Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.

4. Title and Subtitle	5. Report Date
Impending Box Impact Warning System for Prevention of Snowplow-Bridge Impacts	Dec 21, 2008
	6.

7. Author(s)	8. Performing Organization Report No.
Richard R. Lindeke, Hilal Katmale, Ravi Verma	

9. Performing Organization Name and Address	10. Project/Task/Work Unit No.
UMD Mechanical & Industrial Engineering Department University of Minnesota Duluth 10 University Drive Duluth. MN 55812	11. Contract (C) or Grant (G) No.

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
Northland Advanced Transportation Research Laboratory University of Minnesota Duluth 10 University Drive Duluth. MN 55812	Final Report, July 2006 – Dec 2008
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract (Limit: 200 words)

Each year, three or four Mn/DOT snowplows suffer bridge/box collisions while plowing. These collisions can shear off the box and frame damage to the truck. The box then falls onto the road surface where it becomes an immediate life-threatening hazard to traffic. In some cases, the integrity of the bridge may also be compromised. A typical collision of this type requires expenditures of $30,000 to $40,000 and results in potentially dangerous delays in achieving clean pavement status along the affected snowplowing route.

Feasibility of linking on-board GPS technology for Automatic Vehicle Location with the current bridge information database at Mn/DOT, "BrInfo," will be investigated, on a plow-route by route basis, to create collision maps. Collision avoidance then will use some primitive form of map matching. In addition, a prototype warning system that serves as a bridge proximity sensor will be developed to alert the snow plow driver that he/she is approaching a bridge with the box at a dangerous height. This warning system is integrated in an on-board box position sensor so that the driver can be alerted that the box must immediately be lowered. While realizing that additional means for box height control may complicate snowplow maintenance, any system that relieves the driver of cognitive overload, to reduce driver stress and fatigue during plowing operating, when running extended rural plow routes, needs to be implemented.

17. Document Analysis/Descriptors	18. Availability Statement
Collision Avoidance, GPS, On-board Controller, Obstacle Identification, Snowplow Safety	No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		

Impending Box Impact Warning System for Prevention of Snowplow-Bridge Impacts: A Final Report of Investigations

Prepared by:

Richard R. Lindeke, Ph.D.

Hilal Katemale

Ravi Verma

Department of Mechanical and Industrial Engineering
University of Minnesota Duluth

December 2008

Published by:

Center for Transportation Studies
University of Minnesota
200 Transportation and Safety Building
511 Washington Ave. S.E.
Minneapolis, MN 55455

Acknowledgements

The authors wish to thank Dr. E. Kwon for his guidance in keeping us on-track. We also thank the staff at Mn/DOT District 1 in particular: John Cavanaugh, Robert Ege, Kerry Monson (and his maintenance mechanics) as well as the plow truck crew at Nopeming working with John Shallow for running the road testing of the prototype. We also wish to thank the St. Louis County Road Maintenance department staff members in Duluth and Virginia, and in particular Pete Eakmon a staff engineer with the St. Louis County Highway Department, who helped to develop the on-board solution by letting the authors mount a test system on their plow as well.

Table of Contents

List of Tables

List of Figures

Executive Summary

When our team was approached by Mn/DOT and the Northland Advanced Transportation System Research Laboratory (NATSRL) to address the problem of impacts between the sand/chemicals box on a snow plow and fixed obstacles of various types, we began the search for an inexpensive, yet reliable, box/obstacle collision avoidance system for use on Mn/DOT snowplows throughout District 1 and eventually the entire state of Minnesota. This study began with a review of available obstacle detection equipment and included mobile as well as infrastructure based equipment. This search included an exploration of truck mounted onboard GPS (Global Positioning Systems), radar or sonar systems as well as radio based (RFID – Radio Frequency Identification) stationary equipment to mark obstacles. In addition to these solutions, the team considered development of obstacle maps and the use of active GPS devices to explore a vehicle's immediate surroundings for potential collision. When we considered that the primary objective to achieve the goal was to keep the system cost well under $750 for each snowplow, the team's options were limited. A final prototype solution which consisted of a hardened on-board microcontroller, a GPS sensor and a box position sensor was designed at a cost of about $575.

The research team chose the Rabbit Semiconductor OP7200 as the human-machine interface (HMI) because it was easy to program (it was supplied with "Dynamic C" programming language) included a touch screen for operator communication with the controller, runs on a wide range of dc power meaning it will operate using plow truck power and includes a full compliment of digital input and output ports that are directly recognized by the controller. A Garmin 15 H was selected as the GPS, again for its low cost, its ability to accept plow truck (dc) power input, and its ease of programming. The final deciding factor was that this GPS unit has the ability to continue dead reckoning its position for several seconds (up to 30 seconds) even if it loses it connection to the satellite operational constellation. The teams' final hardware choice was the GE Security 166 magnetic switch. This device was selected as the proximity sensor to identify the position of dump box because of its hardened construction and the fact that it was designed for heavy equipment application.

With the system chosen, one that required a reliable obstacle map, the team began to develop this map from Mn/DOT's Br-Info database. It listed bridge clearance heights, widths, and most importantly, physical location in longitude and latitude listed as degree-minute-second (dms) formats. Using this list, which the team copied into MS Access, a Visual Studio 2005 (VBasic.net) program was developed to extract the bridge locations which were written into an obstacle map that could be easily transferred to the onboard controller. When the team obtained the District 1 Br-Info database, the team also obtained two other data sets from officials in Duluth. These were the snowplow route database ("Plowroute" database) which indicated starting and ending mileposts along the highways that Mn/DOT needed to plow in the district and a database ("mileposts" database) that listed longitude and latitude of (nearly) all the mileposts along the state maintained highways in the district. The program began by building a user interface that allows the database manager to observe the process of bridge extract to all of the routes within a given Mn/DOT District. The user begins the program by entering the total

number of plow routes within the district. The VBasic program then opens the plow rote database and extracts each highway on a given route and the starting and ending mileposts along all of the routes. In the case of the plow route databases provided to the team by District 1 personnel, the routes (of which 103 were listed) in this database were, surprisingly, all single roadway maps. While the team suspected that these were not in fact the most recent plow route listing, it simplified the extraction process to our files and therefore, they were used. After the highway numbers and distances (in the form of Starting and Ending mileposts) belonging to each plow route were obtained by extracting information from the "plowroute" database the program entered the "mileposts" database to extract the specific geometry (longitude and latitude) for all of the mileposts along the designated highway for any given plow route. Then, the Br-Info database was searched for any bridges with a clearance greater than zero – indicating that the bridge was in fact an overpass – and whose geometry was between each successive pair of mileposts. Because a one-mile grid is relatively coarse in urban areas and could inadvertently extract bridges that are not alone the highway under consideration (ones that may belong to other plow routes) before entering the Br-Info search, each milepost distance pair was divided by 10 to narrow the search grid to the approximate width of an intersection and then converted to the dms format. Once initial testing was conducted, along a route called 101 in our plow route mapping that ran along I35 through downtown Duluth from West 40th Avenue to East 26th Avenue, the onboard controller was unable to locate many obstacles. As the team further checked into the problem several items were discovered. First, the "plowroute" database we were provided was not, in fact, up to date and only the several I35 route and selected other routes along major trunk highways were single routes, most covered parts of several routes as the team had suspected. Again this was not fatal for testing at the early stages since the plow route the team would test was in fact this same one the team had numbered 101 (even though the number in District 1 records had changed). A further problem was discovered, it seemed that all the bridges (and tunnels) constructed when I35 was extended beyond W. 5th Avenue in Duluth were not included. This problem was traced to the Br-Info database which was an older version rather than the most recent one, a problem related to homeland security issues. Finally, a fatal problem with the original approach to the project was identified: not all potential collision obstacles are bridges! Other permanent obstacles that could have collisions with raised sand boxes like pedestrian crosswalks, highway road signs, overhead power lines, etc. and temporary obstacles like construction equipment and signs, and even tress or sagging power, telephone or cable television lines that would never be found in the a regular database of "hard" Mn/DOT assets (like the Br-Info database) should also be available to the onboard controller. Because of the missing information and presence of many types of obstacles not in Br-Info, the team developed a controller application they designated 'Trainer.' This application was run to build the test routes that were used in all on-road testing actually performed.

The team conducted extensive testing of the IBIAS (Impending Bridge Impact Alerting System) controller both on a test route and then after mounting on a Mn/DOT District I snowplow and one operated by St. Louis County Maintenance Department during winter 2007-2008. While not completely successful, the team discovered many issues which are being addressed, at no further expense at this time to NATRSL or Mn/DOT. Problems focused in two areas: first with the number of obstacles, and resulting computational overhead, and with the action of the box position sensor. While the first problem is not solved, the team has redesigned the sensor mounting system and this new mount is the principal subject of on going testing.

Chapter 1: Introduction

1.1 Research Motivation

It was late into this very long day. The snow has stopped falling but then so is the temperature. Tom, a relief plow operator running his second route of this 16 hour shift, thought as he turned the plow to begin the run back toward the service garage along Route 23. This new route is a bit trickier than the one he normally runs, and this job is way more trying than the lab job that is his normal work. Well only about 30 more minutes of running and he could get out of the truck and stretch. Darn, he thought, looks like the dump box is almost empty – I'll just tilt it up a bit more so the materials will slide back to the spreader. After driving for about three (3) miles he felt that all was well, but the visibility was poor, now, he thought, all we need is this fog setting in. Another two (2) minutes and "BAM" want was that – "oh my Gosh," he thought, "I just rammed the box into that overpass – now there will be hell to pay!"

Sure Tom made a mistake, after raising the dump box he forgot to lower it as he continued to run this unfamiliar route – but can you really blame him, after all those hours in a different job than his regular work. Not really, but still, a plow is now disabled and a route still to be finished, and how much damage can there be to the bridge? Likely the total cost of this accident will run into the thousands, records indicate those costs at anywhere from $30,000 to $40,000 per incident, and they happen three (3) to four (4) times each year in Minnesota alone. Can't a system be designed that will avoid this costly and dangerous problem?

When our team was approached by Mn/DOT and the Northland Advanced Transportation System Research Laboratory (NATSRL) to address this problem, we began the search for an inexpensive, yet reliable, box/obstacle collision avoidance system for use on Mn/DOT snowplows throughout District I and eventually the entire state of Minnesota. This study began with a review of available obstacle detection equipment. This search included an exploration of truck mounted onboard radar or sonar as well as radio based stationary equipment. In addition to these solutions, the team considered development of obstacle maps and the use of active GPS devices to explore a vehicle's immediate surroundings for potential collision. When we considered that the primary objective to achieve the goal was to keep the system cost well under $750 for each snowplow, the team's options were limited. A final prototype solution which consisted of a hardened on-board microcontroller, a GPS sensor and a box position sensor was designed at a cost of about $575. This prototype system was deployed, with mixed results, on both a Mn/DOT District 1 snowplow as well as a St. Louis County Maintenance Department snowplow during the winter of 2007/2008. While proving marginally successful, in part due to sensor problems and a high number of obstacles on the chosen routes, the system appears to be a potential solution worth carrying forward through full system debugging and finalizing for fleet implementation.

An assessment of current "box up" warning systems was made so that improvements in the interaction of driver and information input can be identified. The feasibility of linking on-board GPS technology for Automatic Vehicle Location or AVL as documented by Alexander, et al

2004, Alexander et al 2005, Aono 1998, Dmitriev, et al 2000, and Nookala & Estochen 2002 with the current bridge information database at Mn/DOT, "BrInfo," will be investigated essentially on a plow-route by route basis to create collision maps. The first line of defense for collision avoidance then will be using some primitive form of map matching as suggested in the literature, see Bouju et al 2002, Dmitriev et al 2000, Hofmann-Weilenhof 1997, or Joshi 2002. In addition, a prototype warning system that serves as a bridge proximity sensor will be developed to alert the snow plow and its driver that the unit is approaching a bridge with the box at a dangerous height as suggested by Bostelman et al 2005, Ewald & Wilhoeft 2000, Hong et al 2004, and Watanabe et al 2002. This warning system should also be integrated into an on-board box position sensor so that the driver can be alerted that the box must immediately be lowered, or as a signal to the automatic box controller software system that the box must, temporally, be lower to a safe height to allow the vehicle to pass under the bridge or fixed roadway obstacle. While not the main thrust of this effort, this project will further investigate the types of driver warning (visual, audio, or tactile) that will be the most effective if the semi-automated solution for collision avoidance system is chosen.

While the research team realizes that additional, automated, means for box height control may complicate snowplow maintenance activities, any system that effectively relieves the driver of additional cognitive overload, which should reduce both driver stress and fatigue do to sand and chemical treatment during plowing operating, in particular when the plow drivers are running extended rural plow routes, needs to be to be explored and ultimately implemented.

As part of the work, an economic analysis will be made to determine the economic feasibility of fleet-wide implementation. Previous work at Mn/DOT include a collision avoidance prototype study for low visibility conditions; this included instrumentation of a squad car, ambulance, snowplows and transit buses as documented by Estochen (8), Nookala and Estochen (20) as well as Alexander, et al (1 and 2). Nishi and Takagi (2001) have developed a collision avoidance algorithm, although this is primarily for a potential automobile in which acceleration, braking, and steering can be controlled. The mechanics of a vehicle impacting a concrete bridge girder are summarized in Quio, Yang, and Mosallam (2004); forces, contact time, and deflections were modeled using finite element analysis. Many paper have been prepared in the general topic of Map Matching these in clued studies of complex road following using GPS and Dead Reckoning, see Aona, et al (3) the work of Joshi (15) Bouju, et al (5) and Dmitriev, et al (7) .

Several authors have explored various imminent collision detection method ranging from VORAD radar (Alexander, et al 2004) to Ladar (laser based 'radar'), optical sensors all the way to built-in vision systems for further information see Bostelman et al 2005, Ewald & Wilhoeft 2000, Hong et al 2004, Nishi & Takagi 2001, and Watanabe et al 2002.

System integration, especially with respect to fleet management systems, intelligent vehicles, and information systems, will be addressed in this research. Kroeger and Sinhaa (2004) discuss the analysis of data and information technology with respect to Iowa DOT's winter maintenance operations and note the potential for improving levels of service and reducing costs. Vanderhoe et al. describe the types of data that can be obtained from Wisconsin DOT's winter concept vehicles, integrated with global positioning, in the "Wiscplow" program.

1.2 Report Organization

In this report, Chapter Two will detail the team's search for, and selection of the sensing technology to be employed for this low cost system. Chapter Three will explore the development of obstacle maps starting with the Mn/DOT BrInfo databases and associated information and culminating into a system developed to identify all potential obstacles beyond just the hard assets maintained by Mn/DOT. Chapter Four will detail the design and testing of the system, including hardware and software, and the efforts used to introduce it into the snowplow fleet. Finally, in Chapter Five the team will summarize the project and make suggestions for further work to perfect the system that has been developed.

Chapter 2: An Exploration of available Obstacle detection systems

2.1 Collision Avoidance Systems

Vehicle-mounted collision warning systems (CWS) are deployed to improve the ability of drivers to avoid accidents. These applications use a variety of sensors to monitor vehicle surroundings and alert the driver for conditions that could lead to a collision. Examples of this system include forward collision warning, obstacle detection systems, and road departure warning systems. Figure 1 shows the applications possible using collision avoidance systems.

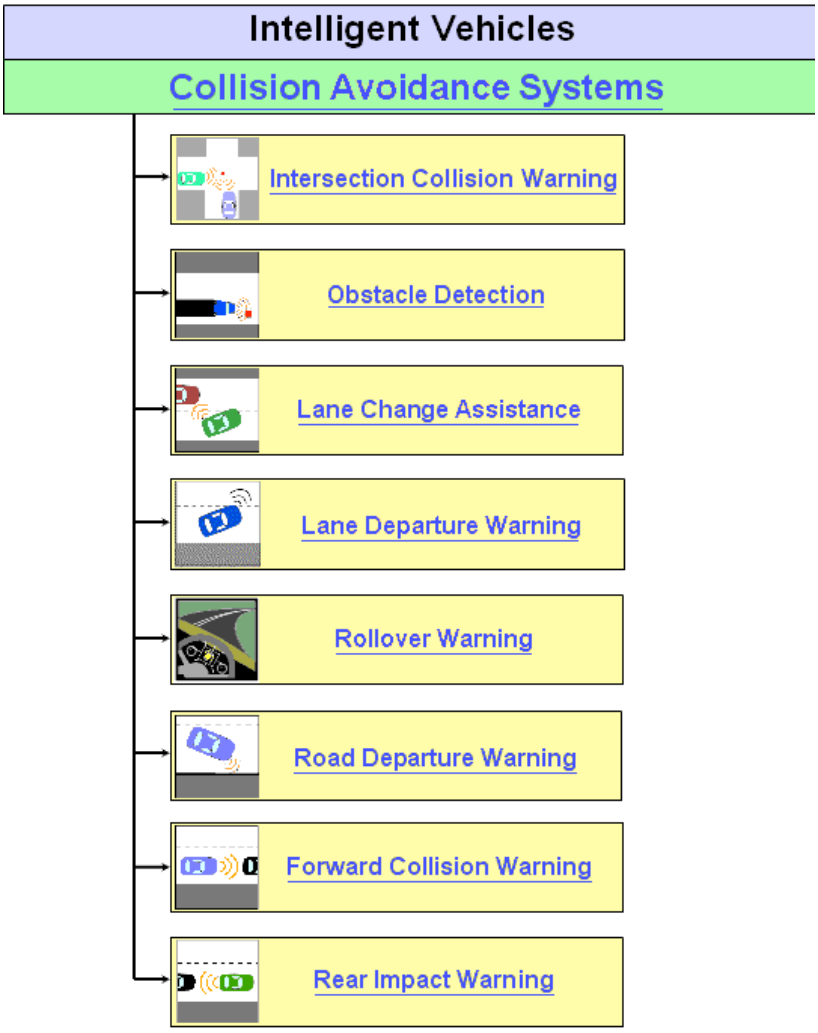

Figure 5: Collision Avoidance Systems (US Department of Transportation, 2007e)

- **Benefits –** The Japan Highway Public Corporation (JH) tested administrative pace-vehicles with millimeter radio wave sensors and GPS technology to lead freeway traffic through heavily fogged areas that were subject to road closures. The sensors were able to detect vehicles or a corrugated paper case 0.375 x 0.475 x 0.375 m through 100 meters of heavy fog. The system attached sensors to leading vehicles and allowed groups of freeway traffic to follow using a warning vehicle in the rear. The emergency management center monitored the ITS-vehicle using GPS and enabled them to track each others position (U.S.Department of Transportation, 2007e).
- **Costs –** Table 1 gives approximate unit costs (adjusted) for equipment for Commercial Vehicle On-Board (CV) (U.S.Department of Transportation, 2007f).

Table 2: Equipment Costs for Commercial Vehicle Onboard Equipment

Unit Cost Element	IDAS #	Life *Years*	Capital Cost *$K, 2006 Dollars (Source Year)*	O&M Cost *$K/year, 2006 Dollars* (Source Year)	Description
Electronic ID Tag	CV001	10	0.5 - 0.9 *(1995)*	0.01 - 0.017 *(1995)*	Includes ID tag, additional software & processing, and database storage. Software is COTS.
Communication Equipment	CV002	10	1.1 - 2.1 *(1995)*	0.007 - 0.011 *(1995)*	Commercial vehicle communication interface and communication device (cell-based radio).
Central Processor and Storage	CV003	10	0.2 - 0.4 *(1995)*	0.005 - 0.01 *(1995)*	Equipment on board for the processing and storage of cargo material.
GPS/DGPS	CV004	10	0.5 - 1.8 *(2004)*	0.12 - 0.6 *(2004)*	GPS for vehicle location. Capital cost depends on features of unit. O&M cost includes annual service fees.
Driver and Vehicle Safety Sensors, Software	CV005	10	0.9 - 1.7 *(1995)*	0.03 - 0.06 *(1995)*	Additional software and processor for warning indicator and audio system interface, and onboard sensors for engine/vehicle and driver. Software is COTS.
Cargo Monitoring Sensors and Gauges	CV006	10	0.13 - 0.27 *(1995)*	0.013 - 0.027 *(1995)*	Optional on-board sensors for measuring temperature, pressure, and load leveling.

Electronic Cargo Seal - Disposable			8 - 10 *(2003)*	0.15 - 0.20	Cost for a disposable radio frequency identification (RFID) E-seal that provides a complete and accurate audit trail of seal status during transport. Low is for passive, and high is for active E-seal communications.
Electronic Cargo Seal - Reusable			209 - 486 *(2002)*		Cost for a reusable radio frequency identification (RFID) E-seal that provides a complete and accurate audit trail of seal status during transport. Low is for passive, and high is for active E-seal communications. Depending on vendor, some E-seals may incur a monthly service charge.
Autonomous Tracking Unit			0.34 - 0.8 *(2003)*	0.138 - 0.4 *(2003)*	Chassis or container mounted unit that tracks location and condition of assets (cost for on-board sensors not included). Higher priced units provide greater functionality, such as polling of location information and increased quantities of sensor data. Annual service charges include the communications link between unit and data center, and information services.

