Assets Management with GPS

Advanced Microprocessor

SEIS 742, Spring 2013

Alex Shtuk

Contents

[1. Introduction 3](#_Toc356131617)

[2. Requirements 4](#_Toc356131618)

[2.1 Functional Requirements 4](#_Toc356131619)

[2.2 Non-Functional Requirements 4](#_Toc356131620)

[3. Product Specification 6](#_Toc356131621)

[4. Software Requirements Specification 8](#_Toc356131622)

[5. Hardware/Software Interface Specification 9](#_Toc356131623)

[5.1 Block Diagram 9](#_Toc356131624)

[5.2 Hardware Architecture 10](#_Toc356131625)

[5.2.1 NXP LPC1769 CortexM3 Microcontroller 10](#_Toc356131626)

[5.2.2 BeagleBone 11](#_Toc356131627)

[5.2.3 66-Channel LS20031 GPS Receiver Module 12](#_Toc356131628)

[5.2.4 XBee 1mW Wire Antenna –Series 1 Wireless Module (802.15.4) from Digi 13](#_Toc356131629)

[5.3 Assembled System 13](#_Toc356131630)

[6. Software Architecture and Design 14](#_Toc356131631)

[6.1 Top Level 14](#_Toc356131632)

[6.2 Gps 15](#_Toc356131633)

[6.3 Bb 16](#_Toc356131634)

[6.4 Tabs 17](#_Toc356131635)

[6.4.1 Distance Tab 19](#_Toc356131636)

[6.4.2 GGA Tab 20](#_Toc356131637)

[6.4.3 GLL Tab 21](#_Toc356131638)

[6.4.4 GSA Tab 22](#_Toc356131639)

[6.4.5 GSV Tab 23](#_Toc356131640)

[6.4.6 RMC Tab 24](#_Toc356131641)

[6.4.7 VTG Tab 25](#_Toc356131642)

[7. Conclusion 26](#_Toc356131643)

[8. References 27](#_Toc356131644)

# Introduction

In this project I will design and implement a prototype for the Assets Management with GPS embedded system using software tools and hardware that I was introduced to during the last three semesters at University of Saint Thomas including the following courses: SEIS740 Real-Time Operating Systems, SEIS 741 Embedded Microprocessor Design, and SEIS 742 Advanced Microprocessor.

Assets Management with GPS is an embedded multi-processor system that allows monitoring an asset using Global Positioning System and notifies an end-user over a TCP/IP network if an asset leaves the boundaries of allowable region. Some of the uses of such system may include monitoring animals, kids, or equipment. System consists of at least two embedded devices:

1. Mobile Asset Attachment – the device that will be attached to an asset including GPS module used to acquire its position and the satellites information from Global Position System and wireless module used to transmit this data to the Base-station.
2. Base-station - the device that provides information to an end-user over TCP/IP network.

My personal goal for this project was to continue learning embedded system design, software/hardware interface, microcontrollers, wireless communication, digital I/O, Real Time Systems, and get a hands-on experience working with embedded Linux and TCP/IP networking.

In this report I will provide the details of the hardware and software design for the Assets Management with GPS system. This document contains the following sections:

1. Requirements - preliminary product requirements and design.
2. Product Specification – detailed description of system components functions.
3. Software Requirements Specification – description of software requirements.
4. Hardware Architecture – hardware block diagram, hardware selection reasons, and estimate of required memory.
5. Software Architecture – software design.
6. Conclusion – learning experience and implementation challenges.
7. References.

# Requirements

## Functional Requirements

1. System should be ready to operate in less than 1 minute.
2. System should indicate when it’s ready for use.
3. System should indicate the current distance from a Reference Point to an asset.
4. System should provide an option to set a distance limits for an asset.
5. System should generate a warning when an asset distance exceeds the range limits or connection is lost.
6. Maximum range for monitoring an asset should be at least 100 ft.

