RFID Asset Tracking System

by

Steven Grier, Michael F. Marchini, and Jeffrey Zimmerman

A Senior Project Report Submitted to the Faculty of
Electrical, Computer, and Software Engineering
Penn State Erie, The Behrend College

Faculty Supervisor(s): Dr. Nelatury, Dr. He, Mr. DeWolf
Industrial Supervisor(s): Dave Hartz
Industry Sponsor: BTC Solutions, Inc., Erie, PA

April, 2007

Table of Contents

Table of Figures

Table of Tables

Abstract

This project proposes a complete asset tracking system for BTC Solutions, Inc., a supply-chain management solutions company based in Erie, PA. Augmenting their current vehicle tracking system, the results of this project allow the company to keep track of the vehicle, its cargo (assets), and where and when assets are loaded and unloaded from the vehicle. This unified solution can help the company record appropriate auditing, security, and statistical data, which can be used to create more efficient asset transportation routes and procedures.

This system employs multiple technologies: Hall-effect sensors, an RFID reader, RS-232 communication, a microcontroller, motion sensors, flash memory, GPS, and GSM communication; all of these technologies are described in detail in Section 1. In this system – when the Hall-effect sensors find the vehicle's cargo door is open – asset data is read by the RFID reader and transmitted to the microcontroller via RS-232 communication. The microcontroller packages this asset data with directional data from the motion sensors and stores it to non-volatile flash memory. Once the Hall-effect sensors tell the microcontroller the cargo door is closed, the flash memory is flushed and streamed via RS-232 to a GPS/GSM unit. Here, the data are transformed into an XML file format. Along with relevant GPS data, the XML file will be transmitted via cellular GSM communication to the cargo vehicle's logistics company for analysis. This system meets the requirements as described in Section 2 and the design is detailed in Section 3.

Multiple tests were performed to ensure the components of this system worked individually as well as in tandem with other components. The system's components passed their unit and integration tests as detailed in Section 4 and the system meets the engineering requirements. We conclude this project in Section 5 with a summary of the entire project experience including the important lessons we have learned, as well as with our recommendations for further work on this system,

1. Problem Statement

1.1 Need Statement

BTC Solutions, Inc. has a need for an autonomous system that tracks a transportation vehicle and its cargo.

Imagine someone whose primary job function is to ensure the products from a manufacturer successfully make their way to their respective consumers. Part of this involves keeping accurate records that verify the products reach their particular milestones along their paths. The difficult aspect of this is a person's ability to follow every package and record the pertinent data. That person cannot possibly track each individual product manually, as they may not take the same route nor travel in the same vehicles to reach their destinations. To better illustrate the magnitude of this situation: in 2002, 15.8 billion tons of items were shipped by various shipping companies. (Imagine one person tracking all of those items.) Autonomous solutions, then, are preferred over manual recording; this helps track the items with a higher degree of accuracy and helps reduce human error in the record-keeping.

To lower costs and increase productivity, the logistics industry is moving toward such autonomous systems; BTC Solutions, Inc., aims to be an early-adopter of this technology. BTC's current systems track only the location of the vehicle and some sensor data, and they feel a more robust system is necessary to track the cargo. As such, they have a need for a system that accounts for not only the vehicle but also the assets in the vehicle during the entire shipping process: from when the item is loaded into the vehicle to when the item is delivered.

1.2 Objective

To meet BTC Solution's need, our objective is to integrate BTC Solutions's GPS tracking system with an RFID reader. The new system will relay the contents and location of the vehicle to a remote host. A remote user can then track, in real-time, where the vehicle goes and when and where items are loaded and unloaded.

1.3 Background and Related Work

1.3.1 Overview

Supply chain management (SCM) involves overseeing an asset as it proceeds from the manufacturer to the end-user through a variable number of intermediaries. Further, SCM involves the efficiency of the product's transition. Walther Bernard, Managing Director of BTC Solutions, Inc., notes that intimate knowledge of how a product moves through the supply chain is quite important. "Since logistics, in general," says Bernard, "is becoming more and more an integral part of the manufacturing cycle, it is vital to know where the truck or trailer currently is, and with it obviously all the goods on or in the truck." Bernard continues, noting that the real-time data supplied by the tracking of mobile assets is critical for optimizing the routing (and rerouting, if necessary) of the product.

The concept of tracking shipments is not novel; it has been realized through the use of barcodes for years. Unfortunately, barcodes are not as robust as the current market demands. For instance, tracking shipments in real-time would require constantly re-scanning the products either manually or with a complex automated system. Currently, there are no systems available

that allow the re-scanned products to be recorded along with the vehicle's location without including the subjectivity of the driver.

A solution that combines a GPS tracking system with an RFID reader is an intelligent solution for achieving real-time shipment tracking. Bernard mentions that "if goods are moved from A to B, and both places have an RFID reader infrastructure (which actually is not cheap to set up), then there is no need for integration. But if – in most cases – there is no infrastructure, and it is important [to have a mobile tracking system]." Bernard continues, providing interesting statistics:

> "A Driscoll & Associates study from 2004 says that out of 20 million vehicles in the US market only 5% are equipped with [autonomous vehicle location] systems. Out of 4.6 million trailers only about 100,000 are equipped with a trailer monitoring system. As a result, there will be an increasing commercial benefit to use the 2 emerging technologies (RFID + GPS/GSM) for improving the efficiency of the logistics processes."

Radio Frequency Identification (RFID) is becoming an increasingly popular method for tracking a company's assets. When combined with a Global Positioning System (GPS), the overall system can track the real-time location of those recorded assets. This complete system helps safeguard the assets in-transit.

Our project focuses on implementing such a tracking system for BTC Solutions, Inc., a supply chain solutions company. The term "supply chain" is defined as "the network of retailers, distributors, transporters, storage facilities and suppliers that participate in the sale, delivery and production of a particular product" [3]. A supply chain solutions company, then, manages the supply chain with innovative tools that ensure specified assets are transported to and from their intended locations.

Specifically, BTC is looking for a robust, integrated system that combines the scanning abilities of an RFID reader and the tracking capabilities of a GPS system. Once sold and installed, the systems will provide BTC's clients with real-time data on their respective vehicle locations and vehicle contents. Further, the system will indicate when and where an asset has entered or left the vehicle.

BTC and its client companies will be early adopters of this tracking technology, placing them in a unique marketing position ahead of their competition. According to the U.S. Bureau of Transportation Statistics, "trucking accounted for $224,464 million in revenue in 2000" [1]. In a recent survey performed by AMR Research, "23% of companies polled are piloting RFID technology, while 38% plan to evaluate RFID technology in the next two years" [5]. Further, AMR research notes that "69% of retail suppliers have plans to evaluate or implement RFID technology" [5]. These statistics help justify BTC's motivation as an early adopter of this technology.

1.3.2 Relevant Technologies

Flash Memory

Also known as Electronic Erasable Programmable Read Only Memory (EEPROM), flash memory allows storage of virtually any type of data (digital voice, images, program data, etc.). Unlike Random Access Memory (RAM) that loses its data when power is removed from it, flash memory is non-volatile; data remains in the memory even if power is lost. This is an excellent technology for use in conjunction with a microcontroller for external storage of critical data, especially in applications where data must be logged for later access.

GPS

The GPS or Global Positioning System is a network of 24 satellites that are in geosynchronous orbit [2]. They are positioned so that transmissions can be read from at least four satellites from any point on Earth. Each satellite is equipped with an atomic clock, transmitter, and computer. The satellites transmit a time code which is picked up and decoded by a GPS receiver. The GPS receiver measures the time delay taken to receive these signals and is then able to calculate its location on Earth.

GSM

GSM stands for Global System for Mobile Communications [7]. The goal of the GSM network is to provide a mobile connection to communication and information services via radio waves. The services include voice communication (telephone), text communication (SMS), and Internet connectivity (IP switching). The identity of a user in a GSM network is established with a Subscriber Identity Module (SIM) card. This card holds the user's subscription information and phonebook. These cards are assigned by the mobile service provider and are placed inside the mobile device.

The GSM network structure is split into three subsystems. First is the Base Station Subsystem. This is where the cell towers are located [6]. They have transceivers and antennae along with a control system. These stations provide the connection from the Network Subsystem to the mobile device via cellular signal (radio wave typically at 900 MHz or 1800 MHz). The Network SubSystem provides the switching service that allows a mobile device to connect to another mobile device or to the Public Switched Telephone Network (landline) [9]. The GPRS Subsystem (General Packet Radio Service) provides IP packet switching services to the Network SubSystem for propagation to mobile devices [7]. Figure 1 shows the interconnection of the three subsystems.

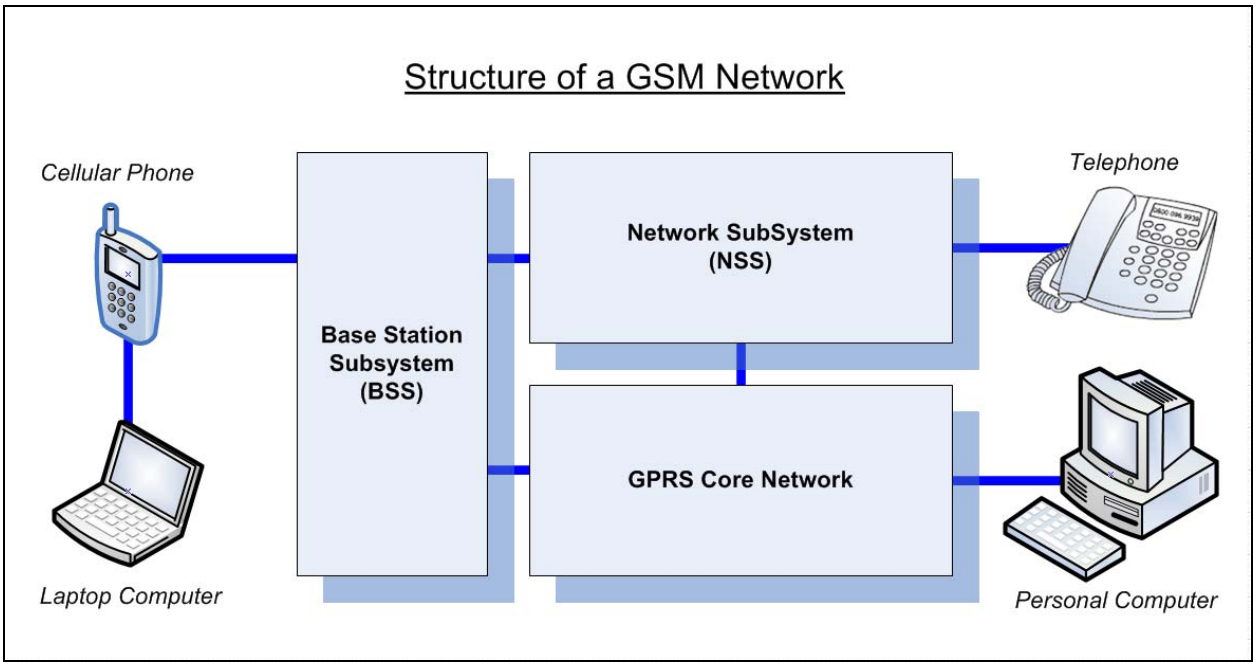

Figure 1: The Structure of a GSM Network [8]

Microcontrollers

A microcontroller unit (MCU) is a programmable integrated circuit that can be used in a nearly infinite number of applications. Microcontrollers often take little power to operate and they feature multiple general-purpose input/output pins that can be set to either receive or send data. Further, most microcontrollers feature timer modules and standard communication interfaces (such as serial RS-232).

RFID

RFID stands for Radio Frequency Identification. According to Symbol Technologies, “RFID is making headlines in the business world as the "next frontier" of supply chain efficiency” [4]. Chris Wassel, an RFID expert from Penn State Erie’s RFID Center, notes that RFID has multitudes of applications, including supply chain management, item/asset identification, and warehouse management [5]. Package tagging is currently done at the pallet and the case levels; once this technology becomes more widely accepted, items will be tagged individually. Figure 2 describes an RFID System as consisting of four main components.