2.2 Automatic Vehicle Location Systems

2.2.1 Introduction

Automatic vehicle location (AVL) is a computer-based vehicle tracking system consisting of devices which can continuously monitor the location and status of vehicles operating in an urban or rural traffic environment (Cutchin, 2005). It is a collection of electronic or electromechanical devices used to acquire information about the location of vehicles. The location information is then relayed to a central location, where it may be further processed, stored, and used to make command and administrative decisions. Typically, vehicle position is stored on the vehicle for a short time of seconds or minutes duration and then relayed to the control center in raw form or processed on-board the vehicle before it is transmitted.

Transit agencies in the U.S. incorporate AVL with other operational functions, including computer-aided dispatch, mobile data terminals and emergency alarms. Increasingly, they are also using AVL for services that directly benefit transit users. These include among others:

- Real-time passenger information
- Automatic passenger counters
- Automated fare payment systems

Other components that are integrated with AVL systems include

- Automatic stop announcements
- Automated destination signs
- Vehicle component monitoring
- Traffic signal priority

US transit agencies have identified four primary objectives for the introduction of AVL systems:

- Improved schedule adherence and timed transfers
- More accessible passenger information
- Increased availability of data for transit management, and
- Increased efficiency/ productivity of transit services

By implementing AVL systems, firms, both public and private, can increase fleet utilization and reduce input factors (fuel, labor and capital). AVL systems help in revenue planning and efficiency through the use of on-board electronic fare collection. They can also provide seamless transfers by implementing and supporting a common or universal fare medium (e.g. a fare card that is accepted by all operators in a specific region). Finally, they can also help to improve safety on-board vehicles by allowing quick location and response to incidents and emergencies.

2.2.2 Vehicle Location Technologies

AVL systems may be implemented using the following basic technologies:

- Signpost and odometer
- Radio navigation/location
- Dead Reckoning
- Global Positioning System (GPS Satellite Location)

A number of systems make use of a combination of one or more of the basic categories to provide system redundancy.

Signpost Systems (Proximity Technology)

In this system some type of "beacon" (the signpost) are installed at regular intervals along the routes (Pletta, Caskey, and Heermann, 1996). The bus carries a transponder that responds to interrogation by the signpost when it is in a close range. The dispatcher is then notified that a particular bus, with a unique ID number in the system, has passed a known location. When the bus reports its location, the distance from the last pole is used to locate the vehicle's position on a route. The system can also be run in reverse, with the transmitter on the bus and multiple receivers mounted along the bus route. However, if the bus leaves the route, there will be no information about the bus, so most transit bodies prefer to have a receiver on the bus. The basic principle of direct proximity location technology is as show in Figure 2**.**

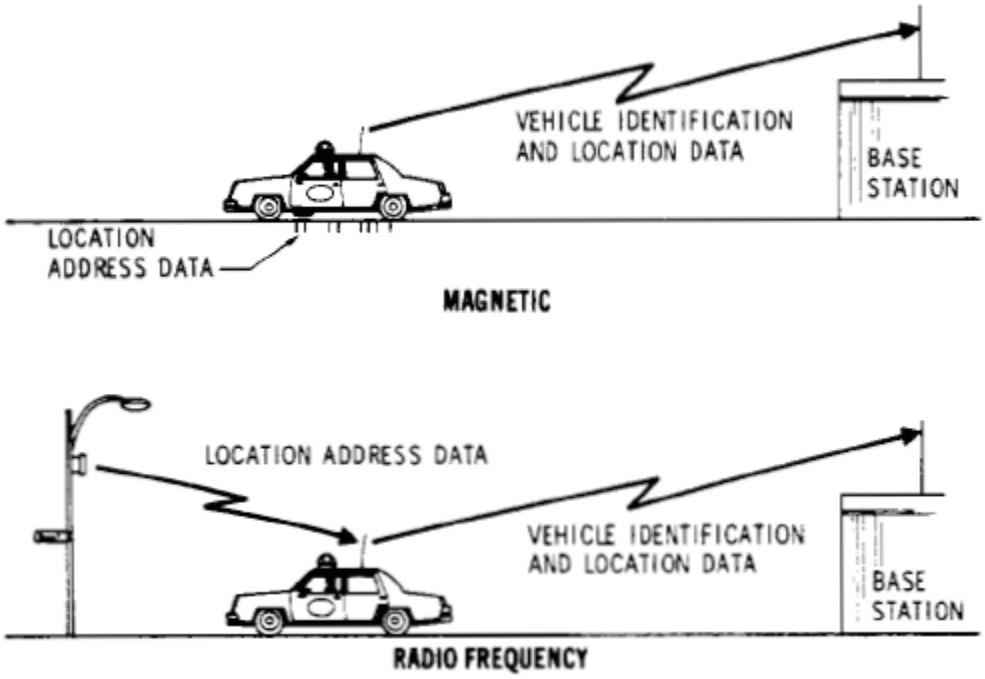

Figure 6: Direct Proximity Concept (Wilson 1978)

Fixed elements (signposts) that are installed at known operational locations provide the vehicle with unique address data in a digital format (Wilson, 1978). The vehicle then relays these data through a conventional mobile radio channel to the control center system where it is translated into a dispatcher-compatible format. The coupling between the fixed elements and the vehicle could be magnetic or electromagnetic. The magnetic coupling system uses magnetic studs imbedded in the road surface. The electromagnetic system requires use of radio frequency devices mounted along the roadway at a particular height.

Advantages of Signpost Systems

- Simple, reliable, low cost vehicle unit implementation

- Minimum base station (control center) computer processing
- Capable of varying system accuracy according to the needs
- Location data is positive and non-ambiguous at all times
- Fail-soft characteristics present (i.e, the failure of any fixed location element does not degrade overall system operation)
- Capability of the system to share fixed location elements data to be shared by both public safety and private users without compromising the law enforcement data.

Disadvantages of Signpost Systems

- The number of fixed elements is inversely proportional to the square of accuracy requirements and directly proportional to the size of the operational area. Hence, large metropolitan area coverage would require the investment in a large number of location elements, with higher initial cost and their inherent continuing maintenance issues and problems.
- Installation of fixed elements could be limited due to local codes or regulations pertaining to the use of the utility or street lighting infrastructure. Also, suitable above-the-ground sites may not be available in some areas. If the proximity device would require power for operation, then high costs would be incurred to provide transformers, watt-hour meters, and labor costs for installation of the system components.
- The fixed location elements would also be susceptible to sabotage or vandalism.

Radio Navigation/ Location Systems

Radiolocation methods locate a vehicle by direct measurements on radio signals traveling between the vehicle and a fixed number of stations. The concept is to determine the relative distance differential of the vehicle from two or more fixed pairs of sites and convert the resulting hyperbolic lines of position to a geographic reference location (Wilson, 1978). The two basic technologies used are: those employing a vehicle transponder (pulse trilateration) and those that employ signals from fixed transmitting stations.

Pulse trilateration technology operation is shown in Figure 3. It makes use of standard navigation networks like the Loran C network.

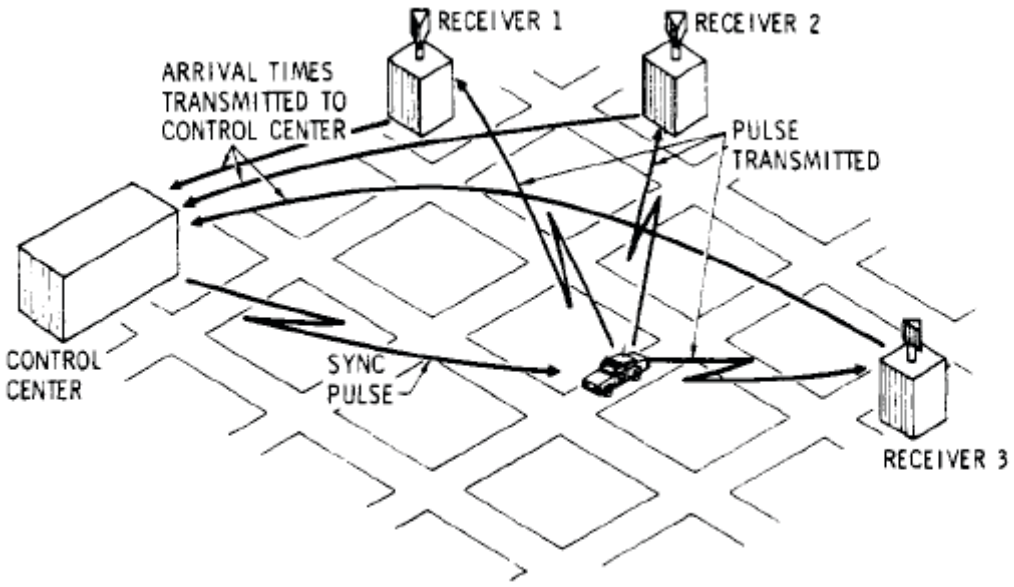

Figure 7: Pulse Trilateration Concept (Wilson 1978)

In this method, the distance between the vehicle and at least three stations is measured by determining the radio frequency travel time from the stations to the vehicle and back to the stations. For this purpose, two-way radio frequency channels are provided or the vehicle is equipped with a transponder which would rebroadcast the signal at some fixed time increment after receiving the incoming signal. This information is then relayed to a central location where the location determination is made.

The concept of using radio signals to determine vehicle location is as shown in Figure 4.

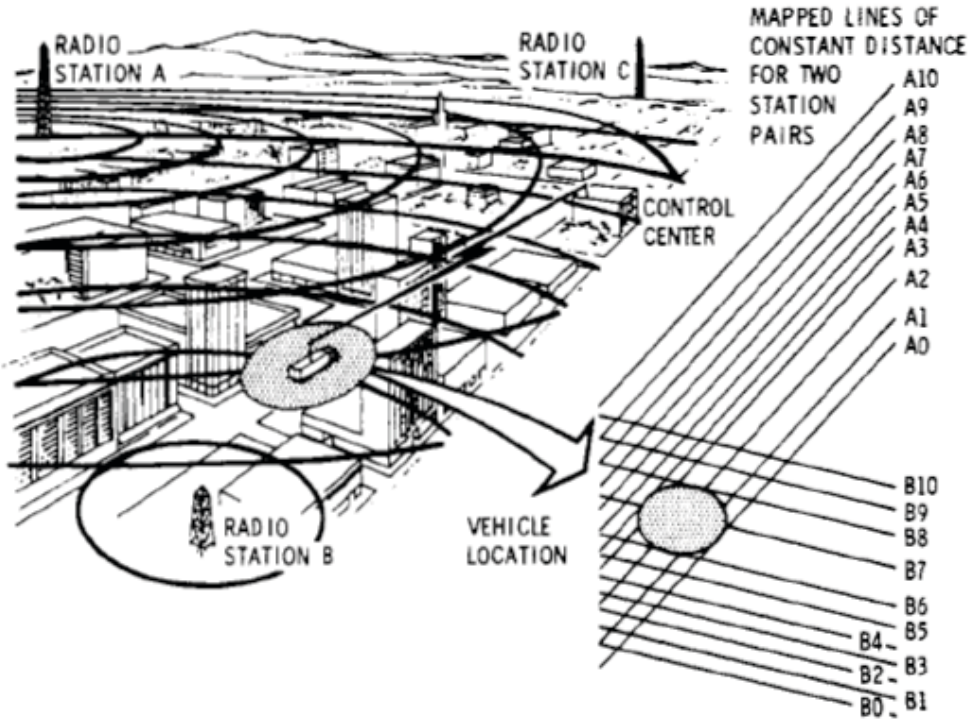

Figure 8: Fixed Transmitting Stations Technology (Wilson 1978)

A receiver is mounted on the vehicle which measures the distance differential between station pairs and transmits these data through a land-mobile communication system to the control center for determining the appropriate geographic location.

Advantages of Radiolocation Systems

- Wide area coverage is suitable for both surface vehicle and airborne operations.
- Capable of providing absolute location.
- Pulse trilateration technology is independent of conventional land-mobile communication links.

Disadvantages of Radiolocation Systems

- Primary errors are related to fading or distortion of radio transmissions.
- Pulse trilateration receivers must be appropriately placed to avoid multipath delay errors.
- In the presence of tall buildings there could be signal shadowing, which may require the use of some direct proximity elements to reestablish location position.

Dead Reckoning Systems

Dead Reckoning methods locate a vehicle by computing its distance and direction of travel from a known fixed point, as seen in Figure 5. A radio link is used to relay this information back

to the base station. The distances are measured using a precision odometer and a compass device to measure the azimuth (Wilson, 1978).

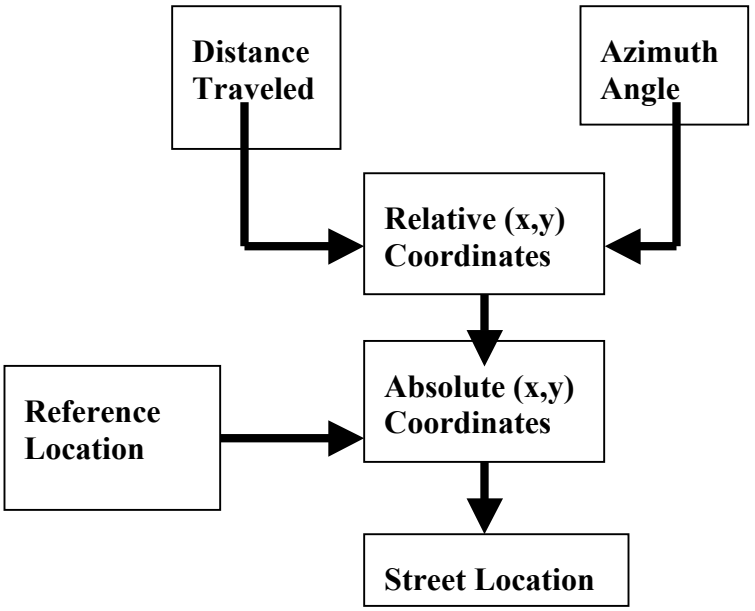

Figure 5: Dead Reckoning Concept (Adapted from Wilson, 1978)

Advantages of Dead Reckoning

- The basic sensors used are rugged and reliable
- The primary location function is contained within the vehicle.

Disadvantages of Dead Reckoning

- Data errors from sensor readings are affected by various factors like any magnetic field in the vehicle, external magnetic variations and deviations, variations in tire pressures, wheel slip, and uneven road conditions. These errors result in a cumulative location error, and hence they require the use of reestablishment reference locations within the control system. Direct proximity devices can be used for a position reset mechanism or the uncorrected sensor data could be transmitted to the control center to compare with city map data to make error corrections. This would necessitate frequent updating of sensor data, the use of voice communications to reestablish vehicle position in case of errors exceeding certain bounds, and an additional land-mobile communications channel for the data transmission function in the vehicle is needed.

Global Positioning Systems

The Global Positioning System (GPS) was developed by the United States Department of Defense (DOD), primarily for military purposes (Moore, 1994). The fundamental concept of GPS is the pseudo-range, which is derived from the measured time of flight of a particular signal

from the satellite to the receiver. With simultaneous measurements from at least four satellites, a GPS receiver is able to instantaneously compute its 3-dimensional position. Although the clocks on board the satellites are in synchronization with the GPS time, the user's receiver clock has an inevitable offset from the GPS time. Since a fourth unknown is needed to model this offset, the use of at least four (4) satellites at a time for computation purposes is required (rather than the three (3) sites needed for triangulation systems). The pseudo-range is measured using either one of the timing codes, which form the basis of the satellite transmission. The civilian 'Standard Positioning Service' (SPS) is based on the Coarse/Acquisition (C/A) code. These signals provide positioning capability with an accuracy of 95% of 100 m horizontally and 150 m vertically. The Precise Positioning Service (PPS) is based on the Precise (P) code. It provides positioning to around 20 m.

Figure 6 illustrates the fundamental operation of a GPS device. It implements a time-difference-of-arrival concept using precise satellite position and on-board atomic clocks that generate navigation messages that are continuously broadcast from each of the GPS satellites. Each GPS satellite employs an on-board computer and navigation message generator to know its own orbital location and system time very precisely (McNeff, 2002). A global network of monitoring stations keep careful track of these parameters. Corrections are uploaded to each satellite, on a daily basis including orbit position projections for each satellite in the constellation, as well as corrections to on-board satellite clocks.

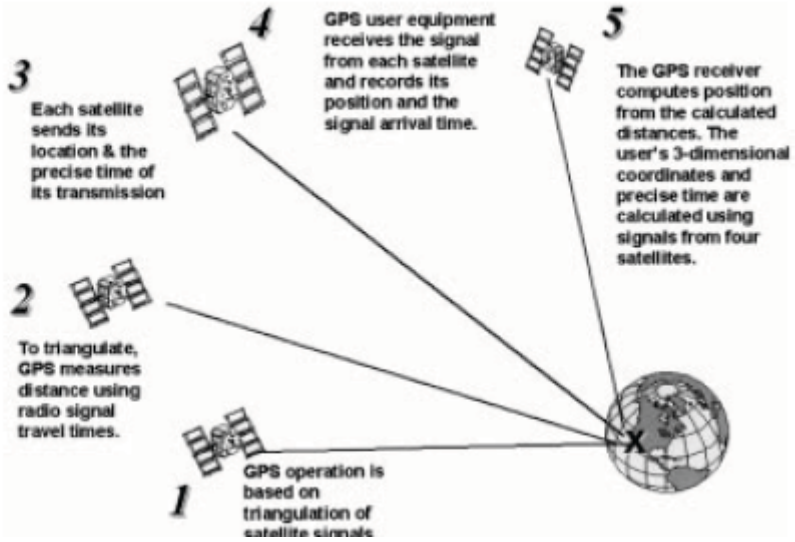

Figure 6: How GPS Works (McNeff, 2002)

System time is maintained on board each satellite by Cesium and Rubidium atomic frequency standards. These on-board clocks are accurate within a few nanoseconds of global coordinated time (UTC) which is maintained by the Master Clock at the U.S. Naval Observatory (USNO) and are individually stable to a few parts in 10^{-13} or better. The four satellites for

simultaneous multi satellite global coverage require a constellation design comprising 24 satellites at semi synchronous altitude (about 11000 nautical miles) in six orbital planes, each of which is inclined at 55° (see Figure 7)

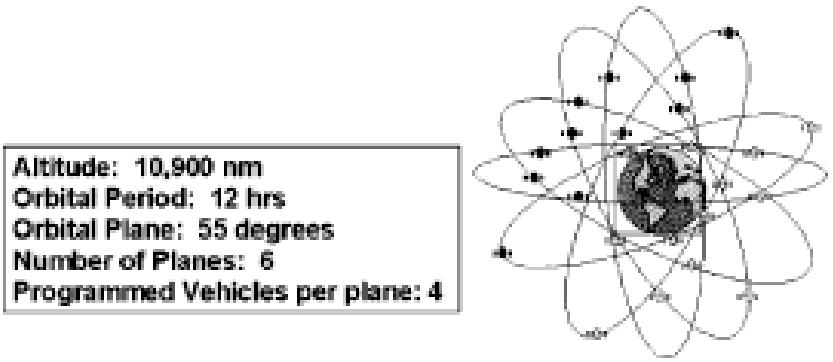

Figure 7: GPS Operational Constellation (McNeff 2002)

The real-time accuracy of pseudo-range positioning is greatly improved by the use of a differential mode of operation using a second, reference, receiver at a known point (Moore, 1994). Since the potential accuracy is limited to a few meters, to achieve higher accuracy it is necessary to change the basic method of measurement. By removing the timing codes from the satellite codes, it is possible to measure the phase of the carrier wave on which the codes are transmitted. With a wavelength of the order of 20 cm, a phase measurement to within a few degrees could lead to a potential range resolution of the order of millimeters. But, this does not lead directly to a range measurement of the same accuracy. The errors due to the clocks, atmospheric disturbances and the satellite orbits sometimes exceed this resolution by several orders of magnitude. Hence, by adopting a differential relative positioning method the problems can be overcome and extremely high position accuracy can be achieved.

Advantages of GPS

- Based on the type of service, corrected or uncorrected signals provide sufficient accuracy for most transit purposes.
- Along with good accuracy, GPS is advantageous as it is available everywhere in the service area; hence there is no distinction between vehicles on-route or off-route.
- Speed and direction can also be determined using GPS, which potentially eliminates the need for additional sensors.

Disadvantages of GPS

- There exists a built in disadvantage to a GPS based system since they require the establishment of line-of-sight references to a minimum of three satellites for 2D position location or at least four satellites for 3D position. Hence, bridges, tunnels, or tall buildings that block the line of sight to the satellite constallation signals can result in a report with no position or one with an inappropriate position.

- Also, signals reflected across buildings can cause multipath reception, and errors equaling from tens to hundreds of feet in position. This typically happens in a "downtown" urban area setting with many tall buildings.
- GPS signals are also attenuated by foliage. Dense overhanging trees along roads and streets present problems to GPS receivers on buses. In a "downtown" urban environment, a backup navigation system in addition to the GPS systems is used, typically dead reckoning systems.

2.3 AVL System Costs

The actual cost of any vehicle location system is dependent on the details of the specific application being considered (Wilson, 1978). To determine these costs and make a comparison:

- Stipulate the system accuracy requirements.
- Determine the number of vehicles that would be pressed into service
- Consider size of the operational area being considered.