## Non-Functional Requirements

1. Low Power for the device attached to an asset that would guarantee long operational hours.
2. Small size of the device attached to an asset.
3. Should be able to power a base station from standard electrical outlet 120v 60Hz.
4. Base-station and attachment device should communicate over wireless link.
5. Base-station should provide information about asset to an end-user over the Internet.

|  |  |
| --- | --- |
| **Characteristics** | **Description** |
| Name | Assets Management with GPS |
| Purpose | To provide a position information about an asset to end-user. |
| Mobile Attachment Device Inputs/Outputs | Inputs:   * GPS Module to acquire position information   Outputs:   * Wireless Module to communicate information to the base station. |
| Base Station Inputs/Outputs | Inputs:   * Wireless Module to communicate with Mobile Attachment device   Outputs:   * Network Connector (either RJ45 Ethernet or IEEE 802.11\* wireless) |
| Functions | * Informs end-user when an assets leaves the boundaries of allowable region or connection lost * Maximum distance for monitoring an asset should be at least 100 ft. |
| Performance | On power up this device should be ready to operate in less than 1 minute |
| Manufacturing Cost | Less than $100 |
| Power restrictions | Low power for Mobile Attachment to ensure long operating time |
| Physical Size and Weight restrictions | Low weight of Mobile Attachment Device |

# Product Specification

The following diagram shows all the components that will be included in Assets Management with GPS system followed by detailed description of each component’s specification



1. GPS Module – receives positioning information from GPS and forwards it to Mobile Attachment microcontroller using serial port.
2. Mobile Attachment Microcontroller – receives information from GPS Module over serial link using UART peripheral and forwards it to Wireless Module over serial link using second UART; sends start/restart and other commands back to GPS Module.
3. Wireless Module(s) – communicate information between Mobile Attachment and Base station microcontrollers.
4. Base-station Microcontroller – receives and stores information from Wireless Module; runs TCP/IP multi-process server to handle end-user network client information requests
5. Ethernet Module – communicates information between Base-station microcontroller and end-user’s network client.

# Software Requirements Specification

1. System should be able to update information to client at 2 Hz
2. System should be able to handle multiple client connections simultaneously.
3. Programming language: C or C++
4. Information should be presented to End-User with Graphics-User Interface.
5. End-User should be able to select Range limits for an asset.
6. End-User should be informed of a current distance between base-station to an asset in green color.
7. End-User should be given a warning in red color when the distance between base-station and an asset exceeds allowable range.
8. End-User should be informed when the connection between base-station and an asset is lost.
9. As an option, End-User can be informed of other GPS communication information and modes of operation.
10. The following software components will be required:
11. UART interrupt handler to receive information from GPS module
12. Serial Communication Components to communicate information between Asset Attachment and Base-station devices
13. TCP/IP server and client components to handle network communication
14. Data structures to store and retrieve GPS information
15. National Marine Electronics Association records parser and generator
16. GPS module MTK commands generator
17. Software with Graphic User Interface to present information to End-User.

# Hardware/Software Interface Specification

## Block Diagram

Internet

ETH0

UART1

XBee Wireless Module

XBee Wireless Module

UART2

UART1

LS20031 GPS Receiver Module

BeagleBone

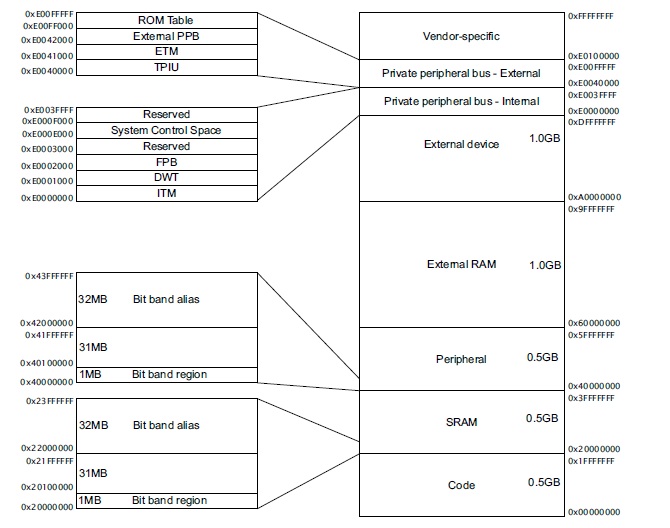
NXP LPC1769 Cortex M3

## Hardware Architecture

### NXP LPC1769 CortexM3 Microcontroller

For this project I decided to go with already familiar NXP LPC1769 Cortex M3 microcontroller to control Asset Attachment. I needed a microcontroller that had at least two UARTs to control GPS receiver and to transmit data to base-station with XBee Wireless module. Also, I needed 1000 byte of RAM to store information received from GPS in circular buffer. I knew from my previous experience working on a project for SEIS 741 Embedded Microprocessor Design that this microcontroller has a lot more of processing power and memory than I needed.