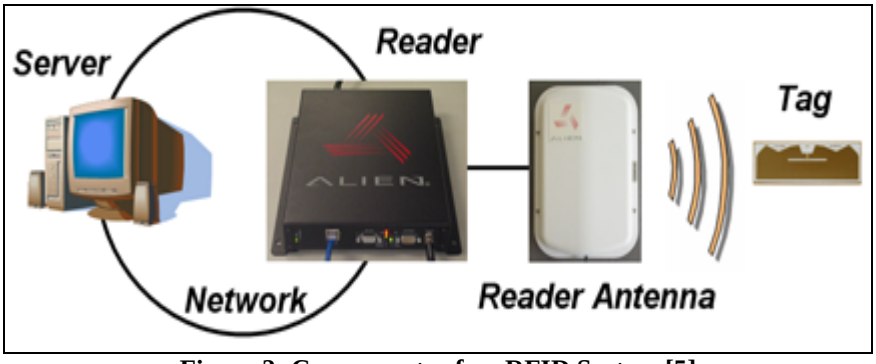

Figure 2: Components of an RFID System [5]

- ***Server –*** Runs the "middleware" – software to control the reader and interface to other applications.
- ***Reader*** – Device used to "interrogate" the tag. The reader obtains the data from the RFID tag and includes a transmitter, receiver, and microprocessor. Some readers can also encode (write to) the tag.
- ***Antenna*** – Propagates (emits) radio waves from the reader's transmitter and accepts radio frequency replies from RFID tags.
    - ***Linear***: The signal is focused to emit perpendicular from the reader; reads tags along a particular axis.
    - ***Circular***: The signal is focused to emit radially from the reader; reads tags in any orientation.
- ***RFID Tag*** – "Harvests" the reader antenna RF energy, powers the microchip, and reflects back its own signals.
    - ***Passive***: Require activation by a reader in order to transmit its data; they do not have their own power source.
    - ***Active***: Have their own battery-based power source and transmit their signal regardless of a reader's presence.

SMS

SMS stands for Short Message Service, and is more commonly referred to as text-messaging [10]; it is available on most mobile phones. SMS messaging sends information to a SMSC or message center. The message center then transfers the message to the intended recipient. SMS messages can contain binary data for mobile process communication.

XML

"Extensible Markup Language (XML) is a simple, highly flexible text format derived from SGML (ISO 8879). XML is playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere" [12]. XML documents can be tailored to the needs of a specific application with relative ease.

1.3.3 Device Objective Tree

Figure 3 represents the Objective Tree for this project. Each of the marketing requirements – as described in Section 2.1 – is functionally decomposed and classified, allowing the viewer to see the relationship between supporting components and their respective requirements. These relationships form the basis of the project's Engineering Requirements.

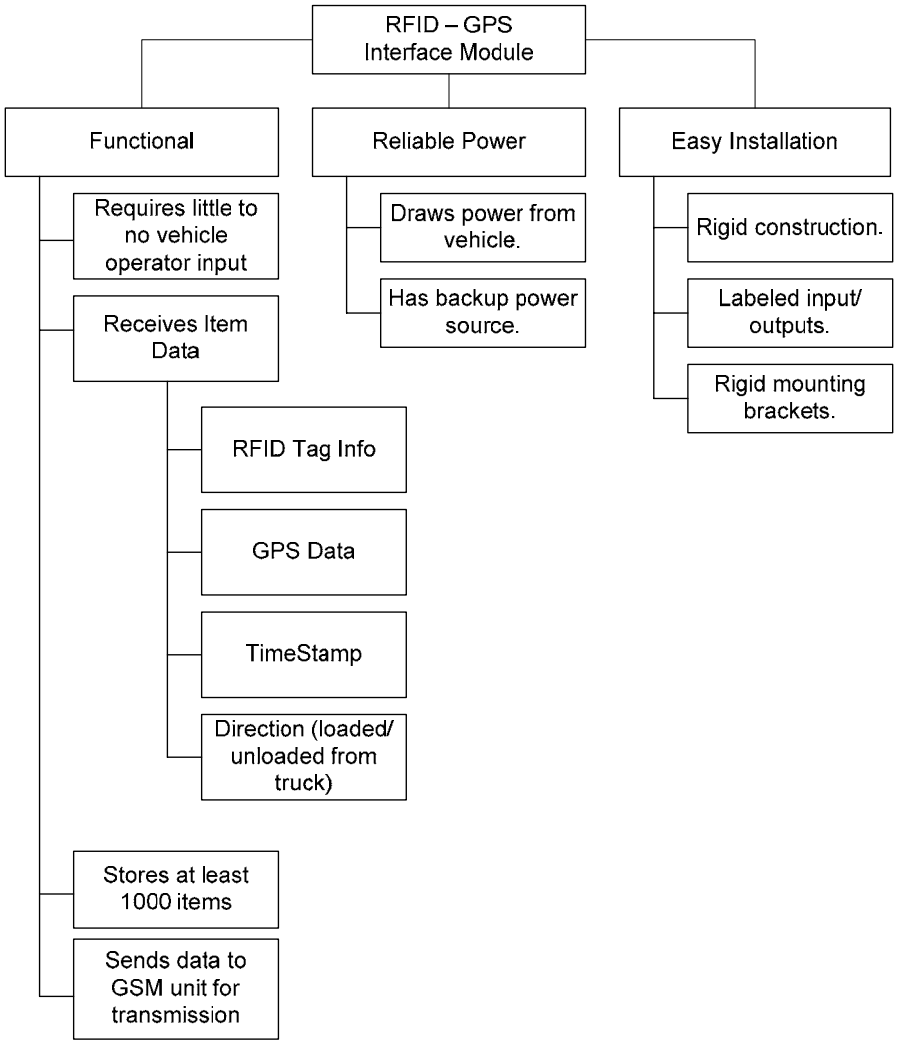

Figure 3: Device Objective Tree

2. Requirements Specification

2.1 The Requirements

The marketing requirements for this project are as follows:

1. Device allows an RFID module to communicate with a GPS tracking system.
2. Device stores data from OEM RFID module for communication to the GPS unit.
3. Device stores RFID tag data with GPS location and time data in memory for transmission to a web server.
4. Detects whether a product is entering or exiting the truck.
5. Utilizes the vehicle's power to operate.
6. Has a backup power source.
7. Device will queue product movements when an unusable (absent or low) GSM signal is present to be transmitted when a reliable signal is acquired.
8. Requires little to no vehicle-operator input for operation.
9. Device is capable of storing at least1000 tags of data.
10. Device can read RFID tags at a range of up to 5 meters.

Table 1 compares the project's engineering requirements with the marketing requirements they cover as well as their respective justifications.

Requirements		**Validation Rationale**
Marketing	***Engineering***	
1, 3, 10	Utilize an ISO 18000-6C - compliant RFID reader that complies with applicable FCC requirements, features anti-tag-collision protocol, and supports at least two antennas.	The utilization of this type of device is required by our industry sponsors. The device must use the ISO 18000-6C standard to read Gen-2 tags (the current standard). The device must support two antennas to improve accuracy of tag data acquisition. The device must have an anti-tag-collision protocol to allow the reader to accurately gather tag data when multiple tags are close together in the RF field. The device must comply with FCC standards for legal use.
3	Utilize a GSM unit to transmit data to a server.	The data collected must be sent to a server for BTC to use. BTC already utilizes GSM communication in their existing systems, and has specified that this device will do the same.
2, 3, 4, 7, 8, 9	The completed system involving the RFID reader and GPS unit must store tag, location, time data, and package direction (on/off the vehicle) for transmission.	The interface device provides a platform on which to store data in regards to current time, location, and tag information given a noteworthy event (item is added to or removed from vehicle). The sponsor has specified that they want to know when, where, and what items have left or entered the vehicle when the vehicle is loading or unloading.

Requirements		Validation Rationale
Marketing	***Engineering***	
5, 6	Regulated power within device specifications will be supplied from vehicle and backup supply to the GPS device.	To help ensure proper operations and to prevent damage to the devices, the required voltages and amperages should be supplied to the system's core components.
9	The system will have at least 3.10 KB of memory specifically for data packets.	The device is required to store at least 1000 packets of data for later transmission in case of a weak signal. Currently, cargo is tagged at the case and pallet level, but a push for unit level tags will be made in the near future. Keeping this in mind, we have selected a minimum of 1000 to allow for the inclusion of unit tag data in the system. Pertinent GPS and RFID data will consume 33 bytes per instance, so 1000 tags will consume 3.10 KB of memory.
1, 3	Utilize a GPS device to gather time, location, and velocity data.	The final design needs time and location data to track where and when packages are loaded/unloaded from the vehicle.

Table 1: Engineering Requirements with Justification

2.2 Constraints

2.2.1 Economic

The final system must not cost more than $2000.

2.2.2 Environmental

The system must store all RFID data on the trailer for an unknown period of time until the truck is attached. The devices on the trailer must be capable of interfacing to any E.Novations Mobile Server that is running the tracking code.

2.2.3 Sustainability

The final system must be reliable for everyday operation by commercial shipping companies to allow BTC to market the device.

2.2.4 Manufacturability

The final system must be modular and easy for BTC Solutions to reproduce.

2.2.5 Social

FCC regulations on RF transmission require less than 4 Watts of power be emitted from any unlicensed source.

2.3 Standards

There are several standards that need to be followed for this project. The first of which pertains to the RFID tag data. Figure 4 is the break-down of an RFID tag’s data layout.

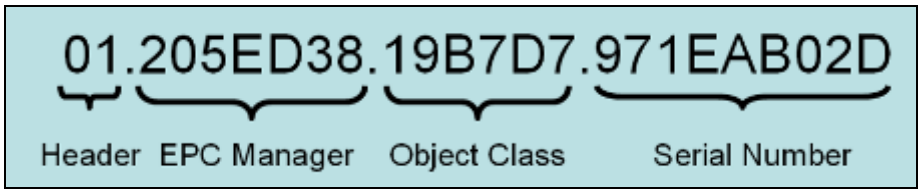

Figure 4: RFID Tag Bit Sample [5]

Another set of standards associated with our project pertains to the GPS data. There are two types of GPS data packets that are used. The first is the GGA protocol.

Table 2 describes the various entries in the following example (white space added for readability):

$GPGGA, 161229.487, 3723.2475, N, 12158.3416, W, 1, 07, 1.0, 9.0, M, , , , 0000*18

Name	Example	Units	Description
Message ID	$GPGGA	-	GGA protocol header
UTC Time	161229.487	-	hhmmss.sss
Latitude	3723.2475	-	ddmm.mmmm
N/S Indicator	N	-	N=North or S=South
Longitude	12158.3416	-	dddmm.mmmm
E/W Indicator	W	-	E=East or W=west
Position Fix Indicator	1	-	See Table 3 Below
Satellites Used	07	-	Range 0-12
HDOP	1.0	-	Horizontal Dilution Of Precision
MSL Altitude	9.0	meters	-
Units	M	meters	-
Geoid Separation	-	meters	-
Units	M	meters	-
Age of Diff. Corr.	-	second	Null Field when DGPS is not used
Diff. Ref. Station ID	0000	-	-
Checksum	*18	-	-
<CR> <LF>	-	-	End-of-message termination

Table 2: NMEA $GPGGA Standard Format [11]

Value	Description
0	Fix not available or invalid
1	GPS SPS Mode, fix valid
2	Differential GPS, SPS Mode, fix valid
3-5	Not supported
6	Dead Reckoning Mode, fix valid

Table 3: Position Fix Indicator [11]

The second GPS data packet is the RMC protocol. As detailed in Table 4, this packet contains the recommended minimum specific data need to obtain a position.

Name	Example	Units	Description
Message ID	$GPRMC	-	RMC protocol header
UTC Time	161229.487	-	hhmmss.sss
Status1	A	-	A=data valid or V=data not valid
Latitude	3723.2475	-	ddmm.mmmm
N/S Indicator	N	-	N=north or S=south
Longitude	12158.3416	-	dddmm.mmmm
E/W Indicator	W	-	E=east or W=west
Speed over Ground	0.13	Knots	-
Course over Ground	309.62	degrees	True
Date	120598	-	ddmmyy
Magnetic Variation2	-	degrees	E=east or W=west
Checksum	*10	-	-
<CR> <LF>	-	-	End of message termination

Table 4: NMEA $GPRMC Standard Format [11]

1. A valid status is derived from the SiRF Binary M.I.D 2 position mode 1.
2. SiRF Technology Inc. does not support magnetic declination. All “course over ground” data are geodetic WGS84 directions.

The last standard is mandated by the FCC. As previously mentioned, FCC regulations on RF transmission require less than 4 Watts of power be emitted from any unlicensed source.

3. Design

Level 0

Figure 5 is the Level 0 Diagram for our project. Here, our system receives four inputs – the RFID Tag Signal, Motion, GPS Signals, and the 12 Volt Power Signal – and produces two outputs – the RF Antenna Signal and the GPRS Signal.