With the help of this information, the annual cost per vehicle can be determined for each technology. Factors that could be considered include:

- Vehicle unit hardware, installation, maintenance, and depreciation costs
- Support Unit (e.g. location elements, base station), hardware, software, installation sites, installation, maintenance, licensing and permit fees, land line leases, training, and depreciation.
- System Integration and testing.

For a chosen system performance, system costs can be expressed as a function of the number of vehicles to be equipped and the size of the operational area. Figure's 8, 9, and 10 give the estimated annual life-cycle costs (for a 300-foot accuracy requirement) of the three representative technologies of Signpost Systems (Proximity Systems), Radio Location (Radio Frequency Positioning) system, and Dead Reckoning Systems installed in operational areas of 25, 100, and 400 square miles. This comparative figure illustrates the effect of area and number of vehicles.

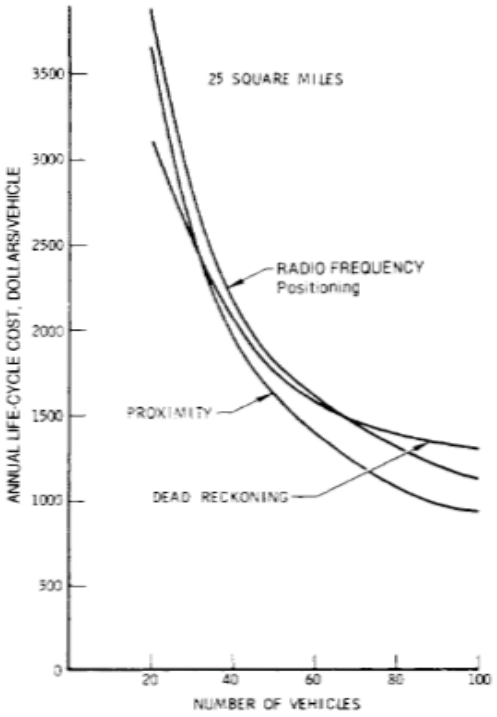

Figure 8: Annual Life Cycle Cost, Dollars/Vehicle Vs No. of Vehicles Over 25 Square Miles (Wilson, 1978)

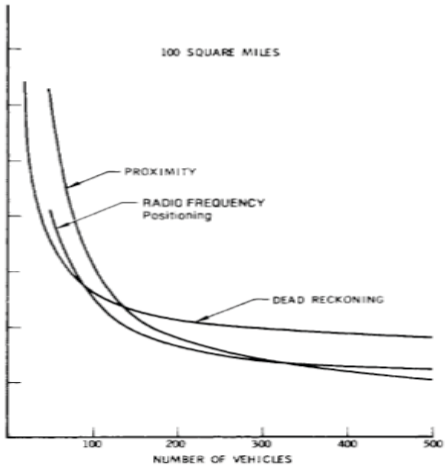

Figure 9: Annual Life Cycle Cost, Dollars/Vehicle Vs No. of Vehicles Over 100 Square Miles (Wilson, 1978)

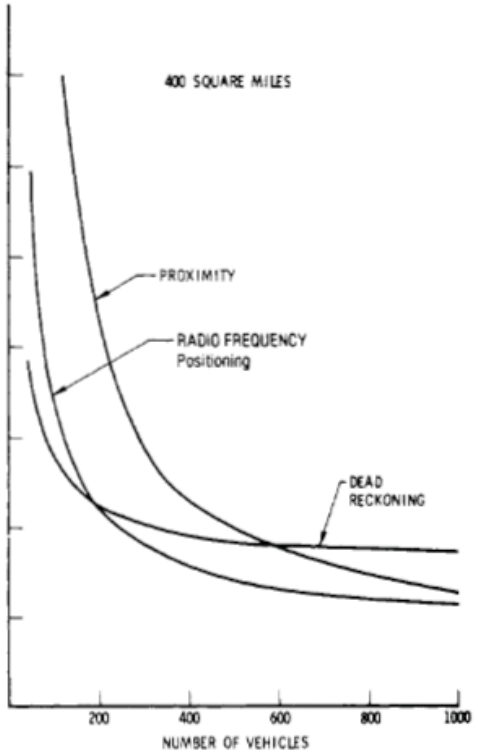

Figure 10: Annual Life Cycle Cost, Dollars/Vehicle Vs No. of Vehicles Over 400 Square Miles (Wilson, 1978)

For a small number of vehicles, the main cost is that of the support elements. As the number of vehicles, increases the proportional cost per vehicle of these fixed elements decreases, and vehicle equipment cost becomes the major factor. Figure 11 depicts the resultant cost effectiveness of each technology as a function of the area under operation and the number of vehicles.

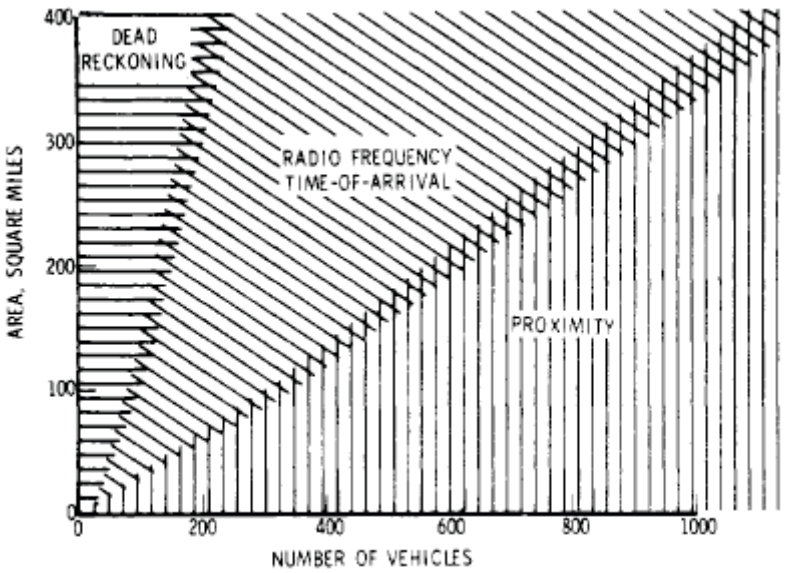

Figure 11: Representative Cost Effectiveness as a Function of Operational Area and Number of Vehicles at 300 feet accuracy (Wilson, 1978)

It is important to note that these cost data are typical, but changes in system performance requirements will change the slopes and relationships of the cost effectiveness curves and the complex nature of the dispatch and communication interfaces significantly impact the annual system operating costs.

2.4 Selection Factors

The following factors play an important role in selecting an appropriate AVL system for the planned operation (Wilson, 1978).

- **Planned Application-** The accuracy and cost of an automatic vehicle location system is largely influenced by its planned use. For example in police patrol operations, system accuracy associated with the general dispatch function will be inadequate for supervision or office safety. Also, it might not be economically feasible to upgrade a system selected for the general dispatch function, should the need for higher accuracy be required in the future.

- **Vehicle Types-** The available mounting space, power and weight constraints would determine the specific type of technology solution to be used for any type of vehicle.

- **Number of Vehicles-** The number of vehicles to be served by the system would have a strong influence on system costs and would significantly impact the complexity of the communications network.

- **Communications-** With the exception of pulse trilateration systems, all automatic vehicle location systems require the use of a land-mobile communications channel, with

one system requiring a dedicated channel. The availability of unused digital communications channel capacity within the spectrum of existing or planned communications system could reduce the location system costs.

- **System Interfaces-** The nature of the location system interfaces with the dispatch system would impact the complexity, cost and selection of the AVL system.
- **Maintenance-** The complexity of the system elements would determine the level of maintenance needed within the existing resources of maintenance personnel. Upgrading of technical skills or subcontracting could increase system life-cycle costs.
- **Installation sites-** A major factor in considering the use of a proximity system is the availability of installation sites for proximity units.

2.5 RFID

2.5.1 Introduction

Radio Frequency Identification (RFID) is the next generation wireless communication technology that is applicable to various areas including distribution, circulation, transportation, tracing, and tracking of products (Min et al, 2007). It is a non-contact technology that identified objects which are attached to the tags. RFID readers obtain the information about objects and surroundings by communicating with the tag antennas. It has the capability to identify mobile objects at high speeds, and can also identify a certain number of tags simultaneously by an anti-collision mechanism. Thus its use in a transportation system as a replacement for traditional "signpost" merits consideration.

RFID is a component of the more general Automatic Identification systems (also referred as Auto-ID systems) (Fuhrer, et al. 2006). The term Auto-ID groups the technologies that help computers to identify objects, animals, or people. Automatic Identification procedures were developed in order to create means of providing information about objects in transit. The historical and technological development of these technologies starts from Barcodes, Smart Cards, and the very recent RFID technology. Figure 12 shows an overview of existing Auto-ID technologies.

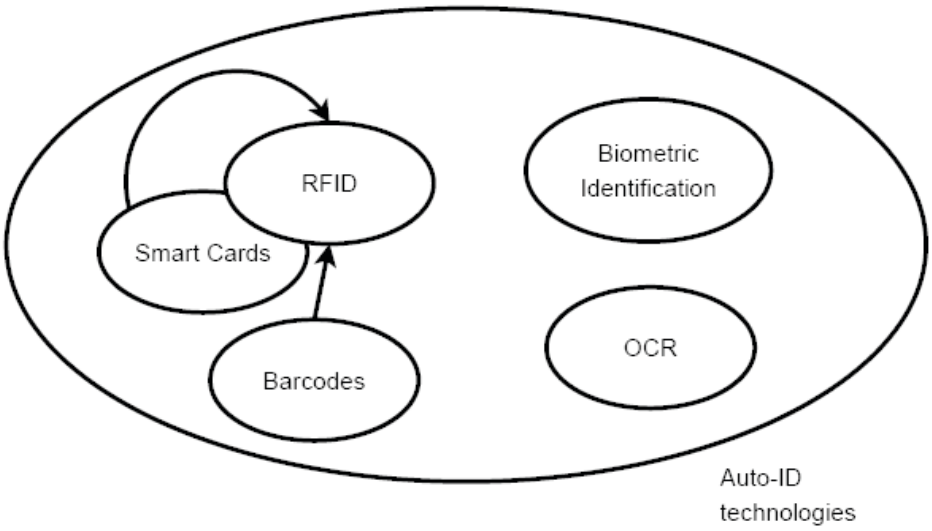

Figure 12: Auto-ID Technologies (Fuhrer, et.al. 2006)

- **Barcodes**- The barcode are comprised of a field of bars and empty spaces, vertically printed on a sticker or a product label. The sequence (bars, gaps) as well as the width of the bars or gaps are converted into an ASCII (American Standard Code for Information Exchange) sequence using optical lasers and a complex set of mirrors. Barcodes standards commonly used are EAN (European Article Number) code, which is an extension of the widely used UPC (Universal Product Code) introduced in the USA in 1974, while many other more sophisticated systems have been devised for higher information density and range of use.

  [On June 26th 1974 in Ohio, USA, the first product using barcodes, a 10-pack of Juicy Fruit chewing gum, was scanned at a check-out counter]

- **Smart Cards**- They were invented in 1974 by Roland Moreno. Smart cards are credit-card sized plastic cards containing a data storage system and a microprocessor. To contact smartcards, the reader initiates a galvanic connection between the reader's metallic contact pins and the card contact surface (usually a gold-plated area of about 1 cm^2). After supplying energy and clock pulses, the reader extracts (or writes) data out of the card.

2.5.2 System Architecture

RFID systems are composed of the three main components (Burdet, 2004);

- **Tags**- RFID Tags, also known as transponders (transmitter/responder), are attached to the objects to count or identify. In its simplest form, a tag receives its power from the interrogation signal, consisting of a microchip to store data and a coiled antenna,.

- **Reader**- RFID Reader, or transceiver (transmitter/receiver) is made up of an RF module and control unit, which sends the interrogation signal to the tags.
- **Data Processing Subsystem –** This is a computational application or database depending on how the data retrieved will be used.

Figure 13 shows a basic RFID system in place.

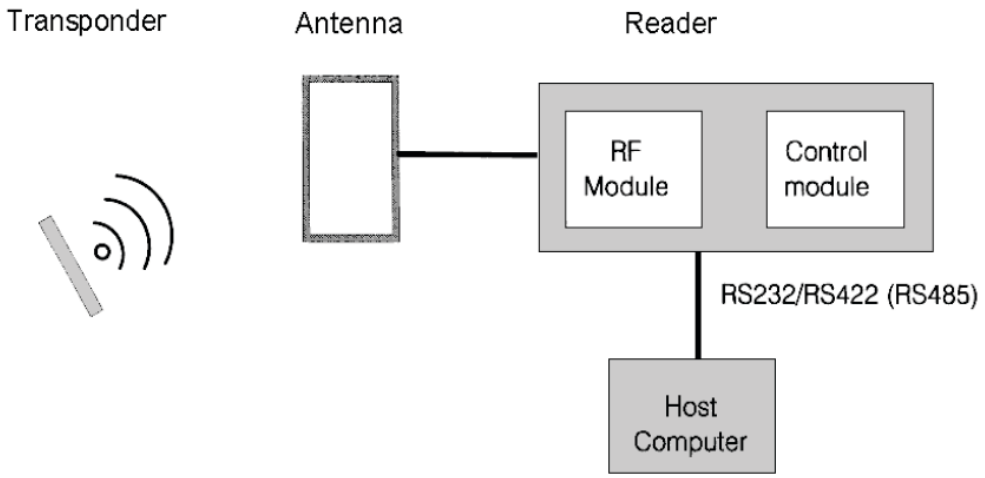

Figure 13: RFID Setup (Hont, 2007)

TAGS / TRANSPONDER

A tag is a label composed of a micro-chip and an antenna. Tags that are made into adhesive labels can be directly attached to parts. Tags in the form of glass cylinders can be implanted under the skin of animals and humans. Passive tags cannot emit information until they receive their power through the electromagnetic waves emitted from the readers. Active tags have their own source of power and are able to send information themselves. Most RFID tags generate power directly from the incident radio waves and also use this carrier wave to "reflect" data back to the reader. This process is known as backscatter [Bitkom, 2005].

The ANSI and ISO standards for tags are defined by five (5) classes (Tully, 2006):

- Class 0 – Factory programmed with a simple ID during manufacturing stage which cannot be updated. They are passive in nature, and can be used for anti-theft applications
- Class 1 – User programmed with a simple ID in the field which cannot be updated. Passive in nature, and can be used for stock control applications.

- Class 2 – Data can be written to the tag and re-written many times. Passive in nature, and usually contain sufficient memory for data logging applications.
- Class 3 – Tags contain on-board sensors for measuring temperature, pressure and motion. Active in nature or battery-assisted as sensors must be monitored when no reader is present. Generally used for sensitive cargo.
- Class 4 – Tags can communicate with each other in the absence of a reader. Active in nature and can support ad-hoc networked applications.

Tags of class 0-3 usually transmit to a reader using inductive coupling (short range) or backscatter (long range) techniques. Class 4 tags contain radio transmitters. Tag information can be encoded to prevent access by unauthorized readers. Encryption can be performed by the reader for Class 0, Class 1, and Class 2 tags. Class 3 and Class 4 tags are able to encrypt locally using public-key cryptography. Denial or jamming is possible using rogue tags deployed within the field of the reader or Class 4 tag. Spread-spectrum radio technology could be used to mitigate jamming.

RFID Readers/ Writers and Antennas

Radio Frequency Identification readers perform the interrogation of RIFD transponders/tags. Readers and transponders are in a master-slave relationship where the reader acts as a master and the transponders act as slaves (Peradovic, and Karmarkar, 2006). A computer application mounted to the transponder reads data from the RFID reader acts as a master unit and sends commands to the reader. In a hierarchical system structure, the application software represents the master while the reader is a slave (Figure 14).

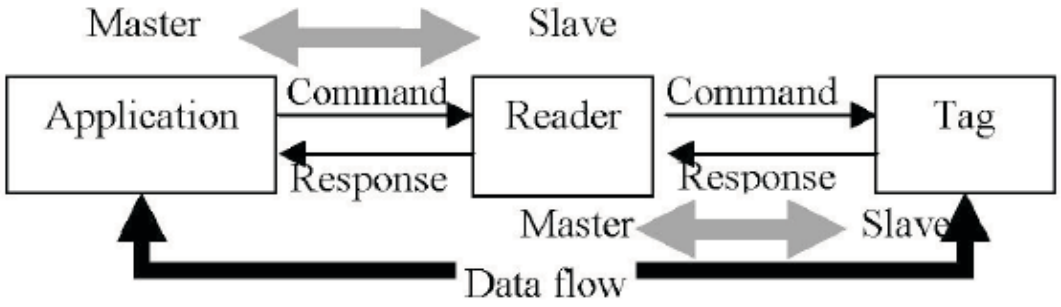

Figure 14: Master-Slave Principle (Peradovic, and Karmarkar, 2006)

RFID readers consist of three main parts that allow them to function in RF and digital systems (Figure 15).

- Control section
- High frequency interface
- Antenna

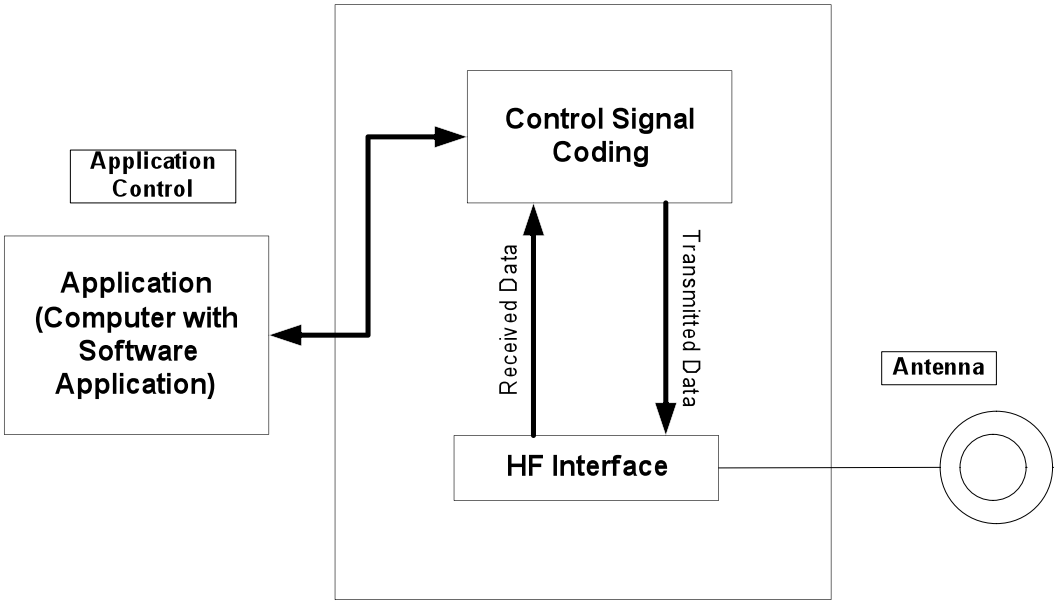

Figure 15: Block Diagram of a RFID Reader (Adapted from Peradovic, and Karmarkar, 2006)

RFID readers can be classified by power supply type, communications interface, mobility, tag interrogation, frequency response, and supporting protocols. The classification of RFID reader based on their power supply is:

- Readers supplied from the power network
- Battery powered (BP) readers

RFID readers classified on the basis of communication interface with peripheral devices and based on readers supplied from the power network are:

- Serial RFID Readers
- Network RFID Readers

RFID readers classified on mobility are:

- Stationary RFID readers
- Handheld RFID readers

RFID readers classified on the transponder frequency responses they listen to are:

- Unique Frequency Response Based
- Non-unique frequency response based

Antennas are connected to electronic control devices; the readers. These generate standing electromagnetic fields through which data is received from or transmitted to RFID tags. Data is transmitted without the need of direct line of sight to the tag. However, unfavorable conditions like metallic environments, or liquids, can cause transmission problems with certain technologies. Tags and readers must have compatible frequencies.

Communication between Reader and Tags

RFID generate and radiate electromagnetic waves. The function of other radio services in the vicinity must not be disrupted or impaired by the operation of the RFID systems (Tully, 2006). It is important that RFID systems do not interfere with nearby radio and television, mobile radio services including police, security services, and industry, marine and aeronautical radio services or mobile telephones. Therefore it is only possible to use frequency ranges that have been reserved specifically for industrial, scientific, or medical applications or for short-range devices. These frequencies are classified worldwide as ISM (Industrial-Scientific-Medical) or SRD frequency ranges. The frequency range along with its standards and specifications is shown in Table 2.

Table 2: Competing RFID Band Allocations (Tully, 2006)

	LF	HF	UHF	Microwave
Frequency Range	<135kHz	13.56MHz	860-930MHz	2.45GHz
Standards	ISO 18000-2	ISO 18000-3	ISO 18000-6	ISO 18000-4
Read range	<0.5m	1m	5m	1m
Advantages	Works well through metals and liquids	Low cost	Long read range of many tags at once	Fast read rates of multiple tags at once
Disadvantages	Large antennae	Short read range of few tags at once	Poor through metals and liquids	Very poor through metals and liquids
Communication	Inductive coupling	Inductive coupling	Backscatter	Backscatter
Power source	Passive	Passive	Active/passive	Active/passive
Applications	Vehicle immobilizers	Access control Payment systems Baggage control	Pallet and box tagging Electronic tolling	Electronic tolling Real time tracking

Electronic Product Code

Electronic Product Code (EPC) is the equivalent of the Universal Product Code used in barcodes and is the internal standard for RFID. The EPC code consists of 64 to 256 bits divided into the four (4) fields as shown in Table 3.