#### Memory Map



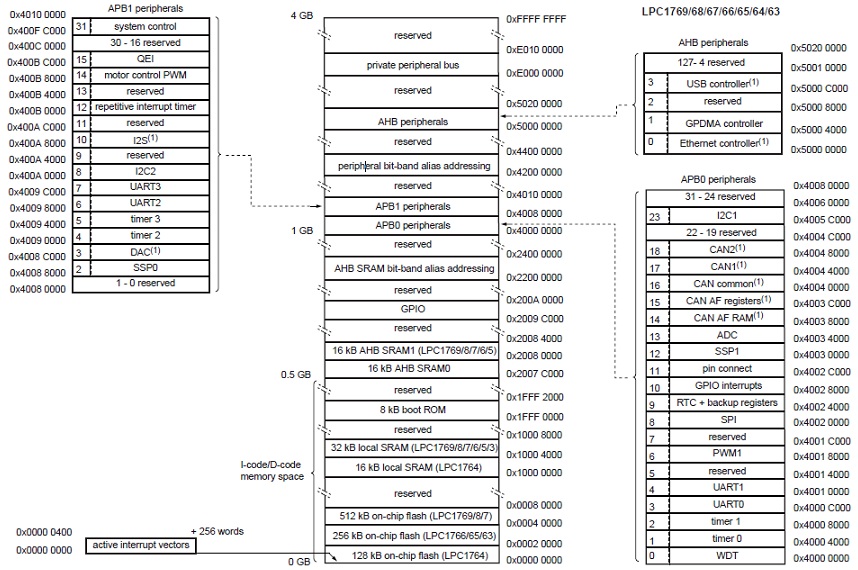
#### Memory Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Text | Data | Bss |
| Consumption (Bytes) | 9852 | 16 | 11980 |

Flash Consumption = Text + Data = 9852 + 16 = 9868 bytes

RAM Consumption = Data + Bss = 16 + 11980 = 11996 bytes

#### IO Map



### BeagleBone

BeagleBone is a very powerful microcontroller based on ARM Cortex A8 processor. I was looking for a microcontroller that would have Ethernet and 3.3v UART ports. Also, I wanted to get some hands-on experience with embedded Linux. For this project the main advantage of having such a multi process operating system running on microcontroller was a very well tested over the years TCP/IP stack and IPC which allowed me to develop a multi-process TCP server to handle multiple client connections that will be described in Software Architecture part.

This is just a few of BeagleBone features that I thought would be important to consider for choosing a microcontroller for a project like this:

* Processor AM3359 500MHZ-USB Powered, 720MHZ-DC Powered
* Memory 256MB DDR2 400MHz
* Debug Support On Board JTAG via USB
* 4 USER LEDs
* Power USB or 5VDC External Jack
* PCB 3.4x2.1 inch
* Ethernet 10/100, RJ45
* Reset Button
* 3.3v IO on all signals
* Available UART ports 1, 2, 4, and 5.

#### Memory Estimation

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Text | Data | Bss |
| Consumption (Bytes) | 7337 | 408 | 32 |

Flash Consumption = Text + Data = 7337 + 408= 7745 bytes

RAM Consumption = Data + Bss =408 + 32 = 440 bytes

### 66-Channel LS20031 GPS Receiver Module

This GPS receiver from Locosys integrates a MediaTek MT3329 66-channel GPS chip with a ceramic antenna. It supports up to a 10Hz update rate.

The main reason I decided to go with this module was availability of documentation and a windows software application from Locosys that I could download. Even though the software source code wasn’t available I could at least test the module with their software or compare the output information. The other advantage among the other modules that I’ve been considering was 3.3v TTL level serial communication that was required for NXP LPC 1769 microcontroller.

Other features include:

