

Remote Device Tracking and Control Using SMS and Google Maps

Daniel Hiranandani

California Polytechnic State University, San Luis Obispo
Computer Engineering Department
Senior Project

Dr. Hugh Smith, Advisor

June 9th, 2009

Table of Contents

List of Tables and Figures.....	4
1. Abstract.....	6
2. Introduction.....	7
2.1 Overview.....	7
3. Technology.....	8
3.1 GPS	8
3.1.a History.....	8
3.1.b Code-Phase vs. Carrier-Phase Synchronization	10
3.1.c Receiver Triangulation.....	10
3.1.d Sources of Error.....	11
3.1.e Advances in Technology.....	13
3.1.f NMEA 0183.....	14
3.2 SMS	15
3.2.a Overview	15
3.2.b GSM Standards	15
3.2.c Nokia Fastbus (F-Bus)	17
3.3 TCP/IP	18
3.3.a Protocol Stack.....	18
3.4 Google Maps.....	21
3.4.a Overview	21
3.4.b AJAX	22
3.5 Microchip PIC Embedded Systems	22
3.5.a Overview	22
3.5.b Olimex Development Boards	22
4. Development	24
4.1 Development Tools.....	24
4.1.a Gnokii.....	24
4.1.b MPLAB.....	25
4.2 Hardware Testing.....	25
4.3 Software Testing	26
4.4 Other Valuable Tools.....	27
5. Implementation	29
5.1 Overview.....	29
5.2 Hardware Block Diagrams.....	34
5.2.a Web Server.....	34
5.2.b Remote Device	34
5.3 Software Flow Diagrams	35
5.3.a Web Server.....	35
5.3.b Remote Device	37
6. Applications	40
6.1 Vehicle Tracking.....	40
6.2 Remote Device Control	40
6.3 Event-Triggered Responses	40
7. Future Improvements.....	41

7.1 Overview.....	41
8. Ethical Considerations	42
9. Conclusion	43
10. Acknowledgements	44
11. References.....	45
12. Appendix.....	48
12.1 Glossary	48
12.2 Parts List	50
12.3 Analysis of Senior Project	50

List of Tables and Figures

Figure 1. This image illustrates the three GPS segments – space, control, and user. ^[2]	8
Figure 2. The GPS constellation is divided into six orbital planes each separated by 60° and all at a 55° inclination. ^[7]	9
Figure 3. The two possible positions of the GPS receiver are shown as stars. One of these can be discarded because only one resides on the surface of the Earth. ^[11] .	11
Figure 4. This illustrated signal multipath caused by a single signal being reflected and reaching the receiver at a different time than the original signal. ^[4]	12
Figure 5. Good satellite geometry is shown on the left due to adequate use of the receiver's Field of View. In contrast, poor satellite geometry is displayed on the right due to the clustering of satellites. ^[12]	12
Figure 6. This calculation demonstrates the magnitude of positional error due to a 1μs clock error. ^[1]	13
Figure 7. This figure shows the contents of a GPRMC sentence contained within the NMEA 0183 standard. ^[16]	15
Figure 8. This is an example of 7-bit packing used to carry data in SMS messages.	16
Figure 9. This is an F-Bus command used to send the SMS message 'hello' to 805-903-2396.....	17
Figure 10. This figure shows the hierarchy of the TCP/IP Stack. ^[25]	18
Figure 11. The contents of an Ethernet frame are shown above. ^[28]	19
Figure 12. Shown above is the structure of an IPv4 packet. ^[29]	19
Figure 13. The TCP Header is shown above. ^[32]	20
Figure 14. This image shows a standard Google Maps screen that was provided by Google. ^[34]	21
Figure 15. This image shows two Atmel ATtiny13 microcontrollers. On the left is the device shown in the Small-Outline Integrated Circuit (SOIC) package, and in the Plastic Dual In-line Package (PDIP) on the right. ^[37]	22
Figure 16. This image shows the Olimex PIC-MAXI-WEB development board that is used as the web server for this project. ^[38]	23
Figure 17. Above is the Olimex PIC-LCD3310 development board used as the remote device in this project. ^[39]	23
Figure 18. This screenshot shows some of the options that Gnokii supports.	24
Figure 19. Microchip's MPLAB IDE is shown above with the code editing and variable Watch windows open.	25
Figure 20. The Nokia F-Bus and M-Bus contact pads are shown above. ^[23]	26
Figure 21. Pictured is the Nokia F-Bus Simulator which generates the basic SMS command messages used for testing.	27
Figure 22. Shown above is the Olimex Web Server board coupled to a breakout board which supplies the connection to a Nokia 3390 cell phone, as well as power.	29
Figure 23. The remote device displaying its main screen is shown connected to a cell phone as well as to the GPS receiver.	30
Figure 24. The remote device's status screen is shown on the left, and the compass screen is shown on the right.	31
Figure 25. A view of the remote device's settings screen.	32

Figure 26. Shown above is a screenshot of the Google Maps webpage with control buttons that was used in this project.	33
Figure 27. Shown above is the hardware block diagram for the web server.	34
Figure 28. The hardware block diagram is shown for the remote device.	34
Figure 29. The flow diagram for the web server's main function.	35
Figure 30. The SMS task function for the web server.	36
Figure 31. The web server's Interrupt Service Routine (ISR) is shown above.	36
Figure 32. The software flow diagram for the remote device main program flow.	37
Figure 33. Above is the software flow diagram for the remote device's SMS Task function.	37
Figure 34. The software flow diagram for the remote device's ISR is shown above.	38
Figure 35. The remote device's SMS ISR	39

1. Abstract

This project documents the creation, modification, and analysis of the devices needed to track a remote location using a Google Maps webpage and SMS through cell phones. The goal of this project was to track a remote device in near real-time and display the location on a Google Maps webpage.

The Short Message Service (SMS) was chosen to provide the data communication because there is cell phone coverage over most places in the world, so it would be possible to track the device practically worldwide. The Global Positioning System (GPS) was chosen because it is a free-to-use service for calculating position on the surface of the Earth. Google Maps was used because of its flexible integration into other webpages through its Application Programming Interface (API). Lastly, Olimex development boards featuring Microchip PIC microcontrollers were chosen because of their features including processing power, memory, number of inputs and outputs (I/O), and also because it would be easy to port code from one environment to the other.

The specifications of GPS, SMS, and the TCP/IP Stack are discussed in detail due to their importance in this project. Additionally, information about Google Maps and Microchip PIC microcontrollers are included to explain their relevancies to the project.

2. Introduction

2.1 Overview

The goal of this project was to track a remote device's location and display a visually-pleasing representation for the position information on the Internet in near real-time. In order to accomplish this, four main components are needed.

A Global Positioning System (GPS) receiver was needed to receive the GPS data coming from satellites orbiting the earth to determine the location of the device. This information pinpoints the location of the receiver to a varying degree of accuracy depending on how many satellites are in view and are being tracked.

A Short Message Service (SMS) communication controller was also necessary, which in this project was a Nokia 3390 cell phone. Once the location of the remote device is determined, this device performs the sending and receiving operations on the data to transport it to and from the web server. This device needs to transmit data over near and far distances because the remote device could be at any location with respect to the web server.

An Olimex PIC-LCD3310 embedded development board was needed as the remote device to process the GPS data and also to communicate with the Short Message Service (SMS) controller. The data coming in from the GPS receiver needs to be formatted for the SMS controller to understand, and then the SMS controller must be instructed when to send the data to the web server. A microcontroller was used to perform these functions because the tasks require little processing power, and topics such as power and size are much more important.

The Olimex PIC-MAXI-WEB embedded development board was needed to perform the web server functions. These operations require more memory and processing capabilities than those of the remote device, but still do not need very much. This board which supports a free implementation of the TCP/IP stack from Microchip was needed to communicate with the SMS controller it is attached to, process the incoming data, and display the location in some way that the user can understand. In this project, the web server displayed the location on a Google Maps webpage.

3. Technology

3.1 GPS

3.1.a History

The Global Positioning System, formerly known as Navigation System with Timing And Ranging (NAVSTAR) Global Position System, began as a research project of the United States Department of Defense (DoD) in the 1960s and 1970s.^[1] The project was to put several satellites containing atomic clocks into orbit to provide navigation and time data for military users on the ground. Since the first launch in 1978, there have been several fundamental changes to the purpose, satellites, and accuracy of the program.

There are three segments to GPS – space, control, and user. The space segment is the constellation network of satellites orbiting the Earth to provide navigation and time information. The control segment consists of five tracking stations located around the world to provide corrections to time, satellite location prediction, and other information to be transmitted up to the orbiting satellites. Both the space and control segments are controlled and maintained by the U.S. DoD. The user segment represents the ground-based receivers in civilian and military applications which acquire and use the GPS signals.

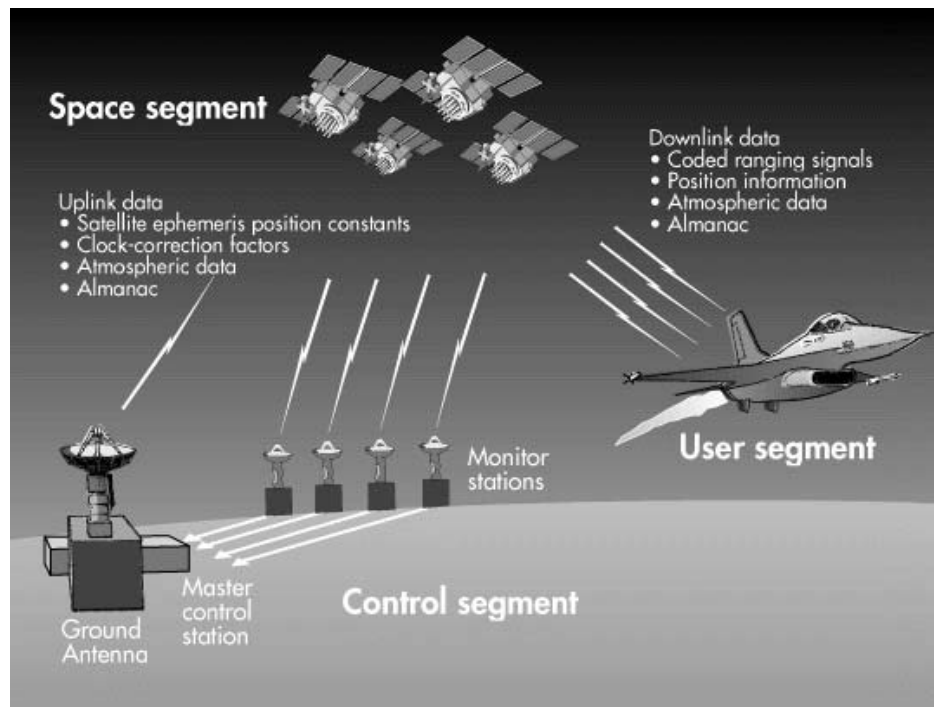


Figure 1. This image illustrates the three GPS segments – space, control, and user.^[2]

A disaster in 1983 where Soviet jets shot down a civilian commercial airplane that mistakenly flew into Soviet airspace prompted President Ronald Reagan to guarantee that GPS data would be provided for free around the world because more accurate navigation

tools could have prevented the tragedy.^[3] When the system became operational in 1995, GPS did provide data to all users around the world at no cost, but the data was only accurate to 100m. This civilian signal (L1) was degraded because “Selective Availability” (S/A) was enabled.^[4] This meant that intentional errors were introduced for the purpose of national security, so it was impossible to achieve accuracy greater than 100m without authorization from the U.S. Government to decode the encrypted precise (P(Y)) code information on the L2 signal. Still operated and maintained by the U.S. DoD and still at no cost, the GPS data is now accessible with S/A disabled which provides a signal accurate to around 10m.^[5]

The design for the constellation of satellites calls for 24 satellites in orbit, but there are currently 31 active and two spare satellites in space.^[6] The 24 satellite design specifies that four operate in each of the six orbital planes with 55° inclination and that each of the planes are separated by 60° which ensures at least six satellites in view at all times throughout the world. Having 31 active satellites makes the constellation non-uniform which improves the coverage over a uniform configuration, especially when a satellite fails.^[7]

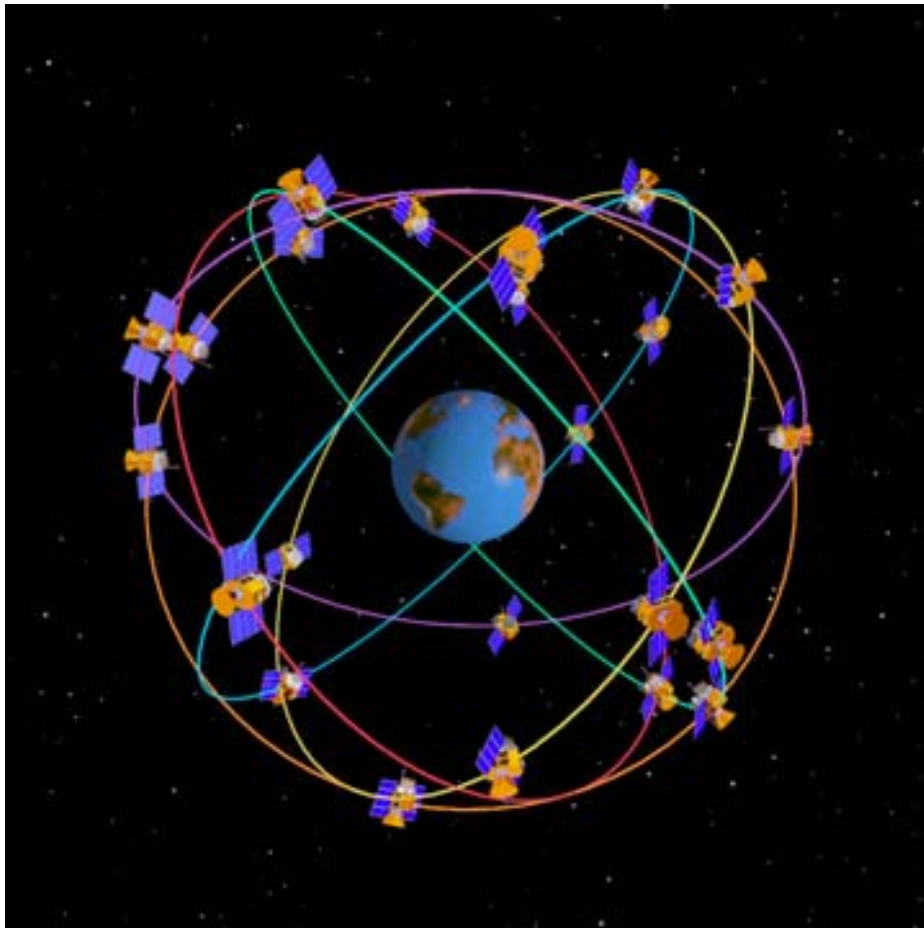


Figure 2. The GPS constellation is divided into six orbital planes each separated by 60° and all at a 55° inclination.^[7]

3.1.b Code-Phase vs. Carrier-Phase Synchronization

A GPS message frame consists of a coarse/acquisition code (C/A) which is pseudo-random code to identify the satellite, ephemeris data to indicate where the satellite should be in space at any time, and almanac data to report on the health of the satellite as well as the current date and time.^[8] The first thing the GPS receiver must do is synchronize its clock with the satellite signals, and there are two ways it can do this.