Table 3: The EPC Code used for RFID Applications (Tully, 2006)

Header	EPC Manager	Object Class	Serial Number
01	0000B36	003BA1	00329DE12

The header defines the EPC code type, which in turn defines the lengths of the remaining fields. The EPC Manager identifies the manufacturer of the product the tag is attached to, the Object Class identifies the type of product while the Serial Number uniquely identifies the specific product. The EPC code may be used by Middleware (called Savants) to obtain information on the product from the Internet via a back-end database or update that information based on reader location or sensor data read from the tag.

2.5.3 Selection Criterion

The selection criterion for RFID systems is based on the following factors (Finzenkeller, 2003):

- RFID systems that use frequencies between approximately 100KHz and 30MHz operate using electric inductive coupling. Microwave systems in the frequency range of 2.45-5.8 GHz are coupled using electromagnetic fields. Microwave systems have a greater range than inductive systems, typically 2-15 m. However, microwave systems require an additional backup battery in comparison to inductive systems.
- **Range –** The require range available to an application is dependent upon:
    - the positional accuracy of the transponder
    - the minimum distance between several transponders in practical operation
    - the speed of the transponder in the interrogation zone of the reader
- **Security Requirements –** The likelihood of a potential attack upon an RFID system by a hacker, to get the money or materials, must be evaluated. The applications are further sub-divided into the following groups:
    - Industrial or closed applications
    - Public applications connected with money and material goods

- **Memory Capacity –** The chip size of the data carrier and hence its price is determined by its memory capacity. Permanently encoded read-only data carriers could be used in price-sensitive mass applications with a low local information requirement. If data has to be written back to the transponder, a transponder with EEPROM or RAM memory technology is required. EEPROM are used in inductively coupled systems. SRAM memory devices with a memory backup are used in microwave systems.

2.6 Collision Warning Systems

Collision Warning Systems (CWS) are in-vehicle systems that monitor the road conditions ahead of the vehicle and alert the driver for a potential risk (U.S.Department of Transportation, 2005). Certain radar-based CWS use specialized algorithms to interpret transmitted and received radar signals that determine the distance, azimuth, and relative speed between the vehicle on which the CWS is installed (host vehicle), and the vehicle or object ahead of the host vehicle in its lane. If a vehicle is within a predefined closing time threshold of the host vehicle, the CWS alerts the driver of an object in its lane with which a collision is eminent. The CWS systems may possibility taking automatic action to prevent collision with the object ahead, but with most systems currently in use, drivers still are responsible to take corrective action after getting the alarm.

As the time interval to the vehicle ahead decreases, CWS issues a progressively more urgent warning. The system's field of view forms an isosceles triangle with its apex at the front center of the vehicle. As the object gets closer to the front of the vehicle, a different range or time interval is sensed, and the system then issues a different type of alarm. These warning thresholds are set by the system manufacturers. CWS also warns the drivers of system malfunctions and can be easily integrated with the on-board cruise control systems. Figure 16 illustrates these 'progressive thresholds' control systems.

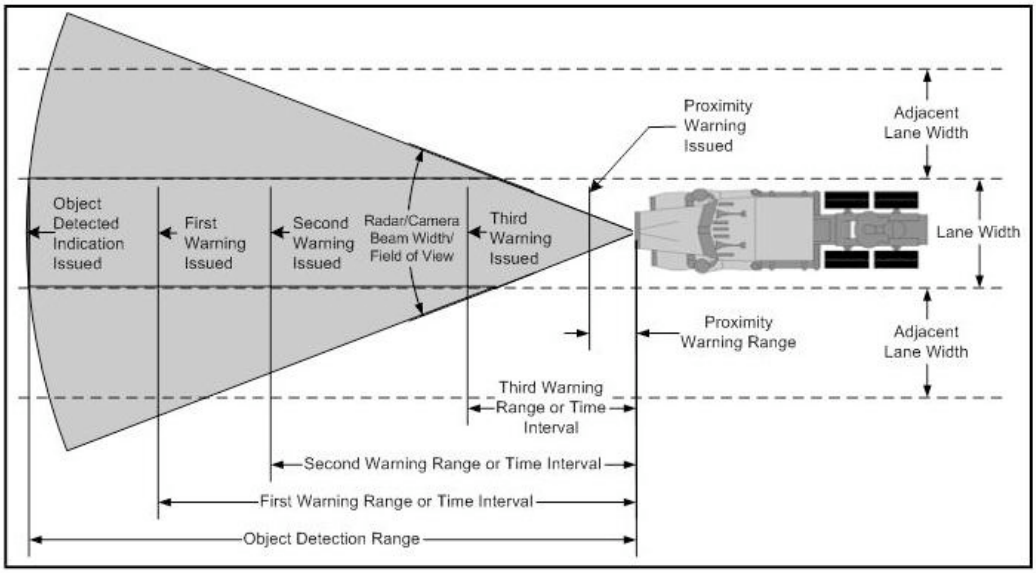

Figure 16: CWS Object Detection Ranges and Collision Warning Thresholds (U.S.Department of Transportation, 2005)

Eaton Vorad EVT-300 Radar System

Eaton Vorad's EVT-300 radar system provides the projection of the location of obstacles relative to the host vehicle (Eaton, 2001). The EVT-300 radar provides range and range rate information to targets, along with azimuth information. Previous generations of vehicle radars did not provide azimuth information, which minimized their utility as a source of information for a conformal driver display. With the azimuth information, the presence of an object along with is location is also available. It then becomes easy to process the location of an obstacle into the Heads Up Display (HUD), given the location relative to the host vehicle.

Figure 17 shows the algorithm dataflow for transforming object location relative to vehicle coordinate frame to global coordinate frame (Gorjestani, et.al. 2003).

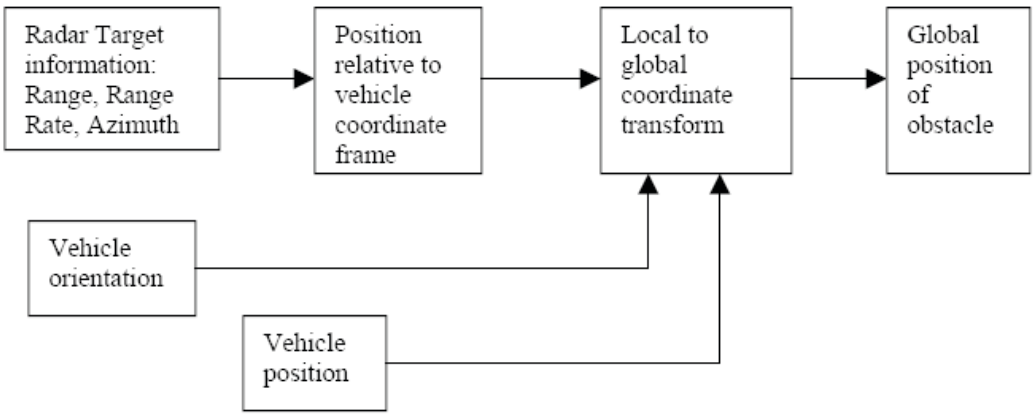

Figure 17: Data Flow Diagram for Transforming Object Location Relative to Vehicle Coordinate Frame to Global Coordinate Frame (Gorjestani, et.al. 2003)

2.7 Case Studies – Technologies

This section provides case studies for the use of various intelligent transportation initiatives that have taken place about public and private organizations. DGPS based gang plowing used by Mn/DOT and other DOT's for snow and ice removal, 3M's Opticom priority control system, and the Southeastern Michigan Snow and Ice Management program are explained in this part of the chapter.

2.7.1 DGPS-Based Gang Plowing

Gang plowing is used by Mn/DOT to increase the productivity of snowplow operations along multilane highways. However, it use is very stressful on the drivers as they struggle keep up with the expected performance level. Factors contributing to this operator stress include the low visibility caused by localized snow clouds created by the lead snowplow and by impatient drivers trying to pass, around and through, the slower moving plows. Tight formations of the snowplows are required to keep the plows together and avoid rogue vehicles coming between, while controlling lateral spacing. When properly formed, the passage of snow from the lead plow to the following plow is very efficient as the gap between the wing on the lead and the front blade on the trailing snowplow is actively controlled at an optimal distance.

To improve the gang plowing process, a Differential Global Positioning Systems (DGPS) based gang plowing system has been developed. This system makes use of the advanced technology to allow a trailing snowplow to automatically follow a lead snowplow at a fixed distance and speed. This form of advanced technology uses DGPS, digital maps (geospatial databases), vehicle-to-vehicle electronic communication, radar, laser, scanner, a driver interface, and steering, brake, and throttle control. The trailing vehicle uses the state of the leading vehicle as a reference from which it follows the lead vehicle at the programmed speed and distance (Alexander, et.al 2005)

Figure 18, below, shows the concept of the gang plowing system. The lead vehicle contains a small computer, a wireless LAN station adapter, and a DGPS receiver. This equipment is installed in a NEMA4 (National Electrical Manufacturers Association) enclosure, and mounted on a roof rack which is installed on the lead vehicle. This minimizes its intrusion into the lead plow cab and provides a weatherproof enclosure. The trailing plow also has on-board, high accuracy geospatial database (digital map). The trailing vehicle can compare its position to the global coordinates of the road and follow the leading plow which has departed the road. The trailing vehicle receives the differential GPS correction from the base station and sends it to the lead vehicle. The lead vehicle sends back its DGPS location.

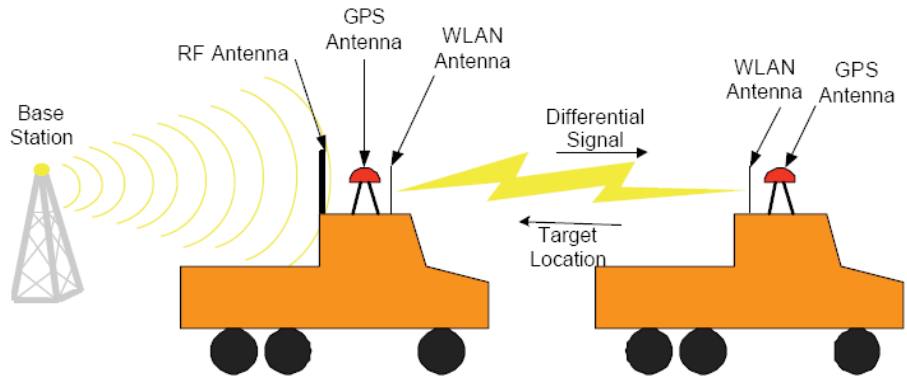

Figure 18: Gang Plow System Diagram (Alexander, et.al. 2005)

Figure 19: Location of DGPS Antenna and NEMA4 Cabinet on the Lead Snowplow (Alexander, et.al. 2005)

Figure 19 shows the location of DGPD antenna and NEMA4 cabinets on the lead snowplow**.**

2.7.2 3M Opticom^{TM} GPS Priority Control System

The 3M Opticom^{TM} GPS Priority Control System is implemented to assist authorized priority vehicles as they navigate through signalized intersection by providing a temporary right-of-way to them. This is accomplished through a vehicle operator interface (to the system) and by using common traffic controller functions. It supports both emergency and transit services, with separate priority levels for signal preemption and transit signal priority, hence eliminating redundancy and a potential conflict at the intersection (3M, 2007). The Opticom GPS system consists of the following components:

- **Vehicle Equipment (Figure 20a)**
    - Radio/GPS unit containing a GPS receiver and a 2.4 GHz transceiver
    - Radio/GPS antenna
    - Control Unit

Figure 20a: Vehicle Components (Control Unit, Radio/GPS Antenna, Radio Unit) (3M Opticom, 2007)

- **Intersection Equipment (Fig 20b)**
    - Radio/GPS unit containing a GPS receiver with antenna and a 2.4 GHz transceiver with antenna
    - Phase Selector
    - Card Rack/Input File

- Auxiliary Interface Panel
- Auxiliary Harness

Figure 20b: Intersection Components (GPS Receiver/Radio, Phase Selector) (3M Opticom, 2007)

The vehicle equipment is mounted on the authorized priority vehicle. The GPS receiver on the vehicle acquires position information from the GPS satellite constellation. This information is used to compute the location, speed, and heading direction of the vehicle. Using this information a priority request and the state of the vehicle's turn signal is broadcast using the 2.4 GHz transceiver.

The transceiver equipment receives the radio transmission from the vehicle equipment. The intersection equipment then compares the information being received from the vehicle to the parameters stored in the intersection equipment's memory. If the vehicle that is heading towards the intersection along a predefined approach corridor and is requesting preemption, and it has met all other programmed parameters, the corresponding phase selector output is activated. This output is then given as a traffic controller preemption input. When activated, the controller cycles to grant the requesting vehicle a right of way by setting a green light in its approach direction, or holding the green light if it is already set. Either condition allows the authorized vehicle to pass through the intersection without stopping.

Power and logic wiring for the phase selector is provided by the card rack/input file which is plugged directly into a slot in the traffic control unit. The auxiliary interface panel provides additional connections for monitoring the green phases and also provides additional priority control outputs. The auxiliary harness is used to provide additional connection for monitoring green phases when the auxiliary interface panel is not required..

Features

- Saves time and money in installation
- Integrates easily into current cabinets
- Uses advanced software to aid implementation and management
- Minimized disruption of traffic flow
- Creates smooth, open path to the future through upgradeable firmware
- Improves safety by eliminating priority conflict at the intersection
- Integrates easily with other on-board devices
- Provides precise activation
- Enhances transit information through expanded coding capability and call history logging
- Enables automated operation by interfacing with AVL systems for conditional priority
- Creates a smoother ride and increases fuel efficiency by reducing stop and go braking and accelerating

2.7.3 Southeastern Michigan Snow and Ice Management- Michigan

The Southeastern Michigan Snow and Ice Management (SEMSIM) project introduced technologies which have resulted in safer roads across the region and more efficient use of resources for the partner agencies, like Road Commission for Oakland County (RCOC), Wayne County Department of Public Services, the City of Detroit, and the Road Commission of Macomb County. The SEMSIM technologies include:

- **Satellite-based GPS vehicle tracking devices** – Each vehicle is equipped with a tracking device that allows satellites to provide real-time vehicle location, direction, and speed. There data is then fed continuously to the snow manager's computer, allowing the management to see where the trucks are at all times.
- **Air and Pavement Temperature Sensors** – Several sensors mounted on each truck provide air and pavement temperatures to help determine if salting is needed. This sensor which uses an infrared beam to check the pavement temperature is mounted on the driver's side rear-view mirror.
- **Plows sensors –** Sensors are installed that provide information on whether the front and underbelly plows are down.
- **Computerized salt spreaders** – These spreaders efficiently regulate the use of salt which negate unnecessary salt applied on roads. The salt data is send back to the snow managers for data review and helps regulate the spread rate based on the vehicle speed.
- **In-vehicle dashboard computer displays –** Trucks are equipped with a dashboard-mounted computer display, known as in-vehicle unit (IVU) that provides two-way

messaging for the driver to receive route instructions, assignments, and provide information about weather conditions, vehicle problems, need for refilling salt. These messages can be sent using push buttons.

- **900 MHz radio system –** Suburban Mobility Authority for Regional Transportation (SMART) is sharing its advanced radio-system with the SEMSIM partners for no additional cost. This state-of-the-art system links the IVU en each vehicle with the computer base stations at the garages. SEMSIM in-vehicle computers continuously send data to the base stations about weather conditions, truck activities, etc. SMART benefits from this by letting its dispatchers know about road conditions during winter storms, helping them to make better routing and scheduling decisions.

- **Customized computer software –** SEMSIM's map-based computer interface provides snow managers with real-time data about each truck. It also helps the playback and review of winter storms to help identify potential improvements. Storm statistics like drive hours, truck miles, and amount of salt used are summarized in tabular reports.

Table 4: Summary of Technological Advantages and Disadvantages of Various AVL systems

Technology	Advantages	Disadvantages
Signpost Systems	Fail-Soft Characteristic (Failure of one fixed location element does not degrade overall system operation)	Fixed Elements are susceptible to vandalism
Radio Location Systems	Pulse Trilateration technology is independent of mobile communication links	Signal shadowing due to urban environment, and needs signal boosting by placing receivers at appropriate locations to avoid multipath delay
Dead Reckoning Systems	Primary Location function is contained in the vehicle and not equipment on external technology like satellites.	Sensor readings from wheel can be distorted by external magnetic fields or road conditions.
GPS	Speed and Direction can also be determined, hence no additional sensors are needed	Signal attenuation in urban environment due to multipathing and in rural environment due to foliage
RFID	RFID allows the improvement of data quality, items management, asset visibility, and maintenance of materiel	Initial investment cost is high for transportation applications.

Table 4 provides a summary of technological advantages and disadvantages of various AVL systems.

2.8 Summary

This chapter provided a review of the literature for the Intelligent Transportation System setup, as outlined by U.S.Department of Transportation. Automatic Vehicle Location technologies like Global Positioning System (GPS) were introduced for the benefits and disadvantages that they provide. RFID as a technology helps in asset tracking and management. Organizations like Mn/DOT, SEMSIM are making use of the AVL technology to improve fleet operations for snow and ice removal. 3M Opticom priority system is capable of providing emergency response vehicles a safe passage for increasing their efficiency. These technologies were explored as part of the literature review effort, to drive towards defining a workable solution for the fleet deployable solutions to the dump box/obstacle collision avoidance problem. While each of the systems explored have their advantages for solving the proposed problem, our team's major driving force, low cost, worked against nearly all of them. To this end, then, the team focused on an initial design that would combine a low cost on-board GPS solution, mated with a backup system that would use passive RFID tags mounted at the approach to hard Mn/DOT assets (bridges, tunnels, pedestrian walkways, etc.) and readers mounted on plow trucks. But, even this simplified system, proved too costly as a reader capable of detecting and reading the passive RFID tags would cost about $2500 per snowplow truck, and a significant investment in infrastructure would be needed to build the network of RFID tags into the roadways. In the final analysis, then the team chose to develop a system that would use a low cost GPS detector and a map coordinates based obstacle map integrated using an on board hardened computer controller. In the next chapter the team will explore the development of obstacle maps initially by extraction of data from various Mn/DOT databases using a Visual Studio 2005 program and then using an controller application that was operated by the on-board controller designed for use during actual plow operations.

Chapter 3
Building Obstacle Maps for Use with the On-board Collision Avoidance Controller

3.1 Introduction

When the team began this project, it was assumed that the best source of obstacles would be the Br-Info database that is maintained by Mn/DOT. This database included relevant information about all of the bridges inventoried in Minnesota, or in our case, the Mn/DOT District 1 region. It listed bridge clearance heights, widths, and most importantly, physical location in longitude and latitude listed as degree-minute-second (dms) formats. Using this list, which the team copied into MS Access, a Visual Studio 2005 (VBasic.net) program was developed to extract the bridge locations which were written into an obstacle map that could be easily transferred to the onboard controller.

3.2 Building the Extraction Software

When the team obtained the District 1 Br-Info database, they also obtained two other data sets from officials in Duluth. These were the snowplow route database ("Plowroute" database) which indicated starting and ending mileposts along the highways that Mn/DOT needed to plow in the district and a database ("mileposts" database) that listed longitude and latitude of (nearly) all the mileposts along the state maintained highways in the district. When the locational information was carefully checked, the team determined that while the Br-Info database bridges were recorded in 'dms' geometry for all of the districts mileposts were stored in a decimal degree format. This required that the extractor program be able to convert between these two designations as it selected bridges that would belong to each route within the district.

The program began by building a user interface that allows the database manager to observe the process of bridge extract to all of the routes within a given Mn/DOT District. The user begins the program by entering the total number of plow routes within the district. The VBasic program then opens the plow rote database and extracts each highway on a given route and the starting and ending mileposts along all of the routes. In the case of the plow route databases provided to the team by District 1 personnel, the routes (of which 103 were listed) in this database were, surprisingly, all single roadway maps. While the team suspected that these were not in fact the most recent plow route listing, it simplified the extraction problem and therefore, they were used. After the highway numbers and distances (in the form of Starting and Ending mileposts) belonging to each plow route were obtained by extracting information from the "plowroute" database the program entered the "mileposts" database to extract the specific geometry (longitude and latitude) for all to the mileposts along the designated highway for any given plow route. Then, the Br-Info database would be searched for any bridges with a clearance greater than zero – indicating that the bridge was in fact an overpass – and who's geometry was between each successive pair of mileposts. Because a one mile grid is relatively coarse in urban areas and could inadvertently extract bridges that are not along the highway under consideration (ones who may belong to other plow routes), before entering the Br-Info search each milepost distance pair was divided by 10 to narrow the search grid to the approximate width of an

intersection and then converted to the dms format. While this computation overheard increased search time, this time was not a problem since the obstacle files were created offline long before they needed to be loaded to the on-board controller.

