) Stream – allows the user to see the current image the WCU sends to the PS
   b) Album – allows the user to scroll through previous images

*2. **Zoom Control** – allows the user to zoom in and out, the "+" sign zooms the camera in, and the "-" sign zooms the camera out

3. **Send Alert E-mail** – allows the user to automatically send the current image as an e-mail attachment to a specified e-mail address.

*4. **Pan/Tilt** – allows the user to view left, right, up, and down. The directional arrow corresponds to the respective direction.

5. **Stop Feed/System Shutdown** – allows the user to stop the system by stopping the server from listening.

* Feature has not been implemented, but will be in Phase 2

The Remote Hawkeye

Volume V
Deliverables

Version 1.0
April 16, 2002

Integrated Product and Process Design
University of Florida

United States Special Operations Command

Dr. Sumi Helal

Eric Grobelny
Doug Holubek
Fenghai Lan
Gregg Mize
Ryan Palacheck
Hady Perez
Matt Potok

Table of Contents

Volume I Final Report **2**

Volume II Product and Process Documentation **15**

Volume III Acceptance Test Results and Report **21**

Volume IV User-Guide **29**

Volume V Deliverables **35**

PRELIMINARY PRODUCT DESIGN SPECIFICATIONS
CONFIGURATION MANGAGEMENT PLAN
RISK TABLE
SOFTWARE TESTABLE SPECIFICATIONS
SOFTWARE PRODUCT DESIGN
SYSTEM LEVEL DESIGN REORT

Preliminary Product Design Specifications

The Remote Hawkeye will be a surveillance system allowing remote observation of targets and network publication of frames deemed important.

The Remote Hawkeye System will consist of the following major components:

- Wireless Camera Unit (WCU)
- Remote Viewing System (RVS)
- Network Interface to View High Priority Frames

Commercial Off-the-shelf Products (COTS) will be used to produce this system. Each major component will be designed to the following specifications:

Wireless Camera Unit	• Weatherproof • Night Vision Capable • 200ft Range • Compression Capability • 500kB Storage • Communicator o Sends Image to RVS o Sends chosen images to net o Performs other interaction with RVS and Internet • Motion sensor to wake up system
Remote Viewing System	• Receive images from WCU • Allow user to select important images for publication to internet
Network Interface	• Interface allowing authorized internet user to view selected surveillance

Configuration Management Plan

Naming Conventions

General requirements and specifications will be labeled and managed as follows:

<R1>
<S1>

The Remote Hawkeye architecture will be divided into the following three parts:

- Wireless Camera Unit (WCU)
- Remote Viewing System (RVS)
- Network Interface (NI)

We will create a set of functions for each of these units in order to fulfill the general requirements and specifications. These will be named as follows:

Hardware:

<WCU.HF1>

Software:

<RVS.SF1>

Wireless Camera Unit

The module contained in the WCU will serve the following functions:

<WCU.F1>	Accept images from camera
<WCU.F2>	Store high resolution images
<WCU.F3>	Compress the images
<WCU.F4>	Send low resolution images to RVS
<WCU.F5>	Accept request from RVS for new image
<WCU.F6>	Accept request from RVS to turn system on
<WCU.F7>	Accept request from RVS to publish image to internet
<WCU.F8>	Send high resolution image to internet
<WCU.F9>	Accept Motion Sensor Input
<WCU.F10>	Send request to RVS to turn on
<WCU.F11>	Manage image files in memory
<WCU.F12>	Control 'waking up' of WCU hardware when requested
<WCU.F13>	Accept Request from RVS to 'go to sleep'
<WCU.F14>	Control 'putting to sleep' of all WCU hardware

This WCU module will be implemented in memory available to the TINI micro-controller and on the Motorola i85 Smart Phone connected to the TINI serially. The Java 2 Micro Edition platform will be used on the i85 phone and TINI's Java Virtual Machine will be utilized on the TINI board.