With Code-Phase synchronization, the GPS receiver listens to the C/A code in the incoming signals and generates the same pseudo-random code on its own. The receiver keeps delaying the time from the last received signal to when it generates its own C/A code until the incoming signal matches the generated C/A code. The Code-Phase name comes from the receiver synchronizing to the data within the C/A code which has a frequency of 1.023 MHz.^[9] This is a very common method for GPS receivers to use because the hardware required to function near 1 MHz is very cheap.

Carrier-Phase synchronization means that the GPS receiver synchronizes its time to the frequency that the C/A is being carried at as opposed to the contents of the C/A message. The L1 frequency is 1.57542 GHz, so the receiver must operate at near this frequency to generate and compare signals which takes significantly more expensive hardware.^[9]

In practice, it is difficult for receivers to perform Carrier-Phase synchronization because of repetitions in the carried signal. At around 1 MHz, one C/A signal cycle would be represented with approximately 1500 cycles in the L1 carried signal. Due to this difficulty, receivers that use Carrier-Phase synchronization use Code-Phase synchronization first to know where they are within the content of the signal, then switch over to Carrier-Phase to hone in on the carrier frequency.^[9]

3.1.c Receiver Triangulation

GPS works on the idea of triangulation. Once the receiver's clock is synchronized with a satellite, it measures the time it takes for the signal to arrive. The time delay can then be converted into a distance from the satellite, because electromagnetic waves normally travel at the speed of light.^[10]

With a GPS receiver getting a signal from one satellite, the receiver is able to determine its location on an imaginary sphere where the radius is exactly the calculated distance from the satellite. Signals from two satellites reduce the possible locations to the circle of intersection between the two imaginary spheres from each of the satellites. Adding a third satellite's information intersects the imaginary spheres once more which leaves two points of intersection. One of these points is not on the surface of the Earth, so it can be discarded in the simplest implementation. If the GPS receiver is able to receive data from more than three satellites, it can use the extra information to reduce the error in its location. Having more than three satellite signals lets the receiver pick the most accurate and discard any others which it can determine based on the idea that all of the signals' imaginary spheres should intersect at the same point if they were all completely correct. If at least three signals meet this criterion, then erroneous signals can easily be detected and discarded.^[10]

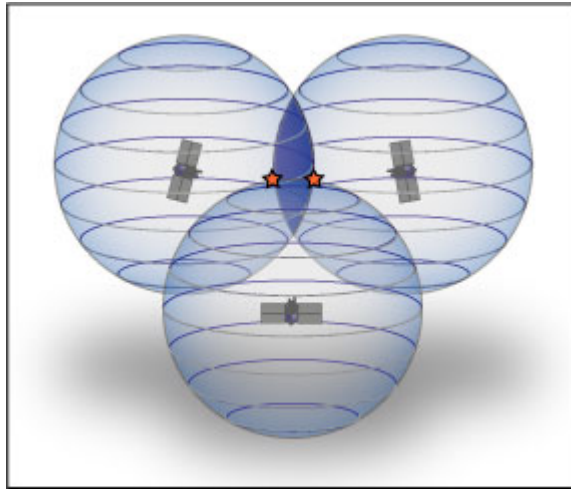


Figure 3. The two possible positions of the GPS receiver are shown as stars. One of these can be discarded because only one resides on the surface of the Earth.^[11]

3.1.d Sources of Error

Even with the massive amount of money spent on GPS, there are still inaccuracies in the technology. Some of the errors include delays while passing through the atmosphere, signal multipath, orbital errors, satellite geometry, receiver clock errors, and the number of visible satellites.^[8]

Atmospheric delays occur in the ionosphere and troposphere when the data is transmitted from the satellite to the user. Since electromagnetic waves travel at different speeds through different mediums, and the atmosphere is not uniform throughout, the time it takes to reach the Earth will vary.^[8]

Signal multipath is when the data coming from the satellite travels on a path that is not a straight line. The signal can reflect off of objects which causes it to take a longer path, so the receiver will see the straightest-path signal first, and then will be bombarded with duplicate signals which cause excess noise in the first signal.^[8]

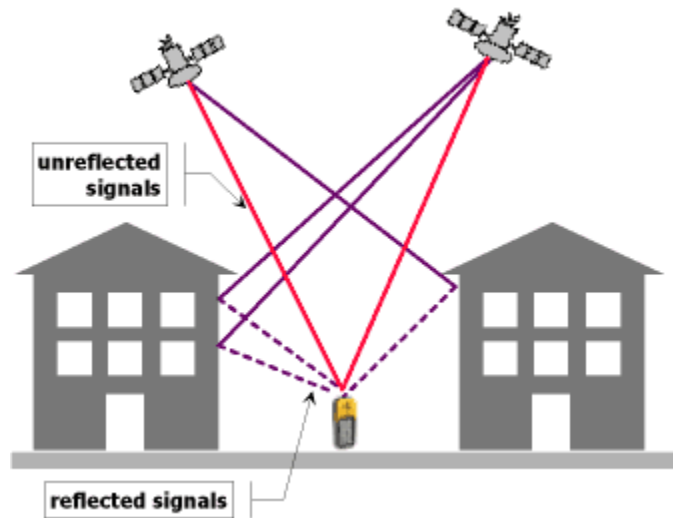


Figure 4. This illustrated signal multipath caused by a single signal being reflected and reaching the receiver at a different time than the original signal.^[4]

Ephemeris, or orbital, errors occur when the satellite's position is not where it is expected to be. Ephemeris data is stored in the GPS receiver and is updated periodically to maintain the recent orbits of the satellites, but this information can be incorrect if the satellite believes it is in a different location than it actually is.^[8] Position calculations on the ground are entirely reliant on knowing where the satellite should be at all times, so the ephemeris data needs to be as accurate as possible.

Satellite geometry is the position of a satellite with respect to the other satellites around it. The accuracy on the surface of the Earth depends on the shape the satellites form because if the satellites are very close together, the data gathered from each signal will be highly correlated with the other signals. In order to produce accurate results, each satellite needs to maintain a distance from the other satellites so that each provides the user with non-overlapping data.^[8]

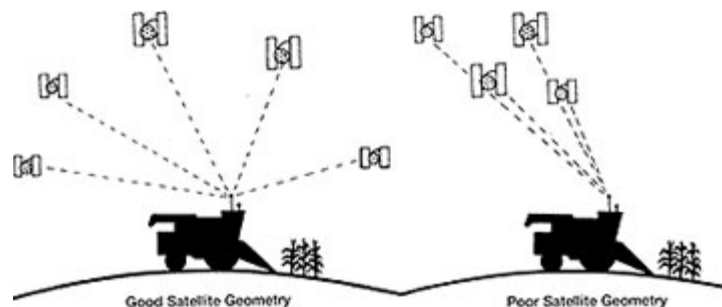


Figure 5. Good satellite geometry is shown on the left due to adequate use of the receiver's Field of View. In contrast, poor satellite geometry is displayed on the right due to the clustering of satellites.^[12]

The user can introduce errors in two ways: an inaccurate clock in the receiver, or having too few satellites in view. Timing is critical in calculating the distance from each

of the satellites, so even though the clock in the receiver can be less accurate than the atomic clocks in the satellites, the results will still be highly dependent on the error of the clock.

The rubidium and cesium satellite clocks are “accurate to 1 part in 10^{12} and 1 part in 10^{13} respectively” while Quartz-based consumer GPS receivers are “accurate to 1 part in 10^8 ,”^[1]. This means that if the receiver’s clock is off by one microsecond, the position calculation would be off by nearly 300m as shown in Figure 6.

$$R_E = T_O * c \quad (5-7)$$

where

R_E = user equivalent range error

T_O = time offset

c = speed of light

(1) The following example shows the calculation of the user equivalent range error (UERE or UR).

$$T_O = 1 \text{ microsecond } (\mu\text{s}) = 10^{-6} \text{ seconds (s)}$$

$$c = 299,792,458 \text{ m/s}$$

From Equation 5-7:

$$\begin{aligned} R_E &= (10^{-6} \text{ seconds}) * 299,792,458 \text{ m/s} \\ &= 299.79 \text{ m} \approx 300 \text{ m user equivalent range error} \end{aligned}$$

Figure 6. This calculation demonstrates the magnitude of positional error due to a 1μs clock error.^[1]

Having more satellites in view of the receiver increases the accuracy of the results because error-ridden transmissions can be discarded in favor of another satellite’s information.

3.1.e Advances in Technology

To combat the sources of error, there have been many different improvements to increase the accuracy of GPS. The first and most general compensation is to model the sources of error. Having a typical estimate for an error such as atmospheric delay will reduce the error significantly, but also inherently introduces a new possible source of noise. The models can only estimate what could be happening as opposed to empirically measuring the error, so it is not a perfect solution. In addition to models, modifications can be made to the space and control segments to improve accuracy.

Differential GPS (DGPS) is a system that is based on the idea of measuring the GPS signals at a base station, calculating the station’s location from the signals, and then comparing the result to its very precise known location. The difference in the positions accounts for all types of error in the space segment and can be broadcast via ground

stations to the user segment. This method improves accuracy to 5m, but is only valid for relatively short distances near the base station because atmospheric delays make up a large source of the error and the delay changes depending on the location.^[13]

The Wide Area Augmentation System (WAAS) developed by the Federal Aviation Administration is a system used to measure variations in the GPS satellite signals, and then broadcast those variations from two WAAS satellites for the user segment. Ground stations first monitor the different GPS signals and then send the data to master stations. The master stations create correction information for ephemeris and clock errors as well as atmospheric delay, and then transmit the data to the WAAS satellites for broadcast. With WAAS enabled, accuracy improves from to less than 3m 95% of the time.^[14]

Currently, there are several new signals being prepared for broadcast in the future including the improved Civilian L1 (L1C) and the new Civilian L2 (L2C) signals.^[15]

The L2C signal will transmit the same data as the current L1 signal, but at a different frequency.^[15] This addition will enable receivers to judiciously calculate the atmospheric delay at their individual locations. Receivers can compensate for errors simply by knowing the frequencies of the two signals because they will be transmitted at the same time, so the majority of any delay would be caused by one frequency traveling at a different speed through the atmosphere. The difference in speed will cause one signal to reach the receiver first, and the difference between arrival times is used to calculate the atmospheric delay.

The major change in the L1C signal is using more power to transmit its signal. A boost in power will decrease signal degradation and in turn make the signal less susceptible to noise. L1C must maintain the same specifications as L1 to ensure backwards compatibility since it will be transmitted at the same frequency as L1, so changes to its specification are limited.^[15]

3.1.f NMEA 0183

NMEA 0183, also known as NMEA, is a specification set by the National Marine Electronics Association (NMEA) that defines how data is to be communicated for use in marine electronic devices. This standard uses serial ASCII text for data transmission, and each message is referred to as a sentence. Sentences must begin with a dollar sign symbol followed by two characters identifying the talker, and then three characters to indicate the type of message. The default specifications for serial communication are: 4800bps, 8 data bits, no parity, and 1 stop bit.

The GPRMC sentence, also known as the Recommended Minimum sentence, contains the latitude, longitude, speed, bearing, and time information among others. This sentence is sufficient for many applications because it gives the receiver the data for navigation, and it is the one used for this project.