After the appropriate bridge geometries were extracted for each plow route to a datatable within the application, the geometry was written to a text file given a name included the specific route number from the "plowroute" database. Because the standard used by most GPS system commercially available generated geometric position in yet a third format for longitude and latitude (degree and decimal minutes) a final conversion on the obstacle geometries was performed before the targets were written to the text file for the plow route. As each of the obstacles in the data table were written into the text file, the letter "B," designating bridge, was appended except for the last obstacle which had the letter "X" appended indicating the last obstacle or the plow is about to exit the route. A full listing of the Visual Studio 2005 (VBasic.net) application is presented in Appendix A.

3.2.1 Problems with the Data Extraction Application

When this application was applied to the three databases supplied by District 1 personnel, several problems arose, these fell into categories: missing individual milepost values or missing highway designations in the "mileposts" databases. While troubling, since tests were to be conducted on a route that were potentially fully covered the team felt that testing could be completed. Once initial testing was conducted, along a route called 101 in our plow route mapping that ran along I35 through downtown Duluth from West 40th Avenue to East 26th Avenue, the onboard controller was unable to locate many obstacles. As the team further checked into the problem several items were discovered. First, the "plowroute" database we were provided was not, in fact, up to date and only the several I35 routes and selected other routes along major trunk highways were single highway routes, most routes included parts of several highways, as the team had suspected. Again this was not fatal for testing at the early stages since the plow route the team was assigned for testing was in fact the same one the team had numbered 101 (even though the number in District 1 records had changed). A further problem was discovered, it seemed that all the bridges (and tunnels) constructed when I35 was extended beyond W. 5th Avenue in Duluth were not included. This problem was traced to the Br-Info database which was an older version rather than the most recent one, a problem related to homeland security issues. Finally, a fatal problem with the original approach to the project was identified: not all potential collision obstacles are bridges! Other permanent obstacles that could have collisions with raised sand boxes like pedestrian crosswalks, highway road signs, overhead power lines, etc. and temporary obstacles like construction equipment and signs, and even tress or sagging power, telephone or cable television lines that would never be found in the a regular database of "hard" Mn/DOT assets (like the Br-Info database) should also be available to the onboard controller. Because of the myriad of problems, the team realized that a more direct means for creating obstacle files would be needed. This solution is termed the "Trainer" or TIBIAS (Trainer for Impending Box Impact Alert System), and the need to develop such an application delayed the completion of this project significantly. Steps taken to build the Trainer will be described below.

3.3 Development of the Trainer Application

The trainer system was designed to cope with the (potentially) frequent changes in the route files stored on the controller. The trainer system should be able to make changes to existing route data on the controller and create new routes and add them to the route database on the controller as needs change.

For the Trainer program the team used a nearly identical code as that of the IBIAS (Impending Box Impact Alert System) system. The difference was that the "Lower Box", "Raise Box" or "Moving" function were not enabled. The Input function was modified to display the current GPS position on the computer screen whenever a certain key on the controller is pressed. To use the trainer program, the controller has to be connected to a personal computer through the same cable used to program the controller. Once the computer is connected to the controller, the system can be mounted on any vehicle, thus it can be used in any service vehicle be it snowplow, service truck, or car.

To modify an existing route, i.e. adding obstacles, the user first has to input the route number to be modified and the computer screen displays the route number on it. Now whenever an obstacle has to be added the vehicle is driven under it and the operator must press any key on the controller. Now the (connected PC) computer screen displays the just entered GPS location with route number. Once completed, the operator just copies the GPS location onto a text file; adds a prefix "B" to the GPS location, this prefix B is used to keep track of the number of obstacles on a route, and appends it to the routes text file. Multiple readings can be taken and added to a route file at the same time. Once all the GPS locations have been modified by adding the prefix, all these readings can be appended to the existing route file. All these values have to be added before the last obstacle in the file (the obstacle identified with the prefix X as noted above). Now the operator must compile the program from the PC back to the OP7200 controller with this modified route file, which loads the revised route file to the controller's memory.

To create an entirely new route, the operator must first make a new text file with the route number as its name. Then compile the program (from PC to OP7200 controller) so that this route file is stored in the controller's memory. Now select the new route number of the route file to be generated. The operator then must again move the vehicle under each of the obstacles that need to be added to the obstacle list, press any key on the controller and hold it down till the current GPS location is displayed on the computer screen. Once done with taking the locations for all the obstacles for the new route add a prefix B these GPS locations as mentioned in the previous paragraph and follow the same steps as above. Since this is a new route, the last obstacle reading should be marked with the prefix X, signifying the end of the file.

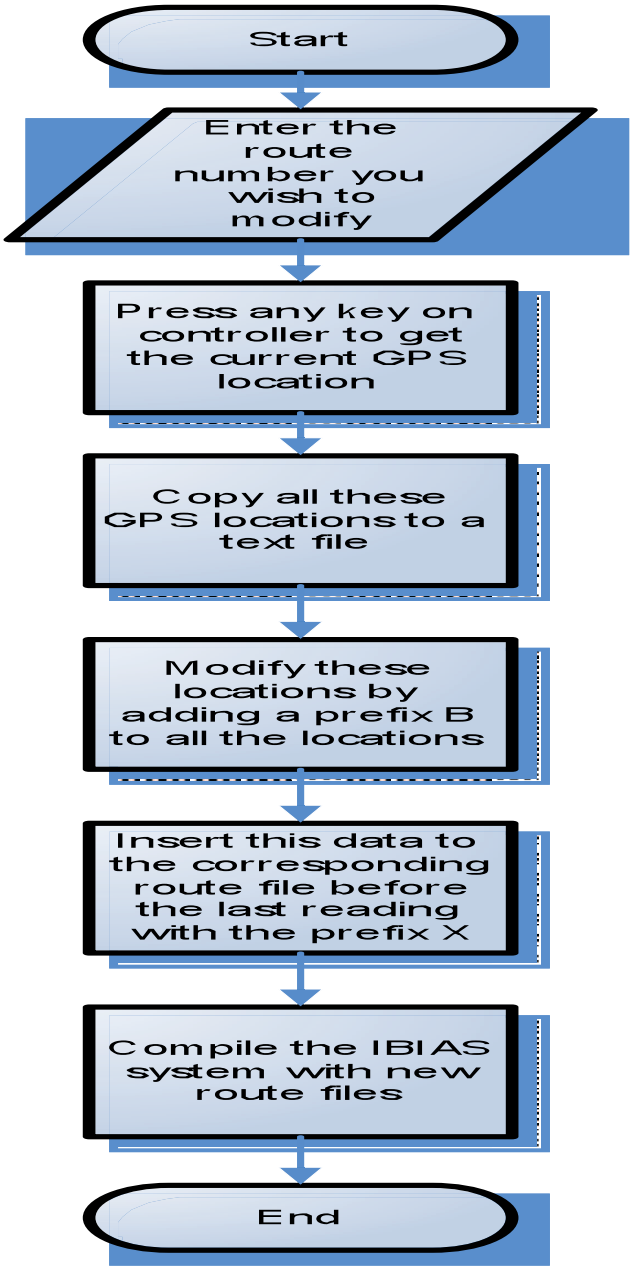

Figure 21: Flowchart for Trainer System

After copying all of the obstacles (with their appropriate prefix) into the local route text file, the operator must compile the program, from PC to OP7200, with this new route file, this would load the new route file to the controller's memory and make it ready to be used to detect obstacles. Figure 21 explains this process with the help of a flowchart. Once the team had good obstacle files it was able to perform testing on actual snowplow routes. In the next chapter a

detailed discussion of the development, design and programming of the on-board system will be presented.

Chapter 4
Development of the IBIAS

Because of limitation in cost and space within the cab of a plow truck, a prototype onboard controller was constructed and tested by the team, it can be seen in Figure 22, below.

Figure 22: IBIAS System Mounted into Mn/DOT Snowplow Being set for Data Entry (right), the Controller at left is the Box/Plow/Chemical Controller used by the Driver

The first task was to assemble the various pieces of hardware together to develop the solution. The solution developed was a fully functional, stand alone unit containing hardware capable of accepting User, GPS and Magnetic switch inputs to alert the plow operator to the presence of dangerous obstacles if and when required.

This controller required the creation of a design to integrate the various hardware products with each other and a software solution to use this hardware as required. This chapter discusses the hardware and software design in detail.

4.1 The Onboard Controller

The research team chose the Rabbit Semiconductor OP7200 as the human-machine interface (HMI) because it was easy to program (it was supplied with “Dynamic C” programming language) included a touch screen for operator communication with the controller, runs on a wide range of dc power meaning it will operate using plow truck power and includes a full

compliment of digital input and output ports that are directly recognized by the controller. A Garmin 15 H was selected as the GPS, again for its low cost, its ability to accept plow truck dc power input, its ease of programming. The final deciding factor was that this GPS unit has the ability to continue dead reckoning its position for several seconds (up to 30 seconds) even if it loses it connection to the satellite operational constellation. The teams' final hardware choice was the GE Security 166 magnetic switch. This device was selected as proximity sensor to identify the position of dump box because of its hardened construction and the fact that it designed for heavy equipment application. A photograph of the magnetic switch mounting is visible in Figure 23 below.

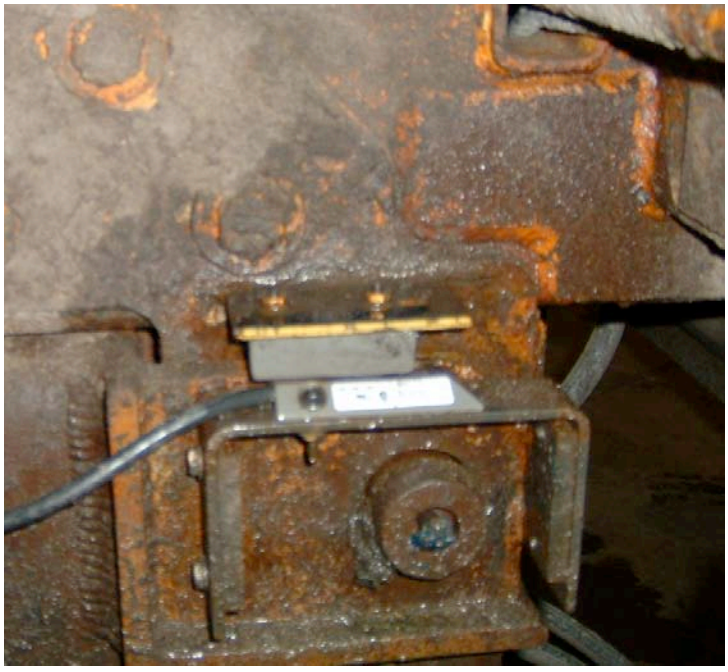

Figure 23: Switch Components Mounted on an Early Model Adjustable Bracket. The Lower Unit is Connected to the Controller (and Truck Frame) while the Upper Unit is a Magnet that Positions the Switch (opened/closed) Depending on Dump Box Position

The following section explains the process of interfacing various pieces of hardware to develop the proposed solution.

The Rabbit semiconductor OP7200 was selected to be the control center for the solution as it has controller and programmable memory. A method for connecting all the supporting hardware to OP7200 had to be developed. As mentioned earlier, OP7200 has 19 digital input ports and eight digital output ports. This section describes how these ports were used to interface each hardware component to form the proposed solution.

Figure 24 shows the wiring diagram of how the team connected the selected hardware to for the IBIAS solution.

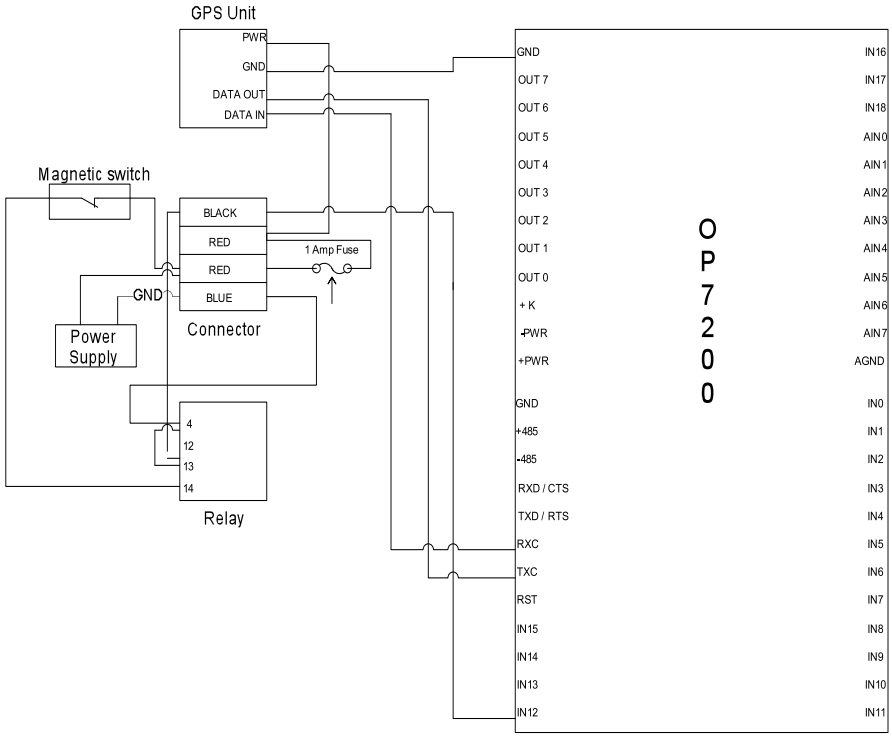

Figure 24: Wiring diagram for the proposed solution.

The hardware selected by the team was identically feasible with power requirements, so the team did not require a separate piece of hardware for power regulation. The OP7200 could use the 12 VDC power supply on the truck; however a one (1) Amp fuse for the GPS unit was added as a measure of precaution for the GPS circuits. The team used a 12 VDC relay and several quick connectors other than GPS, HMI and Magnetic switches as shown as "colored boxes" in Figure 24. The quick coupling connectors were used to wire various pieces of hardware together easily. This made initial connections simple and provided flexibility of changing connections if required. When individual costs were summed, the total cost of the proposed solution was approximately $575 while was slightly above our target figure, the team believes that this is an acceptable overage because the hardware chosen is road hardened and should be usable for several years on a plow truck.

4.2 Software Solution

When the team began to develop their solution, the team envisioned a controller that could be operated in a fully automatic mode that would remove all required dump box control from the driver. To achieve this solution the controller and its associated software would have to be able to sense approaching obstacles, and determine if any action was required to lower the box to a safe level from the position previously set by the driver. If the controller actually forced the box

downward to a safe height, it should then be able to return it to the preset height after the vehicle passed the obstacle. While in this initial solution a means to move the box was not incorporated, the team programmed the application to do the automated action should further automation be made available at a later time. Thus, a modular approach was developed and both a 'lower box' function as well as a 'raise box' function were programmed. By them being encoded as modules, the raise box function could be easily disabled, as it was in intermediate testing, by removing only a single reference line in the program while the module could be easily reactivated should full automatic control become desirable. In this semi-automated edition of the control, the driver is alerted to take action, using a built in buzzer to both lower and raise the box. When the team considered user friendliness for the operator of the plow, the fact that plow route can be operated from multiple operational directions, and considering that routes can even be closed geometries that have single or multiple loops, the team decided to use a non-directed search method to explore upcoming obstacles. This method, which was computationally intensive, meant that a plow route could be entered at any point and from any direction and upcoming obstacles could always be detected. Using this approach, if an plow operator and their vehicle is assigned to plow to bare pavement over their route, they can repeatedly plow the route or backtrack along the route, after a turnabout, without having to reset the control.

The sections below describe the process of software development for the proposed solution. The team chose to develop a modular solution so that functionalities could be added or removed if and when required. The team divided the software development into four (4) phases:

1. Planning.
2. Design /Develop Prototype.
3. Implementation.
4. Testing / Evaluation.

4.2.1 Planning

The first phase of software development is planning. The goal of planning is to understand the objectives and constraints of the proposed system. First of all the team had to identify the stake holders involved in this project. Then the team held interview sessions with the stake holders to gather the requirements. After conducting interviews with all the stake holders the requirements were finalized with the consent of the stake holders. The objective was to develop a collision avoidance system that uses the current GPS coordinates and bridge coordinates from a database to identify upcoming obstacles. If an obstacle is coming up and the dump box is higher than the safe clearance height, it should be lowered. The HMI controller had to be programmed to handle the GPS data acquisition, bridge database, I/O from the snow plow and a user interface to accept user input and alert the user. One of the major constraints for the proposed system was installation cost. For the pilot system more expense was allowed (Pelzer et al., 2007). Another constraint for this project was extreme weather conditions. The snow plows operate in extremely cold and wet conditions, thus limiting the type of technology that can be used. The snow plows spread a magnesium chloride mixture which is very corrosive. This influenced the team selection of hardware on the vehicle exterior; it had to be corrosion resistant.

4.2.2 Developing the prototype

Once all the objectives and constraints were identified, the team started developing a prototype for the proposed solution. The team began with developing a flow chart shown in Figure 25.

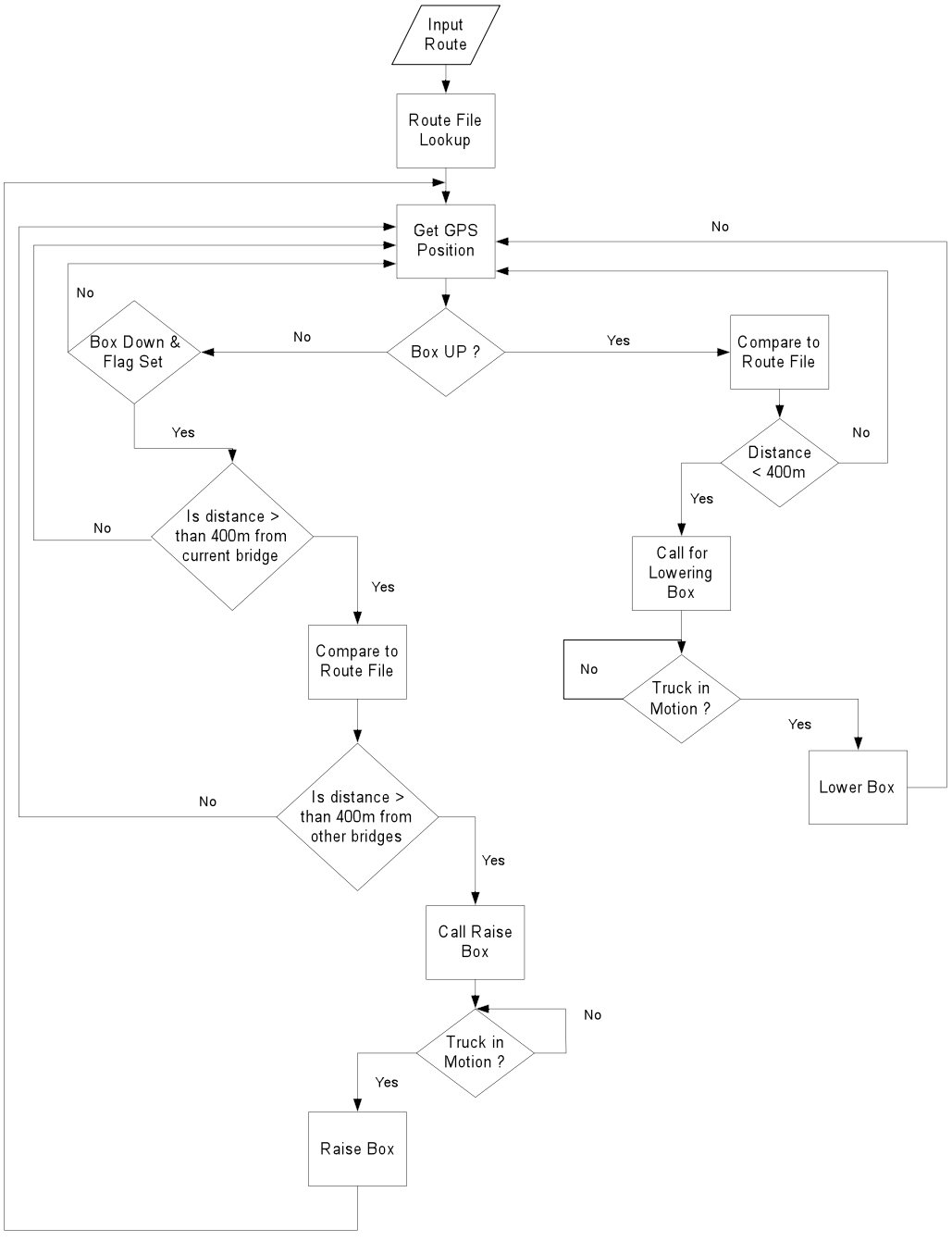

Figure 25: Prototype system Flowchart

The flowchart states that a 400 meter offset distance was used as the value to initiate lower and raise the box actions for collision avoidance. The value of 400 meter is based on the following calculations:

Time to lower the box from fully raised position to fully lowered position (t) = 15 sec.

Buffer time added for safety (bt) = 5 sec.