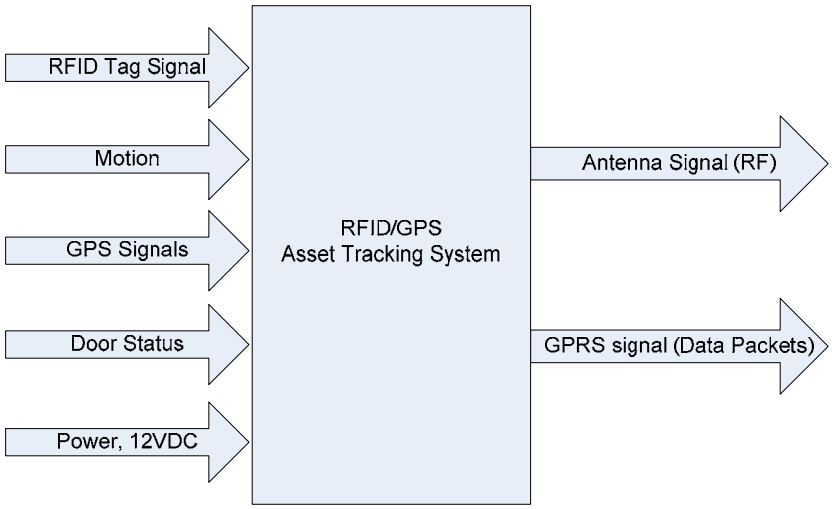

Figure 5: Level 0 Diagram

Level 1

Figure 6 is the Level 1 Diagram for our project. With this diagram, we can see the interaction of the components that lie within the system. The E.Novation Mobile Server (HCP65-G) contains customized software to control the gathering of GPS Signals and the transmission of data packets via GPRS. The PIC24F series microcontroller unit (MCU) can turn the RFID Reader on or off via the Control signal. The RFID reader, in turn, energizes the RFID Antennas which transmit their Antenna Signals. If an RFID tag crosses an Antenna Signal, the tag transmits an RFID Tag Signal, which the RFID Antenna receives and forwards to the RFID Reader. Once the RFID Reader interprets the RFID Tag Signal, the corresponding RFID Tag Data is sent to the MCU. The MCU also receives Motion Sensor Signals from Motion Sensors which are activated by the Motion of anyone or anything within the Sensors' fields of view. The MCU will store the RFID tag data with a direction of movement flag. The data stored on the MCU will be sent to the Mobile Server via serial connection. Lastly, the Mobile Server packages and sends the Data Packets via a GPRS Signal as determined by its customized software.

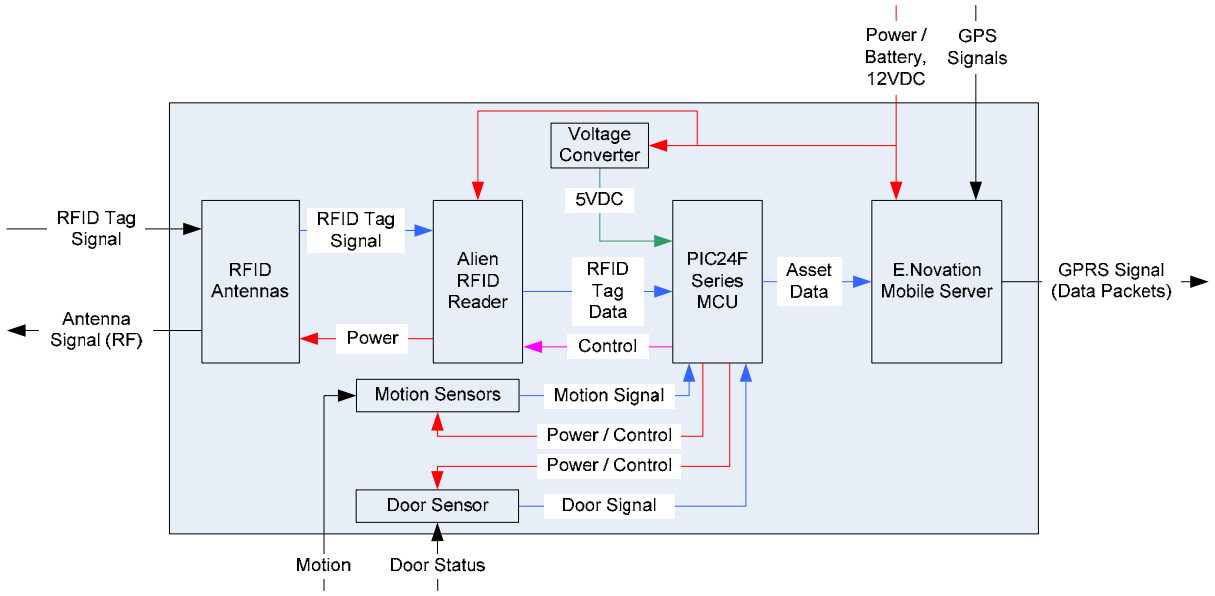

Figure 6: Level 1 Diagram

As shown in Figure 7, data is sent from our system to the remote host (a company, in our diagram) via the following steps:

1. Sensors attached to the door inform the MCU of the door's status.
2. The MCU activates the RFID Reader when a direction sensor is tripped (thus energizing the RFID Antennas).
3. An RFID Tag Signal is sent by a nearby RFID Tag.
4. The RFID Antennas receive the RFID Tag Signal and forward the signal to the RFID Reader.
5. The RFID Reader interprets the signal and sends RFID Tag Data to the MCU.
6. The MCU combines the RFID Tag and direction data from the sensors in to Data Packets.
7. The GPS unit receives data from the MCU and GPS Satellites and transmits the data via GSM.
8. The GSM signal is received by T-Mobile (BTC Solution's Internet Service Provider) and their equipment uses GPRS to make the data available at the IP packet level.
9. The Data Packets are then forwarded to the Company for interpretation and analysis.

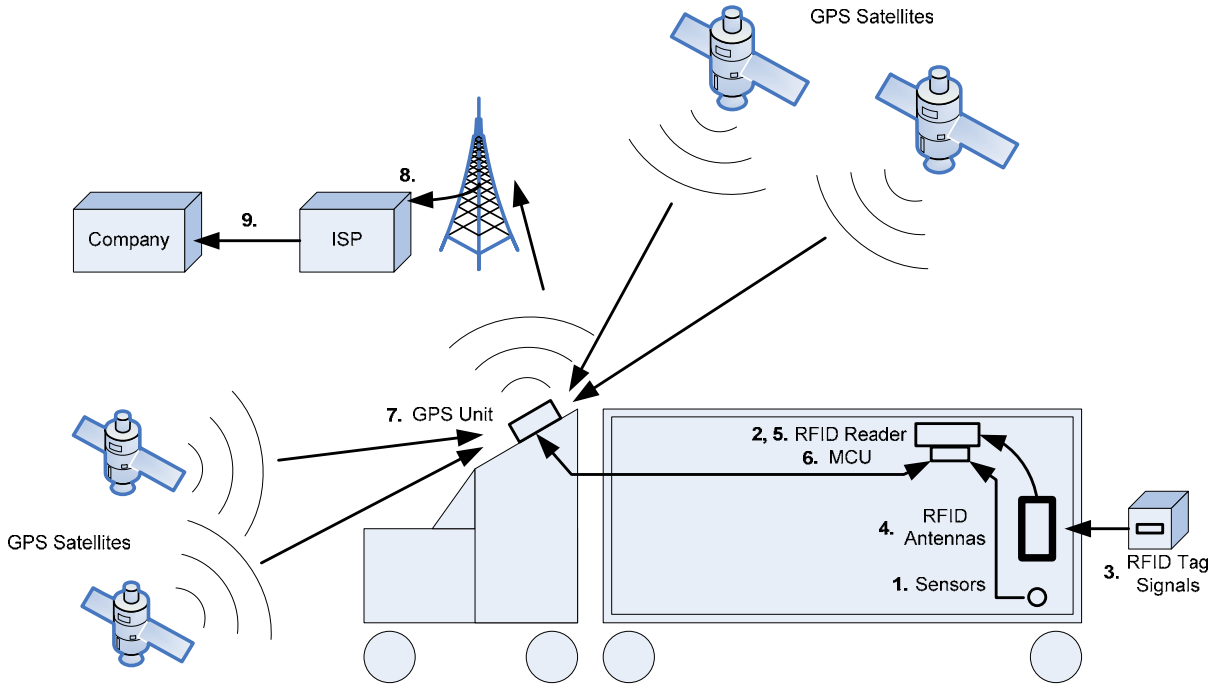

Figure 7: Schematic of the System's Data Flow

Level 2

Level 2 of our system involves the customized software within the E.Novation Mobile Server and the MCU. Figure 8 is a flowchart that describes the flow of the MCU's core software. The software is based on the concept of the current GPS tracking system that gathers location data every minute, and transmits every twenty minutes. The MCU will be responsible for gathering RFID Tag data and direction data. The MCU will send the data it collects to the GPS unit when a valid connection exists (the trailer is attached to the truck). Following the state chart, the MCU software will operate as follows:

1. Begin in the system startup state.
    a. If the door is open, transition to the Store Door Event state.
    b. If the door is closed, transition to the Wait Door Closed State.
2. In the Store Door Open state.
    a. Store the Door Open Event to memory and jump to the Wait Door Open State
3. In the Store Door Closed state.
    a. Store the Door Closed Event to memory and jump to the CommCheck State
4. In the Wait Door Open state.
    a. If a tag is detected, jump to the Get Tags From RFID state,
    b. If the Door is closed, jump to the Store Door Closed State
5. In the Get Tags From RFID state.
    a. Retrieve the tags from the RFID reader and place them into the Microcontroller RAM.
    b. If the tag data is done being stored, move to the Store Tags To Flash state.
6. In the Store Tags To Flash state.

a. Write the tag data to the flash memory buffer. If the buffer is full, write the buffer to the flash memory.
b. Move to the Wait Door Open state when done.

7. In the Comm Check state.
    a. Attempt to establish a connection with the GPS unit
    b. If a connection is established, transition to the Send Data state.
    c. If the door is open, transition to the Store Door Open state.
    d. Otherwise, remain in the CommCheck state.
8. In the Send Data state.
    a. Send the data in the memory to the GPS unit via RS-232.
    b. Purge the flash memory.
    c. Transition to the Wait Door Closed state.
9. In the Wait Door Closed State.
    a. When the door is opened, move to the Store Door Open Event State.

The two direction sensors are wired to external interrupt pins on the microcontroller. Using interrupts, the direction of the package can be determined.

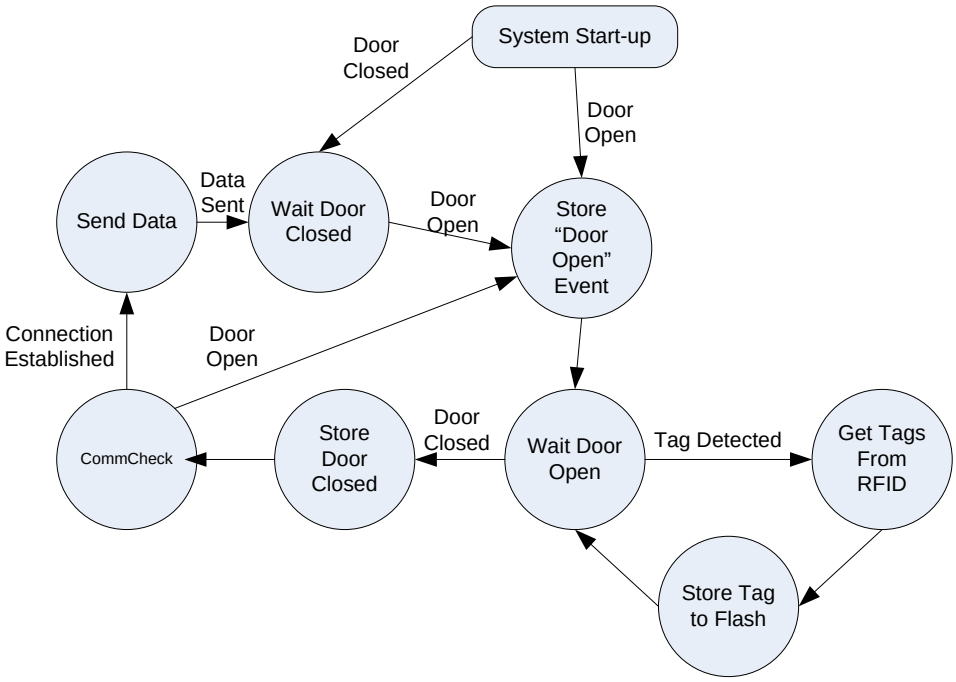

Figure 8: MCU Finite State Machine Diagram

The E.Novations Mobile Server (HCP65-G) gathers the GPS data and controls the GSM modem. The MCU will be connected to the HCP65-G via a serial connector (DB9) when the trailer is attached to the truck for hauling. There will be a power line that will be attached with the serial connector to recharge the battery in the trailer. As stated above, the MCU will check for a valid connection when no other actions are required of it. It will do this by sending an 8-bit request over the serial connection. If an 8-bit reply message is received, then a valid connection

exists and the MCU will begin to send any information in the list of actions since the last valid connection was lost. The operation of the HCP65-G software will operate as follows:

1. The system begins its operations as soon as it is powered-on, and continues until power is lost.
2. The software will start-up the GPS receiver and GSM modem hardware.
3. Concurrent tasks of gathering GPS data and asset data begin.
    a. The software will start a task to read/write over the serial port.
        i. The task will listen for a request command and reply with a reply message.
        ii. Following the reply message the task receives a set of asset data packets (RFID Tag data and direction) preceded by a header containing a 16-bit length field.
        iii. The task will append the packets to an activity log.
        iv. The task will start listening for a request command again (Step 3.a.i).
    b. The software will start a task to manage the GPS hardware.
        i. This task was created by BTC and simply returns NMEA GGA and RMC standard data (see report section 2.3 for NMEA standard reference table).
        ii. The data is written to a log every 30 seconds.
    c. The software checks for an activity file. If located, the file is parsed into XML.
4. The software will start a task to manage the GSM hardware.
    a. This task was created by BTC and will send all the log files after a defined amount of time (20 minutes) to BTC's server via a GPRS gateway provided a strong signal is present.
    b. If a signal is not present, the data is archived for later transmission.