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A
```

Where:

RMC	Recommended Minimum sentence C
123519	Fix taken at 12:35:19 UTC
A	Status A=active or V=Void.
4807.038,N	Latitude 48 deg 07.038' N
01131.000,E	Longitude 11 deg 31.000' E
022.4	Speed over the ground in knots
084.4	Track angle in degrees True
230394	Date - 23rd of March 1994
003.1,W	Magnetic Variation
*6A	The checksum data, always begins with *

Figure 7. This figure shows the contents of a GPRMC sentence contained within the NMEA 0183 standard.^[16]

3.2 SMS

3.2.a Overview

The Short Message Service was developed for GSM networks to allow for small amounts of text data to be sent between cell phones. Since the GSM standard was created with voice communication in mind, SMS was designed to send data through a separate path that was already in use for signaling the start and end of voice calls. By sending the messages over the signaling network, there was no added cost to the providers because the paths were only used when a phone call was initiated or terminated so the resources were quite often unused.^[17]

SMS messages were originally limited to 128 bytes to fit into the existing message format, but this was later extended to 140 bytes by cutting out unnecessary information from the message frame.^[17] To make even better use of space, message frames can carry up to 160 ASCII characters by using 7-bit packing.^{[18][19]} Other alphabets are also supported, but transmit fewer characters in each message because they need space to identify the encoding scheme.^[20]

SMS messages are divided into three categories: Mobile Originating (MO), Mobile Terminating (MT), and Cell Broadcast (CB). MO is when a mobile device sends a message, MT is when a mobile device is the destination, and CB is when one message is sent to many mobile devices within an area.^{[18][19]}

3.2.b GSM Standards

The GSM standards documents are what define SMS communication, and there are several aspects of the messages outlined in these documents including: contents of a message frame, encoding scheme for values, language alphabet for data being transferred. The five GSM documents that cover the main specifications of SMS are GSM 03.38, GSM 03.42, GSM 03.40, GSM 03.41, and GSM 07.05.

GSM 03.38 defines the default alphabet for SMS messages which must be supported by all cellular devices. The default alphabet is a 7-bit encoding scheme used to convert the 8-bit ASCII alphabet into 7-bits. The ASCII alphabet only uses the lower 7-bits for the Latin alphabet, numbers, control characters, and symbols, so the standard

ASCII alphabet can be preserved by eliminating the extraneous bit. In a full 140 byte message, an extra 20 characters can be included simply by using 7-bit encoding.^[20]

ASCII	h	e	l	l	o
ASCII hex	0x68	0x65	0x6c	0x6c	0x6f
ASCII 8bit binary	0b01101000	0b01100101	0b01101100	0b01101100	0b01101111
ASCII 7-bit binary	0b1101000	0b1100101	0b1101100	0b1101100	0b1101111
	(swap order of bytes due to shifting least significant bits (LSb) of a higher order byte into the most significant bits (MSb) of a lower order byte)				
Byte swap	0b1101111	0b1101100	0b1101100	0b1100101	0b1101000
7-bit encoded	0b00000110	0b11111101	0b10011011	0b00110010	0b11101000
7-bit encoded hex	0x06	0xfd	0x9b	0x32	0xe8
Byte swap					
7-bit encoded message	0xe8	0x32	0x9b	0xfd	0x06

Figure 8. This is an example of 7-bit packing used to carry data in SMS messages.

GSM 03.42 describes the use of Huffman Coding for compressing the data in an SMS message. The document details the two implementations of both parties knowing the compression dictionary prior to data exchange which would be based on common transmissions, or by creating and sharing the dictionary dynamically.^[21] Largely due to the limited number of characters that can be sent in an SMS message, this standard is not often used.

GSM 03.40 details the heart of SMS including specifications about SMS Centers (SMSC) which handle message processing and relaying, message frame fields and values, and SMS transmission including MO and MT but not CB. The idea is that any SMS message originating from a mobile device will be sent to an SMSC and any message destined to a mobile device will first be sent to an SMSC. The SMSC acts as a distribution center to process the incoming message, bill the user(s) accordingly, and then transmit the message to its next hop or to the destination. A message sent to or from an SMSC can contain the following information: validity period, service center timestamp, priority, messages waiting, alerts, and status reports among other fields.^[18]

The validity period is a value to designate how long the message is guaranteed to be saved in the service center's memory before the delivery has been made to the mobile device. The service center timestamp indicates the time the message arrived at the service center. Priority can be used to force-send a message to the destination regardless of the device being known as "temporarily absent", or having no free memory available.^[18]

Alerts are used to inform the service center that the destination device is now reachable and/or that it has memory available, or to inform the sender that the destination is unable to receive the message due to lack of availability or available memory.^[18]

Lastly, status reports can be used to indicate if the message was successfully delivered, or if it was unable due to temporary reasons like an unavailable service center, or permanent reasons including an expired validity period or invalid destination number.^[18]

The CB specifications are contained within GSM 03.41, and they describe how one message can be broadcast to many users within a particular geographical area. These broadcast messages originate from entities connected to Cell Broadcast Centers (CBC), and are then forwarded out to the individual areas. Luckily for users, the broadcast messages are categorized into classes based on the message content and language so they can be filtered based on what the user chooses to receive.^[19]

GSM 07.05 outlines the aspect of SMS pertaining to communication between the cellular device and a computer. The three protocols described are Block Mode, Text Mode, and PDU Mode. Block Mode is a binary protocol which includes error protection, Text Mode is based on the AT Command Set which uses characters for communication, and PDU Mode is a character-based communication but encodes the data in hexadecimal.^[22]

Concerning reliability, SMS is a best effort protocol, so the source assumes the message will be delivered once it sends the message to its next hop. SMSCs improve on this by using the validity period and alert messages, but messages still are able to arrive out of order, much later than they were sent, or never arrive at the destination at all.

3.2.c Nokia Fastbus (F-Bus)

The Nokia F-Bus is a protocol supported by most Nokia cell phones for communication between the mobile device and a computer. The F-Bus follows the M-Bus which was a bi-directional, half-duplex bus that transmitted data at 9600bps through a single pin. The newer F-Bus transmits at 115,200bps and is full-duplex which greatly increases the speed of data transfers.^[23]

When a cell phone is tethered to a computer through F-Bus, a multitude of commands can be sent to and received from the phone ranging from SMS send requests, SMS messages received, GSM network status update, phone reboot requests, and many more.

1	5	10	15	20	25	30
1E 00 0C 02 00 31 00 01 00 01 02 00 07 91 31 21 13 94 18 F0 00 00 00 00 11 00 00 00 05 0B 91						
	35	40	45	50	55	
81 50 09 23 93 F6 00 00 00 00 A9 00 00 00 00 00 00 E8 32 9B FD 06 01 40 00 18 90						
Bytes 01–06: Send a type-SMS message over cable from PC to phone						
Bytes 14–20: SMSC is 1-312-314-9810 stored in nibble-swapped order (i.e. 3121 = 1312)						
Bytes 31–37: Destination is 805-903-2396, also stored in nibble-swapped order						
Bytes 49–53: Message to send is ‘hello’						
Bytes 55–58: Sequence number, even byte padding, even and odd byte checksums						

Figure 9. This is an F-Bus command used to send the SMS message ‘hello’ to 805-903-2396.

3.3 TCP/IP

3.3.a Protocol Stack

The TCP/IP Stack can signify two things: a description of the functions that should be performed by the protocols, or a library of files that perform the functions that the protocol describes. The protocol is broken up into five layers inspired by the OSI Model. These layers are generally independent of each other which allows for different implementations at each level and still allow data to be communicated between all of the layers.^[24]

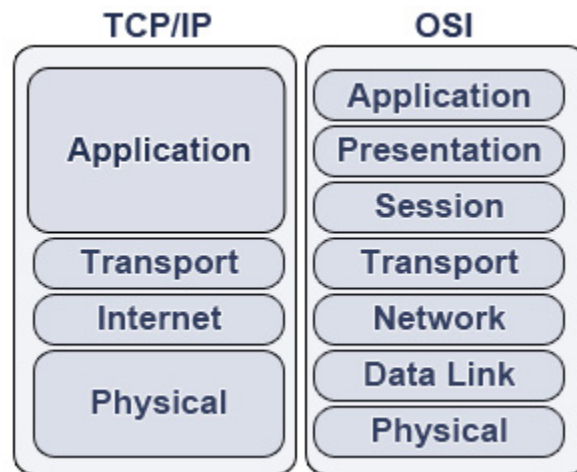


Figure 10. This figure shows the hierarchy of the TCP/IP Stack.^[25]