The speed of a snow plow while in operation (max) (v) = 45 mph

*Safe distance (sd) = (15 + 5) * (45 miles / hr) * (hr / 3600 sec)*

= .25 miles ≅ 0.4 km. or 400 m

Using these calculations the team made sure that if an obstacle was identified within about 400 meters of the snowplow, with the dump-box fully raised, that collision can definitely be avoided. The driver would almost never have the box in its completely raised position while moving and it would not be required to be completely lowered to avoid collision. Thus 400 meters provides the controller solution plenty of time to lower the box once an obstacle is identified.

4.2.3 Implementation

The program begins with calling the Input function "inputfunc()" which is used to accept a route number that has to be plowed from the user. This function uses the touch screen capability of the OP7200 to accept user input. This function returns an integer value to the main program which in turn is used to extract the correct obstacle list from the database for the routes. Figure 26 shows excepts from inputfunc().

```
while (!tscVKBAttributes(vkbX,1,500,100,1));


// Create 5 buttons to be displayed and used on the LCD


btnCreateText(userX,1,40,10,230,80,1,1,&fi10x12,"ENTER A\nROUTE
NUMBER");

btnCreateText(userX,2,40,150,230,80,1,1,&fi10x12,"ENTER THE \n NEW
BRIDGE");

      btnAttributes(userX,1,0,0,0,1);

      btnAttributes(userX,2,0,0,0,1);

   btnMsgBox(0,0,320,240,NULL,"",1,0);


final = 1;

   while (final)

      {

      costate

            {
```

Figure 26: Excerpts from inputfunc()

When the main function receives the route number it passes it to function named "Relate()". This function returns a variable of type long. This function then uses a switch statement to identify the proper route file from the database. This route file is copied into the cache memory. Figure 27 shows excerpt from Relate(). Once the entire route data is stored in cache the GetPosition() function is called by main. This function extracts the current location from the GPS receiver and converts it to the desired format. The team identified Degrees/Decimal Minutes as the desired format for all GPS readings. All of the Mn/DOT

databases used different formats to store GPS location information.

```
long relate(int ip)

{


      fontInfo fi14x16,fi10x12,fi;

   glXFontInit(&fi, 17, 35, 32, 127, Font17x35);


    switch(ip)

   {
```

Figure 27: Excerpts from Relate()

```
GPSPosition getposition()

{

      char sentence[MAX_SENTENCE];

      int input_char, count;

      int string_pos;

      char dir_string[2];

   float distance;

   serCopen(4800);

   string_pos = 0;

      dir_string[1] = 0;


      //receive and parse GPS data
```

Figure 28: Excerpts from GetPosition()

The main function checks if the box is in raised position. If the box is up then the main function compares the value received from the GetPosition() function with each obstacle on the route stored in the cache memory, Figure 28. If any obstacle is less than 400 meters away, then the main program calls another function called "Moving". The moving function checks if the truck is in motion. One of the assumptions made by the team was if the truck is moving at less than five (5) mph, it might not be safe to lower the box. This was a precaution taken by the team to avoid injury to workers working around the truck. To check this the moving function accepts two readings from the GPS at an interval of 1 sec. If the distance between these two readings is less than two (2) meters it means that the speed of the truck is less than five (5) mph. If the truck is not moving the driver would not be alerted to lower the box. It was added to make sure that nobody gets hurt while performing repair or maintenance on the truck. Figure 29 show excerpts of the Moving() function.

```
int moving(void)

{

      char sentence[MAX_SENTENCE];

      int input_char, count;

      int string_pos;

      char dir_string[2];

   float distance;




      //calculate distance from known coordinates

      GPSPosition ar1, ar2;

      serCopen(4800);

      string_pos = 0;

      dir_string[1] = 0;
```

Figure 29: Excerpts from Moving()

If the truck is moving then the LowerBox() function is called. The function alerts the user by flashing a message on the LCD screen and with a beep. It keeps on beeping at the user until the box is lowered to the safe height. Figure 30 shows excerpts from LowerBox function.

```
void lowerbox(void)

{

mov = moving();

   if(mov == 1)

   {

   do

   {

     digOut(1,0);

      costate

      {

            glSetBrushType(PIXXOR);

            glPrintf(100,25,&fi,"Lowering");

            glPrintf(135,65,&fi,"Box");
```

Figure 30: Excerpts from LowerBox()

Now if the box is alerted down by the controller, and the next obstacle is more than 400 meters away, and the truck is moving, then the main function calls the “Raise box” function. This function alerts the driver to raise the box by flashing a message on the LCD display and beeping with a different signal than lower box. Figure 31 shows excerpts from RaiseBox().

The entire code including the functions discussed in this section can be found in Appendix B.

```
void raisebox(void)
{
   mov = moving();
   printf("in raise");
   if (mov == 1)
   {
   do
   {
      digOut(2,0);
      costate
      {
         glSetBrushType(PIXXOR);
         glPrintf(100,25,&fi,"Raising");
         glPrintf(135,65,&fi,"Box");
         waitfor (DelayMs(500));
         for(i=0; i < 3; i++)
         {
                  for(delay = 0; delay < 2000; delay++);
                  buzzer(1);
                  for(delay = 0; delay < 3000; delay++);
                  buzzer(0);
          }
       }
   }while(digIn(13)==0);
```

Figure 31: Excerpts from RaiseBox()

4.3 Testing / Evaluation of the Software

While testing and evaluation is the last step in software development cycle, the team spent a large block of time performing these functions. Testing was completed in several steps and lead to the discover on many of the problems already explained in Chapter Three and ultimately the development of the TIBIAS solution and the construct of a test route away from busy highways.

After the IBIAS software was developed the team had to test it to make sure everything was working in the predicted and desired manner. This section discusses the process of software testing performed by the team. The basic idea of testing is to make sure that:

- The software is in compliance with the customer needs.
- The software runs without any errors.

The main task of this sub-section is to make sure that the program covers all the requirements of the system. These requirements were gathered by interviewing various stakeholders at Mn/DOT and represent the essential system requirements. These requirements are listed in Table 5.

Table 5: Controller System Requirements

I.	Provide some kind of alarming system to alert the driver.
II.	Alert the driver to lower the box if an obstacle is coming up.
III.	The system should alert the driver with enough time to lower the box.
IV.	Alert the driver when it is safe to raise the box.
V.	The system should not alert the driver to lower or raise the box when stationary.

To lower or raise the box some kind of beeper, for the purpose of alerting the driver, was required. The team decided to use the speaker built into the OP7200 controller. Two different types of beeps were created so that the driver could distinguish when they were to lower or raise the box. Lower box functionality was the more important of the two functions considering the safety factors the team was addressing. So the beep for the lower box had to sound more urgent. At the same time a visual message on the controller screen was also displayed. This message would indicate “lower box” or “raise box” thus making sure the users know exactly what to do when they hear beeping. This beeping functionality is a part of lower box and raise box functions. The lower box used the beeping to alert the user if an obstacle was less than 400 meters away. Another important functionality was to provide a way to alert the driver to raise

the box. The team developed a function dedicated to this functionality. This function becomes active once the box is lowered by the controller. This function performs two comparisons:

1. The plow has passed the current obstacle by more than 100 meters.
2. There is no obstacle less than 400 meters away.

If either of these two conditions is not satisfied then it is not safe to raise the box. If both of these conditions are satisfied then this function alerts the driver to raise the box by sounding an alarm. This alarm is different from the one to lower the box. The function also flashes a message on the controller output screen.

Another requirement was the system should not alert the driver to lower or raise the box if the truck is stationary. This was a safety feature as someone might be working under the dump box performing maintenance or repair functions while parked within 400 meters of an obstacle. From interviewing stakeholders at the workshop, the team realized that the truck might even be moving a bit while performing maintenance operations. Based on this input the team decided that if the truck is moving slower than five (5) mph it should still be considered stationary. To provide this functionality, the programmer created a function which would extract two GPS readings, at an interval of 1 second then compute the distance between these, and compute the speed of the vehicle. If the computed speed is more than five (5) mph, the program should not alert the users to raise or lower the box, thus another user requirement was met.

4.3.1 Testing for Programming Errors

The team created a list of all the functions in the software. While keeping software engineering design in mind, various functionalities were built up as separate modules. Modularity provides the team with the option to add or remove certain functionality if required without redefining the fundamental program structure. This also made the testing and debugging easier to perform, which will be explored in this section. Table 6 includes a list of all the important functions that needed testing.

Table 6: A List of All Important Functions That Needed To Be Tested

I.	**Input function** User interface for route to be worked
II.	**Get** (long./lat.) **Position** of snowplow to compare
III.	**Lower Box** since snowplow is approaching obstacle
IV.	**Raise Box** since obstacle is cleared
V.	Check that Snowplow is **Moving**

In this section each of the functions is tested individually for logical and programming errors.

Input Function

The main task of the input function is to receive an input from the user, store this input, and based on this input load the appropriate text (.txt) to the controller's cache or active memory. To test this function the team would check if the controller was able to store the user input correctly, and if it was able to load the appropriate route file based on the user input. To check the user input the controller was required to print the variable (to the screen) in which the user input was stored. Next, the team had to make sure that proper route file is being pulled up and stored in the cache. To check this, the team required the controller to print the route file out on the screen so it could be compared to the respective route file in the route database. Once both these tests were successfully performed, the programmer was confident that no programming errors existed in this function.

Get Position

This function is used to acquire data values from the GPS unit, store it in a local variable, extract the GPS coordinate from this stream, and convert it into the required GPS coordinates format i.e. Degrees and Decimal Minutes. This function is called whenever a GPS position is needed; this makes it one of the most important functions in the program. First, it assured that the stream of data received from the GPS were not 'garbage values', so this value was printed out on the controller screen and it was compared with the stream of data received from the same location directly by the GPS software. Once both consistently agreed, indicating the controller was processing the data stream correctly, the next step was to make sure that proper GPS coordinates were extracted from this data stream. Once extracted from the data stream they were compared to the coordinates received from a hand held GPS unit. Once the coordinates correctly agreed the next step was to make sure that they were in the correct required format: Degrees and Decimals Minutes. Once all of these tests had been successfully performed, the Get Position function was declared to be working error free.

Moving

The main task of the moving function was to obtain two GPS positions readings at a certain interval. Once the controller had these readings the next task was to compute the distance in meters between these two GPS positions. If this distance is zero meters, that means the vehicle is stationary and not moving. If this distance is more than zero meters that means that the vehicle is moving. This function was added to the controller program, as stated above, to provide safety to operators and to avoid accidents of any kind while servicing or repairing the vehicle. If the function returns a value that signifies that the vehicle is not moving, then the system would not alert the driver to raise or lower box. Based on input from the drivers, and personnel at the Mn/DOT workshops it was decided that if the vehicle is moving at a speed greater than five (5) mph then the chances of someone being under the dump box is zero. The team determined that if the vehicle moves more than two (2) meters in 1 second its speed exceeds five (5) mph. The programmer had created two temporary variables called: T1 and T2; T1 stores the first GPs reading and T2 stores the reading taken after one second. To test, the programmer received a

reading into T1 and stored a known GPS position in T2. The programmer then checked to see if the proper distance value was computed, then the routine was tested the other way round by replacing T1 instead of T2 with a fixed value. Once the team was satisfied that the controller was receiving proper values in T1 and T2, the programmer wanted to check if the function was returning a proper value. To do that, the programmer first hard coded the same values in T1 and T2, and checked if the function returned a zero. Then the programmer replaced T1 and T2 with different known values and checked if the functioned returned a one. Once all these tests were performed the programmer was sure there were no bugs in this function.

Lower Box and Raise Box

The lower box function is called by the main function if there is an obstacle less than 400 meters away while it is moving. The lower box function first checks if the box is in the raised position, then calls the moving function to check if the vehicle is moving. If both these conditions are true then the function should alert the driver by sounding the beeper present on the controller. There were only two things that had to be checked in this function. First, was the function receiving and storing proper values from the magnetic switch. Second, did the beeper work as desired and when required. The input from the magnetic switch was received at port 12 of the controller and stored in a variable say called 'X'. So first the programmer checked if variable X stored a one when the switch is open and a zero when the switch is closed. Once the team was sure that this functionality was flawless, they had to make sure the beeping functionality was working properly. The function should beep if the vehicle is moving and the box is up. To test this logic the team made a change to this condition, i.e. it should beep if the box is up (even with the controller stationary). This test made sure that the beeping functionality was working properly.

The raise box function is called by the main function only after the box is lowered by the lower box function. The main task of the raise box function is to alert the driver when it is safe to raise the box again. The raise box function basically checks two conditions: if it has passed the last bridge (obstacle) by more than 100 meters, and the next bridge (obstacle) on the route is more than 400 meters away. If both these conditions are true, then it alerts the driver to raise the box. In this function the programmer had to make sure that the function computes the 100 meter and 400 meter distance to raise the box correctly. The tests for this function were the same as the lower box function; the only difference was that the function had to stop beeping after 25 seconds, as desired by the plow drivers.

4.3.3 Stationary Test Runs

Once all the functions were working individually as expected, the team had to make sure that all these functions worked correctly with each other. The basic difficulty in doing that was the program is not supposed to work at all if it is not placed on a moving vehicle, because of the presence of the safety driven Moving function. Again some changes were necessary to perform the stationary test runs. The Moving function must be disabled so that the program could work even without moving. It was not very difficult as the program was modular and the moving function was separate allowing it to be enabled or disabled very easily. Once the moving function was disabled the first stationary test run could be conducted. This test run obviously did not provide many results; but, it made sure that all the functions were running well with each

other. After a successful stationary test run the team was ready to take it out for an actual test run. The following section covers all the testing that was performed on the road in real time conditions.

4.3.4 First test run of the system

After all the testing mentioned in the previous chapter was done, the team was now ready to take the testing to the next level or take their solution live. These live tests were performed in a car instead of a Mn/DOT snowplow, so many conditions had to be emulated by the team. First of all the magnetic switches which will be mounted on the snowplow dump box were handled manually to emulate raising and lowering of the box. The team decided to use a cigarette lighter plug to draw the power supply from the car for the controller. Route "101" from the original test data set was used as the test route. The route started at 26th Ave East on to I-35 all the way to 40th Ave West on I-35. Obstacles along this route included sign posts, bridges, over-head pedestrian crossings and railway bridges. A test was run to analyze if the solution correctly detected all the obstacles on the route and alerted the driver in time to lower the box or raise the box. In this test run all the functions were tested again but in real time.

4.3.5 Difficulties

Once the software went live, many abnormalities started to show up. Table 7 lists all the errors and abnormalities observed by the team.

Table 7: List of Errors and Abnormalities

I.	The software was not able to find all the obstacles on the route. The route was ran a couple of times, but the software was still not able to identify all the obstacles on the route.
II.	The lower box function was working as expected but the raise box function would not alert the driver properly.
III.	Many times the program would hang while in the raise box function.
IV.	After a few test runs the program started identifying fewer and fewer obstacles.
V.	The program would hang after finding a few obstacles and start printing garbage values for the distance between the current location and the next bridge.

These were some of the difficulties faced right away after the first few test runs. This section describes the cause of the difficulties identified by the team.

The first problem was that the solution was not able to identify all the obstacles on the route. To solve this problem, the team counted the total number of obstacles on the route and compared them to the number of obstacles on the corresponding route file. There were many more obstacles on the route than the ones present in the route file. By analyzing this data the team realized that the route file only had bridges built before ca. 1990. All the bridges constructed after this time were missing from the route file as noted above in Chapter Three (3). The route also only included the bridge data. The data for pedestrian bridges, railway bridges and signposts was missing from the route file. Thus, the team realized that some kind of system was required which could be used to store location of all the missing obstacles on the route file.

The second problem was that the lower box function worked perfectly but the raise box function did not work properly. Consistently, the system would not alert the user to raise the box or alerted the user to raise the box while an obstacle was coming up and already in range. The team also noticed that when there were a several obstacles close to each other, the system would hang while trying to raise the box. The team realized that the raise box function would be trying to raise the box even when there is an obstacle less than 400 meters away and at the same time the lower box function was trying to lower the box because there was an obstacle less than 400 meter away. This would make the lower box and raise box functions go into an infinite loop thus causing the system to crash.

The third problem was that the raise box function would try to raise the box even before it had crossed the obstacle and sometime not raise the box even after passing the obstacle by more than 100 meters. This problem was a relatively straight forward logical error. The software was not saving the current obstacle separately so when the raise box function was looking for bridges less than 400 meters away after crossing the bridge it would find the same bridge it just crossed as it was not excluded from the search.

Another problem was that after a few runs the system would became inconsistent i.e. it would find certain obstacles in one run, but would not find the same obstacles in the next run. The team identified the cause of this problem by printing out the list of obstacles on the screen. They noticed that some of the obstacles appeared multiple times on the screen while they appeared only once in the original route file. This helped the team to identify the problem. The problem was due to poor software engineering. The programmer neglected to clear the buffer in which the obstacles were temporarily stored after each load iteration. This lead to multiple entries of the same obstacle (early in list) and no room for obstacles from later in the route file.

While printing the distance of all the obstacles from the current location, sometimes these distance values were very large or just “garbage” and sometimes a segmentation fault error occurred and the program would terminate. The segmentation fault occurs when a program tries to access a memory location that it is not allowed to access or attempts to access an invalid memory location. The team realized that several times after the pointer reached the last entry in the obstacle list would not roll back to the first entry. Thus it would keep receiving garbage value from those locations or run out of bounds. One of the reasons for this error was that the team was keeping track of the numbers of obstacles in the route file and used that number as the counter to

browse through route file. The error occurred if the obstacle comparison began from anywhere but the beginning of the list. To counter this error the team made a small change in the route file format. A prefix X was added to the last entry of our route file to indicate the end of list. So whenever an X would be encountered the pointer would be reset to the first location in the route file.

4.3.6 Troubleshooting

By now the team had identified many errors in the system including both logical and programming errors. The team had realized that because of the incomplete obstacle data some kind of mechanism was required so that the missing obstacles could be added to the route file. The team planned to design another system which would work with the current system to keep it up to date. Chapter Three (3) had already discussed the need and design of such a system.

The team had to change the Raise box function significantly because while designing the function the team did not expect to encounter so many obstacles so close to each other. When this actually happened the function crashed. There was another change that had to be made to the program. The program needed to store a pointer for the obstacle the box was lowered for, and exclude that obstacle while doing the calculations to raise the box.

After all the changes were made, the team had to take the system out on the route again for testing. It was very risky to test the system on the actual route along I 35 which is one of the most heavily traveled highways in Northeast Minnesota. So the team decided to create an alternate route and simulate actual route-like conditions on this controlled route. The team created this route using the trainer program. The trainer system was used to create a route with multiple obstacles. Most live tests were performed on this route, a route that wrapped around UMD's upper campus starting at stadium drive and St. Marie Street. Once confident that the system was working properly the team decided to perform a final proof test on the actual route. Using the test route created by the trainer program helped the team to debug the program in real route like conditions, but it was less risky then testing it out on the actual highway. This made troubleshooting easier as, the driver could stop, take an exit or slow down as required to test the system which would not be possible on the real highway. Another advantage of using this test route was that the team could add or remove obstacles for testing convenience to simulate real life conditions. This helped the team in testing and debugging the system extensively. This kind of extensive testing might not have been possible by running just a single route, as all these simulated conditions might not even arise on that particular route but might arise on some other route.

4.4 Road Testing the IBIAS System on Snowplow Trucks

Once initial controller testing cycle was complete, as described in the preceding sections, the team installed the IBIAS system on one of the Mn/DOT District 1 snowplows and tested it in the actual working conditions. The following sections detail the results of performance testing after the team installed the IBIAS system on a snowplow. When the team took the system to Mn/DOT District maintenance headquarters in Duluth to mount it on a plow truck, they found that the truck chosen would not be used on "Route 101" but rather a route that covered part of

I35 (26th Ave. W to 26th Ave E.) but then continued along Mn 61 (London Road) all the way out to 61st Ave E. This new route would provide all types of the hard obstacles that were discussed in the earlier sections and added a large number of low hanging power and TV cable lines stringing across London road. The number of obstacles along the route was about 80, a number much higher than with any test route the team had explored. In addition to the Mn/DOT snowplow, St. Louis Country agreed to allow the team to place an IBIAS system on one of their maintenance plows that ran a rural route outside of Virginia, Mn. This plow maintained a mostly low speed loop with over 90 obstacles, nearly all low hanging cables as it circled a lake past many residence homes. This route was normally plowed at a lower speed so the team was able to relax the obstacle recognition distance to 200 meters (from 400 meters) but even with this change, the obstacle list would prove difficult to handle.