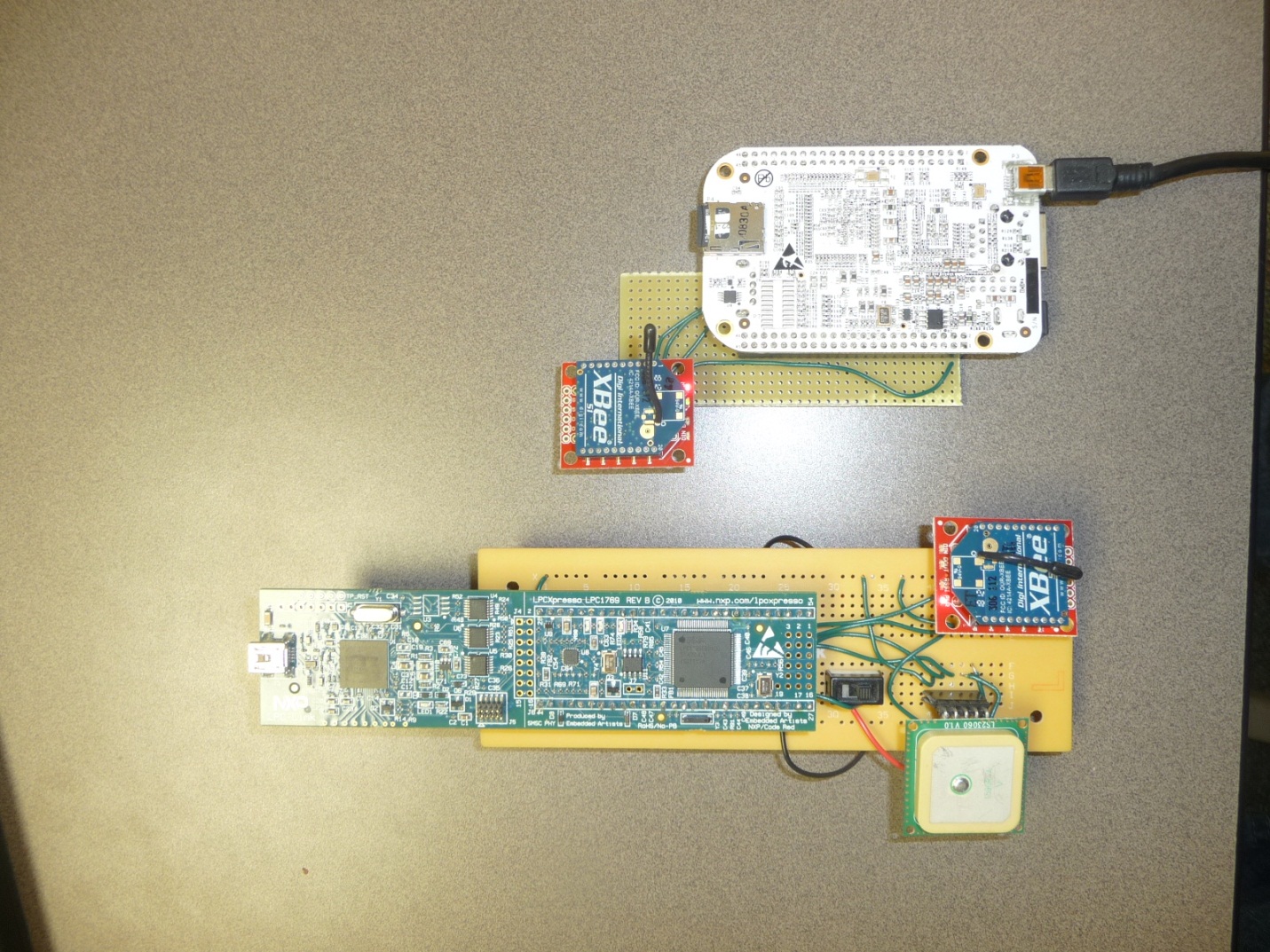
* 35 second cold start, less than 2 second hot start
* 3 to 4.2 v operating range
* Built-in rechargeable battery to preserve system data for rapid satellite acquisition
* Red LED indicator for GPS fix or no fix
* Capable of SBAS (WAAS, EGNOS, MSAS)
* Supports assisted GPS (AGPS)

### XBee 1mW Wire Antenna –Series 1 Wireless Module (802.15.4) from Digi

The following features were the reason this module was selected for this project:

* 3.3V power requirements
* 250 kbps Max data rate
* 300ft maximum operating range
* serial communication
* available documentation, tutorials
* price

## Assembled System



# Software Architecture and Design

Communication with GPS turned out to be not very complicated. LS20031 GPS receiver module generates six different sentences containing positioning and satellites information in the following format:

$GPGGA,053740.000,2503.6319,N,12136.0099,E,1,08,1.1,63.8,M,15.2,M,,0000\*64

First, it includes the message identifier which can be one of the following:

$GPGGA – Global positioning system fixed data

$GPGLL – Geographic position – latitude/longitude

$GPGSA – GNSS DOP and active satellites

$GPGSV – GNSS satellites in view

$GPRMC – Recommended minimum specific GNSS data

$GPVTG – Course over ground and ground speed

Then the data contained in each sentence is separated by ‘,’ delimiter and can be later parsed and translated using Locosys Technology, Inc. Datasheet of GPS smart antenna module, SL20031. Star ‘\*” indicates end of data load followed by XOR checksum that can be used to verify that data load was not corrupted during transmission.