The first layer is a combination of the OSI's Physical and Data Link layers, called the Link layer. With respect to the Physical layer, the Link layer encompasses the physical aspect such as the medium that the data travels through, and specifications including information such as the types of cabling and maximum distances. It also includes how to get to the next destination, or hop, in the path to the final destination.^[24] A common Link layer for wired networks is Ethernet, and contained within the Ethernet specification is the Media Access Control (MAC) protocol.^[26]

MAC is a specification that enables multiple network nodes to communicate with each other typically in a local area, and also to detect and/or avoid frame collisions. For the communication aspect, data can be sent to one destination or to all local devices using a special – broadcast – address.^[26]

To prevent collisions, the Carrier Sense Multiple Access / Collision Detection (CSMA/CD) protocol is used. CSMA/CD specifies that the Ethernet devices must listen to the wire before attempting to transmit to avoid data collisions. If a collision does happen, the offending devices must send a jam signal indicating a collision happened so that no other devices attempt to transmit data and cause more collisions. After sending the jam signal, the two devices that caused the original collision wait a random amount of time before retransmitting so that they will hopefully send at different times and avoid another collision.^[27] This layer also provides a trailing checksum for error detection.^[28]

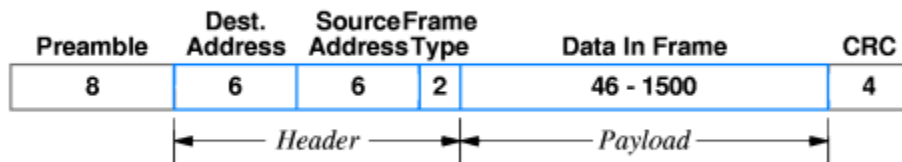


Figure 11. The contents of an Ethernet frame are shown above.^[28]

The second layer is the Internet or Network layer, and in the Internet is commonly the Internet Protocol (IP). One main function of the Internet layer is that it allows multiple Link layer networks to communicate with each other. A core function of the Internet layer is to route, and this means that it locates the final destination and determines the path(s) to take across the Link layer networks in order to get to the final destination. IP is a best effort protocol, so it is possible for packets to be lost, corrupt, out of order, or duplicated when they arrive at the destination. IPv4 does include a checksum in the header to detect corrupt packets, but the checksum has been omitted from the newer IPv6 header because IPv6 assumes layers above and below provide their own checksums.^[24]

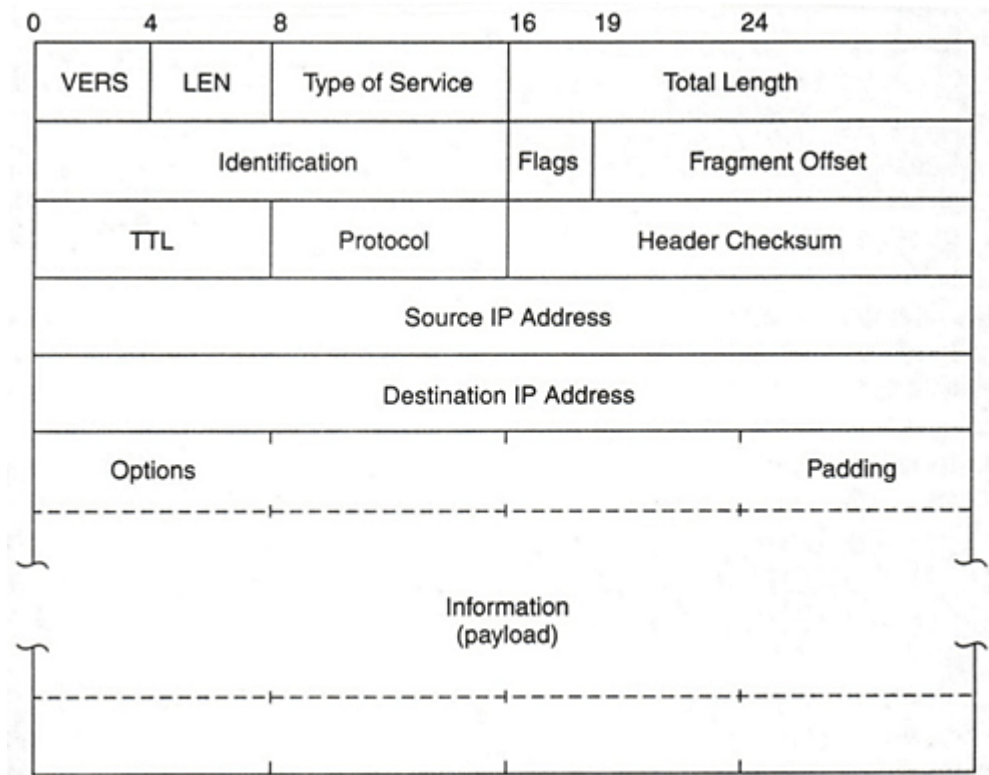


Figure 12. Shown above is the structure of an IPv4 packet.^[29]

The Transport layer is the third layer, and can provide services such as: reliability, flow control, error recovery, and ordering of packets at the destination. The two main

protocols at this layer are the Transport Control Protocol (TCP) and the User Datagram Protocol (UDP).^[24]

TCP is a connection-oriented protocol meaning that a connection must be established between the source and the destination before any data can be transmitted. The connection setup process is called the Three Way Handshake which consists of a Synchronization (SYN) packet, followed by a Synchronization-Acknowledgement (SYN-ACK) packet, followed by a final Acknowledgement (ACK) packet. The SYN, SYN-ACK, ACK handshake allows the source and destination devices to agree on sequence numbers they will be attaching to data in order to keep track of which packets arrived correctly and the correct order. In addition to providing a reliable connection, TCP also provides error detection, error recovery, and flow control.^[30]

UDP is a connectionless protocol without flow control and is best-effort, so the protocol doesn't care if the data does or doesn't reach its destination after it has been sent, whereas TCP certainly does care. The advantage UDP has is that it can send data to a destination faster than TCP, because it doesn't need to set up a connection before data can be transmitted.^[31]

In the Internet, TCP is used because recovery from corrupt or lost packets is essential for the end users, and flow control is also needed to prevent any device from using more bandwidth than it should.

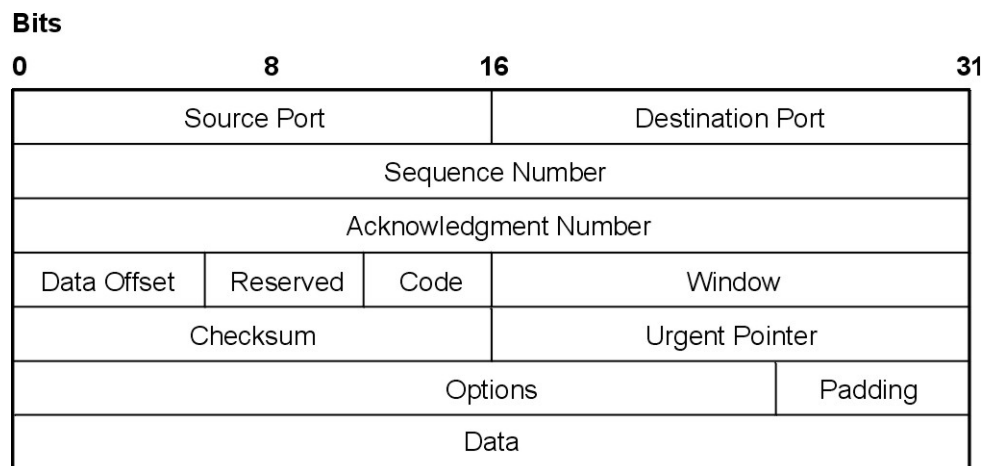


Figure 13. The TCP Header is shown above.^[32]

Finally, the fourth layer is the Application layer.^[24] This layer uses the work of its lower layers, and if using TCP, to access data directly knowing it is error-free and in order. Common Application layer protocols are the Hyper Text Transfer Protocol (HTTP) for Internet browsing, Post Office Protocol 3 (POP3) for email, File Transfer Protocol (FTP) for file transfers, and Domain Name System Protocol (DNS) for translating text-based domain names or Uniform Resource Locators (URL) into IP addresses for the computers to understand. In this project, HTTP is the most extensively used Transport layer protocol because it is what carries data between the web server and the client's browser.

3.4 Google Maps

3.4.a Overview

Google Maps is a free web application that shows maps of a requested area in several views showing information including street name overlays, satellite and aerial images, topographic maps, and even images from city streets.^[33] Google Maps can be used for many purposes including virtual exploration, and generating driving directions, but there are also many third party implementations because Google offers an Application Programming Interface (API) for developer use.

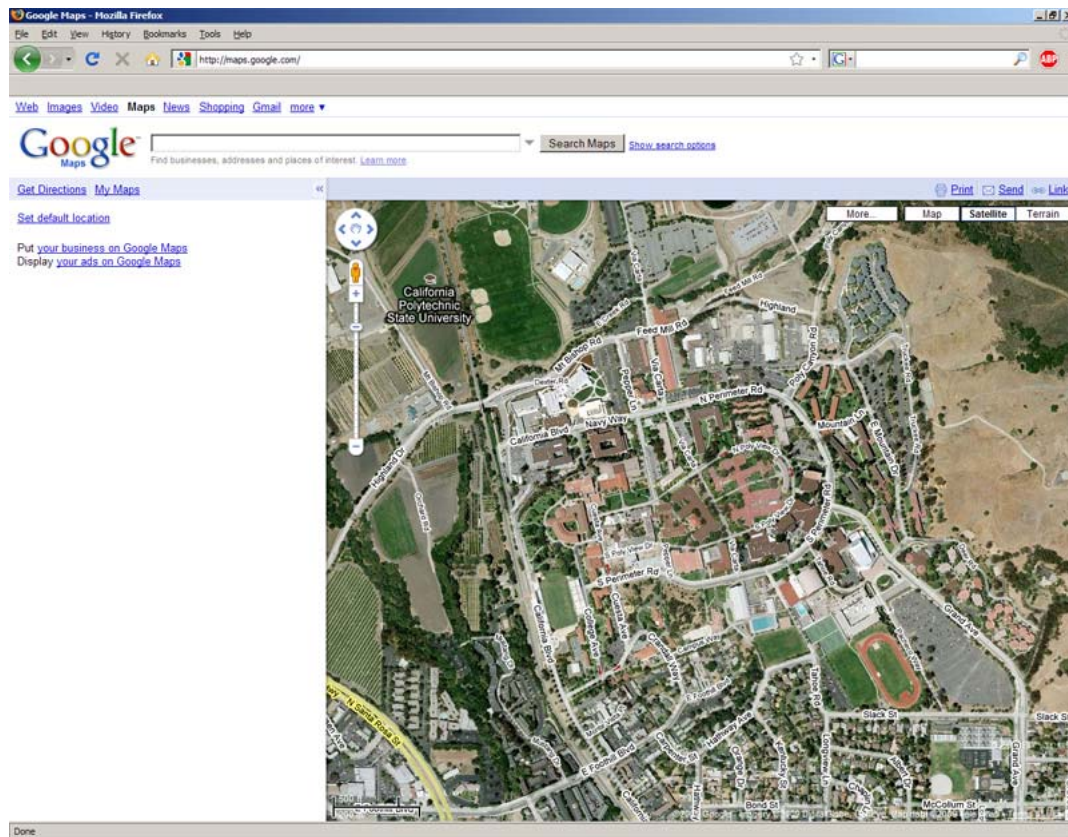


Figure 14. This image shows a standard Google Maps screen that was provided by Google.^[34]

The Google Maps API allows developers to interface to Google's code and create customized Google Maps on their own websites.^[33] Each map displays a location based on GPS coordinates, but the API has many capabilities because Google Maps is paired with Google's search engine. A user can enter a street address, business name, or many other types of points of interest, and Google Maps will translate the text into coordinates and display the location.

3.4.b AJAX

Google Maps relies heavily on the use of Asynchronous JavaScript and XML (AJAX) which is the idea of grouping of different web programming techniques to create highly dynamic webpages.^[35] By mixing JavaScript, Extensible Markup Language (XML), Hypertext Markup Language (HTML), and Cascading Style Sheets (CSS), content on the webpages can be updated individually or all at once without the user having to issue a page refresh request.

3.5 Microchip PIC Embedded Systems

3.5.a Overview

Microchip PIC microcontrollers are very small devices that have a wide range of applications. They contain a processor, memory, inputs and outputs (I/O), and code to run which are four characteristics needed to be a computer.^[36] Microcontrollers are very flexible because of their size and power consumption so they can be used in many ways, but they usually serve only a few functions because of the limited resources they have.



Figure 15. This image shows two Atmel ATtiny13 microcontrollers. On the left is the device shown in the Small-Outline Integrated Circuit (SOIC) package, and in the Plastic Dual In-line Package (PDIP) on the right.^[37]

3.5.b Olimex Development Boards

The two development boards used in this project are made by Olimex Ltd, and they both use the PIC18 series of microcontrollers because these chips have a large number of I/O, memory, and can run at relatively high frequencies.

The web server board, PIC-MAXI-WEB, uses the PIC18F97J60 and boasts 128KB flash memory, built-in Ethernet MAC and PHY, a 16x2 character LCD, relays, buttons, LEDs, and most importantly, support for Microchip's free TCP/IP Stack.

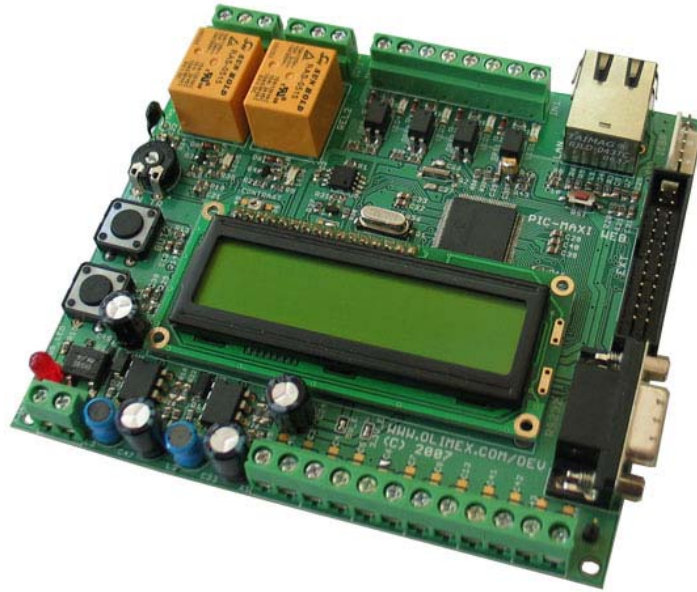


Figure 16. This image shows the Olimex PIC-MAXI-WEB development board that is used as the web server for this project.^[38]

The remote board, PIC-LCD3310, uses the PIC18F67J60 chip which also has 128KB flash memory, but the board incorporates a Nokia 84x48px LCD, a joystick, 3-axis accelerometer, LEDs, USB and SD/MMC memory card interfaces. This project mostly focuses on the web serving functions of the web board, and the LCD on the remote board, but the features certainly leave room for future additions.



Figure 17. Above is the Olimex PIC-LCD3310 development board used as the remote device in this project.^[39]

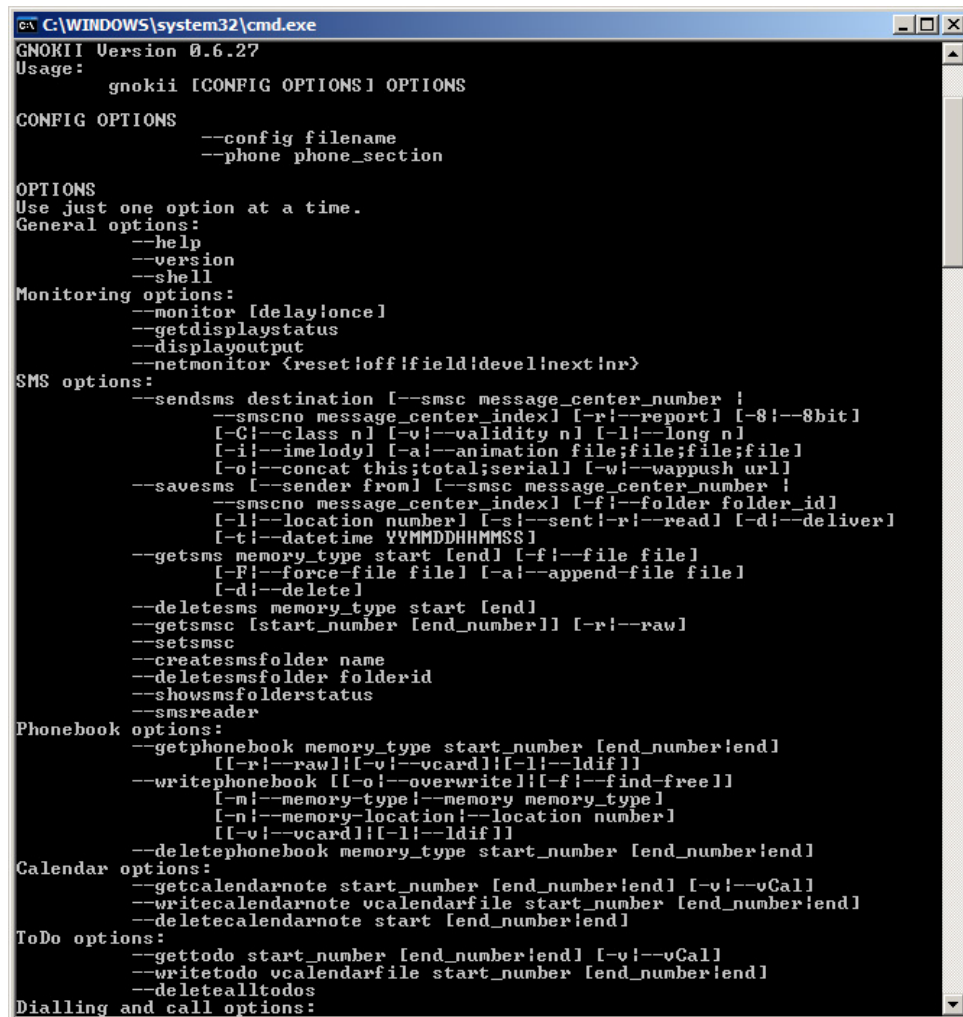
4. Development

4.1 Development Tools

In order to successfully debug this project, I used several tools which proved to be invaluable.

4.1.a Gnokii

Gnokii is an open source project that was developed to allow a PC to communicate with several versions of Nokia cell phones. It supports communication over USB, serial, and infrared connections, and most importantly supports a very large number of phone commands. The Gnokii developers also included documentation on the different commands for families of cell phones which is the most extensive documentation I found that is available to the public.