The flowchart in Figure 9 describes the process the E.Novations server will follow:

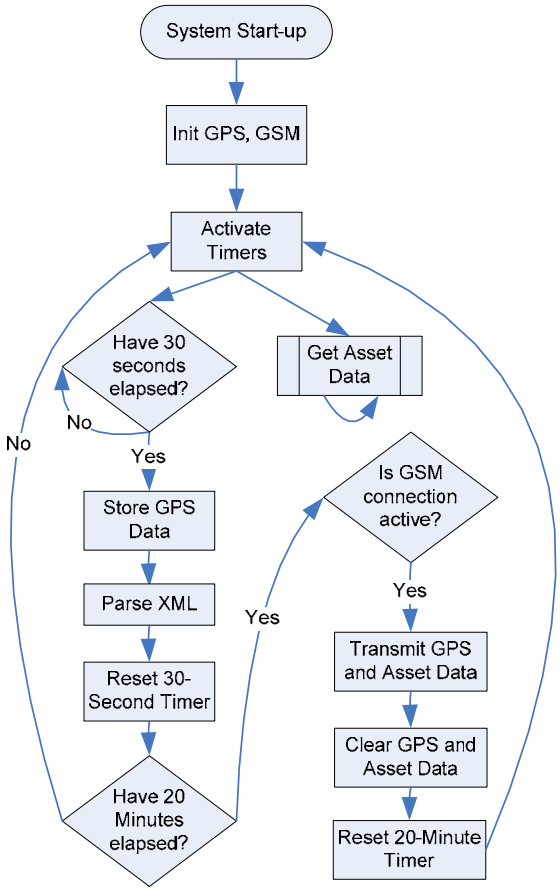

Figure 9: E.Novations Server

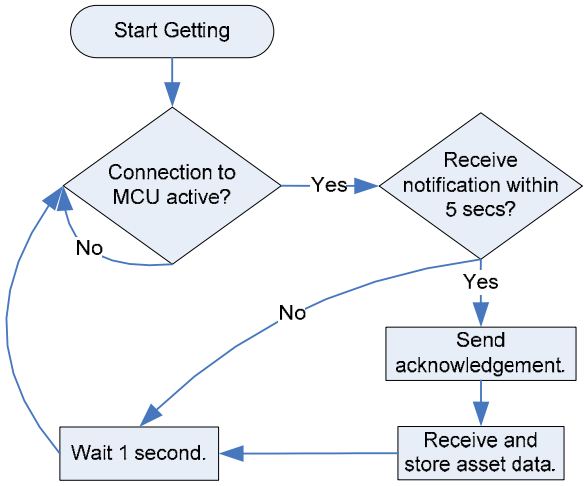

Figure 10: Get Asset Data Process Flowchart

The process block in Figure 10 involves receiving data from the MCU. If the connection with the MCU is active and a data notification has been received, an acknowledgement is sent. Then, the asset data is received and stored. After a one-second delay, the process repeats.

Device Selection

The selection of primary devices (the GPS/GSM unit and the RFID reader) was done by BTC Solutions before this project began. The use of the E.Novation mobile server was a constraint; as was the use of the Symbol MR400 RFID card. However, the specified RFID card was not available for use in a timely manner and a substitution was made for an Alien

Technologies RFID reader. The remaining devices to be selected are the MCU, the RFID antennas, the power supply, the sensing equipment used to detect package direction, and door open/close sensor.

Antennas

The Alien RFID reader is compatible with Alien RFID antennas. As requested by BTC Solutions, we will use two of these antennas to detect RFID tags.

MCU

The MCU selection was constrained by several aspects: it needed to support two serial connections and be compatible with the PICKit2 serial programmer. The MCU was not included in the original design and had no place in the budget, so we limited the selection to MCU's that can be obtained for free as a sample. We chose to use Microchip's PIC series MCU's because the design team was already familiar with these types of MCU's. The PIC24FJ128 fits all of the above requirements.

Direction Sensor

There are several ways to detect packages moving through the door of the truck. The three that were considered for this project were photoelectric eyes, hall-effect sensors, and digital image processing. Two photoelectric devices will be positioned horizontally so that one will be "tripped" before the other. The order in which the sensors are "tripped" will give us the direction data needed for operation. The hall-effect sensor set-up involves the sensors being positioned along the bottom (floor) of the truck bed to detect items passing over top of them. The sensors will be placed in two planes parallel to the door of the truck. The order of detection will determine object direction. The digital image set-up involves a camera taking image samples from above the truck door (looking down). The images would be compared to determine the direction of objects moving on/off the truck.

There are four types of photoelectric set-ups: direct reflective, reflection w/reflector, polarized reflection w/reflector, and thru-beam; the details of each are explained below and are detailed in Table 5. The direct reflective set-up involves an emitter/receiver unit that emits a signal and waits for the reflection off of the object to be detected. This is limited in what it can detect. The object must be reflective and the environment non-reflective (the opposite of our design environment). The benefit of this setup is that the emitter/receiver is the only piece of equipment needed.

The reflection w/reflector set-up involves an emitter/receiver and a reflector. The emitter emits a signal at the reflector while the receiver waits for the reflection. The presence of an object is determined by a signal that fails to return to the receiver. This limits the objects that can be detected to non-reflective object. This will not work for our design since some packaging materials are reflective.

The polarized reflection w/reflector set-up involves a polarized emitter/receiver and a polarized reflector. The emitter emits a polarized signal that the reflector waits to receive. The polarized reflector is design to maintain the polarity of the signal emitted. This allows the eye to

detect both reflective and non-reflective packages. The limit of this approach is the range of the sensor. This set-up is limited to 2m, while the width of the truck's trailer is 2.5m.

The thru-beam set-up involves a separate emitter and receiver. The emitter emits a signal directed at the receiver. Objects passing between the two devices will break the beam, causing detection. This set-up allows the detection of both reflective and non-reflective objects. This set-up also allows longer detection ranges (4m, 6m, or 15m). The down side of this setup is the cost of separate emitter and receiver units.

The hall-effect sensor detects magnetic fields. The idea relies on packages being loaded/unloaded using a fork-lift that will cause induction when passing the sensor arrays. This idea was discarded due to both the limited range of hall-effect sensors and the requirement of a fork-lift being used.

The digital image processing approach is potentially the most accurate approach. It would provide information on not only direction of objects crossing its view field, but the size and structure of the object (possibly detecting the difference of a human walking on/off the truck and a package being loaded/unloaded). This approach was not taken due to the significant cost and overhead of processing digital images for packages that could be any size, shape, or color.

Sensing Techniques	Category	Pros	Cons
Photoelectric	*Direct Reflection*	• One emitter/receiver combo unit. • No other equipment necessary.	• Cannot detect non-reflective objects (fatal).
	Reflection w/reflector	• One emitter/receiver combo unit. • Detects breaking of signal.	• Cannot detect reflective objects (fatal).
	Polarized reflection w/reflector	• One emitter/receiver combo unit. • Can detect both reflective and non-reflective objects.	• Short detection range (2m) (fatal).
	Thru-Beam	• Long detection range (4m, 6m, 15m). • Can detect both reflective and non-reflective objects.	• Separate unit for emitter and receiver (cost).
Hall-Effect	-	• Sensors are inexpensive. • Sensors have long life-spans.	• Detection range is very limited (fatal).
Digital Image Capture/Processing	-	• Very accurate.	• Very costly. • Complex processing overhead.

Table 5: Comparison of Sensor Types

The final design will be implemented using thru-beam sensors. This is because it is the lowest cost implementation that covers all cases that we need to detect. The device we need to use needs a minimum range of 2.5m. AutomationDirect supplies a 4m thru-beam photoelectric tandem for $53.00 (the least expensive of all that meets the 4m range requirement).

Door Sensor

The possible implementations for the door sensor are a magnetic proximity sensor or a limit switch. In comparison, both options would satisfactorily relay the door position. The magnetic proximity sensor would require a magnet on the door (unless it was magnetic). The limit switch would require a more robust mounting bracket to hold it in place. The cost of a set of magnetic door sensor and magnet is $6.97 (Cherry product rated for durability and harsh environment operation). The cost of a limit switch is $32.21 (Omron enclosed). The difference in cost is the reason the final design will consist of two magnetic proximity sensors (one for each door).

Power Supply

The MR400 will be powered by the 12V electrical system of the Semi. According to the MR400 specifications, the device needs a regulated 5V +/-5% with a peak current of 1.5 amps. In order to power this device, a 12V to 5V step down converter will be needed.

There are 2 main types of power supplies that can be used in this application. The first is a DC to DC converter which implements a controller to control the switching of the input voltage at a high frequency and regulate the output voltage. This provides an efficient and precise control, while increasing cost and physical size.

The second type of converter is a linear regulator. This regulator steps down the output voltage by converting the excess voltage by dissipating it as heat. This is a very inefficient design, as a good portion of the power is dissipated as heat. However, it is very cost effective and small in size. The comparison is shown below in Table 6.

Property	DC-DC (Buck converter)	Linear
Efficiency	**High**	Very Low
Cost	High	**Low**
Output Power	**High**	Med
Noise regulation	**Med-high**	**High**
Size	Large	Small

Table 6: Power Supply Comparison

The main factors in choosing the power supply are efficiency and cost with more emphasis on efficiency. We chose the buck converter because the efficiency of the linear power supply was too low to ensure reliable operation.

The final system did not contain the MR400, as the original design had intended. As a result, a second approach was made for the final system. The Alien RFID reader needed 120VAC power. A DC/AC Inverter was run off of a car battery to supply the RFID reader with the needed power. This is not an ideal solution, since the battery is in the way of the

loading/unloading process, but the sudden substitution from the MR400 did not allow time to implement a more well-designed power supply. The aforementioned power supply was still used to supply the microcontroller, direction sensors, and door sensors.

System Installation

A typical trailer is 53’ long, 101” wide and 110” high. The antennas were placed on either side of the entrance to provide an optimal read rate. Each was mounted on a bracket like the one in Figure 11. These brackets allowed the antennas to be as close to the wall as possible, so they would not restrict the loading and unloading of the truck.

Figure 11: Antenna Mounting Bracket

Figure 12 shows the set-up of the directional sensors, mounted on both sides of the trailer below the RFID antennas.

Figure 12: Mounted Antenna and Direction Sensors

In Figure 13, the RFID reader, the microcontroller, and their supporting hardware are attached to the wall of the trailer. This ensured the device was out of the way during loading and unloading.

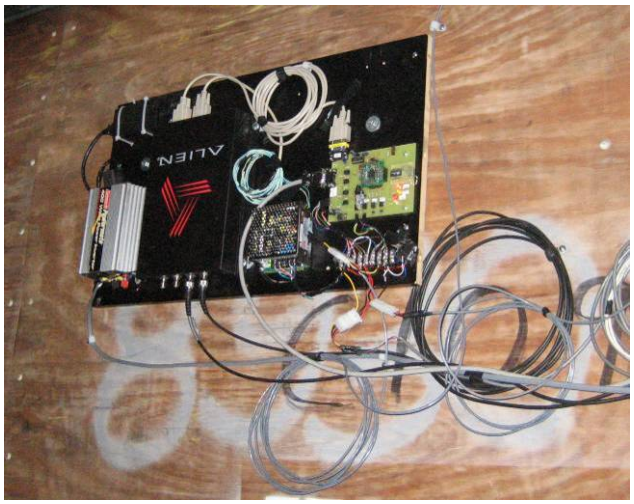

Figure 13: Wall-Mounted System Hardware

The power connection for the system was originally intended to be tapped-onto the power for tail lights. Unfortunately, the distance from the cab created too large of a voltage drop to run our system. So, the system's power needed to come from the 12V car battery in Figure 14, which was located in the trailer.

Figure 14: System Power Source

The physical layout of the system's components is shown in Figure 15.