4.4.1 Mounting the System

Many challenges surfaced once the team actually started to implement the system even after the extensive testing program. The team needed to mount the magnetic switches below the dump box at the back of the truck's frame near the pivot point of the single hinge. As the controller was originally conceived, one switch would check if the box is in raised position higher than the safe height, the other to indicate if it's down lower than the safe height. While installing the switches the team realized that there wasn't enough room to place two switches near the box pivot location. This led to one of the major changes in the system design. The team decided to use only one switch instead of two, this change had a beneficial side effect since improved programming eliminated the need troublesome second switch which reduced the cost of the system by about $75. The remaining switch had to be placed in such a way that the circuit does not break (the switch does not open) until the box crosses the safe height threshold. This brought in another challenge: determining an exact position to place the switch. The specifications of the switch states that the magnets have to be more than one inch apart to break the circuit. The team calculated the expected switch location, based on the one inch specification and box height angle. But when the switch was actually placed on the truck at the position determined by the calculations, the team noticed that it took more than 1.5" of movement for the switch to open. At the same time the magnet had to be less than 1" from the switch unit before the switch reconnected. There could be many reasons for the switch not working as specified but the one affecting our system could most likely be the way magnets move away from each other, and the proximity to the mass of steel in the truck frame and box. The team noted a curved motion as the path followed by the switch halves mover apart to break the circuit. So the switch had to be moved closer to the pivot but the problem here was that an exact position could not be found. To counter this problem the team designed an adjusting mechanism on which to mount the switch, so that minor positional changes could be made. The team design a fixture pair employing two separate mounting units. One-half of the sensor pair (the magnet) was mounted on the top adjustable bracket (which is attached to the dump box) and the other (switch body) was mounted on the lower adjustable bracket, attached to the truck frame. Since the active sensor was wired directly to the controller this mounting arrangement minimized the potential physical damage to the switch wiring as a result of repeated motion. Figure 32 and Figure 33 show models for the upper and lower mounting units as designed in CATIA R17.

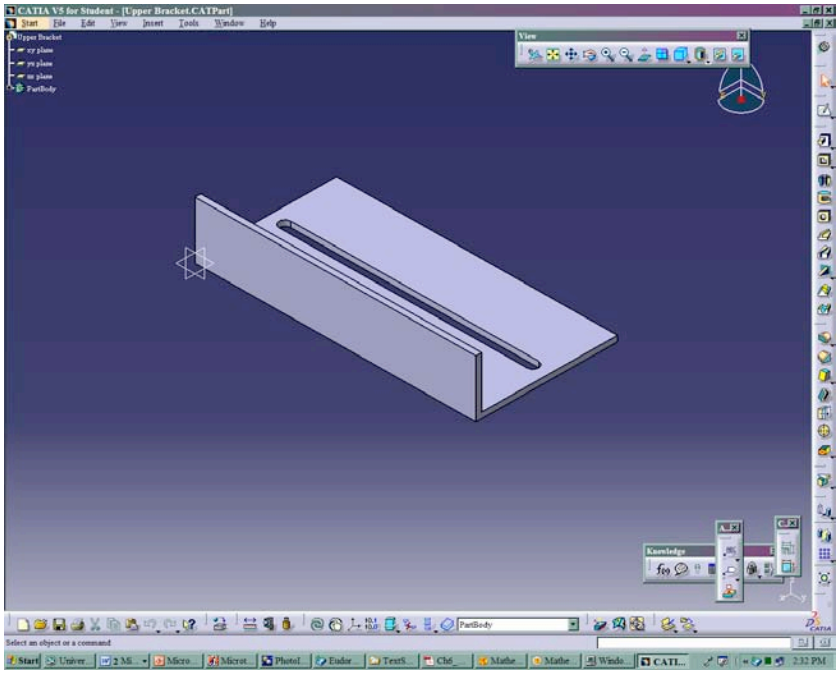

Figure 32: Upper Adjustable Mounting Bracket

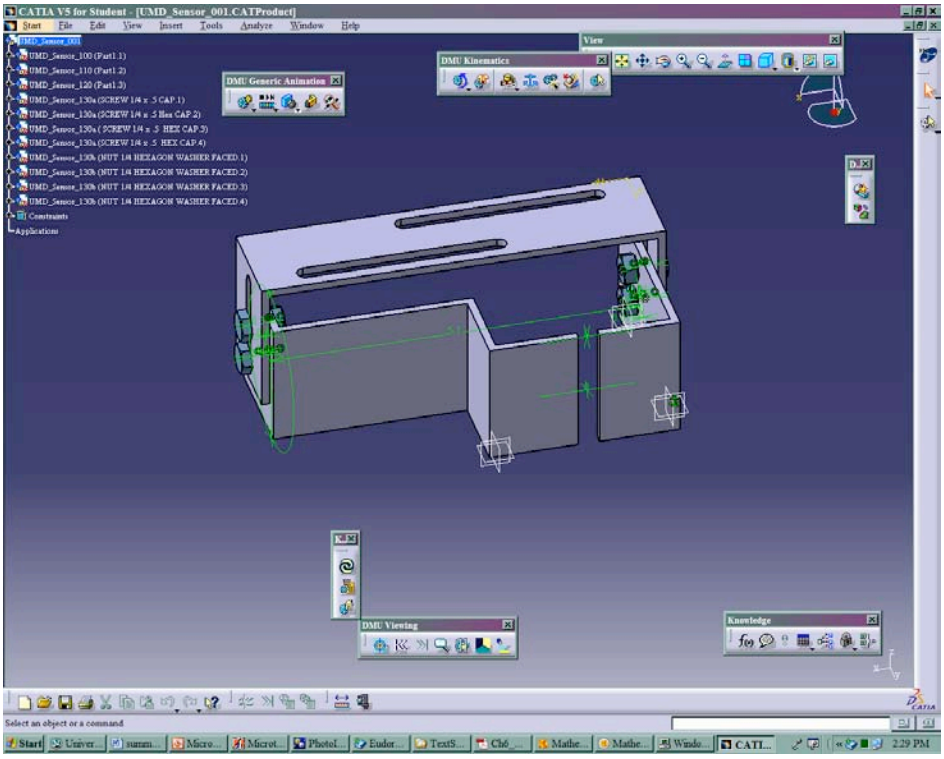

Figure 33: Lower Adjustable Mounting Bracket

Another problem was to identify an appropriate position to place the controller inside the cabin. During initial testing was an as yet unsolved problem as the cabin was already loaded with many other pieces of control equipment. However, relying on the help of Mn/DOT Duluth workshop personnel a small metal mounting platform was made and welded near the driver's seat inside the cabin, see Figure 21. After the plow operators drove the plow truck on a few early test runs, they felt that the position was poor and requested that it be moved to a position closer to the back wall of the truck cab. The original position had been chosen so that the operator could easily see the user interface to enter their plow route and read any messages, however, since the audio alerts were very loud and the obstacles were considered continuously, the operator really had no need to view the screen on the Op7200 controller once they began plowing operation. During phase two (2) testing, currently being conducted, the mounting platform has been move to a less conspicuous position at the rear of the cab.

4.4.2 Testing the system in the Snow Plow

Once the system was mounted on the truck, the team had to again test the system. Since the snowplow truck provided by District 1 was assigned to a specific plow route, and not the one the team studied extensively, one team member joined the operator on the actual route for which it was to be used. While the driver ran the route, the team member took readings using the TIBIAS system to develop the required obstacle map. After this obstacle map was loaded back to the controller, the driver was asked to run the route with the box raised to a level at which they would usually operate the plow truck. The IBIAS system was able to identify all the obstacles, but on both test routes there were too many obstacles the constant alerting to upcoming obstacles proved to be very irritating for the driver. This was a serious issue that had to be handled, because if the user of the system was not comfortable he would not use it and that would defeat the purpose of developing the system. While the problem is yet unresolved and is the subject of on-going testing, the team first decided to increase the safe height for the box by adjusting the switch position and in fact designing a new switch mounting device that moves the magnetic switches away from the earlier hard mounts in contact with the steel of the box and frame. This new switch mounting system is also part of the on-going test phase. The team notes that these continuing tests are being conducted by the MIE department of UMD at no further cost to NATRSL or Mn/DOT at this time. The department is conducting the further testing as a part of their educational mission to Northeast Minnesota.

The team faced another problem while testing the IBIAS system in Virginia, MN, on the St. Louis county snowplow route. The test route in this location had a many obstacles all fairly close to each other as noted earlier. This forced the system to beep nearly continuously either to lower the box or raise the box. This proved very frustrating for the snowplow operators. The team decided to make changes to the system to improve user friendliness and user acceptability. A change was made in the raise box function so that it stops beeping after 30 seconds if the driver does not raise the box. This change was in addition to the one noted above (reducing the safe closing distance to 200 meters). These changes helped the user feel more comfortable with the system, as it was not beeping at them all the time, especially when safe. The team discussed making a similar change to the lower box function. However, they decided against it as the lower box function was more safety critical as compared to the raise box function.

When phase I testing was reviewed after the 2007-08 winter season, the team was not satisfied with the system's performance. Several issues were uncovered during this testing that should make the system more robust with further refinements. The team was encouraged that the use of a GPS system seemed to be satisfactory as a sole mapping technique but the signals were very disturbing to the drivers, a solution seems to be the use of a dash mounted warning light: "lower Box" with a audio signal that lasts only five (5) to 10 seconds in duration, a task being explored in second phase testing. Because of the apparent problems with the operation of the magnetic switch, the team performed a second redesign of its mounting system. The team felt that this problem was the most damaging problem that was observed during phase I testing. In the new design, the switch will be directly pulled apart, following a linear path, rather than by rocking apart as was noted with the initial designs. Additionally, with the new mount, the switch and magnet has been mounted in a plastic case on an aluminum base that is offset away from the truck' frame and not connected directly to the steel dump box. These improvements are expected to eliminate the hanging switch, or a switch that once opened after the box has been raised would not close (reset) when the box was lowered to a safe height.

Chapter 5: Summary and Conclusions

5.1 Summary of the Project Activities

A usable snowplow truck mounted collision avoidance controller has been developed and testing during snow plow operation along the highways of Mn/DOT District 1. As the primary goal of this project, the team can declare that a successful conclusion to the project has been achieved. The controller that has been designed and constructed, based on the goals set for the project, is one that can be an aid to Mn/DOT maintenance staffs statewide. After phase I testing, several nagging concerns with the solution are still evident, and are being address through a continuing testing phase by members of the faculty, staff, student body of MIE at UMD. However, the team can report that the project has been concluded on a successful note. The team began the project by performing a study of applicable means for sensing and mapping a set of obstacles that would be required by any on-board controller. From this study, and the target cost communicated by Mn/DOT officials, a controller system for mobile deployment that included an OP 7200 hardened programmable controller, a Garmin 15L GPS receiver/controller and a GE Guardian 166 magnetic proximity switch sensor was designed. The prototype system that was designed totals about $575 in it prototype design. The system can be seen in Figure 22 in this report.

During the system design activities, the team developed a tool for creating obstacle files. In the original project proposal, the team had proposed using Mn/DOT's Br-Info database as the source of this list. A Visual Studio 2005 software application was built to extract bridge geometric (locational) information from this database to write an obstacle file that could be loaded into the OP 7200 controller for use along the roadways of District 1. The application developed used two additional databases provide by Mn/DOT personnel: a Plow route database that included road and milepost information for each snow plow route in the District and a database of locational information for most of for the highway mileposts within the District. When the application was executed, a total of 104 text files were built to cover the plow routes in the District, as specified by the plow routes database. After later analysis, it was found that the set of plow routes was in fact no longer current. However, the routes that were operated along I35 from Pine County through Duluth were correctly correct in position, but they were no longer called by the identifying labels as had been provided by District personnel in the database.

During preliminary testing one of the routes, called "Route 101" by the research team, a route that ran along I35 from W. 40th Avenue to E. 26th Avenue in Duluth, the team found that all the bridges built after ca. 1990 were not coded in the files created by the Visual Studio application. Careful review of the Br-Info database that had been provided confirmed that these bridges, which would have been part of the final construction phase to complete I35 were not included. The team suspected that, due to homeland security requirements, current Br-Info database data is restricted in its use. While making the early test runs, the research team noticed that other types of obstacles were passed and they were not in the Br-Info database. These included highway signs, pedestrian crosswalks, low hanging power cables and even temporary overhead obstacles that may be present due to roadway maintenance activities or construction. While they could be struck by the plow dump box if it is raised, they would never be found in the Br-Info database environment which is used for other purposes. The team decided to develop a

separate controller application called the "TIBIAS" or trainer application to allow the system manager to build route obstacle files that can be easily loaded to a plow truck controller. This application can be executed by a trained operator, in any vehicle, simply by running along a new or existing plow route and harvesting all the obstacles, be they permanent or temporary. After minor adjustment to the data, all the locations so harvested can be written to a text file including the route label and then recompiled to the appropriate snowplow truck's controller.

The controller software that is executed during plow operation was designed to be used in an automated manner that would relieve the driver of all box positional control. While currently deployed as a semi-automated version where the driver is alerted to move the box down to a safe height as an obstacle approaches, the modular design would be able, with suitable modification to operate in the envisioned automatic control mode. After extensive testing, including the construction of a test route near UMD in Duluth, the obstacle detection system or "IBIAS" was mounted on a Mn/DOT snowplow and tested during actual plowing activities during Winter 2007-2008 along a route in Duluth. The team was also able to mount an IBIAS controller on a St. Louis County Maintenance plow during this time frame and the team was able to gather vital information that will lead to a much improved IBIAS after further refinements during phase two testing being performed by MIE personnel at UMD. The team notes that these continuing tests are being conducted by the MIE department of UMD at no further cost to NATRSL or Mn/DOT at this time. The department is conducting the further testing as a part of their educational mission to Northeast Minnesota.

As with all NATRSL project that have been conducted at UMD, one of the most important components is the education impact of transportation based research. This project was no different and touched many students' educational lives. The project formed the basis for two Master of Science degree theses for the co-author's of the project final report: Hilal Katmale and Rami Verma (Katmale 2007, Verma 2009). Additionally, one of the MIE programs capstone senior design project teams was used to provide design work during 2007 (Pelzer, et. al. 2007). Finally, when it was determined that the magnetic switch mounting system was problematic, Dr. Lindeke asked his class in kinematic's design during spring semester 2008 to analyze and build alternatives to the mechanism used to move the sensor units apart, linearly. Two teams suggested designs and one was chosen for phase II testing and is mounted on the plow at this time.

5.2 Observations, Conclusions and Recommendation for On-board Collision Warning Systems

Having completed phase one on-board testing and assessed the results of this project, the team can report the following issues observation, results and recommendations:

- With the proper GPS unit, one capable of internal dead reckoning, a single obstacle detection sensor could be used in open areas of District 1 highways. While this was a concern at the outset of the project, road testing suggested that a single sensor solution method can be implemented successfully. This was refreshing news in light of the costs of additional reliable sensor systems. When the sensors and controller hardware were assembled into a system for use on the snowplows, a cost of about $575 (with potentials for lower costs if the system is

build in larger quantities) was found. This cost is very close to the target of about $500 for each plow that was suggested by Mn/DOT personnel at the projects beginning.

- The use of modular controller software for the IBIAS controller was implemented. By using this approach the current, semi-automated, system could be modified, with small effort, to one that is fully automated to free the driver from routine obstacle tracking activities during long and fatiguing plowing shifts.

- The method used to track obstacles in which had each obstacle in the obstacle list is checked each time a new reading was obtained from the GPS unit (at 1 second intervals) was problematic on routes with a high number of potential obstacles (like the two that formed the test cases during winter 2007-2008). However, by checking obstacles this way, which was computationally intensive, a plow route could be entered at any point and from any direction and upcoming obstacles could always be detected. Using this approach, if an operator is assigned to plow to bare pavement over their route, they can repeatedly plow from end to end along the route or backtrack along the route, after a turnabout, without having to reset the control.

- Additionally, because a plow could be diverted from its regular route to another in case of break downs during snow emergencies, all plow routes within a district are loaded into the IBIAS controller as a default case. Then, if the driver is asked to move to a different route, all they need do is enter a function key on the controller, after finishing their current route and entering the identification for the new route to which they have been assigned.

- Because of issues with the Br-Info databases supplied, and the presence of obstacles along the plow routes that would never be listed in this Mn/DOT database, the team developed a companion application called the Trainer or TIBIAS. This application, which would be loaded on a similar OP 7200 based controller like those used for the IBIAS system, when coupled to a laptop computer can be used, and in fact was used, to create obstacle locational files that can be used to maintain the currency of the information for any and all plow routes.

- Significant educational activities have been conducted by this project at the undergraduate as well as graduate level including undergraduate course projects, a senior undergraduate capstone design project and two Master of Science degree projects.

Bibliography and References

Alexander, Lee, Pi-Ming Cheng, Max Donath, Alec Gorjestani, Bryan Newstrom, Craig Shankwitz, and Walter Trach, Jr. "DGPS-based Lane Assist System for Transit Buses," *IEEE Intelligent Transportation Systems Conference*, October 2004, Washington, DC pp. 755-760.

Alexander, Lee, Alec Gorjestani, Craig Shankwitz, *DGPS Based Gang Plowing*, Final Report of Investigations, Minnesota Department of Transportation,St. Paul, Mn, April 2005.

Alexander, Lee., Gorjestani, Alec., Shankwitz, Craig., *DGPS- Based Gang Plowing*, Center of Transportation Studies, University of Minnesota, Minneapolis, Mn (April 2005), Retrieved November 22, 2007, from http://conservancy.umn.edu/handle/993

Aono, T., K. Fujii, S. Hatsumoto, T. Kamiya, "Positioning of Vehicles on Undulating Ground using GPS and Dead Reckoning," *IEEE International Conference of Robotics and Automation*, May 1998, Leuven, Belgium.

Bitkom, (2005), *RFID White Paper Technology, Systems, and Applications*, Retrieved November 15, 2007, from: http://www.rfidconsultation.eu/docs/ficheiros/White_Paper_RFID_english_12_12_2005_final.pdf

Bostelman, R.V., Hong, T.H., Madhavan, R., "Towards AGV Safety and Navigation Advancement Obstacle Detection using TOF Range Camera," *ICAR'05 Proceedings, 12th International Conference on Advanced Robotics*, July 2005, pp. 460-467.

Bouju, Alain, A Stockus, A. Bertrand, P. Bousier, "Location Based Spatial Data Management in Navigation Systems," *IEEE Intelligent Vehicle Symposium*, June 2002, pp. 172-177.

Burdet, Luc A, *RFID Multiple Access Methods*, ETH Zurich, (2004), Retrieved Nov 15, 2007, from http://www.vs.inf.ethz.ch/edu/SS2004/DS/reports/06_rfid-mac_report.pdf

Cavanaugh, J. Personnel Officer, Mn/DOT District I, Private Communication, Duluth, Mn, 1Aug, 2006,.

Cutchin, Carli, *Automatic Vehicle Location*, California Center for Innovative Transportation, UC Berkely & Caltrans (2005), Retrieved October 20, 2007, http://www.calccit.org/itsdecision/serv_and_tech/Automatic_vehicle_location/avl.html

Dmitriev, S.P., O.A. Stopanov, B.S. Rivkin, D.A. Koshaev, D. Chang, "Optimal Map-Matching for Car Navigation Systems," *Gyroscopy and Navigation*, v. 29 #2 2000, pp. 57-69.

Eaton Vorad, *EVT-300 Technical Highlights* (2001), Retrieved November 20, 2007, from http://path.berkeley.edu/~cychan/Research_and_Presentation/Pedestrian_Detection_TO5200/Sensors_Information/EVT-300.pdf

Estochen, Brad, "Specialty vehicle platform results for intelligent vehicle initiative: Minnesota field operational test," *Transportation Research Record*, n 1826, 03-2672, 2003, pp. 45-52.

Everson, C. K., "Dynamic Demand for Utilization, Maintenance, Installations, and Retirements of Railroad Freight Cars," *International Economic Review*, Vol. 23, No. 2, June 1982, pp. 429-446.

Ewald, A., V. Willhoeft, "Laser Scanners for Obstacle Detection in Automotive Applications," *IEEE Intelligent Vehicles Symposium*, Oct. 2000, Dearborn, Mi pp. 682-687.

Federal Highway Administration [http://www.fhwa.dot.gov], Sep 2007.

Finkenzeller, Klaus, *RFID Handbook: Fundamentals and Applications is Contactless Smartcards and Identification*, Wiley 2003, New Yory, NY, Retrieved November 12, 2007, from http://www.d.umn.edu/lib/ebooks/index.htm

Fuhrer, Patrik., Guinard, Dominique., & Liechti, Olivier, *RFID: From Concepts to Concrete Implementation*, (2006) , Retrieved Nov 15, 2007, from http://www.guinard.org/unifr/docs/rfid_internal.pdf

Garmin Corporation, *GPS 15H & 15l Technical Specifications, Rev D*, Retrieved May 7, 2007, from: http://www8.garmin.com/products/manual.jsp?product=010-00240-12

Garmin Corporation, *Installation Instructions,* (2008, July). Retrieved September 11, 2008, from http://www8.garmin.com/support/download_details.jsp?product=010-00240-11

Gendreau, Michel., and Potvin, Jean-Yves, "Issues in Real-Time Fleet Management," *Transportation Science*, Vol.38, No.4, (November 2004), pp. 397-398.

Givens, Joella and Arlis Heath, *Snowplow Routing Optimization Project (P264): Technical Findings*. Internal Mn/DOT Report 9-28-2000.

Gorjestani, Alec., Alexander, Lee., Newstorm, Bryan., Cheng, Pi-Meng., Sergi, Mike., Shankwitz, Craig., and Donath, Max, *Driver Assistive Systems for Snowplows*, (March 2003), Intelligent Vehicles Lab ITS Institute, University of Minnesota, Mpls. Mn, Retrieved November 23, 2007, from http://www.lrrb.org/pdf/200313.pdf

Hofmann-Weilenhof, B., H. Lichtenneggar, J. Collins, *Global Positioning Systems Theory and Practice*, 4th ed., Springer-Verlag Berlin/NewYork, 1997.