```
C:\WINDOWS\system32\cmd.exe
GNOKII Version 0.6.27
Usage:
    gnokii [CONFIG OPTIONS] OPTIONS

CONFIG OPTIONS
    --config filename
    --phone phone_section

OPTIONS
Use just one option at a time.
General options:
    --help
    --version
    --shell
Monitoring options:
    --monitor [delay|once]
    --getdisplaystatus
    --displayoutput
    --netmonitor <reset|off|field|devel|next|nr>
SMS options:
    --sendsms destination [--smc message_center_number !
        --smcno message_center_index] [-r!--report] [-8!--8bit]
        [-C!--class n] [-v!--validity n] [-l!--long n]
        [-i!--imelody] [-a!--animation file;file;file;file]
        [-o!--concat this;total;serial] [-w!--wappush url]
    --savesms [--sender from] [--smc message_center_number !
        --smcno message_center_index] [-f!--folder folder_id]
        [-l!--location number] [-s!--sent!r!--read] [-d!--deliver]
        [-t!--datetime YYMMDDHHMMSS]
    --getsms memory_type start [end] [-f!--file file]
        [-F!--force-file file] [-a!--append-file file]
        [-d!--delete]
    --deletesms memory_type start [end]
    --getsmc [start_number [end_number]] [-r!--raw]
    --setsmc
    --createsmsfolder name
    --deletesmsfolder folderid
    --showsmsfolderstatus
    --smsreader
Phonebook options:
    --getphonebook memory_type start_number [end_number!end]
        [[-r!--raw]![-v!--vcard]![-l!--ldif]]
    --writephonebook [[-o!--overwrite]![-f!--find-free]]
        [-m!--memory-type!--memory memory_type]
        [-n!--memory-location!--location number]
        [[-v!--vcard]![-l!--ldif]]
    --deletephonebook memory_type start_number [end_number!end]
Calendar options:
    --getcalendarnote start_number [end_number!end] [-v!--vCall]
    --writecalendarnote vcalendarfile start_number [end_number!end]
    --deletecalendarnote start [end_number!end]
ToDo options:
    --gettodo start_number [end_number!end] [-v!--vCall]
    --writetodo vcalendarfile start_number [end_number!end]
    --deletealltodos
Dialling and call options:
```

Figure 18. This screenshot shows some of the options that Gnokii supports.

4.1.b MPLAB

MPLAB is an Integrated Development Environment (IDE) provided by Microchip for free that allows users to write and compile code, program chips, and debug all in the same application. Combined with the In Circuit Debugger 2 (ICD2), it is possible to debug code while running on the chip in real-time. The ICD2 has the capability to have three hardware breakpoints, step through C and/or ASM code line by line, and view register, data, and program memory contents.

Microchip also provides a free version of C18 which is their C compiler, assembler, and linker for the PIC18 series of microcontrollers. C18 comes with many libraries similar to those found in a Unix environment.

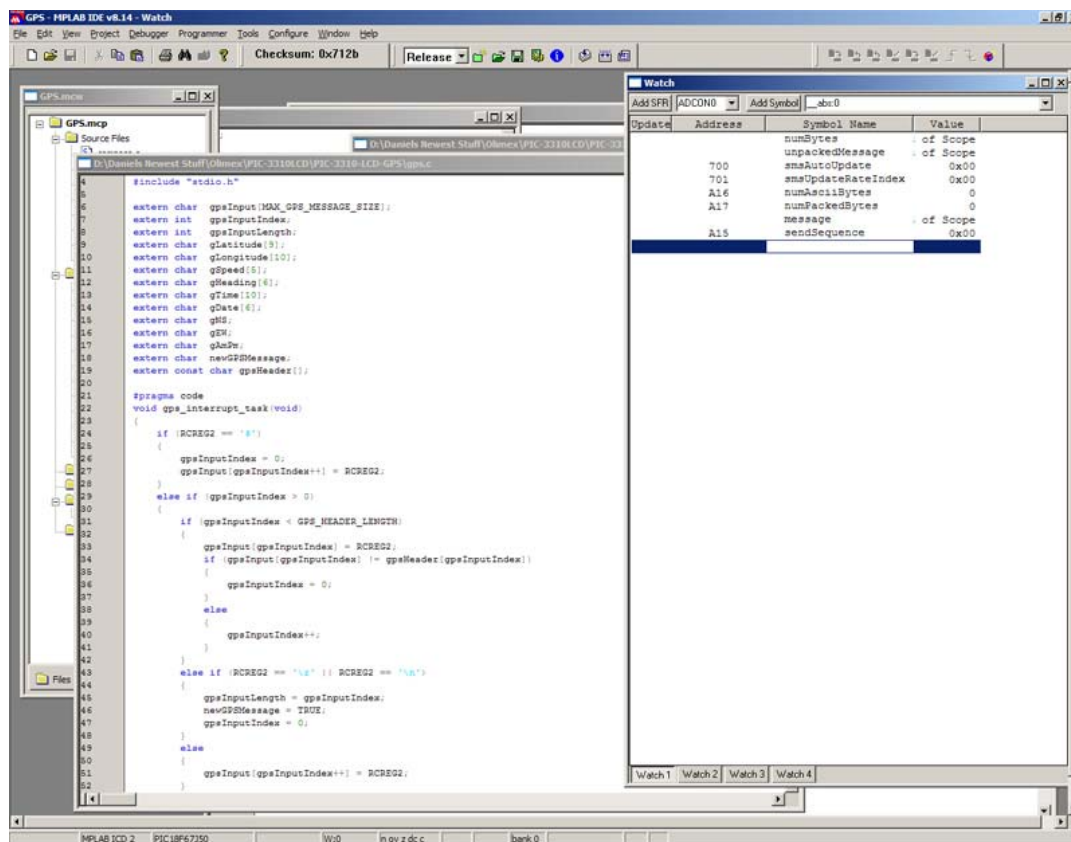


Figure 19. Microchip's MPLAB IDE is shown above with the code editing and variable Watch windows open.

4.2 Hardware Testing

In order to test the cell phones, I soldered wires onto the contact pads in the back of the phones buried under the battery. These contacts are the Tx, Rx, and Gnd needed to communicate with the phone over F-Bus. The M-Bus pin is not used because F-Bus is much faster.



Figure 20. The Nokia F-Bus and M-Bus contact pads are shown above.^[23]

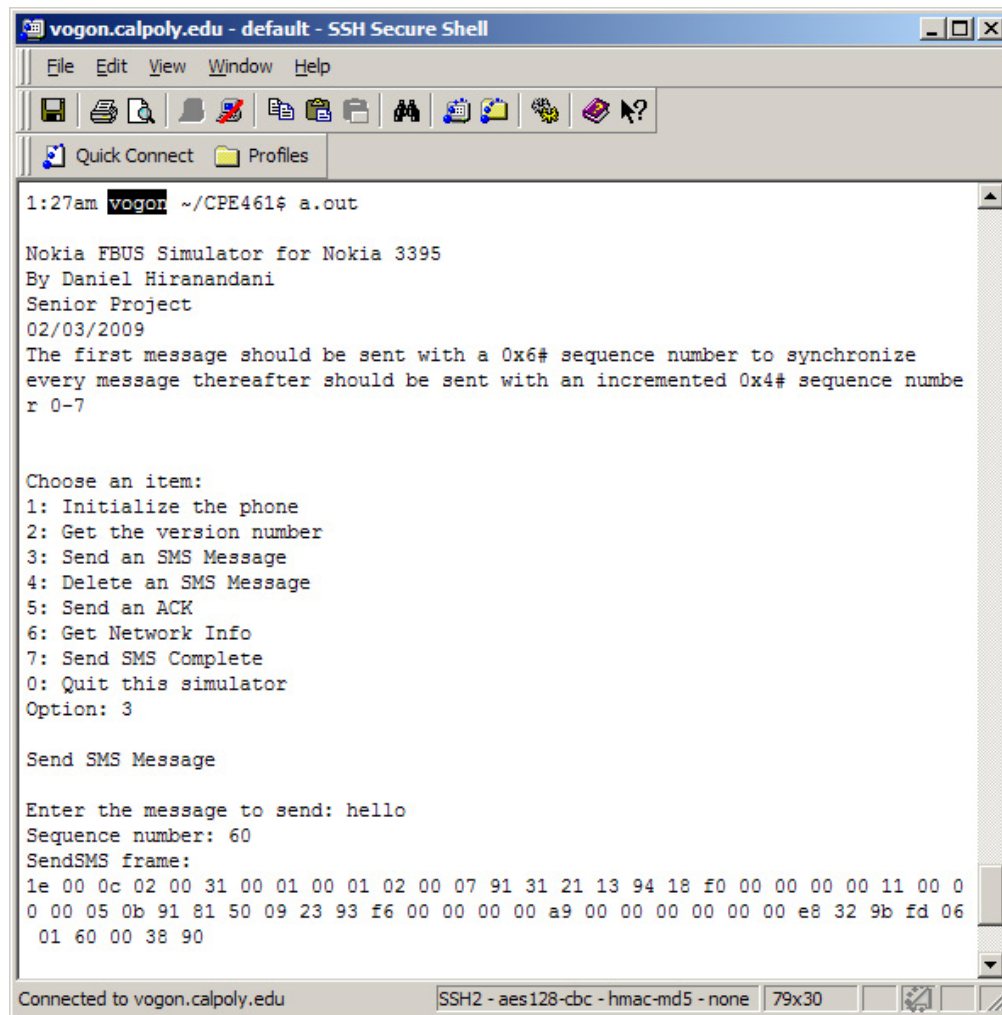
With a connection directly to the phone, I coupled the phone to a RS-232 to TTL voltage shifter which allowed me to have my computer talk to the phone without having to develop any test software or additional circuitry. I used Gnokii to communicate with the phone and sent test SMS messages to verify that it was capable of receiving commands through the F-Bus.

Testing the GPS receiver was a very similar process to testing the cell phones, except I only had to connect the receiver to the RS-232 shifter and then open up HyperTerminal. The GPS receiver has a status LED that stays solid when it is acquiring a GPS fix and flashes when it has a fix, so the only other step was to ensure that data was being transmitted from the device which HyperTerminal confirmed that it was.

Testing the two development boards was also quite easy because Olimex ships them with test code. Out of the box, I was able to have the web server displaying a test webpage along with the remote board flashing LEDs and displaying different screens on the LCD.

4.3 Software Testing

One of the biggest challenges with this project was perfecting the SMS communication code. Relying heavily on the Gnokii documentation for the Nokia 6110 family of cell phones which the 3310s fall into, I created a primitive F-Bus simulator. The simulator responded only to the commands needed for the project, and it was a great way to test the 7-bit packing algorithm before implementing it on hardware.