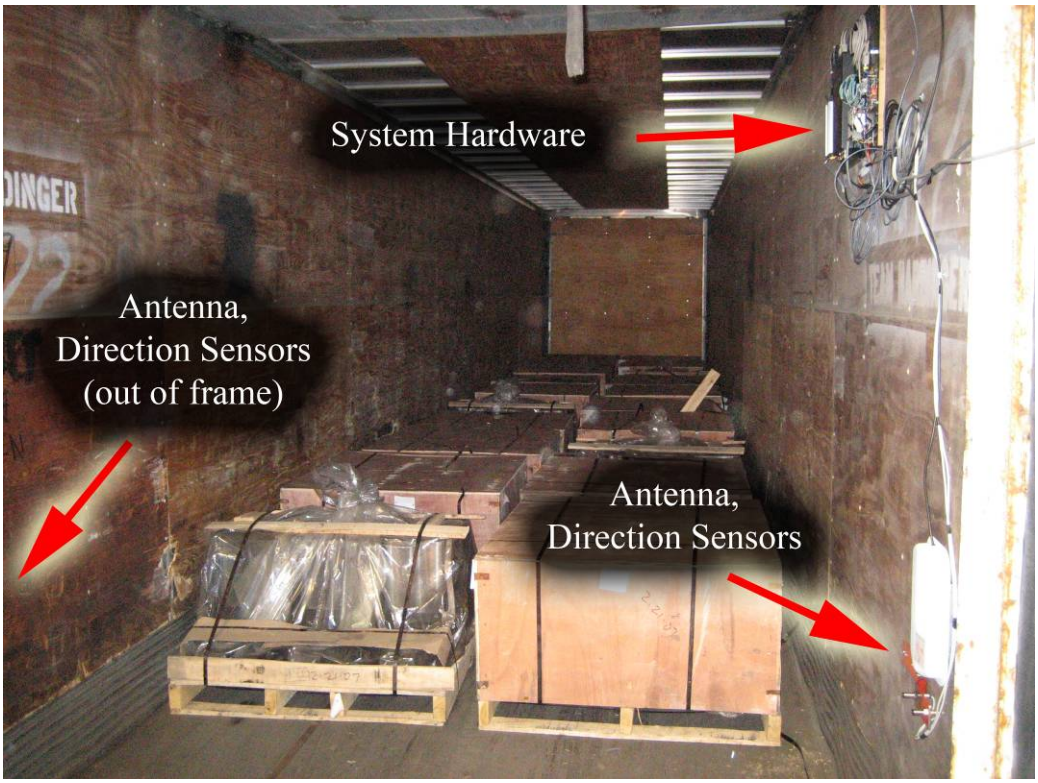

Figure 15: System Setup

The mobile server was installed in the cab of the truck, hidden in the glove box and powered directly from the cab's fuse panel, as shown in Figure 16:

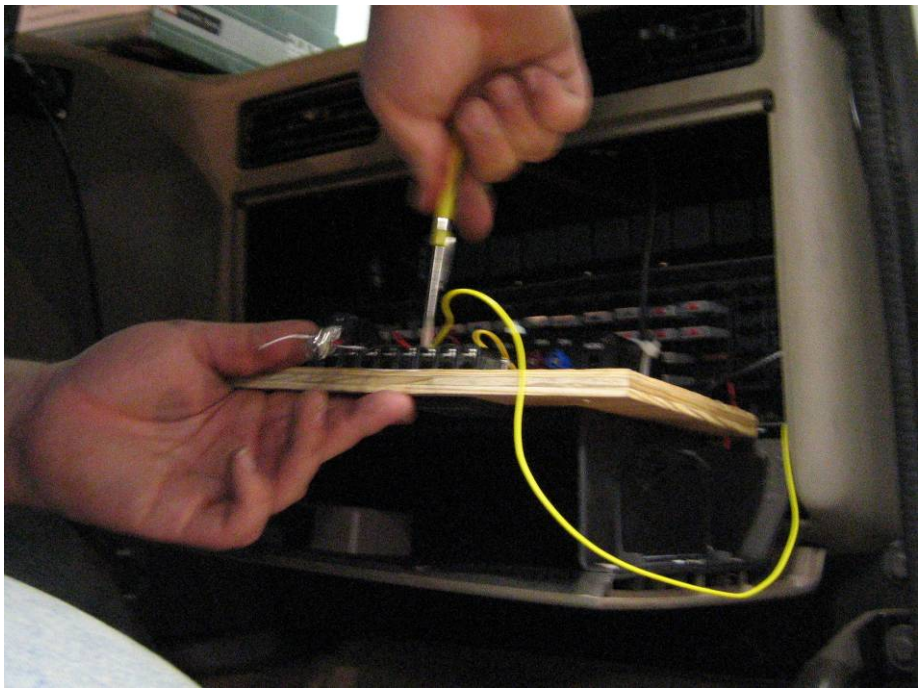

Figure 16: HCP65-G Installation

Alternative Approach Consideration

The E.Novation mobile server's use in this design was required by BTC Solutions. This device contains the GPS receiver (means to record time/location data), GSM modem (means to transmit stored data), and the hardware that will run Java code (means to control various devices). Originally, we were only going to use the E.Novation server to handle all control and data collection. This approach was optimal since all software, data collection, and control would reside on one device. The problem with this approach is that a trailer is not usually connected to the truck cab (where the E.Novations servers must reside) during loading and unloading operations. We chose to use an MCU as opposed to an FPGA for simplicity (C code vs. VHDL) and for the USART subsystem that is incorporated in the MCU.

4. Design Verification

4.1 Test Results

4.1.1 Testing the GPS

When the HCP65-G device was received, we conducted a quick test to see if the GPS unit was functioning. After modifying the code provided by BTC, we were able to write the GPS data to a log file (Figure 17).

```
20070124224706 - $GPGGA,193623.159,4206.9359,N,07958.7142,W,1,04,22.0,75.5,M,-33.8,M,0.0,0000*42
20070124224706 - $GPRMC,193623.159,A,4206.9359,N,07958.7142,W,0.000,123123,280107,9.5,W*42
<END>
20070124224709 - $GPGGA,193625.159,4206.9287,N,07958.7134,W,1,04,22.0,67.5,M,-33.8,M,0.0,0000*44
20070124224709 - $GPRMC,193625.159,A,4206.9287,N,07958.7134,W,0.000,,280107,9.5,W*47
<END>
```

Figure 17: Sample GPS Log Data

As shown in Figure 18, we created a simple LabVIEW interface to decode GGA and RMC packets in the log file. These coordinates could then be imported into a mapping program like Google Earth to see the coordinate relation with respect to aerial images. The GPS coordinates were also compared to a Garmin model 72 handheld GPS device to verify accuracy; based on said comparison, the results were accurate.

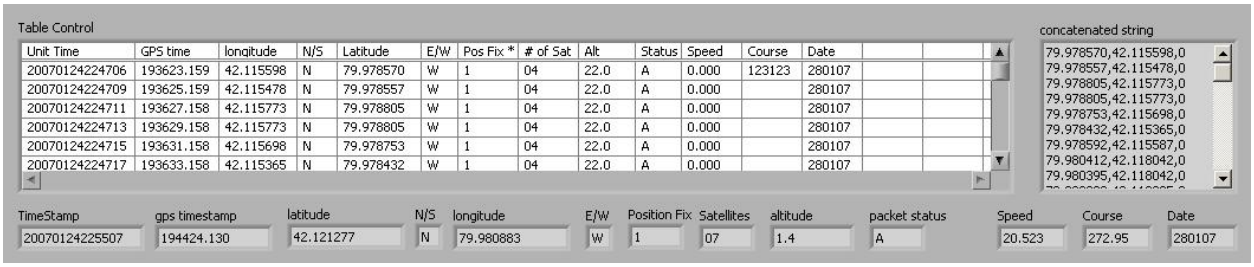

Figure 18: LabVIEW Interface for GPS Data

4.1.2 Testing the Microcontroller

The microcontroller was set-up to be programmed by the PICKit2 programmer, so we wrote and built a small program that would allow us to test the MCU's digital input and output.

Applying a voltage to an input would cause a corresponding LED to activate, and removing that voltage would turn off the LED. Following the success of this test, we needed to test both serial ports. The compiler we purchased for this project (the Mikroelektronika mikroC dsPic compiler) included an intuitive library for activating and using the MCU's serial ports. A simple test program confirmed the MCU could send and receive data across both ports.

4.1.3 Testing the Sensors

This system involves two types of sensors: one for the door of the cargo vehicle and one set for determining the direction of an asset's motion. The door sensor was a hall-effect sensor that closes its circuit when a magnet is nearby. This was attached to one of the digital inputs to the MCU and test code was set-up so that a state change (circuit open to circuit closed and vice versa) would cause an interrupt service routine to print a notification to the PC's terminal. An LED would also correspond to the state of the door sensor: if the circuit was closed, the LED would be off; if the circuit was open, the LED would be on. This simple test was successful in the lab but, as seen in the field test in Section 4.1.9, the hall-effect sensor was not a very effective solution. This sensor was later substituted for a proximity sensor that produced the same results and was much more reliable.

Each of the two direction sensor receivers was on its own isolated input to the MCU. Through code, we set-up the input pins to trip individual interrupt service routines (ISRs) when the corresponding motion sensor's beam was broken. We tested this by printing a notification to the PC's console when the ISR was tripped. Since this test was successful with both sensors, we needed to ensure the software could determine the proper direction of motion. We set the software to print a "+" to the PC's console when an asset moved onto the cargo vehicle and a "-" when the asset moved off of the vehicle. This test, too, was successful.

4.1.4 Testing the Real Time Clock

The real-time clock (RTC) was tricky to work with because setting it up was a time-sensitive operation; this required using assembly code while initializing the MCU. Once set-up, we tested the RTC by initializing it to 00:00:00 (zero hours, zero minutes, zero seconds) and printing the value to the PC's console every second. When compared with a web-based atomic clock, the timekeeping proved to be accurate.

4.1.5 Testing the RFID Reader

The Alien RFID reader is an intelligent reader. That is, it can be configured to operate in various modes. It also allows the user to apply various types of output formatting to the tags the reader finds. Using its console interface, we set the reader to raise a notification flag on its input/output port when a tag has been detected. We also set the reader so it will dump its list of tags to its serial port when a digital input to the reader is toggled. Sample data from the RFID reader is shown in Figure 19.

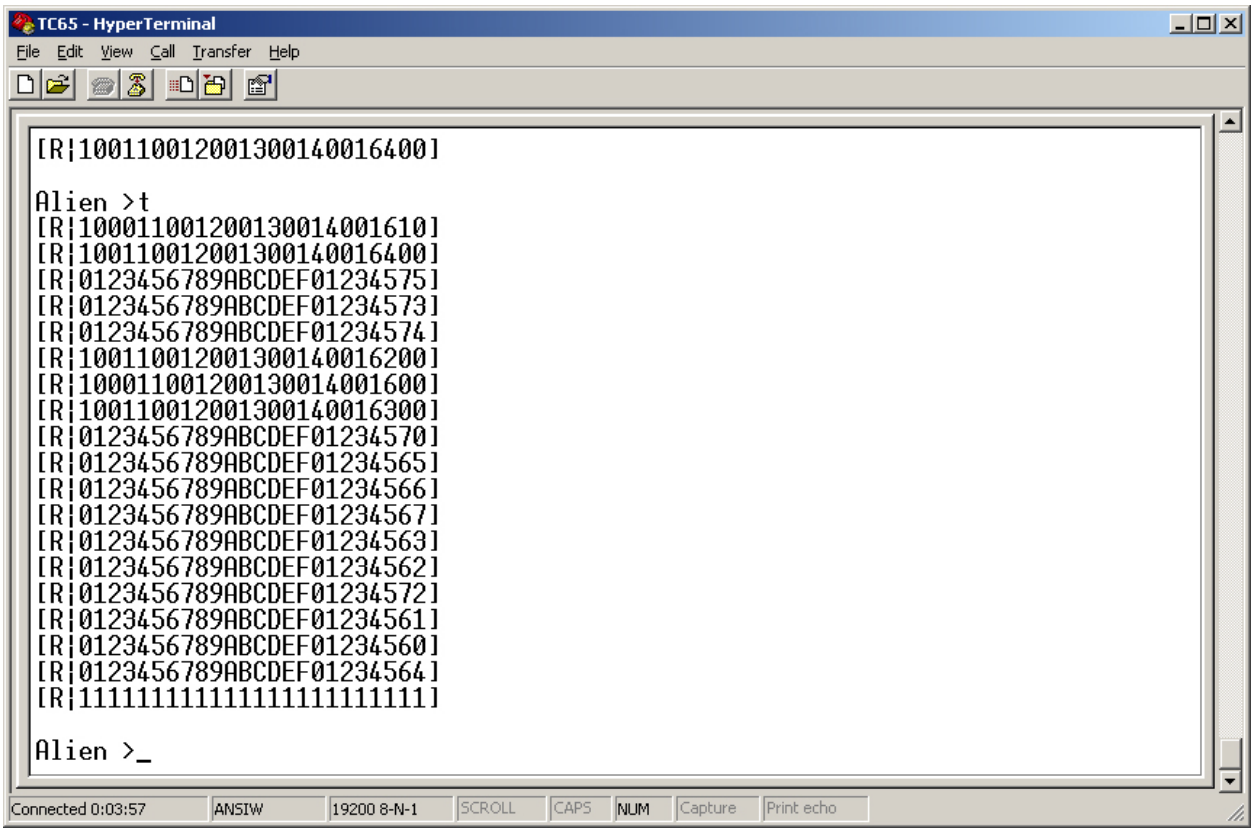

Figure 19: RFID Reader - Tag Read Test

We designed a small function in the microcontroller's code to control the reader, as well as an interrupt to notify the microcontroller that a tag has been found. Through a PC's terminal window, we were able to observe all of the tags we passed between the reader's antennas. With the real time clock, the direction sensors, and the RFID reader all producing valid data, we were ready to begin storing that data to the flash memory.