Remote Viewing System

The module contained in the RVS will serve the following functions:

<RVS.SF1>	Accept images from WCU
<RVS.SF2>	Display images from WCU
<RVS.SF3> <RVS.SF3.1> <RVS.SF3.2> <RVS.SF3.3> <RVS.SF3.4>	User Interface Allow User to request new image Allow User to request an images publication to net Allow User to 'wake up' system Allow User to put system 'to sleep'
<RVS.SF4>	Send request to put system to sleep
<RVS.SF5>	Send request to wake system up
<RVS.SF6>	Accept request to wake up, display user options

Network Interface

The module contained in the Network Interface will serve the following functions:

<NI.SF1>	Accept images from WCU
<NI.SF2>	Store images from WCU
<NI.SF3>	Display images from WCU
<NI.SF4>	User Interface

Risk Table

Risk	Impact	Impact Assessment	Probability	Mitigation Strategy	Owner	Status
HARDWARE						
Camera does not allow remote power control.	High	Can't turn system on remotely → poor power management → high system downtime.	Medium	Selection of camera with this feature. Set switch between battery and camera.	Ryan	New
Camera power not triggered by motion detection.	Medium	Must provide external feature to turn on camera in case of activity.	High	Plan for external motion detection and power control.	Eric	New
Motion detection signal not available to our system.	Medium	Must provide external feature to turn on camera in case of activity.	High	Plan for external motion detection and control.	Eric	New
Phone does not have sleep mode.	High	Poor power management → high system downtime or short usage time.	High	Select phone with feature or purchase battery with necessary life.	Ryan	New
Resolution of Remote Unit is not sufficient to customer specifications.	High	Poor function of device.	Medium	Select device with adequate resolution.	Matt	New
Speed of cellular network does not allow for adequate function.	High	Poor function of device. Poor resolution of image or long delay between frame display.	Medium	Select fast enough network if available. If not, design system to balance speed and quality with input of customer.	Hady	New
SOFTWARE						
Picture encoding process not accessible.	High	Can't extract picture files from camera information stream.	Medium	Select manufacturer willing to cooperate with our project by providing encoding information.	Fenghai	New
Availability of understandable API's	High	Need extra time to code.	Medium	Research API's before making final component selections.	Doug	New
INTEGRATION						
Phone signal cannot turn on TINI/Camera	High	Poor power management → high system downtime.	Medium	Design logic device to control TINI power.	Fenghai	New
TINI cannot turn on camera	High	Poor power management → high system downtime.	Medium	Design switch between battery and camera allowing for TINI to turn on camera.	Matt	New

<u>Software Testable Specifications</u>

1.1. Overview

Along with the hardware specifications, there are several testable specifications of the software that must be defined for the system.

1.2. Specifications

The functionality of the system is largely defined by the development of key software components working together across several devices.
These testable software specifications are as follows:

System Component	Testable Specifications
1) Wireless Camera Unit	a) Can the software of the WCU control the following functionality: -Timer delay of still images -Changing resolution
	b) Store high quality images for later transmission to the internet.
	c) Receive requests from the Remote Viewing System and send high quality images to the internet.
	d) Continuously transmit images to the RVS.
2) Remote Viewing System	a) Receive images from the WCU and display them with the appropriate software plug-in, decompression, and/or file translation.
	b) Provide functionality to allow RVS user to select image from the WCU queue and request that the WCU publish the image onto the internet.
	c) Allow interactive sleep mode of the WCU to be controlled by the RVS.

Software System Requirements and Product Design

I. Product Architecture

A. WCU

1) Hardware - Motorola i85 smart phone capable of TCP/IP communication and including a serial connection to the micro-controller.
2) Platform - J2ME, Java 2 Micro Edition, which runs on top of KVM on the smart phone.

B. RVS

1) Hardware – Compaq iPaq 3760
2) Platform – Jeode, a Sun Java certified 3rd party Java platform which runs the bytecode on EVM for the Compaq iPaq 3760

II. Requirements

A. WCU

<WCU.SR.1> The WCU software must send an image, which has been compressed to the RVS across the GSM/ GPRS network.
<WCU.SR.2> The WCU must accept interrupts for image selection from the RVS.
<WCU.SR.3> Once the WCU receives an command for a selected image, it then pauses submission of the images, retrieves the image from the micro-controller and sends that image to the internet.