```
vogon.calpoly.edu - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
1:27am vogon ~/CPE461$ a.out

Nokia FBUS Simulator for Nokia 3395
By Daniel Hiranandani
Senior Project
02/03/2009
The first message should be sent with a 0x6# sequence number to synchronize
every message thereafter should be sent with an incremented 0x4# sequence number 0-7

Choose an item:
1: Initialize the phone
2: Get the version number
3: Send an SMS Message
4: Delete an SMS Message
5: Send an ACK
6: Get Network Info
7: Send SMS Complete
0: Quit this simulator
Option: 3

Send SMS Message

Enter the message to send: hello
Sequence number: 60
SendSMS frame:
1e 00 0c 02 00 31 00 01 00 01 02 00 07 91 31 21 13 94 18 f0 00 00 00 00 11 00 0
0 00 05 0b 91 81 50 09 23 93 f6 00 00 00 00 a9 00 00 00 00 00 e8 32 9b fd 06
01 60 00 38 90

Connected to vogon.calpoly.edu SSH2 - aes128-cbc - hmac-md5 - none 79x30
```

Figure 21. Pictured is the Nokia F-Bus Simulator which generates the basic SMS command messages used for testing.

The rest of the software testing was done while running on the development board hardware because inter-device communication was a key aspect of the project. My methods of testing were to monitor the I/O of the boards through my PC's serial port, and use the ICD2 to set breakpoints in the code and view variable contents at runtime. With these two methods, I was able to resolve the vast majority of the memory leaks, logic errors, and wiring issues I encountered.

4.4 Other Valuable Tools

To help with debugging, I found that testing without these three tools would have been much more time consuming as well as frustrating.

An RS-232 to TTL Voltage shifter was a key component in developing and testing code for the different serial communications in this project. I used two from Sparkfun Electronics to listen to commands sent to the cell phone from the development board as well as to listen to messages received back from the phone. Having two voltage shifters for the command/response

style protocols was essential because I could look at both streams of data at the same time as well as review the timing between the data exchanges. These devices take a power input as a reference for what voltage to convert up or down to which makes them very versatile because they can be used with circuits using voltages other than 3.3V.

Also specifically for serial communication, I used several different terminal applications to view and log data streams. Microsoft's HyperTerminal proved to be the easiest to use for displaying GPS data since it arrives in plain ASCII, SerialPort Tool's Comm Operator was used for hexadecimal data transmission and especially for its timestamps, and Virtual Integrated Design's RS232 Hex Com Tool was also used for viewing hexadecimal communication. RS232 Hex Com Tool was the most reliable for monitoring hexadecimal data because I was unable to change the carriage return/line feed character defaults in Comm Operator, and 0x0D 0x0A is a valid sequence of bytes in the F-Bus protocol. When the phone would transmit 0x0D 0x0A as data and not as a carriage return/line feed, RS232 Hex Com Tool would display it with ease while Comm Operator would start logging on a new line instead of displaying the two bytes. RS232 Hex Com Tool also has the capability to run simple macros, so I was able to test sending commands with static delays to the cell phone before I had an interface on the development boards.

In testing all of the different components, having breadboard jumper wires were extremely useful to have. These are simply short wires with male or female header connections on the ends, but they made it very easy to change between testing the separate SMS and GPS devices.

5. Implementation

5.1 Overview

In order to complete this project, the first step was acquiring all of the pieces of hardware and setting up the GSM accounts. Since I had been planning the details for a few years, I had time to research the different components prior to starting construction.

I chose the PIC-MAXI-WEB as the web server because I became familiar with Microchip's TCP/IP Stack through an internship, as well as for the board's memory and wide variety of I/O. The choice for a remote development board was fairly simple because I wanted a platform where I could easily port any code between the two devices. The PIC-LCD3310 was a perfect candidate because it uses a PIC18 series microcontroller, it was small enough to be portable, and because it has an LCD which I used to display coordinate information.

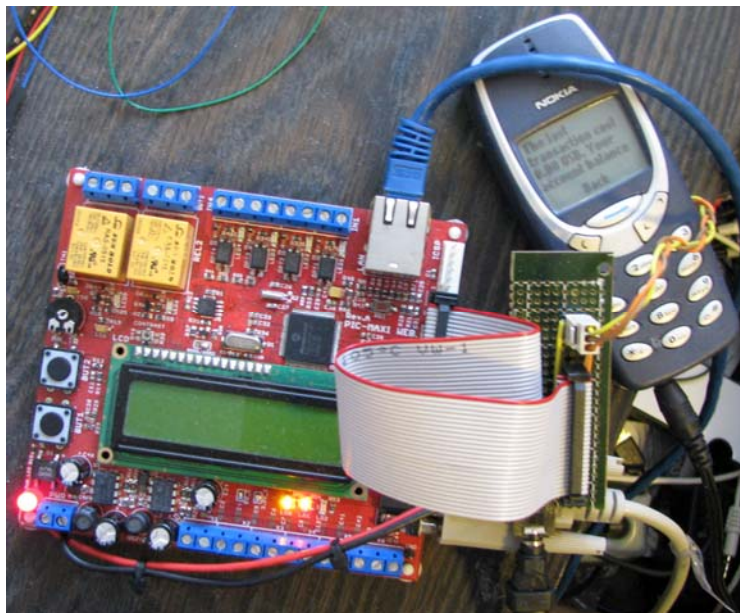


Figure 22. Shown above is the Olimex Web Server board coupled to a breakout board which supplies the connection to a Nokia 3390 cell phone, as well as power.



Figure 23. The remote device displaying its main screen is shown connected to a cell phone as well as to the GPS receiver.

For data communication, I decided to use SMS because it is a fairly inexpensive option for the features it provides. The biggest advantage for using SMS is the coverage, and having coverage almost anywhere in the world allows the remote device to be tracked continuously. The Nokia 3390s became the phone of choice because they supported Nokia F-Bus, the F-Bus contact pads are fairly easy to access under the battery, and they are inexpensive.

Once each of the hardware pieces were tested individually, I created a simple Google Maps webpage that would get coordinates from the web server, and display the location with a marker. The ten test coordinates were pre-programmed into the web server's memory and would increment after a few seconds to test the Google Map's ability to automatically update the marker location. I made use of the XMLHttpRequest function calls to retrieve the latest coordinate information from the web server without having to issue a page refresh command which gives seamless updates. In the displayed web page, the GPS information is retrieved from the web server memory one field at a time because this is what the Microchip's TCP/IP Stack v4.18 supports.

The next step was to attach the GPS receiver to the remote device through the one of the Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) ports which was used like a Universal Asynchronous Receiver Transmitter (UART). A UART is used for serial communication which is exactly how the GPS receiver transmits data. I configured the UART for 4800bps communication, connected the transmit pin (Tx) on the GPS receiver to the receive pin (Rx) on the PIC-LCD3310, and vice versa for the receiver's Rx pin, then wrote the software to decipher the NMEA GPRMC sentence.

I found the easiest way to decode the GPS messages was to have any change on the remote device's Rx pin generate an interrupt and then act based on the current character received along with the prior characters received.

Every time an interrupt is generated, the interrupt service routine (ISR) takes the current character and compares it to what the value should be if it is a GPRMC sentence. The remote device starts storing the contents of a new message when a '\$' is received, and continues storing characters if the message is a GPRMC sentence, otherwise waits for another '\$' to start over. Once the GPRMC header has been matched, the rest of the data is then stored until a newline character is received signaling the end of the sentence.

The remote device then processes the entire sentence including converting the latitude and longitude values from hours, minutes, seconds notation into decimal notation. It also converts the speed from knots to miles per hour and adjusts the UTC time to PST among other tasks. These values are stored in memory, and also updated for the LCD screen display.

The LCD displays two GPS information screens that show compass information and raw status information. The compass displays the heading and speed along with a visual representation of a compass, and the status screen displays the raw latitude, longitude, heading, and speed values.



Figure 24. The remote device's status screen is shown on the left, and the compass screen is shown on the right.

Along with the compass and status screens, there is a settings screen which allows the user to change the automatic update settings for how often to send coordinate data to the web server if at all, as well as options to force a single update or to reboot the cell phone in case of errors.



Figure 25. A view of the remote device's settings screen.

The remote device also is connected to a Nokia 3390 cell phone through a second EUSART port which is also used as a traditional UART. In order to communicate with the cell phone, the remote device must first send an initialization sequence of 128 0x55 characters which allows the phone to send and receive data through the F-Bus. With the help of Gnokii's documentation as well as software testing platform, I found that there is a certain sequence of commands needed in order to send an SMS message as opposed to simply sending the phone a send SMS command. The somewhat convoluted sequence is as follows:

- Send 128 bytes of 0x55 to the phone to initialize phone communication
- Enable the extended command set
- Request the International Mobile Equipment Identity (IMEI) code which uniquely identifies the phone
- Enable the extended command set
- Request information about the phone including model and firmware version
- Enable the extended command set
- Request information about the phone including hardware version
- Request the SMSC number
- Request network status information
- Request to send an SMS message which includes the message to be sent

Initially I had problems reliably sending data from the development boards, but following this sequence produces very reliable results.

When an SMS message is received, the remote board and the web server board perform different functions, but the handling process is the same for both and is similar to how the GPS data is processed. First, the phone automatically sends the data to the development board for processing. When a byte is sent over the F-Bus, an interrupt is generated and the ISR compares the current byte to expected values of messages that should be processed. If there is a match, then the board continues to store the data until the number of expected bytes is filled. When the

entire message has been received, it is then processed to convert the 7-bit packed message into ASCII, and then the web server and remote board handle the data in the following ways.

The web server only receives data messages, so it updates coordinate information in its memory with the latest GPS data when a new message is received. In contrast, the remote device only receives control messages, so it processes any incoming message by setting automatic update settings including turning automatic update on or off and the frequency of updates, as well as has the capability to enable or disable outputs on the remote device.

In addition to receiving a control message to change the automatic update settings, the remote board settings can also be modified through the webpage displayed by the web server. When a user clicks on one of the Update, Start, Stop, or Single Update buttons, an SMS message is generated by the web server and sent to the remote device to change the automatic update settings.

As a result of all the hardware and software integrated into one system, the user would view a webpage shown below:

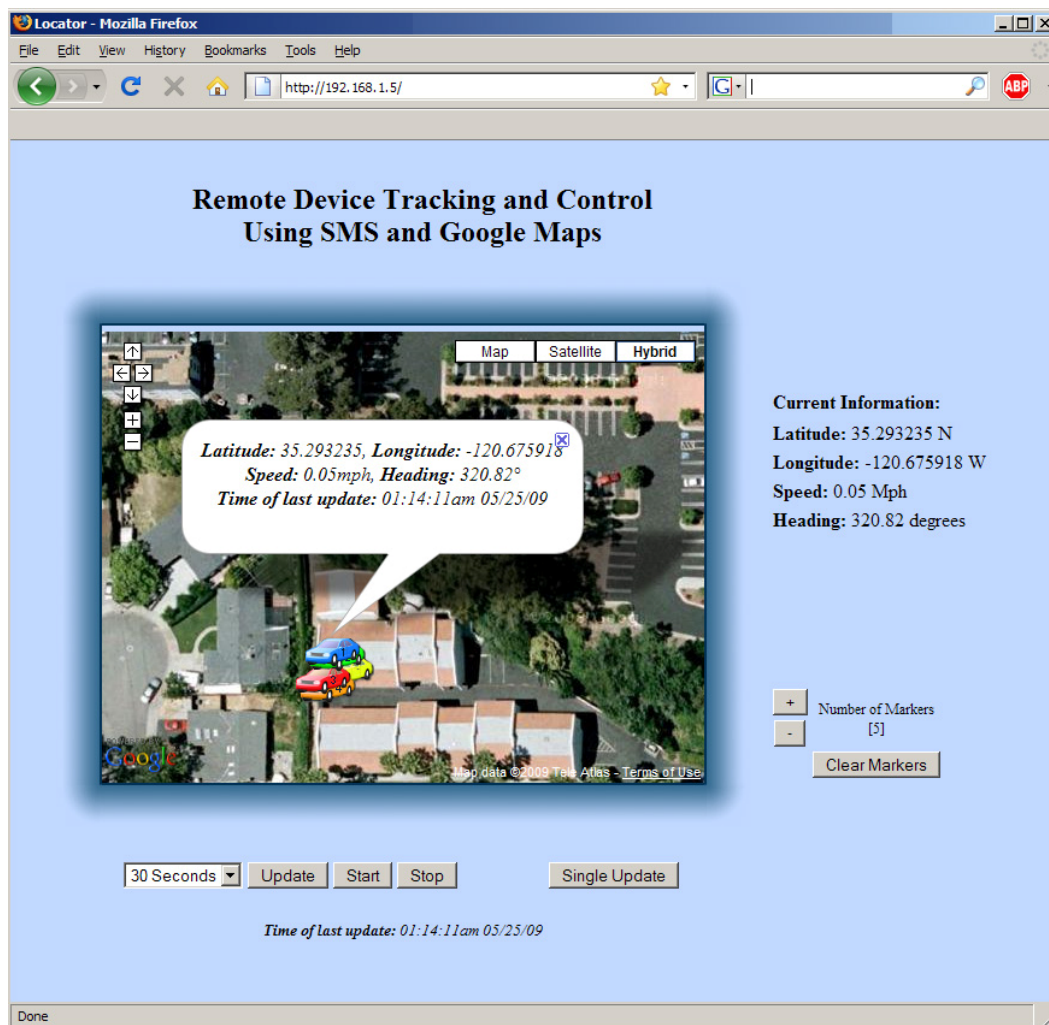