4.1.6 Testing the Flash Memory

The Amtel flash memory chip is composed of 2048 pages, each of which is 264 bytes wide. Figure 20 is a diagram of this structure:

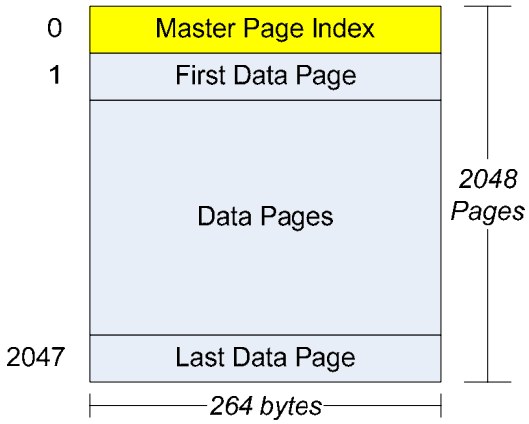

Figure 20: Flash Memory Layout

We needed to develop a small library of functions to work with this memory, including commands to write to the memory's buffers and reading directly from the memory's pages. Once the library was complete, we developed a memory test that would write a given string of characters to varying numbers of pages. Once written, the data was read to a PC's terminal window via RS-232 communication. Then, the data in the given set of pages was erased and the memory was read again to verify the pages were, indeed, erased. The tests proved successful for all of the pages in the memory. The data received from the RFID reader and the motion sensors was then rerouted to the flash memory for storage.

4.1.7 Testing Data Transfer

The next test involved a PC listening to its serial port and outputting any data received from the MCU. We decided to set-up a handshake for data transmission between the MCU and the GPS device. When the MCU is ready to send data to the GPS unit, it sends an 'H' character across the serial port. If it received an 'I' in response, the MCU will spool all the data in its memory to the serial port. In Figure 21, we tested this manually with user interaction and the data was spooled properly.

Figure 21: Data Transfer Test Via Console

We then tested it with a test program built in C# that would listen to the PC's serial port and send an 'I' whenever an 'H' was received. The resulting data stream was dumped to a file so XML parse testing could proceed. Figure 22 shows the output from the execution of the test program, while Figure 23 shows the file generated by the test program.

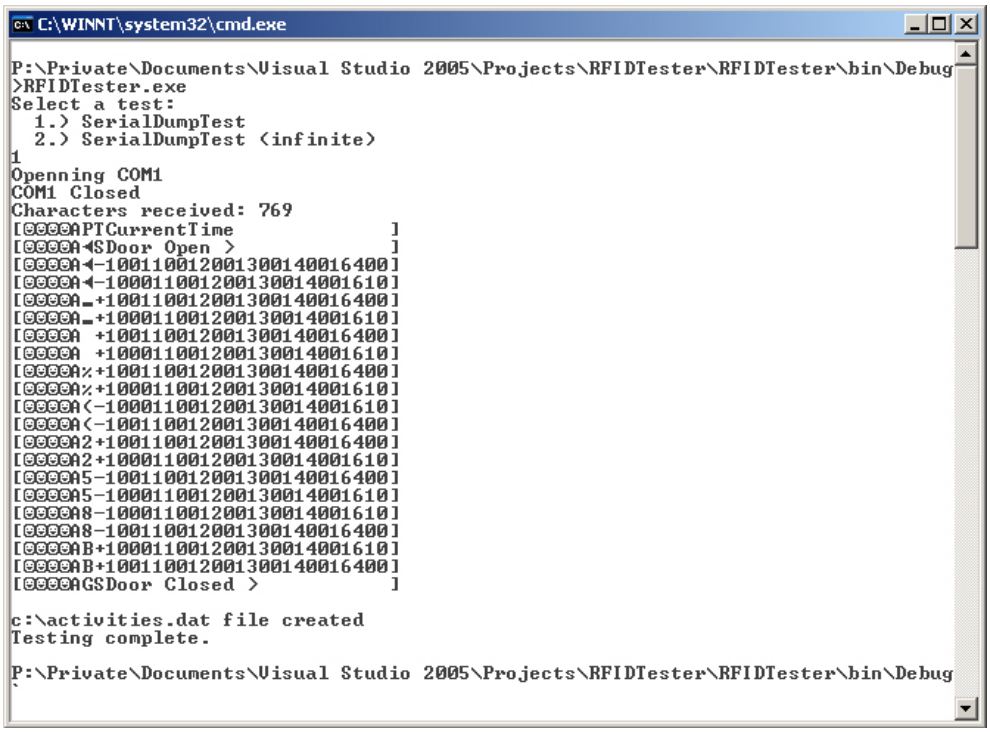

Figure 22: Automated Data Transfer Test

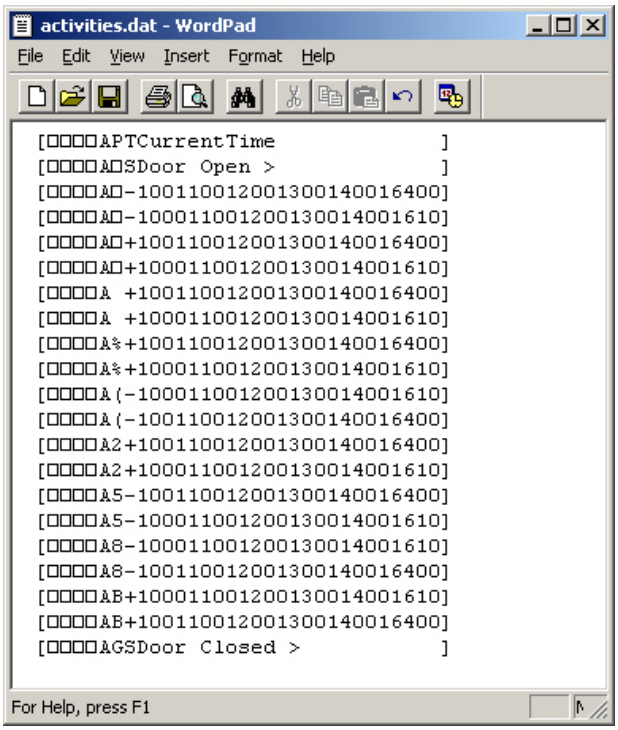

Figure 23: Generated Data File - Data Transfer Test

4.1.8 Testing XML Parsing

The XML formatting appears in Section 4.3.2. The XML parsing feature was tested by supplying the HCP65-G with a text file (activities.dat) containing a sample set of operations.

The text file is formatted with the packet structure that the PIC is configured to use. Figure 23 shows the data file used in this test. Figure 24 shows the XML file generated by the HCP65-G. The resulting xml file can be opened by any xml editor. This test was conducted in the lab so the GPS time and location data were not available, hence the zeroes. The final test (Section 4.1.9) has an XML file with valid GPS data (Figure 27).

```
<?xml version="1.0" encoding="UTF-8" ?>
- <activity>
  - <connection time="00000000000000.000 = 01/01/01 01:41:50" gps="Lat: 0.0; Long: 0.0">
    - <timeStamp time="01/01/01 01:41:07">
        <event status="Door Open >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:12">
        <tag status="Off" tagData="100110012001300140016400" />
        <tag status="Off" tagData="10001100120013001400 1610" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:16">
        <tag status="On" tagData="100110012001300140016400" />
        <tag status="On" tagData="10001100120013001400 1610" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:20">
        <tag status="On" tagData="100110012001300140016400" />
        <tag status="On" tagData="10001100120013001400 1610" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:25">
        <tag status="On" tagData="100110012001300140016400" />
        <tag status="On" tagData="10001100120013001400 1610" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:28">
        <tag status="Off" tagData="10001100120013001400 1610" />
        <tag status="Off" tagData="100110012001300140016400" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:32">
        <tag status="On" tagData="100110012001300140016400" />
        <tag status="On" tagData="10001100120013001400 1610" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:35">
        <tag status="Off" tagData="100110012001300140016400" />
        <tag status="Off" tagData="10001100120013001400 1610" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:38">
        <tag status="Off" tagData="10001100120013001400 1610" />
        <tag status="Off" tagData="100110012001300140016400" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:42">
        <tag status="On" tagData="10001100120013001400 1610" />
        <tag status="On" tagData="100110012001300140016400" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:41:47">
        <event status="Door Closed >" />
      </timeStamp>
    </connection>
  </activity>
```

Figure 24: XML Test Result (ActivityTest.xml)

4.1.9 Testing the System – Field Testing

The final test of the design was to gather data on the device's performance. The goal of the field test was to show the performance of the system its final environment, the back of a trailer. Two field tests were conducted, each consisting of two runs from a warehouse, where individual parts were stored, to the plant in which they were assembled. Team Hardinger is a local logistics company that let us track shipments from their local warehouse to General Electric (GE).

The system was installed in a truck and trailer together. The RFID reader, PIC microcontroller, direction sensors, and door sensor were mounted in the back of the trailer near the door. The HCP65-G (GPS and GSM unit) was installed in the glove box of the truck, and wired to the truck's fuse box. The original plan was to run a serial cable from the PCB in the trailer up to the HCP65-G in the cab. Due to hardware issues involving the serial port on the HCP65-G, data could not be transferred between the two devices. The firmware on the HCP65-G did not recognize the serial port connector while a java program was running. Since the connection would be useless, the serial ports of the PCB and HCP65-G were not wired together. The truck that we were assigned for the test had a sliding door (similar to a typical garage door). This caused a modification to the door detection.

The testing procedure involved detecting skids (or pallets) being loaded to or unloaded from a trailer at both Team Hardinger's warehouse and GE's receiving dock. The products that were shipped were each tagged with a unique RFID tag. The device collected data, following the operation described in Section 3. Once all the skids were loaded, we used a laptop to act as the HCP65-G, and stored all communication to a file. This file was loaded onto the HCP65-G at the warehouse dock. The truck then transported the trailer to GE's receiving dock. The tagged skids were unloaded, and the laptop was used to gather a second file that was loaded on to the HCP65-G. The truck returned the trailer to the warehouse where the Activity.log file was downloaded from the HCP65-G with the results the test.

Two separate tests were conducted. The HCP65-G was not programmed correctly for the first test (GPS data was not being detected), so it was omitted from the test. The data from test 1 is in the form of the activities.dat files. Test 1 consisted of a run between Team Hardinger and GE. Sixteen skids were tagged for the first test. Of the sixteen tags, six were detected during loading (Figure 25) and three during unloading (Figure 26). The direction accuracy was 50% (3 of 6) for the loading and 67% (2 of 3 considering the last occurrence of a tag read) for the unloading. The team determined that the RFID reader needed to be actively searching for tags for a longer period of time after a direction sensor was changed. During the unloading phase, the fork lift was backing out of the trailer with the product (and tag). The back end of the fork lift would trip the direction sensor and the RFID reader would complete its search before the tag entered the field. Also the truck was shaking as the fork lift drove onto it, causing the door sensor to give false door changes (as can be seen in Figure 26) and the direction sensors to give bad directions.