B. RVS

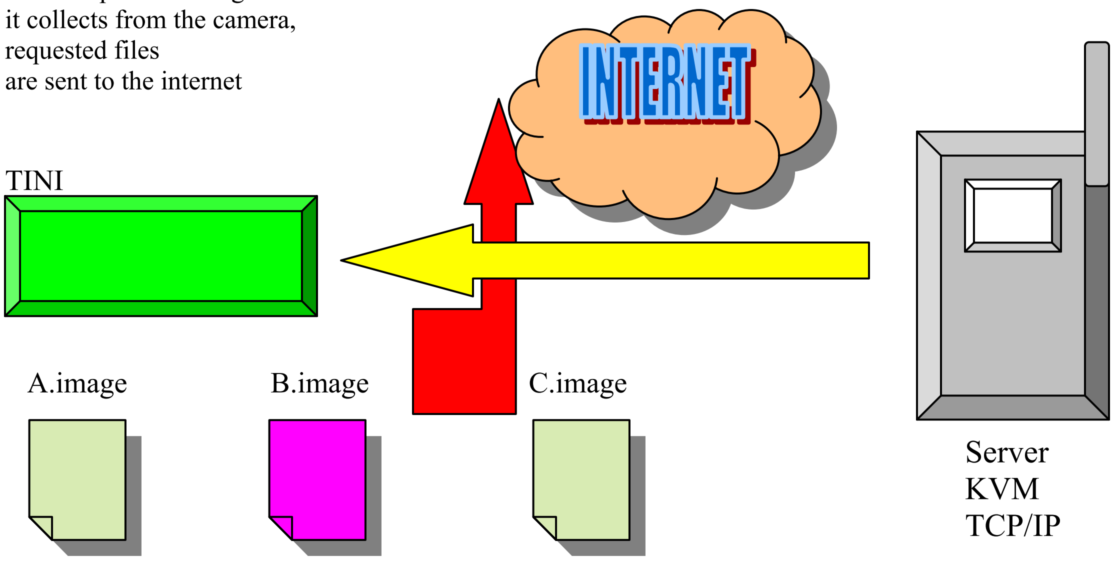
<RVS.SR.1> The RVS software will include a GUI interface.
<RVS.SR.2> This interface will have two separate modes of operation, an image slide show and queue selection.
<RVS.SR.3> These modes of operation will run concurrently on the handheld.
<RVS.SR.4> The handheld receive compressed images from the server and decompress them in real time and put the into an image queue in the device's memory.
<RVS.SR.5> The handheld will receive the images at a delay of some specific quantum (for example every 5 seconds) depending on the image size, rate of compression/decompression, and bandwidth of the network.
<RVS.SR.6> The images will be decompressed with a ZIP utility, JAR command or 3rd party decompression software, depending on how the camera or micro-controller compress the images.
<RVS.SR.7> Once decompressed, they will be placed in queue of a fixed length (for example 50 images) and the oldest image will be disposed of in memory when a new image is inserted at the back of the queue.
<RVS.SR.8> The selection mode will enable the user to view the images in the present queue, and then choose an image and request the server to send the image to the internet. Note: the WCU should pause transmission of images (see part 5 of previous section) so that the queue is not shuffling images and disposing of them.