Figure 26. Shown above is a screenshot of the Google Maps webpage with control buttons that was used in this project.

5.2 Hardware Block Diagrams

5.2.a Web Server

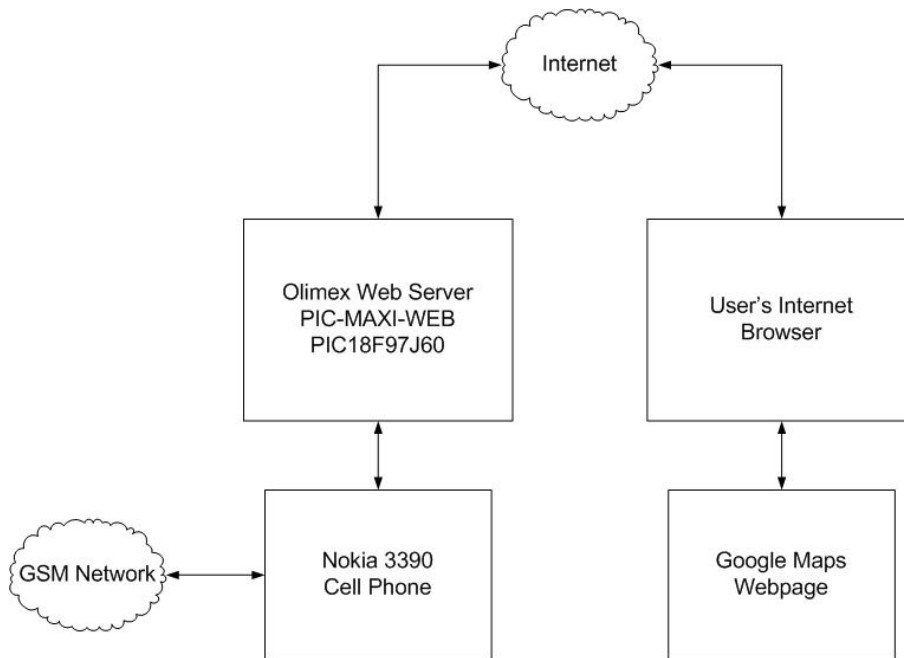


Figure 27. Shown above is the hardware block diagram for the web server.

5.2.b Remote Device

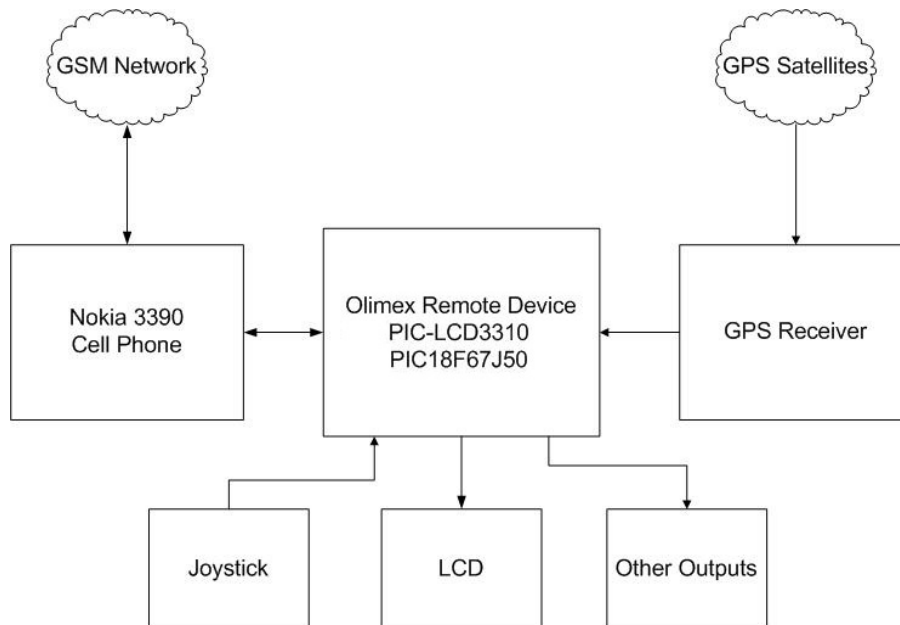


Figure 28. The hardware block diagram is shown for the remote device.

5.3 Software Flow Diagrams

5.3.a Web Server

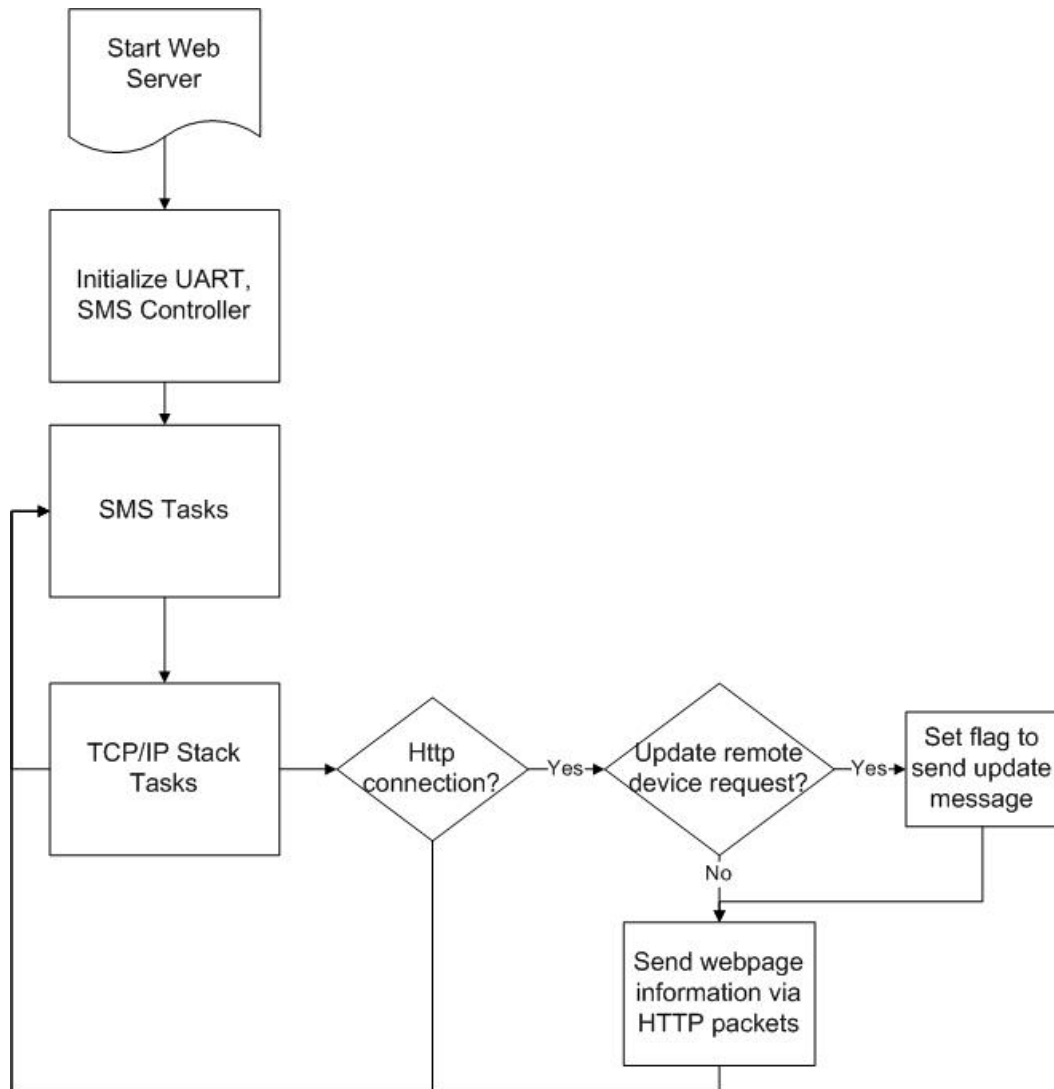


Figure 29. The flow diagram for the web server's main function.

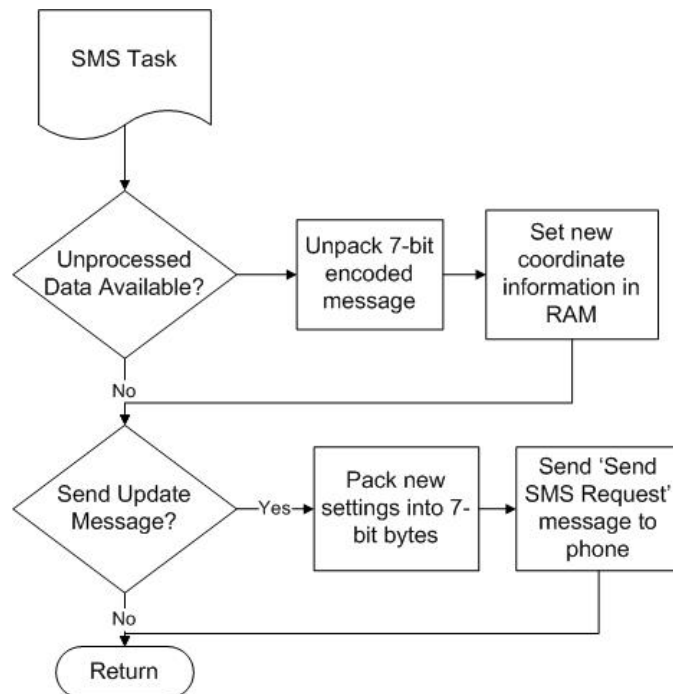


Figure 30. The SMS task function for the web server.

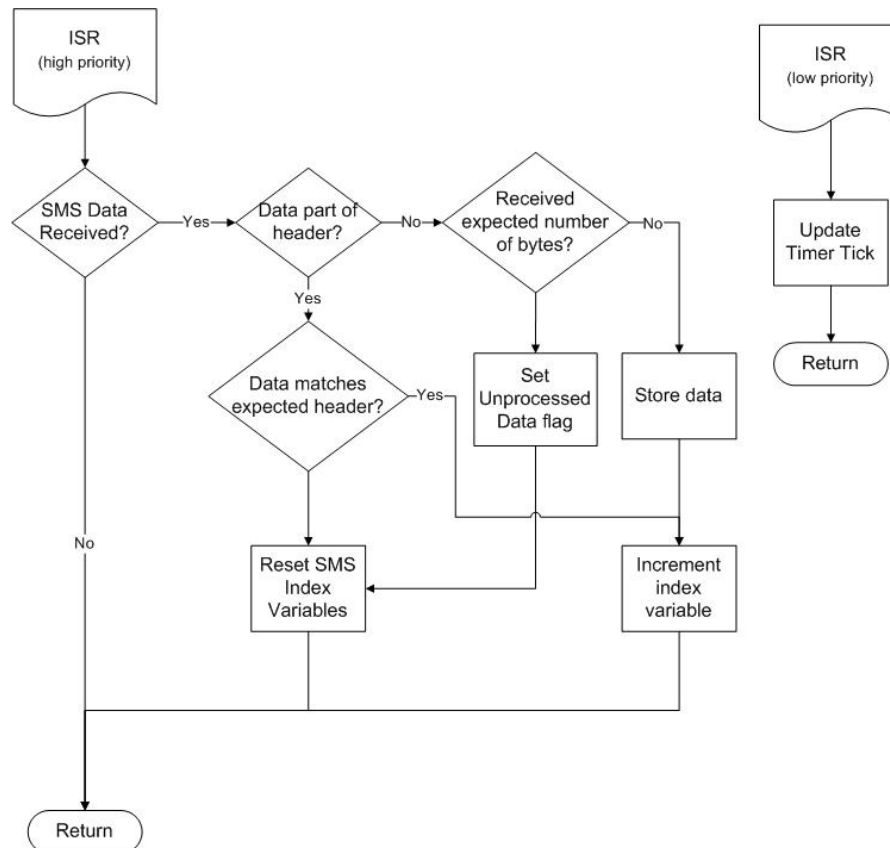


Figure 31. The web server's Interrupt Service Routine (ISR) is shown above.

5.3.b Remote Device

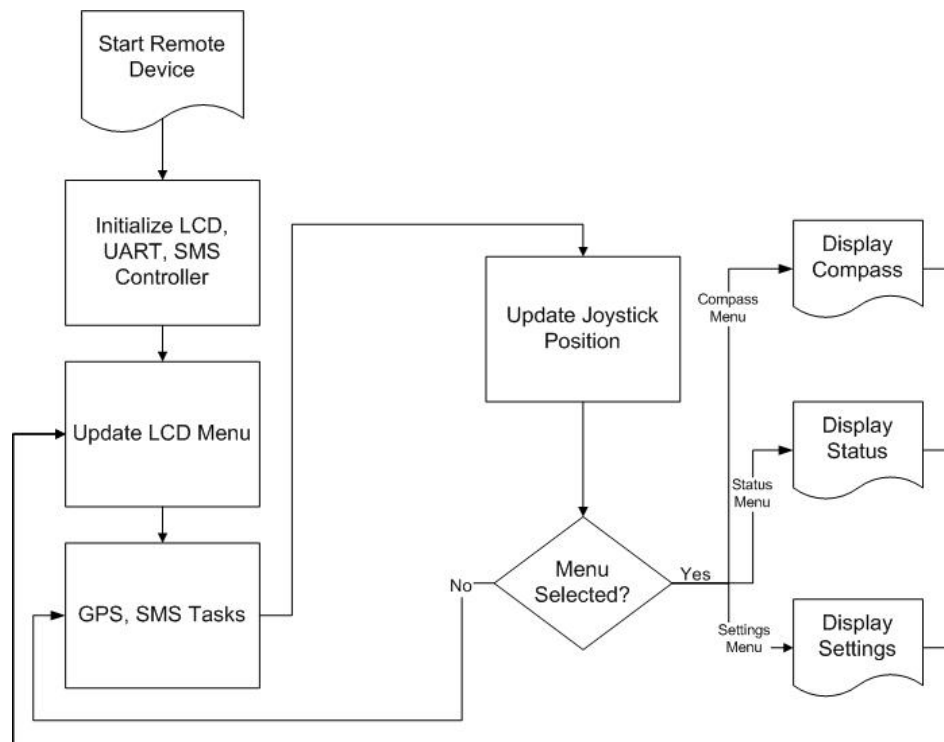


Figure 32. The software flow diagram for the remote device main program flow.

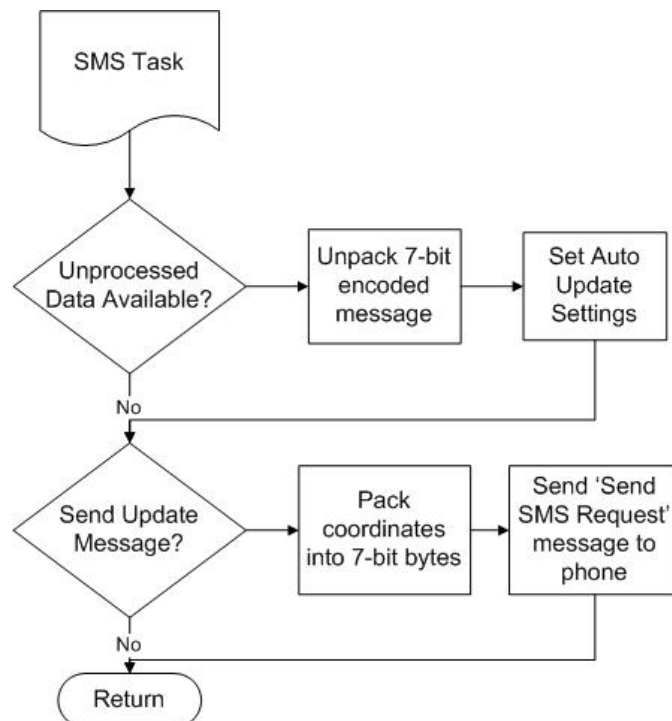


Figure 33. Above is the software flow diagram for the remote device's SMS Task function.

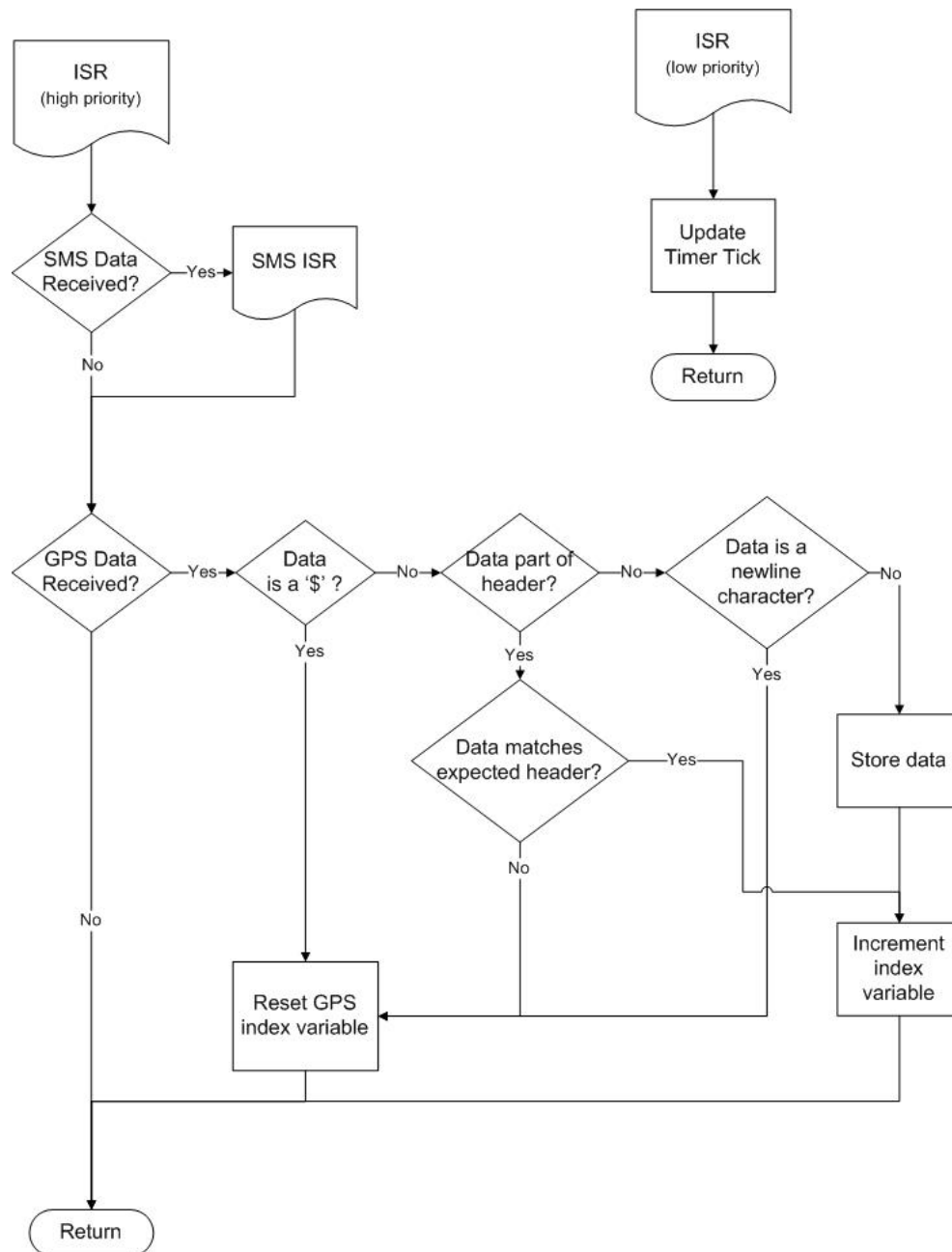


Figure 34. The software flow diagram for the remote device's ISR is shown above.

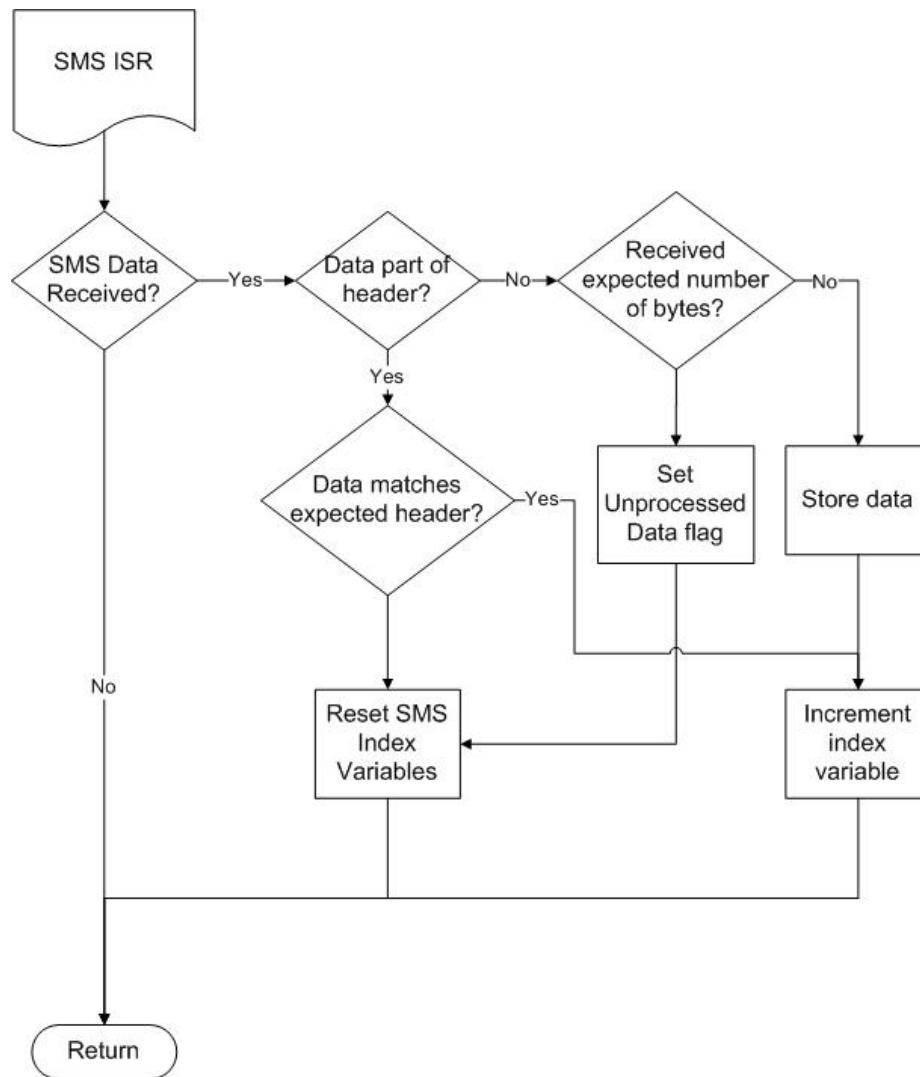


Figure 35. The remote device's SMS ISR

6. Applications

6.1 Vehicle Tracking

The first, and most obvious, application of this is automobile tracking. Making the device compact, powered by the vehicle, and hidden would be easy to do if manufactured. It would also be straightforward to connect I/O pins from the remote device to the car's security system, electric windows, horns, lights, or even Electronic Control Unit (ECU) through the On-Board Diagnostics II (OBD-II) port to have access to view or control many aspects of the vehicle.

6.2 Remote Device Control

Any electronic device has the capability to be controlled by this system. The remote device has many outputs that are capable of enabling or disabling the device, or even sending data to a device that could be carried in the SMS message.

6.3 Event-Triggered Responses

It is possible to have event-triggered responses generated by the remote device such as proximity to an area, exceeding a speed threshold, or even exceeding an acceleration threshold. The speed and location information would come from the GPS data, and the acceleration information would be generated by the accelerometer that is built into the remote device. The accelerometer would provide more accurate instantaneous data compared to the acceleration data that would be interpolated from the GPS data which helps increase the accuracy of data.

7. Future Improvements

7.1 Overview

While working on this project, I have kept a list of future improvements which continues to grow.

The TCP/IP Stack used in this project is Microchip's Stack v4.18. There are currently versions up to v5.00 which support more complex features such as dynamic XML. This would mean all of the GPS information could be updated in a single step in the web page compared to one field at a time that is currently implemented.

Another feature would be to make the SMS controller more generic than only using the Nokia F-Bus. The AT Command Set is supported by most new phones, and uses ASCII for the command messages instead of hexadecimal like the F-Bus. This would be a much more user-friendly command set to work with from a developer's standpoint, and would also support many more phones than just Nokias.

One possibility would be to connect an image capturing device to the remote board, and have the capability to send images over SMS as a Multimedia message. This would be very beneficial for security applications such as in a car, home, or office because the notification would normally be within a minute of an event triggering the remote device such as an alarm system.

Another nice-to-have would be to explore the Google Maps application that is now supported in many cell phones. Instead of using a web server, the user could have coordinates sent directly to their phone and automatically update the location within the Google Maps application. This would be much more viable as a consumer product because the system wouldn't require the extra hardware and configuration for the web server.

8. Ethical Considerations

As with many new applications of technology, there are a few ethical concerns that are raised from this project.

The first concern is that it would be possible to track a user without their knowledge or consent. If a user knows that the device is present, they would have the ability to disable the device in software by choosing a 'Disallow Updates' option, or through hardware simply by turning off the components. However, if the target has no knowledge of the device in the car, there would be no indication that they were being tracked.

Another concern is power consumption because power inevitably impacts the environment. The devices used in this project are fairly low power devices to begin with, but the area of improvement would be to put the devices to sleep when they are not being queried. The remote device could go into a low power mode until it is requested to update its location, and the web server could do the same until it receives a TCP socket connection request.

Accuracy of the location is another issue, because if a person is relying on this information, they will expect it to be correct. GPS has its own accuracy measure down to a few meters, but there are times when the data can become skewed especially when few satellites are in view. One issue in particular is the calculated speed. Since the speed is calculated from difference in the previous location and the current location divided by the time interval, the speed will inherit the error margins from both of the calculated locations which could lead to artificially high maximum speed measurements.

Lastly, accessibility of the information needs to be considered. With the position and time data updating to the Internet, it would be possible for anyone to view that information. Due to the nature of this system, knowing the location of a person or object with a resolution of twenty seconds could be considered private information that should be protected. Implementing a secure web server that uses encryption and authentication could solve most aspects of this problem, but there will always be a possibility that the information could be accessed since it is connected to the Internet.

9. Conclusion

This project is the culmination of my undergraduate learning at Cal Poly. In the classes I have taken, I have become very interested in embedded systems, computer networks, and wireless communication. This project is dependent on all three of these topics, which is why I have been very eager to implement the design for the last several years.

Over the course of this project, I put my research, programming, and debugging skills to great use, and certainly learned quite a few things along the way. I had a high-level understanding of the F-Bus, GPS, and SMS specifications before I started, and now feel like I have a fairly good grasp of the details that went into designing these systems.

I consider this project a success in that it met all of the goals of the project proposal, and because of the knowledge and pride I gained from the experience.

10. Acknowledgements

Cameron Leslie was a great source of ideas and inspiration for this project. He and I first thought of the idea to use SMS for data transmission and GPS tracking several years ago, and he has encouraged me ever since we first thought up the idea for this project.

Ken Tsoi was very generous for giving me the two cell phones and their accessories to use for this project. Ken went out of his way to make sure I had everything cell-phone-related that I needed for the project which was a great help.

Dr. Hugh Smith guided me through this process and gave me great advice throughout. He kept me on track to be able to finish the project and paper, as well as encouraged me with his support.

My parents also supported and encouraged me through the duration of this project. They aided me financially and mentally, as well as physically by taking me out to lunch.

11. References

- [1] United States of America. Department of the Army. US Army Corps of Engineers. NAVSTAR Global Positioning System Surveying. 1996.
- [2] The Three GPS Segments. Digital image. GPS Report - What is GPS? <<http://infohost.nmt.edu/~mreece/gps/segments.gif>>.
- [3] "Global Positioning System (GPS)." AviationExplorer.com. <http://www.aviationexplorer.com/Global_Positioning_System_GPS.html>.
- [4] "GPS explained: Error sources." 19 Apr. 2009. <<http://www.kowoma.de/en/gps/errors.htm>>.
- [5] "GPS Frequently Asked Questions." U.S. Coast Guard Navigation Center. <<http://www.navcen.uscg.gov/faq/gpsfaq.htm>>.
- [6] Lefkow, Chris. "GPS satellites not 'falling out of the sky': Air Force." PhysOrg.com. 21 May 2009. <<http://www.physorg.com/news162133400.html>>.
- [7] Massat, Paul, and Wayne Brady. "Optimizing Performance Through Constellation Management." The Aerospace Corporation. 2002. <<http://www.aero.org/publications/crosslink/summer2002/03.html>>.
- [8] "What is GPS?" About GPS. Garmin. <<http://www8.garmin.com/aboutGPS/>>.
- [9] "GPS Tutorial." Code-Phase GPS vs. Carrier-Phase GPS. Trimble. <http://www.trimble.com/gps/sub_phases.shtml>.
- [10] "GPS Tutorial." Measuring Distance. Trimble. <<http://www.trimble.com/gps/howgps-measuring.shtml>>.
- [11] Intersection of calculated-distance spheres from GPS satellites. Digital image. GPS: The Basics. 1 June 2004. <http://www.sco.wisc.edu/gps/gps_graphics/inter_spheres.jpg>.
- [12] Pfost, Donald, and William Casady. "Precision Agriculture: Global Positioning System (GPS)." University of Missouri Extension. Nov. 1998. <<http://extension.missouri.edu/publications/DisplayPub.aspx?P=WQ452>>.
- [13] Chivers, Morag, and Trimble. "Differential GPS Explained." ESRI. 2003. <<http://www.esri.com/news/arcuser/0103/differential1of2.html>>.
- [14] "How WAAS Works." 2005. Federal Aviation Administration. <http://www.freeflightsystems.com/waas_howitworks.htm>.
- [15] Hudnut, Kenneth. "Future Navigation Needs Your Input - GPS World." GPS World. 1 Nov. 2005. <<http://www.gpsworld.com/gpsworld/article/articleDetail.jsp?id=192974>>.
- [16] DePriest, Dale. "NMEA data." <<http://gpsinformation.org/dale/nmea.htm>>.
- [17] Milian, Mark. "Why text messages are limited to 160 characters." 3 May 2009. Los Angeles Times.

- <<http://latimesblogs.latimes.com/technology/2009/05/invented-text-messaging.html>>.
- [18] GSM 03.40 v7.4.1. Global System for Mobile Communications. Technical realization of the Short Message Service (SMS). 1998.
 - [19] GSM 03.41 v7.4.1. Global System for Mobile Communications. Technical realization of Cell Broadcast Service (CBS). 1998.
 - [20] GSM 03.38 v5.3.1. Global System for Mobile Communications. Alphabets and language-specific information. 1996.
 - [21] GSM 03.42 v7.1.1. Global System for Mobile Communications. Compression algorithm for text messaging services. 1998.
 - [22] GSM 07.05 v5.5.0. Global System for Mobile Communications. Use of Data Terminal Equipment - Data Circuit terminating; Equipment (DTE - DCE) interface for Short Message Service (SMS) and Cell Broadcast Service (CBS). 1998.
 - [23] "Nokia F-Bus Protocol made simple." Embedtronics. 9 Apr. 2005. <<http://www.embedtronics.com/nokia/fbus.html>>.