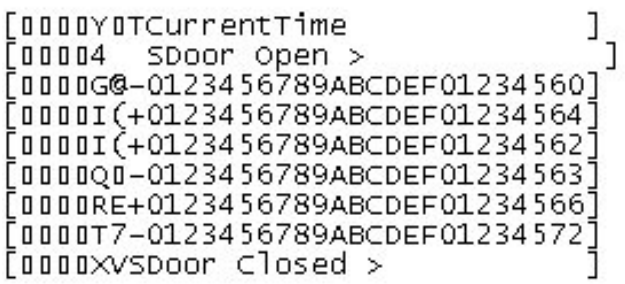

Figure 25: Test 1 Loading Data

```
HH
[□□□□%(TCurrentTime                ]
[□□□□□0□SPower Up >                ]
[□□□□□0□SDoor Closed >             ]
[□□□□1ASDoor Open >                ]
[□□□□1ASDoor Closed >              ]
[□□□□1ASDoor Open >                ]
[□□□□1ASDoor Closed >              ]
[□□□□1SSDoor Open >                ]
[□□□□1SSDoor Closed >              ]
[□□□□1TSDoor Open >                ]
[□□□□1TSDoor Closed >              ]
[□□□□3□SDoor Open >                ]
[□□□□3□SDoor Closed >              ]
[□□□□□0□SPower Up >                ]
[□□□□□0□SDoor Closed >             ]
[□□□□□0"SDoor Open >               ]
[□□□□□0"SDoor Closed >             ]
[□□□□□0#SDoor Open >               ]
[□□□□□0#SDoor Closed >             ]
[□□□□□04SDoor Closed >             ]
[□□□□□05SDoor Open >               ]
[□□□□□05SDoor Closed >             ]
[□□□□1□□SDoor Open >               ]
[□□□□1□□SDoor Closed >             ]
[□□□□1□□SDoor Open >               ]
[□□□□1□□SDoor Closed >             ]
[□□□□1$SDoor Closed >              ]
[□□□□1$SDoor Open >                ]
[□□□□1$SDoor Closed >              ]
[‹Y›□□£@0123456789ABCDEF01234575]
[‹Y›□□£@0123456789ABCDEF01234574]
[□□□□□0□SPower Up >                ]
[□□□□□0□SDoor Open >               ]
[□□□□□0□+0123456789ABCDEF01234575]
[□□□□□0□+0123456789ABCDEF01234575]
[□□□□□0#+0123456789ABCDEF01234574]
[□□□□□0#+0123456789ABCDEF01234575]
[□□□□□0Q-0123456789ABCDEF01234574]
[□□□□T$-0123456789ABCDEF01234562]
[□□□□XUSDoor Open >                ]
[□□□□□1SDoor Closed >              ]
[□□□□□1SDoor Open >                ]
[□□□□□!SDoor Closed >              ]
[□□□□□!SDoor Open >                ]
[□□□□□□□SDoor Closed >             ]
[□□□□□□□SDoor Open >               ]
[□□□□□□ESDoor Closed >             ]
[□□□□#5SDoor Closed >              ]
[□□□□#5SDoor Closed >              ]
[□□□□#6SDoor Open >                ]
[□□□□#6SDoor Closed >              ]
[□□□□#7SDoor Open >                ]
[□□□□%□SDoor Closed >              ]
```

Figure 26: Test 1 Unloading Data

The second test was much more successful. There were several coding changes made after considering the results of the first test. The state machine was revised to its current form, and the RFID reader was reconfigured to search while any direction sensor was tripped. These changes in conjunction with a revised door sensor scheme that used a proximity sensor in place of the hall-effect sensor greatly improved the read rate for the tagged products. The code that operated the HCP65-G was fixed, so the device was included in the second test. The data collected from the microcontroller was loaded onto the HCP65-G at each location so the device could attach GPS time and location data to the loading and unloading processes.

Test 2 involved a shipment of 8 skids loaded two at a time. The loading process (Figure 27) showed a 25% (2 of 8) read rate with 100% direction accuracy. The power up entry in the xml table show that the device lost power during this test. Looking at the data for the unloading process (Figure 28), it can be seen that the first tag read leaving the truck was one of the tags read during the loading process. The first skid leaving the truck is the same as the last skid loaded onto the truck. This shows that the power up event happened 3 seconds before the loading of the last skids. This explains the first test's poor read rate, as power to the device appears to have been interrupted during the test. The unloading process (again Figure 28) shows a 88% (7 of 8) read rate with 100% direction accuracy. All of the tags read in the unloading case were grouped in the same time stamp. All of the skids were not unloaded at the same time. The door open event in the same time stamp is also unusual. Our conclusion is that the either RTC froze, or more likely the time was not updated correctly (probably due to the state machine since the door closed event and all subsequent events contain changing time stamps).

```
<?xml version="1.0" encoding="UTF-8" ?>
- <activity>
  - <connection time="20070420172302.321 = 01/01/01 01:31:21" gps="Lat: 42.125893; Long: -80.025081">
    - <timeStamp time="01/01/01 01:39:16">
        <event status="Door Open >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:44:20">
        <event status="Door Closed >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:30:01">
        <event status="Power Up >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:30:02">
        <event status="Door Open >" />
        <event status="Door Closed >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:30:04">
        <event status="Door Open >" />
        <tag status="On" tagData="100011001200130014001570" />
        <tag status="On" tagData="100011001200130014001580" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:31:16">
        <event status="Door Closed >" />
      </timeStamp>
    </connection>
  </activity>
```

Figure 27: Test 2 Loading Data

```
<?xml version="1.0" encoding="UTF-8" ?>
- <activity>
  - <connection time="20070419201618.290 = 01/01/01 01:43:13" gps="Lat: 42.118538; Long: -79.980695">
    - <timeStamp time="01/01/01 01:42:13">
        <event status="Door Open >" />
        <tag status="Off" tagData="1000110012001300140015800" />
        <tag status="Off" tagData="1000110012001300140015200" />
        <tag status="Off" tagData="1001100120013001400162000" />
        <tag status="Off" tagData="1000110012001300140015400" />
        <tag status="Off" tagData="1000110012001300140015600" />
        <tag status="Off" tagData="300833C2DDD9014028050006" />
        <tag status="Off" tagData="1000110012001300140015000" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:42:17">
        <event status="Door Closed >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:42:18">
        <event status="SDoor Open >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:42:19">
        <event status="Door Closed >" />
      </timeStamp>
    </connection>
  </activity>
```

Figure 28: Test 2 Unloading Data

4.2 Requirements Verification

Table 7 lists each of the engineering requirements from Section 2.1 with respective verifications of how the requirements have been met.

Requirement	Verification
Utilize an ISO 18000-6C - compliant RFID reader that complies with applicable FCC requirements, features anti-tag-collision protocol, and supports at least 2 antennas.	For the purposes of this project, this requirement is fulfilled by an RFID reader from Alien Technologies.
Utilize a GSM unit to transmit data to a server.	This requirement is fulfilled by the HCP65-G. The data it receives is sent via GSM to a server hosted by BTC Solutions.
The completed system involving the RFID reader and GPS unit must store tag, location, time data, and package direction (on/off the vehicle) for transmission.	Once the appropriate asset data is gathered by the microcontroller, it is packaged to be sent to the HCP65-G for further processing. This data is converted to an XML format and stored for later transmission (see Section 4.1 for test results).
Regulated power within device specifications will be supplied from vehicle and backup supply to the GPS device.	The GPS device is attached to a backup power source during installation in the cab of the transport vehicle.
The system will have at least 3.02 KB of memory specifically for data packets.	The HCP65-G has 1.7 MB of space to store log files in, and the microcontroller is interfaced with a 4 Mbit flash chip.

Utilize a GPS device to gather time, location, and velocity data.	The HCP65-G has a GPS receiver, and we have verified that it works properly (see Section 4.1.1 for test results).

Table 7: Engineering Requirement Verification

All of the specific system requirements have been met in the final implementation. However, the interlink between the HCP65-G and the microcontroller is not functional. Data cannot be sent directly from the MCU to the mobile server; it must be manually loaded. This is due to how the HCP65-G operates. When the mobile server's Java code is running, the external serial port is shut-off. While working with the top engineers at BTC Solutions (the advocates of the device), we were unable to devise a solution to this issue. Our industry sponsor, Dave Hartz, recognizes our project as a proof of concept for this type of asset tracking system and recognizes the HCP65-G's lack of functionality; he sees this project is a forerunner to a new version involving components specifically built for two-way communication under Java. As such, he considers the project successful.

4.3 Standards

To allow for efficient data storage and processing, this system follows specific criteria when dealing with asset data. Pertinent asset data includes the time at which the asset crosses the sensors, the direction of the asset (whether it is being loaded or unloaded), and the asset's unique tag data.

4.3.1 Data Storage in the Flash Memory

Table 8 describes how the data is packaged in the microcontroller's memory.

Data Type:	*Start*	*Time Data*	*Indicator*	*Event Data*	*End*
Sample Data:	[	mmddyyhhmmss	+	abcdef0123456789abcdef01	]
Data Size (bytes):	1	6	1	24	1

Table 8: Sample Asset Data

This data structure uses a left bracket ([ ) to denote the beginning of a tag and a right bracket ( ]) to denote the end of a tag; Using special, non-data characters as delimiters proves useful for parsing asset data later in the system, namely in the E.Novations server.

Following the start character, the time at which the asset was loaded or unloaded is stored in a condensed format, where the month, day, year, hour, minute, and seconds are stored one byte apiece. Next is one byte for the asset's direction, which is described by either a '+' (the asset was loaded) or a '-' (the asset was unloaded). This byte could also be an 'S' to indicate a system event (Power Up, Door Open, etc.), a 'T' to indicate the RTC value when a connection was established, or an '@' to indicate an invalid or undetermined direction. Following this are 24 bytes that represent the RFID tag's data. Lastly, the ending character is appended. This data structure is 33 bytes in length and the flash memory's pages are 264 bytes wide, allowing for 8 tags to be stored in each page.