LS20031 also accepts PMTK commands in the following format

$PMTK000\*32<CR><LF>

It always starts with $PMTK followed by command number and flags separated by comma ‘,’ delimiter. As with NMEA sentences, data load is separated from check sum with star ‘\*’. Checksum can be verified using same XOR algorithm. These commands can be used to start the module, set data frequency rate, the type of the data to be supplied by the module, set power saving mode, etc.

The software will be submitted with the project report.

## Top Level

The software for Assets Management with GPS system consists of three packages:

1. Bb
2. Gps
3. Tabs

Each package is built and loaded on a different hardware and will be described in more details in the following subsections.



## Gps

Gps package is software that runs on NXP LPC1769 microcontroller and performs a function of Asset Attachment. When LPC1769 powers up, it sends LS20031 GPS Receiver Module start command and awaits data from the module on UART1 port. When UART1 interrupts, data from GPS receiver is buffered in CIRC\_BUFFER. After UART1 interrupt is done, new data in CIRC\_BUFFER is forwarded to UART2 where it’s picked up by XBee Module and sent to the Base-station.



## Bb

Bb package software is loaded onto BeagleBone microcontroller which performs a function of the Base-station in the system. It start by initializing Inter Process Communication data structures – Shared Memory where it stores NMEA information from GPS, and a semaphore to control an access to shared memory. After initialization, Bb forks a new process and starts a TCP server. In parent process it handles serial port communication where it receives GPS data from NXP LPC1769 microcontroller using XBee Wireless module and stores information in a table located in shared memory. The TCP/IP server process initializes master socket for TCP communication and blocks while listening to a connection from client. After receiving connection, it forks a new process to handle communication session with client and goes back to waiting for other connections, which allows simultaneous communication sessions with multiple clients. Client is allowed to ask only for one NMEA sentence at a time to limit communication channel load, but it can ask for multiple sentences during TCP session.



## Tabs

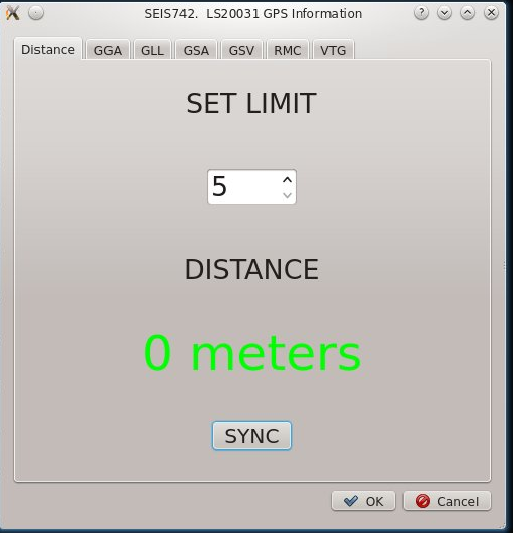
Tabs is a Graphic User Interface application that can run on a Linux box. Because it’s built using Qt4 cross-platform application framework, it shouldn’t take too much time to port it to Microsoft Windows operating systems if needed. Basically, Tabs is a client application. When it executes, it starts a TCP client that connects to BeagleBone and pulls NMEA information for End-User review.

The UML diagram on the next page shows the design of Tabs. It’s a little messy, but the main idea is that Tabs application contains the tabs for each NMEA sentence that can be received from LS20031 GPS Module – GGA, GLL, GSA, GSV, RMC, and VTG that will be explained later. One additional tab, Distance, is dedicated to monitoring an asset. When a new tab is selected, the software sends a request to base station twice per second to update information for selected page. Since Distance tab uses latitude and longitude information from GPS, it sends a request to update GGA.



### Distance Tab

This is a main page of Tabs. The SYNC pushbutton allows setting current LS200031 GPS receiver module location as a reference or a zero point. So, in other words the distance is always computed between zero (reference) point and asset’s current location. Usually, base station location is a designated reference point for this system. The SET LIMIT scroll box allows setting the allowable distance range limit. If the computed distance is greater than the selected limit, the distance readout displayed in green will be changed to red color as a warning. If connection between an asset and base station lost, the distance readout will NO DATA AVAILABLE message.