III. Subsystem Design
(needs to be defined, how the above functions can be implemented in Java)

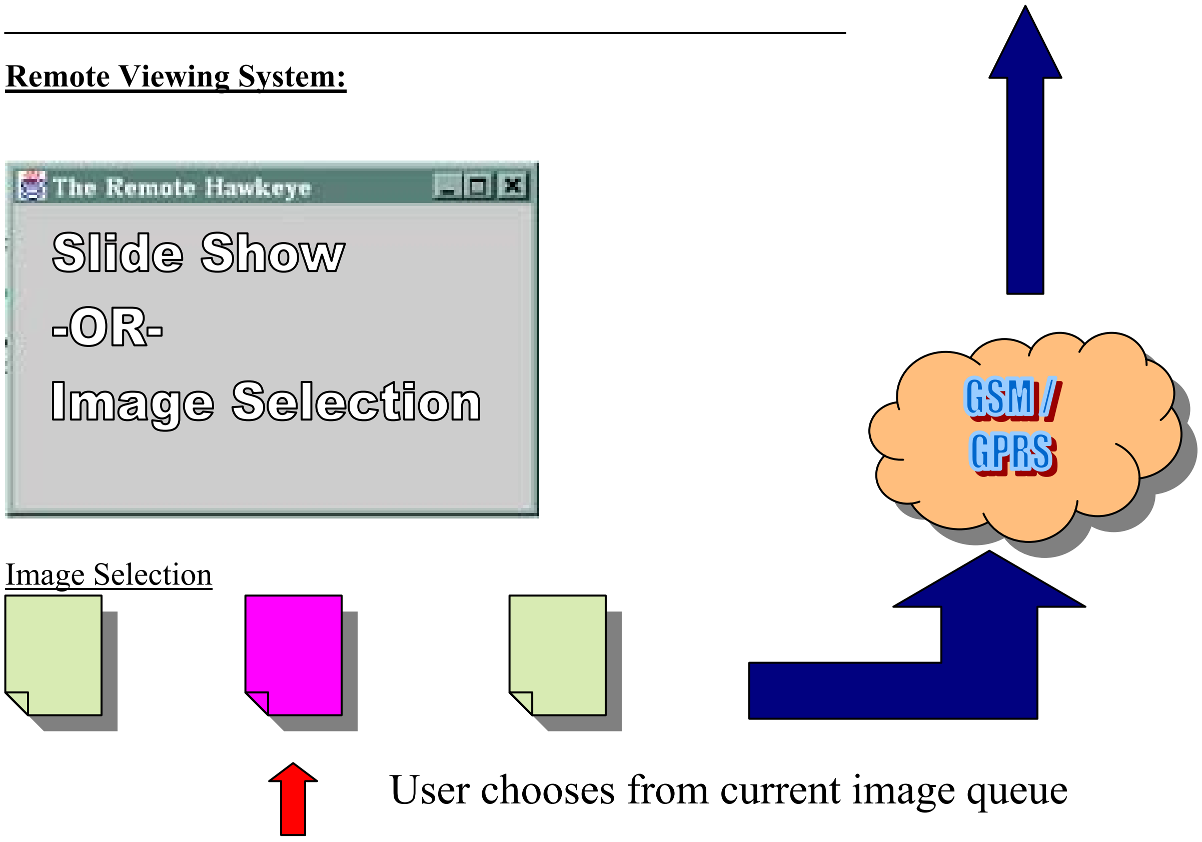
SOFTWARE PRODUCT DESIGN

Wireless Camera Unit:

Tini microcontroller
stores a queue of image
it collects from the camera,
requested files
are sent to the internet



Remote Viewing System:



The Technical Strategy

The following components and interfaces will be used in the USSOC project:

1) **Client-server model** – The communication between the camera and officer will be mediated using the client-server model, in which the server will be the TINI chip and the clients will be the officer's PDA/smart-phone/notebook PC and the camera.

2) **COTS** – COTS in the acronym for Commercial Off The Shelf. One of the project's requirements is that all components are pre-made and can be purchased from a retailer. Therefore, the camera, battery, micro-controller, communication devices, and other components can be readily integrated with minimal modifications.

3) **HTTP and IP** – HTTP (HyperText Transfer Protocol) and IP (Internet Protocol) will be used as network interfaces in this project. In order to send data to and from the officer and camera, a smart-phone and another communication device (TBA) will use IP packets. If data is to be published on the Internet, HTTP will be used to do this.

4) **Java KVM** – Java KVM is a micro-version of the full-fledge version of the Java Virtual Machine. This virtual machine was developed to have smaller memory requirements and smaller libraries to accommodate smaller hardware devices with low-bandwidths and limited processing power such as PDAs and smart-phones. Programs will be written under the KVM platform in order to control input and output of the smart-phone(s) and/or PDA that will be used to control the camera.

The following programming languages will be used in the USSOC project:

1) **Java KVM language** – This language will be used to develop applications that will allow the smart-phone(s) and/or PDA to communicate with each other in order to control functions such as camera movement, camera activation, capturing pictures from the video feed, and publishing pictures onto the Internet.

2) **HTTP** – This programming format will be used to control the TINI chip and its functions, such as controlling camera movements, storing captured pictures, and transimitting data to and from the camera and smart-phone.

The following tools will be used in the USSOC project:

1) **The PC** – Personal computers will be a vital tool in the success of this project. They will accommodate most of the testing of the software in order to make sure all components are working properly. They will also provide a means for communication with the customer, coach and development team.

2) **Simulators** – Simulators will be the testing tools of the project. Simulations of the Java KVM programs on the smart-phone(s) or PDA will be necessary in order to easily test the integrity and operability of the applications written by the development team.

3) **Electrical components and equipment** – Electrical components, such as resistors, transistors, and capacitors, may be required in order to provide different power requirements to the various devices used in the project. Electrical equipment will be used to connect and manipulate these components with the necessary devices.