4.3.2 Data Storage in the E.Novations Server

Once a connection is established between the microcontroller and the E.Novations server, the stored asset data is streamed to the server and parsed into XML format. This format was specifically requested by the engineers at BTC Solutions. Figure 29 shows example XML code generated on the E.Novations server.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <activity>
  - <connection time="20070420172734.456 = 01/01/01 01:31:21" gps="Lat: 42.136266; Long: -80.030183">
    - <timeStamp time="01/01/01 01:39:16">
        <event status="Door Open >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:44:20">
        <event status="Door Closed >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:30:01">
        <event status="Power Up >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:30:02">
        <event status="Door Open >" />
        <event status="Door Closed >" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:30:04">
        <event status="Door Open >" />
        <tag status="On" tagData="1000110012001300140015 70" />
        <tag status="On" tagData="1000110012001300140015 80" />
      </timeStamp>
    - <timeStamp time="01/01/01 01:31:16">
        <event status="Door Closed >" />
      </timeStamp>
    </connection>
  </activity>
```

Figure 29: Sample XML Data

Table 9 details the structure of our XML files.

XML Tags	*Description*	*XML Terminator*
<?xml version="1.0" encoding="UTF-8" ?>	This tag denotes the beginning of an XML file. This includes the XML version and the encoding of its characters.	None
<activity>	This tag begins the document (the list of activities).	</activity>

<connection time="**20070420172734.156 = 01/01/01 01:31:43**" gps="**Lat: 42.136266; Long: -80.030183**">	This tag denotes an established connection between the E.Novations server and the microcontroller. This includes the GPS date and time of the connection as well as the GPS coordinates.	</connection>
<timeStamp time="**01/01/01 01:30:01**">	This tag denotes a timestamp. This includes the date and time of the event.	</timeStamp>
<event status="**Power Up >**" />	This tag denotes an important event. This includes the system's status during that event.	None
<tag status="**On**" tagData="**1000110012001300140015 70**" />	This tag denotes a tag event. This includes the tag's status (whether it is moving onto or off of the cargo vehicle) as well as the tag's data.	None

Table 9: XML Asset Data Format

5. Summary and Conclusions

Our plan to develop an asset tracking system was successful. With constant and accurate communication with our industry sponsor, we were able to develop this plan; with guidance from our faculty advisors and persistence as a team, this plan came to fruition. We are proud to say this project represents the culmination of our unique skills, especially those developed during our academic tenure at Penn State Behrend.

In discussions with the engineers of BTC Solutions, we learned our system is the proof of concept for another system that will be developed in the coming years. This system will involve two compact units, one in the cab of the transport vehicle and the other in the trailer. These units will communicate wirelessly, and will be built solely upon multi-threaded Java Virtual Machines. Once the next generation system is developed, it will be utilized by logistics control companies worldwide to help secure and expedite their supply chains.

If any further work were considered for the device we developed, we suggest working with different types of motion sensors and door sensors, as more costly devices may yield more consistent results. Also, using the originally intended Symbol MR400 RFID Card would be advised so that creating a low-power system would be practical.

This project was an excellent experience for the three of us. We learned how to apply input from industry leaders and faculty advisors in developing a useful system while maintaining focus on the core system requirements. We also learned that no plan is perfect and that being flexible and thinking radically is often the key to achieving success. Other lessons we learned in the course of this project include:

- Ideal lab conditions are not the best conditions for preparing for real-world testing.
- There are many ways to reach the same results.
- Prototypes and products that are not currently in production should not be included in device selection considerations.
- Technical documentation is great for reference during implementation, but verifying the device's intended usage with its company during the design phase is always wise.
- Hand drawings are excellent aids for well-structured programming.
- Automated testing can reveal hidden bugs.
- Reliable bugs can be a programmer's best friend.
- Bypass capacitors can help with strange system behavior.
- Creating custom function libraries makes working with external devices much more tolerable.
- No test is perfect, there is always one more aspect to verify and validate.
- Breaking the system is the first step in enhancing it.
- Even odd explanations have their roots in reality.

The most rewarding part of this project was seeing its components work in unison; seeing that made all the time and effort spent in the laboratory and the classroom a worthwhile investment.

6. References

[1] Bureau of Transportation Statistics, “Table 3-18: Total Operating Revenues”, dated 2000. Available at <http://www.bts.gov/publications/national_transportation_statistics/2002/html/table_03_18.html>.

[2] HowStuffWorks.com, “How GPS Receivers Work”, copyright 1998-2006 by HowStuffWorks, Inc. Available at <http://electronics.howstuffworks.com/gps.htm>.

[3] Investorwords.com, “supply chain”, copyright 1997-2005 by WebFinance, Inc. Available at <http://www.investorwords.com/4823/supply_chain.html>.

[4] Symbol Technologies, “RFID – A Revolution in Asset Management”. Available at <http://www.symbol.com/products/rfid/rfid.html>.

[5] C. Wassel, and A. Onda “RFID Technology Briefing,” Electronic Presentation, Center for eBusiness and Advanced IT, copyright 2006.

[6] “Base Station Subsystem”, Wikipedia, last modified 22 October 2006. Available at <http://en.wikipedia.org/wiki/Base_Station_Subsystem>.

[7] “GPRS Core Network”, Wikipedia, last modified 3 October 2006. Available at <http://en.wikipedia.org/wiki/Network_and_Switching_Subsystem>.

[8] “Global System for Mobile Communications”, Wikipedia, last modified 20 October 2006. Available at <http://en.wikipedia.org/wiki/Global_System_for_Mobile_Communications>.

[9] “Network Switching Subsystem”, Wikipedia, last modified 10 October 2006. Available at <http://en.wikipedia.org/wiki/Network_and_Switching_Subsystem>.

[10] “Short Message Service”, Wikipedia, last modified 23 October 2006. Available at <http://en.wikipedia.org/wiki/Short_message_service>.

[11] “NMEA Data”, last modified 27 February 2006. Available at <http://www.gpsinformation.org/dale/nmea.htm>.

[12] “Extensible Markup Language (XML)”, World Wide Web Consortium, last modified 9 September 2006. Available at <http://www.w3.org/XML/>.

Appendix A: Project Management Plan

A.1: Work Breakdown Structure

Table 10 is the work breakdown structure for this project; it shows who was responsible for which part of the project and in what order events needed to occur. Some entries are struck-through due to the substitution of RFID readers.

ID	Activity	Description	Time (days)	Predecessors	People	Deliverables	Lab Resources
1	***Material Arrival***			-	-	-	-
1.1	Mobile Server Arrival	E.Novation Mobile Server arrives.	0	-	Dave Hartz (BTC Solutions)	-	-
~~1.2.1~~	~~Symbol RFID Reader Arrival~~	~~Symbol Technologies' MR400 RFID Reader arrices.~~	~~x~~	-	~~Manpreet Singh (Symbol Tech.)~~	-	-
1.2.2	Alien RFID Reader Arrives	Alien Technologies' RFID Reader arrives.	0	-	Chris Wassel (Penn State Behrend)	-	-
1.3	Device Initial Testing	Testing the operations of the E.Novation Mobile Server and RFID reader.	2	-	Jeff, Mike, Steve	Basic understanding of device I/O.	Power supply, oscilloscopes, multi-meters
1.4	Parts Arrival	Sensors, power supplies, batteries, wiring, MCUs and flash memory arrive.	7	-	-	-	-
1.5	Test Truck Available	Truck that will be sent with an actual shipment to GE is available for mounting, wiring	0	-	Dave Hartz (BTC Solutions), Frank Przybycin (Team Hardinger)	-	-
1.6	Material Arrival Milestone			1.1 – 1.5	Jeff, Mike, Steve	-	-
2	***Software Development***			-	-	-	-

ID	Activity	Description	Time (days)	Predecessors	People	Deliverables	Lab Resources
2.1	MCU	Write code to utilize the MCU's basic I/O functions.	1	1.4, 3.1	Jeff, Mike, Steve	MCU's I/O functional.	Power supply, oscilloscope, multi-meters, computer.
2.2	Sensor Interface	Interface the MCU with the door and direction sensors.	1	1.4, 2.1	Jeff, Mike, Steve	Sensors interact with the MCU.	Power supply, oscilloscope, multi-meters, computer.
2.3	Flash Memory Interface	Interface the MCU with the flash memory chip.	5	1.4, 2.1	Jeff, Mike, Steve	Flash memory interacts with the MCU.	Power supply, oscilloscope, multi-meters, computer.
~~2.4.1~~	~~Symbol RFID Reader Interface~~	~~Interface the MCU with the Symbol MR400 RFID Reader.~~	~~5~~	~~3.4.1~~	~~Jeff, Mike, Steve~~	~~MR400 communicates with the MCU~~	~~Power supply, oscilloscope, multi-meters, computer.~~
2.4.2	Alien RFID Reader Interface	Interface the MCU with the Alien RFID reader.	2	3.4.2	Jeff, Mike, Steve	RFID reader communicates with the MCU.	Power supply, computer.
2.5	Mobile Server Interface	Interface the MCU with the mobile server.	3	3.3	Jeff, Mike, Steve	Mobile server communicates with the MCU.	Power supply, computer.
2.6	XML Parsing	Create an XML parsing routine on the mobile server.	3	3.3	Steve	Mobile server parses data into XML.	Power supply, computer.
2.7	Software Completion Milestone			2.1 – 2.6			
3	***Hardware Development***			-	-	-	-
3.1	PCB for MCU	Design and mill a printed circuit board for the microcontroller and its interfaces.	7	-	Jeff	PCB for MCU created and populated.	Soldering station, milling machine, computer.

ID	Activity	Description	Time (days)	Predecessors	People	Deliverables	Lab Resources
3.2	PCB for Flash Memory	Design and mill a printed circuit board for the flash memory chip.	2	-	Jeff	PCB for flash memory created and populated.	Soldering station, milling machine, computer.
3.3	Mobile Server Setup	Set-up the mobile server with the necessary wiring.	1	1.1, 2.1	Jeff	Mobile server is ready to receive code and data.	Computer.
~~3.4.1~~	~~Symbol RFID Reader Setup~~	~~Set-up the physical interface between the MR400 and the MCU.~~	~~3~~	~~1.2.1, 2.1~~	~~Jeff~~	~~MR400 physically interfaced with MCU~~	~~Oscilloscope, multi-meters.~~
3.4.2	Alien RFID Reader Setup	Set-up the physical interface between the Alien RFID reader and the MCU.	2	1.2.2, 2.1	Jeff	Alien RFID reader physically interfaced with the MCU.	Oscilloscope, multi-meters.
3.5	Component Setup (Lab)	Set-up a simulation environment for lab-based testing.	1	1.6, 3.1, 3.2, 3.3, ~~3.4.1~~, 3.4.2	Jeff	A repeatable setup has been created for in-lab testing.	-
3.6	Component Setup (Field)	Set-up the system components for in-field testing.	1	3.5	Jeff, Mike, Steve	Components are set-up in the test vehicle.	-
3.7	Hardware Completion Milestone			3.6	Jeff, Mike, Steve	-	-
4	***Testing***			-	-	-	-
4.1	GPS	Test the mobile server to ensure it receives valid GPS data.	2	3.3	Jeff, Mike, Steve	Data logs of GPS data.	Computer.
4.2	MCU	Test the I/O of the MCU.	1	2.1	Jeff, Mike, Steve	MCU I/O confirmed operational.	Oscilloscope, multi-meters.

ID	Activity	Description	Time (days)	Predecessors	People	Deliverables	Lab Resources
4.3	Door Sensor	Test the door sensor's integration with the MCU.	1	2.2	Jeff, Mike, Steve	Door sensor produces repeatable results.	Computer.
4.5	Direction Sensors	Test the direction sensors' integration with the MCU.	2	2.2	Jeff, Mike, Steve	Direction sensors produce repeatable results.	Computer.
4.6	Real-Time Clock	Test the real-time clock on the MCU.	5	2.1	Jeff, Mike, Steve	The RTC produces accurate, repeatable results.	Computer.
4.7	Flash Memory	Test the flash memory's integration with the MCU.	5	2.3	Mike, Steve	Data can be written to and retrieved from all memory positions.	Computer.
4.8	Data Transfer	Test data transfer between the MCU and the mobile server.	3	2.5	Mike	Data transfers successfully between the MCU and the mobile server.	Computer.
4.9	XML Parsing	Test the XML parsing routine on the mobile server.	3	2.6	Steve	Data from MCU is parsed successfully into a valid XML document.	Computer.
4.10	System	Test the entire system in the field.	5	4.1 – 4.9	Jeff, Mike, Steve	The system produces real-world results.	-
4.11	Test Completion Milestone			4.10	Jeff, Mike, Steve	-	-
5	***Documentation***			-	-	-	-
5.1	Test Results	Document and analyze test results.	2	4.10	Jeff, Mike, Steve	Test results are recorded and analyzed for all tests.	Computer.

ID	Activity	Description	Time (days)	Predecessors	People	Deliverables	Lab Resources
5.2	Create Final Project Report	Generate the report for the entire project.	15	5.1	Jeff, Mike, Steve	Report generated for submission.	Computer.
5.3	Create Final Presentation	Create the final presentation based on the final project report.	2	5.2	Jeff, Mike, Steve	Presentation created for review by advisors.	Computer.
5.4	Revise Final Presentation	Revise the final presentation based on input from advisors.	1	5.3	Jeff, Mike, Steve	Presentation finalized.	Computer.
5.5	Project Completion Milestone			5.4	Jeff, Mike, Steve	-	-

Table 10: Work Breakdown Structure

A.2: Member Contributions

Steve

Our resident Java programmer, Steve was responsible for creating the XML parser and debugging the code that drives the E.Novations mobile server. Steve was also our primary spokesman and record keeper during meetings with our advisors and industry sponsors.

Mike

Our software engineer, Mike was responsible for commenting, cleaning, and simplifying code as well as creating test programs for the various system components. Mike was also responsible for error-checking and formatting the final report and maintaining good communication with Dave Hartz of BTC Solutions.

Jeff

Knowledgeable about printed circuit board fabrication, our “hardware guy” Jeff was responsible for designing, creating, and populating the PCBs necessary of the project. Jeff was also responsible for maintaining good communication with Frank Przybycin of Team Hardinger Transportation / Warehousing and arranging appropriate field tests.

As a Team

Working as a team, we wrote and debugged code, identified system problems and discussed solutions, wrote our design reports and related documents and, of course, gave the required presentations to reflect this project’s progress.

A.3: Development Costs

Aside from a lot of time, pizza, and Mountain Dew, this project incurred certain development costs. Table 11 describes the costs related to the development of the final system.

Component	Price	Part Number	Supplier	Comments	Qty	Ext
Mikroelektronika mikroC dsPIC compiler.	$249.00	Version 3.0	Mikroelektronika	This compiler is much more robust than the compilers available on-campus and is includes libraries of functions that proved quite useful during development. Donated by Mike.	0	$0.00
Microcontroller.	$5.02	PIC24FJ128GA006	Microchip	Free sample.	1	$0.00
PICKit2 Programmer.	$34.99	-	Microchip	Donated by Jeff.	1	$0.00
DB9 Connectors.	$0.49	DB-9S	All Electronics	DB9 connector	2	$0.98
Optoisolator	$0.32	160-1300-5-ND	Digikey	4N25 isolator	8	$2.56
MAX3232 Chip	$5.04	MAX3232ECPE+-ND	DigiKey	Max232 (3.3V compatible)	1	$5.04
Surf Board for PIC	$0.90	64PINLQFP	Futurlec	64-Pin surfboard	1	$0.98
Flash Memory Chip	$5.70	AT45DB041B-RI-2.5-ND	DigiKey	4Mbit flash memory	1	$5.70
Printed Circuit Board	$0.00	-	-	Free From PCB Lab	1	$0.00
12V Car Battery	$60.00	-	-	Donated by Jeff.	1	$0.00
DC/DC converter.	$40.95	175661PS	Jameco	12v (truck power) to 5vdc +/- 5% @1.5A peak.	1	$40.95
Door sensor (sensor).	$3.59	CH407-ND	Digi-key	-	2	$7.18
Door sensor (magnet).	$3.38	CH413-ND	Digi-key	-	2	$6.76

Component	Price	Part Number	Supplier	Comments	Qty	Ext
Direction Sensor (emitter).	$22.00	DME-00-1A	AutomationDirect	Photobeam emitter.	2	$44.00
Direction Sensor (receiver).	$31.00	DMR-0N-1A	AutomationDirect	Photobeam receiver.	2	$62.00
Alien RFID Reader.	$1800.00	9780	Alien Technologies	Donated by RFID Center.	1	$0.00
Alien RFID Antenna.	$700.00	-	Alien Technologies	Circular Antenna, donated by RFID Center	2	$0.00
				Total Cost:		$176.15

Table 11: Development Cost Itemization

Appendix B: Software

The disc below contains all the software for this project, as well as relevant technical documentation, and test data (photographs, videos, and raw data). The code for the microcontroller is in C, while the code for the HCP65-G is in Java.