### GGA Tab

GGA Tab shows information from GGA output sentence that contains Global Positioning System Fixed Data.



### GLL Tab

GLL Tab contains information provided by GPS in Geographic Position Lat Lon sentence.



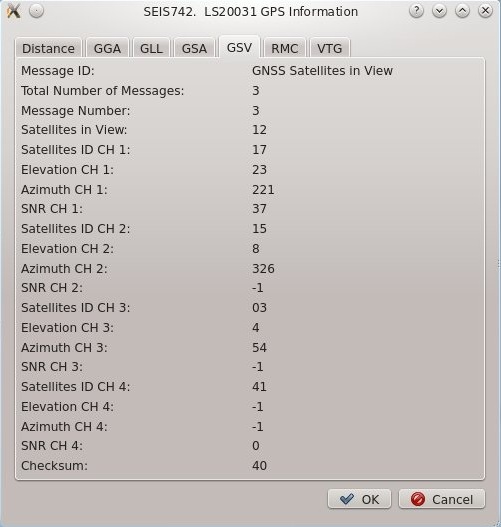
### GSA Tab

GSA tab provides information found in GNSS DOP and Active Satellites record.



### GSV Tab

GSV tab provides information from GNSS Satellites in View record.



### RMC Tab

RMC tab shows Recommended Minimum Specific GNSS Data information.



### VTG Tab

VTG tab is used to monitor Course Over Ground and Ground Speed information.



# Conclusion

The main challenge of this project was inability of communication of GPS receiver with GPS inside any building. Even though the datasheet says that 12 minutes maybe required in wrose case situation to acquire a signal I couldn’t get a signal inside my house, work, or a school lab. So, all testing and debugging had to be done outside sitting in a car. The other problem with this module is that it can work great for hours, but then it loses satellites and can’t acquire a fix no matter what. Or other way around – it can’t capture a fix at the beginning, but as soon as it gets it, it works great for hours again inside moving car or sitting on a driveway.

Wireless communication was the other challenge. To be honest I thought it would take more time for me to setup wireless communication between devices. But as soon as I figured out how to setup UART port on BeagleBone and NXP LPC1769, set the communication channel and baud rate, it just magically worked. But as oppose to GPS receiver, wireless communication worked great inside the house (different rooms, different levels), and not outside. I was inspired by the idea of trying GPS and wireless communication during someone’s presentation in our previous class, but I’m not sure if XBee Series 1 modules would work great to control an RC plane that we may build with my son this summer. And again, sometimes it worked great and other times it wouldn’t establish a connection over 40 ft.

On a positive side I never got a corrupted message from GPS module or invalid command notification. And I never got a corrupted message over wireless link when the communication was established.

BeagleBone programming was very stable. All IPCs worked well. I haven’t noticed any bugs in my software related to that YET.

Creating client GUI application was definitely a time consuming task. I haven’t tracked the time spent for this project, but I think this task was most expensive even though GUI doesn’t look that fancy at all. Just typing all the data translation routines, and setting many fields and labels took bunch of time even though I’ve tried to follow good design guidelines.

The most exciting and interesting part was writing a parser and generator for NMEA records and commands, and later looking what kind of information is coming out from the GPS receiver thinking how much technology is involved in this project only to change a color of Distance Readout. Thanks for reading!

# References

1. Datasheet of GPS smart antenna module, LS200031. LOCOSYS Technology Inc
2. MTK NMEA Packet User Manual, Revision: 0.3 Linked FW Version: 1.902\_06, ETEK Navigation, INC
3. LPC17xx User Manual, UM10360, NXP
4. XBee Builder Tutorial, <http://bildr.org/2011/04/arduino-xbee-wireless/>
5. XBee Buying Tutorial, <https://www.sparkfun.com/pages/xbee_guide>
6. BeagleBoard.org web-site, <http://beagleboard.org/Products/BeagleBone>
7. Advanced Programming in the UNIX Environment (2nd Edition), by Richard Stevens
8. INTERNETWORKING with TCP/IP Client-Server Programming and Applications, Douglas E. Comer, David L. Stevens
9. Movable Type Scripts, Calculate distance, bearing and more between Latitude/Longitude points, <http://www.movable-type.co.uk/scripts/latlong.html>
10. BeagleBone RevA6 System Reference Manual, BeagleBoard.org