 - [24] Yanowitz, Jason. "Under the hood of the Internet: An overview of the TCP/IP Protocol Suite." ACM Crossroads 1 (2004). The Association for Computing Machinery. <<http://www.acm.org/crossroads/xrds1-1/tcpjmy.html>>.
 - [25] "How the Application Layer Works." Learn Networking. 27 Jan. 2008. <<http://learn-networking.com/tcp-ip/how-the-application-layer-works>>.
 - [26] "Ethernet: IEEE 802.3 Local Area Network (LAN) protocols Overview." Javvin Network Management & Security. <<http://www.javvin.com/protocolEthernet.html>>.
 - [27] Bryant, Chris. "How Ethernet CSMA/CD Works." MC MCSE. <<http://www.mcmse.com/cisco/guides/csma.shtml>>.
 - [28] Ethernet frame structure. Digital image. Hardware Addressing. 24 Nov. 2005. <<http://penguin.dcs.bbk.ac.uk/academic/networks/data-link-layer/hardware-addressing/ethernet.gif>>.
 - [29] A typical IP packet. Digital image. Learning Guide: How does VoIP work? 2 June 2006. <http://media.techtarget.com/digitalguide/images/Misc/voip_5.gif>.
 - [30] Darpa Internet Program. RFC 793. Transmission Control Protocol. 1981.
 - [31] Postel, Jon. RFC 768. User Datagram Protocol. 1980.
 - [32] TCP Packet Header. Digital image. System Management Guide: Communications and Networks. <http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/IBMp690/IBM/usr/share/man/info/en_US/a_doc_lib/aixbman/commadmn/figures/comma38.jpg>.
 - [33] "Google Maps API." Google Code. <<http://code.google.com/apis/maps/>>.

- [34] Google Maps satellite view of Cal Poly, SLO campus. Digital image. Google Maps. <<http://maps.google.com>>.
- [35] "History of Ajax Components." Learn-AJAX. AjaxWith.com. <<http://www.ajaxwith.com/History-of-Ajax-Components-.html>>.
- [36] Brain, Marshall. "How Microcontrollers Work." Howstuffworks. <<http://electronics.howstuffworks.com/microcontroller1.htm>>.
- [37] Two Atmel microcontrollers shown with a match for scale. Digital image. Electronic components. <<http://www.guboweb.de/i1317675-1.jpg>>.
- [38] PIC-MAXI-WEB. Digital image. Olimex Ltd. <<http://www.olimex.com/dev/images/PIC/PIC-MAXi-WEB.jpg>>.
- [39] PIC-LCD3310. Digital image. Olimex Ltd. <<http://www.olimex.com/dev/images/PIC/PIC-LCD3310-2.jpg>>.

12. Appendix

12.1 Glossary

- AJAX (Asynchronous JavaScript and XML) – A grouping of several programming techniques used to create dynamic webpages.
- API (Application Programming Interface) – The developer's interface into a software library.
- C18 – Microchip's Compiler for PIC18 microcontrollers
- Cloud – In computer networks, a cloud is an idea used to describe any type of network. It can be of any topology and simply represents an unknown section of the network.
- CSS (Cascading Style Sheets) – A language used to describe the style and formatting of a document or webpage.
- EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter) – A communication channel on a microcontroller used for sending and receiving serial data.
- F-Bus (Fast-Bus) – Nokia's communication protocol for transfer information between a cell phone and a computer. 115200 bits per second, 8 bits per byte, no parity bit, 1 stop bit.
- GPS (Global Positioning System) – A constellation network of satellites used by the military and civilians for the purpose of calculating a user or object's location on the Earth.
- GSM (Global System for Mobile Communications) – A standard for telecommunication used in mobile phones.
- HTML (Hypertext Markup Language) – The language used to create webpages to be displayed by a web browser.
- HTTP (Hypertext Transfer Protocol) – A layer 4 protocol in the TCP/IP Stack that is most known for carrying webpage information.

- ICD2 (In Circuit Debugger 2) – A device used to debug and program software onto Microchip microcontrollers. A useful tool for testing software on the hardware to find logical errors.
- IMEI (International Mobile Equipment Identity) – The unique number that identifies a cell phone; similar to a serial number.
- IP (Internet Protocol) – Layer 3 in the TCP/IP Stack. IP is used to route and determines which paths to take to get to the destination.
- MAC (Media Access Control) – A protocol that runs at the data link layer and provides addressing for the connected devices.
- M-Bus – Nokia’s protocol that predates the F-Bus. Running at 9600bps with 8 data bits, 1 odd parity bit, and 1 stop bit, it is a single pin, bi-directional, half-duplex bus.
- Microcontroller – A computer consisting of a processor, memory, input and output, and software. These devices are generally very specialized and perform a small number of functions, so they can be small, inexpensive, and use low power.
- NAVSTAR (Navigation System with Timing And Ranging) – The original name of the U.S. DoD’s research project now known as GPS.
- RMC (Recommended Minimum sentence) – A GPS data sentence containing latitude, longitude, speed, heading, time, and date information.
- RS-232 (Recommended Standard 232) – The serial communication standard that is commonly found on personal computers.
- S/A (Selective Availability) – An artificial error introduced to the civilian GPS signal to keep accuracy low at around 100m.
- SMS (Short Message Service) – Also called Text Messaging, this is a way to send short text messages over the cellular network. SMS is generally used to send messages between cell phones.
- SMSC (Short Message Service Center) – A location that receives SMS messages that are to be transmitted to other locations or devices.
- TCP (Transmission Control Protocol) – Layer 4 in the TCP/IP Stack. TCP has a three-way-handshake for connection setup which includes the SYN-SYN/ACK-ACK, and also has a connection teardown.

- XML (Extensible Markup Language) – A dynamic markup language that is used to create webpages with classified or organized data.

12.2 Parts List

Part	Quantity x Unit Cost		Total Cost
Olimex PIC-MAXI-WEB Web Server Board	1	\$128.00	\$128.00
Olimex PIC-LCD3310 Remote Device Board	1	\$50.95	\$50.95
Olimex ICD2 Programmer/Debugger	1	\$106.95	\$106.95
EM-406A SiRF III GPS Receiver -Needed replacement after over-powering the first	2	\$59.95	\$119.90
RS-232 Voltage Shifter -Needed replacement after over-powering the first	2	\$13.95	\$27.90
3-12V Power Adapter	1	\$11.00	\$11.00
Nokia 3390 Cell Phones	2	\$0.00	\$0.00
AT&T Unlimited SMS (monthly)	2 x 5mo.	\$19.99	\$199.90
Various cables, wires, connectors, etc.	N/A	\$50.00	\$50.00
Total Cost:			\$694.60

12.3 Analysis of Senior Project

Summary of Functional Requirements

The main function is to track a person or object anywhere in the world, and to display the location in a manner that is visually understandable by a human without having to further process the position data. Additionally, the person viewing the location has the ability to send control messages to the remote location to perform some other function.

Primary Constraints

Learning and understanding several specifications, protocols, and programming environments made this project challenging. I needed to understand serial communication, SMS command messages, GPS data sentences, TCP/IP and AJAX for webpage display, and be able to implement all of them using the Microchip C18 compiler for embedded systems. I had experience with all of them at a fairly high level prior to starting this project, but I needed to delve much deeper into all of them to successfully complete the project.

Economic

- Original cost estimate:

Part	Quantity x Unit Cost		Total Cost
Olimex PIC-MAXI-WEB Web Server Board	1	\$128.00	\$128.00
Olimex PIC-LCD3310 Remote Device Board	1	\$50.95	\$50.95
Olimex ICD2 Programmer/Debugger	1	\$106.95	\$106.95
EM-406A SiRF III GPS Receiver	1	\$59.95	\$59.95
RS-232 Voltage Shifter	1	\$13.95	\$13.95
Nokia 3390 Cell Phones	2	\$5.00	\$10.00
AT&T Unlimited SMS (monthly)	2 x 3mo.	\$10.00	\$60.00
Various cables, wires, connectors, etc.	N/A	\$20.00	\$20.00
Total Cost:			\$449.80

- Final bill of materials:

Part	Quantity x Unit Cost		Total Cost
Olimex PIC-MAXI-WEB Web Server Board	1	\$128.00	\$128.00
Olimex PIC-LCD3310 Remote Device Board	1	\$50.95	\$50.95
Olimex ICD2 Programmer/Debugger	1	\$106.95	\$106.95
EM-406A SiRF III GPS Receiver -Needed replacement after over-powering the first	2	\$59.95	\$119.90
RS-232 Voltage Shifter -Needed replacement after over-powering the first	2	\$13.95	\$27.90
3-12V Power Adapter	1	\$11.00	\$11.00
Nokia 3390 Cell Phones	2	\$0.00	\$0.00
AT&T Unlimited SMS (monthly)	2 x 5mo.	\$19.99	\$199.90
Various cables, wires, connectors, etc.	N/A	\$50.00	\$50.00
Total Cost:			\$694.60

- Cost for one device with one month of service (excluding development tools):

Part	Quantity x Unit Cost		Total Cost
Olimex PIC-MAXI-WEB Web Server Board	1	\$128.00	\$128.00
Olimex PIC-LCD3310 Remote Device Board	1	\$50.95	\$50.95
EM-406A SiRF III GPS Receiver	1	\$59.95	\$59.95
3-12V Power Adapter	1	\$11.00	\$11.00
Nokia 3390 Cell Phones	2	\$0.00	\$0.00
AT&T Unlimited SMS (monthly)	2 x 1mo.	\$19.99	\$39.98
Various cables, wires, connectors, etc.	N/A	\$25.00	\$25.00
Total Cost:			\$314.88

I estimated that I would have the project portion fully completed by the end of the first quarter by spending 10-20 hours per week, and then finish the paper portion leisurely over the course of the first half of the second quarter. The actual time spent was closer to 20-50 hours per week for both quarters, and the paper was written while finishing the project near the end of the second quarter. The total estimated number of hours for the project is 400 hours for research, implementation, and documentation.

If manufactured on a commercial basis

I estimate 1000 devices to be sold per year with a manufacturing cost of \$125, and a purchase price of \$175. This would result in a yearly profit of \$50,000 minus overhead costs, so I estimate the total profit per year to be \$35,000. The cost for a user to operate the device per month is \$10 for up to one hundred location updates, \$20 for up to five hundred updates, or \$40 for unlimited location updates. These monthly costs are determined by the cell phone service provider, and these rates reflect the current AT&T prices for two phones.

Environmental

The components in the system are RoHS compliant, so the major impact would be disposal.

Manufacturability

I do not foresee any challenges to manufacturing this design.

Sustainability

There are no abnormal issues with this system regarding sustainability. The only upgrade would be to implement a low-power mode where the devices sleep until they are queried to perform a task.

Ethical

There is the possibility of this system being used to track a person who does not know they are being tracked. For a user who knows they are carrying the device, the user would have the option to mark a 'Disallow Updates' flag so that no further coordinate updates would be sent to the web server.

Health and Safety

There are no abnormal concerns; the components are RoHS compliant.

Social and Political

People might be opposed to the idea of the project, especially when used unethically, however it poses little threat to anyone when used as intended. When used as intended, the user would know they are being tracked and would also know to secure the Internet connection so that only authorized viewers could access the information.

Development

I learned a huge amount about the SMS standard, communicating with a cell phone over F-Bus, the GPS system and how it works, Microchip's C18 compiler, and the PIC memory model, all of which were learned independently. Also, I am now more familiar and comfortable with reading various types of specification datasheets.