Hong, T., R. Bostelman, R. Madhavan, "Obstacle detection using a TOF Camera for Indoor AGV navigation," *2004 Performance Metrics for Intelligent Systems Workshop PerMIS* 2004, Gaithersberg, Md.

Joshi, Rajashri R., "Novel Metrics for Map-Matching in In-Vehicle Navigation Systems," *IEEE Intelligent Vehicle Symposium*, June 2002, pp. 36-43.

Katmale, Hilal., and Wyrick, David., "Difficulties in Fleet Management: Intangible Factors, Information Overload, and Data Integrity," *American Society for Engineering Management (ASEM) National Conference Proceedings*, October 2006, Huntsville, Florida

Katmale, Hilal *Technological assessment for safe snow plow operation (plow box obstacle detection avoidance)*, Master Thesis Collection, University of Minnesota Dultuth, 2007.

Keene, Doug, "An Overview of Fleet Management," *Public Works*, July 2000, pp. 42-46.

Kroeger, Dennis A. and Reggie Sinhaa, "Business Case for Winter Maintenance Technology Applications," *Sixth International Symposium on Snow Removal and Ice Control Technology*, paper Snow04-034, *Transportation Research Circular #-C063*, June 2004, pp. 323-331.

Leegard, Paul, "Review of Fleet Management Systems within Mn/DOT," Preliminary draft report, Mn/DOT, St. Paul, Mn (2001).

Lee, Tony., and Ghosh, Sumit, *Intelligent Transportation Systems: New Principles and Architectures*, CRC Press Boca Raton, Fl, Retrieved November 13, 2007, from http://www.d.umn.edu/lib/ebooks/index.htm.

McNeff, Jules, "The Global Positioning System," (March 2002), Retrieved October 31, 2007, from http://ieeexplore.ieee.org/iel5/22/21335/00989949.pdf

Min, Zhang., Wenfeng, Li., Zhongyun, Wang., Bin LI., and Xia, Ran, " A RFID-based Material Tracking Information System," *IEEE International Conference on Automation and Logistics* (August 2007), Retrieved Nov 15, 2007, from http://ieeexplore.ieee.org/iel5/4338502/4338503/04339081.pdf?tp=&arnumber=4339081&isnumber=4338503

Moore, Terry., *An Introduction to the Global Positioning System and its Architecture*, Institute of Electrical Engineers, London (1994) Retrieved November 1, 2007 from http://ieeexplore.ieee.org/iel3/1332/7038/00284635.pdf

Nookala, Marthand and Brad Estochen, "Minnesota, USA Intelligent Vehicle Initiative," *IEEE Intelligent Vehicle Symposium*, Versailles, Fr., June 2002, pp. 533-36 (v.2).

Nishi, Tatsuya and Tomohiro Takagi, "A proposal of collision avoidance algorithm for driving support system," *IECON Proceedings (Industrial Electronics Conference)*, v 3, 2001, pp. 80-83.

Pelzer, T., Strezisher, J., Wiegers, B., (2007, May). *Mn/DOT Box Automation System. Final Report* of University of Minnesota Duluth, Duluth, Mn, UMDMIE-CD-2007-JVDK-23.

Peradovic, Stevan., and Karmarkar, Nemai, "RFID Readers: A Review," ICECE 2006, Retrieved Nov 16, 2007, from http://ieeexplore.ieee.org/iel5/4178390/4139314/04178418.pdf?isnumber=4139314&prod=CNF&arnumber=4178418&arSt=100&ared=103&arAuthor=Preradovic%2C+S.%3B+Karmakar%2C+N.C.

Pletta, J Bryan., Caskey, David L., and Heermann, Philip D, "Performance Assessment of Automatic Vehicle Location (AVL) Systems," (April 1996) Retrieved October 31, 2007, from http://ieeexplore.ieee.org/search/srchabstract.jsp?arnumber=509083&isnumber=10819&punumber=3679&k2dockey=509083@ieeecnfs&query=%28%28avl+systems%29%3Cin%3Emetadata%29&pos=19

Qiao, Pizhong, Milja Yang, and Ayman S. Mosallam, "Impact analysis of I-Lam sandwich system for over-height collision protection of highway bridges," *Engineering Structures*, v 26, n 7, June 2004, pp. 1003-1012.

Rabbit Semiconductor, *Dynamic C product manual,* Retrieved Mar 10, 2007 from: http://www.rabbit.com/documentation/docs/manuals/DC/DCUserManual9/2intro.htm

Rabbit Semiconductor, *Dynamic C*. Retrieved March 10, 2007, from http://www.topleveltech.com/dynamic-c.htm

Rabbit Semiconductor, *Dynamic C development software*. Retrieved March 10, 2007, from http://www.impulse-corp.co.uk/display_product.asp?id=69

Rabbit single board computers, *OP7200 EDISPLAY*. Retrieved August 20, 2007, from http://mouser.com/catalog/635/53.pdfRoth, Roberta., & Wood, William, "A Delphi Approach to Acquiring Knowledge from Single and Multiple Experts," Association for Computing Machinery (1990), Retrieved November 24, 2007, from http://portal.acm.org/citation.cfm?id=97709.97731

Tully, Alan., *Pervasive Tagging, Sensors and Data Collection*, (2006), Retrieved November 15, 2007, from http://www.foresight.gov.uk/Previous_Projects/Intelligent_Infrastructure_Systems/Reports_and_Publications/Intelligent_Infrastructure_Futures/Reviews/Tagging_Sensorsa.pdf

Verma, Ravi, *Snowplow Dump Box Collision Avoidance System,* Plan B Project report, submitted to the Mechanical and Industrial Engineering Department (Engineering Management) University of Minnesota, Jan 2009.

Vonderhoe, Alan, et al., "GIS-Based Analysis of Intelligent Winter Maintenance Vehicle Data," Sixth International Symposium on Snow Removal and Ice Control Technology, paper Snow04-029, *Transportation Research Circular #-C063*, June 2004, pp. 348-362.

Walker, Kelly, *Mn/DOT Fleet and Shop Operations Summary of Audit*, March 2002.

Watanabe, Masahiro, Okazaki, K.; Fukae, J.; Tamiya, N.; Ueda, N.; Nagashima, M.; "An Obstacle Sensing Radar System for a Railway Crossing Application: 60GHz Millimeter Wave Spread Spectrum Radar," *IEEE Microwave Symposium Digest*, 2002, v2, pp. 791-794.

Wilson, Geoffrey, "Automatic Vehicle Location System Selection," *Vehicular Technology Conference, 28th IEEE* (March 1978), Retrieved Nov 1, 2007, from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1622504

Yonas, Albert., and Zimmerman, Lee, *Improving the Ability of Drivers to Avoid Collisions with Snowplows in Fog and Snow*, Retrieved October 31, 2007, from http://conservancy.umn.edu/bitstream/627/1/200629.pdf.

3M, *Opticom™ Priority Control Systems*, Retrieved November 15, 2007, from http://solutions.3m.com/wps/portal/3M/en_US/Traffic_Safety/TSS/Offerings/Systems/Opticom/

OP7200 eDisplay - Models OP7200, OP7210, (2007). Retrieved August 15, 2007, from http://www.rohnertbiz.com/Op_Inter/OP7200.html

OP7200 SBC & ¼ VGA operator control panel, Retrieved August 15, 2007, from http://www.industrysearch.com.au/Products/OP7200_SBC_and_%C2%BC_VGA_Operator_Control_Panel-24996

Minnesota Department of Transportation, (2003). *Strategic plan*. Retrieved October 18, 2007, from http://www.dot.state.mn.us/information/statplan00/pdf/strategicplan.pdf

Minnesota Department of Transportation, (2005, June). *Benefit-Cost analysis for transportation projects*. Retrieved October 12, 2007, from http://www.oim.dot.state.mn.us/EASS/index.html

Minnesota Department of Transportation, (2006). *SAIL-2 Evaluation* (Mn/DOT Contract No. 86353). Saint.Paul, MN.

U.S.Department of Transportation. (2007a). ITS Joint Program Office. Retrieved October 19, 2007, from http://www.itsoverview.its.dot.gov/

U.S.Department of Transportation. (2001). Regional ITS Architecture Guidance, "National ITS Architecture Team". Retrieved October 31, 2007, from http://www.itsdocs.fhwa.dot.gov/JPODOCS/REPTS_TE/13598.pdf.

U.S.Department of Transportation. (2007b). ITS Joint Program Office/CVO. Retrieved October 19, 2007, http://www.itsoverview.its.dot.gov/green_level.asp?System=CVO

U.S.Department of Transportation. (2007c). ITS Joint Program Office/Benefits. Retrieved October 19, 2007, http://www.itsbenefits.its.dot.gov/its/benecost.nsf/ID/0EFDDBCB0D902643852569610051E2A3?OpenDocument&Query=BApp

U.S.Department of Transportation. (2007d). ITS Joint Program Office/Costs. Retrieved October 19, 2007, http://www.itscosts.its.dot.gov/its/benecost.nsf/ID/0670EBC92B2F61E685256E1D004E9961?OpenDocument&Query=CApp

U.S.Department of Transportation. (2007e). ITS Joint Program Office/Collision Avoidance Systems/Benefits. Retrieved October 19, 2007, http://www.itsbenefits.its.dot.gov/its/benecost.nsf/ID/346B56233657D7AB85256A6A00691A0D?OpenDocument&Query=BApp

U.S.Department of Transportation. (2007f). ITS Joint Program Office/Collision Avoidance Systems/Costs. Retrieved October 19, 2007, http://www.itscosts.its.dot.gov/its/benecost.nsf/SubsystemCostsAdjusted?OpenForm&Subsystem=Commercial+Vehicle+On-Board+(CV)

U.S.Department of Transportation, Federal Motor Carrier Safety Administration (2005). *Forward Collision Warning Systems (CWS)*, Retrieved November 22, 2007, from http://www.fmcsa.dot.gov/facts-research/research-technology/report/forward-collision-warning-systems.htm

Appendix A: Listing of the Visual Studio 2005 (Vbasic.Net) File Extraction Application for Creating a Bridge Obstacle Data Listing for Use by IBIAS

Main Application:

```
Public Class Form1
    Dim Br_Con As New OleDb.OleDbConnection()
    Dim daMilepost As OleDb.OleDbDataAdapter
    Dim dtMilepost As New DataTable
    Dim cbMilePost As OleDb.OleDbCommandBuilder
    Dim strRoute As String
    Dim da_GetBridges As OleDb.OleDbDataAdapter
    Dim cb_GetBridges As OleDb.OleDbCommandBuilder
    Dim dt_GetBridges As New DataTable
    Dim objFile As System.IO.StreamWriter
    Dim BrFile_R As System.IO.StreamWriter
    Dim daRoutes As OleDb.OleDbDataAdapter
    Dim cbRoutes As OleDb.OleDbCommandBuilder
    Dim dtRoutes As New DataTable

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
Dim str_ConString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Documents and
Settings\Administrator\My Documents\Transportation Research\Bridge.mdb"
        Dim I, II, W, B, Q, BD, RCount, Long1, Long2, Lat1, Lat2, RC, loc_in_Array As Integer
        Dim LongLat(,) As String
        strRoute = "'" & txtNo_Rts.Text & "'"
Dim StrippedLong, StLong2, stStartMP, stEndMP, Fst_MP_Long, Fst_MP_Lat, Lst_MP_Long, Lst_MP_Lat As
String
        Dim StrippedLat, StLat2, Z As String
Dim ROUTE_CNT, L_Long, S_Lon, L_Lat, S_La, Tom, Mary, TomTen, Maryten, TomSgn, MarySgn As Integer
        Dim Long_To_DMS, Lat_To_DMS, Tr_DMIN_LONG, Tr_DMIN_Lat As Integer
        Dim Start_MP, End_MP As Single
        Dim StLatSec, StLatDecMin, StLongSec, StLongDecMin As String
        Dim StLatDegMin, StLongDegMin As String
Dim DecLatMin, DecLongMin As Decimal
        ROUTE_CNT = CInt(txtNo_Rts.Text) + 1
Dim Fst_MP_Long_DM, Fst_MP_Long_DMin, Fst_MP_Lat_DM, Fst_MP_Lat_DMin As String
Dim Lst_MP_Long_DM, Lst_MP_Long_DMin, Lst_MP_Lat_DM, Lst_MP_Lat_DMin As String
Dim QueryString1 As String = "select ID, ROADWAY_NA, FROM_TRUE_, TO_TRUE_MI FROM PlowRoute
Where OBJECTID > 0 AND OBJECTID <" & ROUTE_CNT
Dim dFst_MP_Long_DMin, dLst_MP_Long_DMin, dFst_MP_Lat_DMin, dLst_MP_Lat_DMin As Decimal
Dim iFst_MP_Long_DMin, iLst_MP_Long_DMin, iFst_MP_Lat_DMin, iLst_MP_Lat_DMin As Integer
        Using Br_Con As New OleDb.OleDbConnection()
            Br_Con.ConnectionString() = str_ConString
            Br_Con.Open()
            daRoutes = New OleDb.OleDbDataAdapter(QueryString1, Br_Con)
            cbRoutes = New OleDb.OleDbCommandBuilder(daRoutes)
            daRoutes.Fill(dtRoutes)

            For RC = 0 To dtRoutes.Rows.Count - 1
              Start_MP = dtRoutes.Rows(RC)("From_TRUE_")
              End_MP = dtRoutes.Rows(RC)("TO_TRUE_MI")
              strRoute = "'" & dtRoutes.Rows(RC)("ROADWAY_NA").ToString & "'"
                'add it here
              txtMPostNum.Text = strRoute

Dim QueryString2 As String = "Select XUnits, YUnits From MilePosts Where ROUTE =" & strRoute & "and TM
>=" & Start_MP & "and TM <=" & End_MP
```

```
            daMilepost = New OleDb.OleDbDataAdapter(QueryString2, Br_Con)
            cbMilePost = New OleDb.OleDbCommandBuilder(daMilepost)
            daMilepost.Fill(dtMilepost)
            txtRecordCnt.Text = dtMilepost.Rows.Count - 1
            If dtMilepost.Rows.Count > 0 Then
               ReDim LongLat((dtMilepost.Rows.Count - 1) * 10, 1)
            End If
    'Dim objFile As New System.IO.StreamWriter("c:/test1.txt", True)
            Z = CStr(0)
            loc_in_Array = 0
            For I = 0 To dtMilepost.Rows.Count - 2

StrippedLong =   Microsoft.VisualBasic.Left(dtMilepost.Rows(I)("XUnits").ToString, 6)
StLong2 = Microsoft.VisualBasic.Left(dtMilepost.Rows(I + 1)("XUnits").ToString, 6)
StrippedLat = Microsoft.VisualBasic.Left(dtMilepost.Rows(I)("YUnits").ToString, 6)
StLat2 = Microsoft.VisualBasic.Left(dtMilepost.Rows(I + 1)("YUnits").ToString, 6)
                Long1 = CInt(StrippedLong)
                Long2 = CInt(StLong2)
                Lat1 = CInt(StrippedLat)
                Lat2 = CInt(StLat2)
                If I = 0 Then
                   Fst_MP_Long = DectoDMS(Long1)
                   Fst_MP_Lat = DectoDMS(Lat1)
          Fst_MP_Long_DM = Microsoft.VisualBasic.Mid(Fst_MP_Long, 2, 4)
          Fst_MP_Long_DMin = (Microsoft.VisualBasic.Mid(Fst_MP_Long, 6, 2))
                   dFst_MP_Long_DMin = CDec(Fst_MP_Long_DMin)
                   dFst_MP_Long_DMin = dFst_MP_Long_DMin * 100 / 60
                   iFst_MP_Long_DMin = CInt(dFst_MP_Long_DMin)
                   If iFst_MP_Long_DMin < 10 Then
                      Fst_MP_Long_DMin = Z & CStr(iFst_MP_Long_DMin)
                   Else : Fst_MP_Long_DMin = CStr(iFst_MP_Long_DMin)
                   End If
                   Fst_MP_Long = Fst_MP_Long_DM & Fst_MP_Long_DMin

          Fst_MP_Lat_DM = Microsoft.VisualBasic.Mid(Fst_MP_Lat, 2, 4)
          Fst_MP_Lat_DMin = (Microsoft.VisualBasic.Mid(Fst_MP_Lat, 6, 2))
                   dFst_MP_Lat_DMin = CDec(Fst_MP_Lat_DMin)
                   dFst_MP_Lat_DMin = dFst_MP_Lat_DMin * 100 / 60
                   iFst_MP_Lat_DMin = CInt(dFst_MP_Lat_DMin)
                   If iFst_MP_Lat_DMin < 10 Then
                      Fst_MP_Lat_DMin = Z & CStr(iFst_MP_Lat_DMin)
                   Else : Fst_MP_Lat_DMin = CStr(iFst_MP_Lat_DMin)
                   End If
                   Fst_MP_Lat = Fst_MP_Lat_DM & Fst_MP_Lat_DMin
                End If
                If I = dtMilepost.Rows.Count - 2 Then
                   Lst_MP_Long = DectoDMS(Long2)
                   Lst_MP_Lat = DectoDMS(Lat2)
        Lst_MP_Long_DM = Microsoft.VisualBasic.Mid(Lst_MP_Long, 2, 4)
        Lst_MP_Long_DMin = (Microsoft.VisualBasic.Mid(Lst_MP_Long, 6, 2))
                   dLst_MP_Long_DMin = CDec(Lst_MP_Long_DMin)
                   dLst_MP_Long_DMin = dLst_MP_Long_DMin * 100 / 60
                   iLst_MP_Long_DMin = CInt(dLst_MP_Long_DMin)
                   If iLst_MP_Long_DMin < 10 Then
                      Lst_MP_Long_DMin = Z & CStr(iLst_MP_Long_DMin)
                   Else : Lst_MP_Long_DMin = CStr(iLst_MP_Long_DMin)
```

```
            End If
        Lst_MP_Long = Lst_MP_Long_DM & Lst_MP_Long_DMin

Lst_MP_Lat_DM = Microsoft.VisualBasic.Mid(Lst_MP_Lat, 2, 4)
Lst_MP_Lat_DMin = Microsoft.VisualBasic.Mid(Lst_MP_Lat, 6, 2)
            dLst_MP_Lat_DMin = CDec(Lst_MP_Lat_DMin)
            dLst_MP_Lat_DMin = dLst_MP_Lat_DMin * 100 / 60
            iLst_MP_Lat_DMin = CInt(dLst_MP_Lat_DMin)
            If iLst_MP_Lat_DMin < 10 Then
               Lst_MP_Lat_DMin = Z & CStr(iLst_MP_Lat_DMin)
            Else : Lst_MP_Lat_DMin = CStr(iLst_MP_Lat_DMin)
            End If
            Lst_MP_Lat = Lst_MP_Lat_DM & Lst_MP_Lat_DMin
         End If
         If Long1 > Long2 Then
            L_Long = Long1
            S_Lon = Long2
            TomSgn = -1
         ElseIf Long1 = Long2 Then
            L_Long = Long1
            S_Lon = Long2
            TomSgn = 0
         Else
            L_Long = Long2
            S_Lon = Long1
            TomSgn = 1
         End If
         If Lat1 > Lat2 Then
            L_Lat = Lat1
            S_La = Lat2
            MarySgn = -1
         ElseIf Lat1 = Lat2 Then
            L_Lat = Lat1
            S_La = Lat2
            MarySgn = 0
         Else
            L_Lat = Lat2
            S_La = Lat1
            MarySgn = 1
         End If
         Tom = L_Long - S_Lon
         TomTen = Tom \ 10
         Mary = L_Lat - S_La
         Maryten = Mary \ 10
         For II = 0 To 9
            Long_To_DMS = Long1 + TomTen * II * TomSgn
            Lat_To_DMS = Lat1 + Maryten * II * MarySgn

            LongLat(loc_in_Array, 0) = DectoDMS(Long_To_DMS)
            LongLat(loc_in_Array, 1) = DectoDMS(Lat_To_DMS)

            'objFile.Write(LongLat(loc_in_Array, 0))
            'objFile.WriteLine(LongLat(loc_in_Array, 1))
            loc_in_Array = loc_in_Array + 1
         Next
         'Me.Refresh()
```

```
            'For W = 1 To 1000000
            'Next
            'txtLong.Text = ""
            'txtLat.Text = ""
            'Me.Refresh()
        Next
        'objFile.Close()
        'objFile.Dispose()
Dim Tab_Name, Br_ID, Br_Long, Br_Lat, Br_Vert, Br_DeWidth, Br_NoSpans As String
Dim P, RecordCount, Cur_Bridge_Ct, Sum_Br_Count, long_zone, lat_zone As Integer

'If (MessageBox.Show("Would You Like To Create at SnowPlow Route Table", "Table Name",
MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) =
Windows.Forms.DialogResult.Yes) Then
        'Tab_Name = InputBox("Enter A SnowPlow Route Name")
        'If Tab_Name = "" Then
        'MsgBox("You Entered Nothing")
        'Else
        'MsgBox("You Entered " & Tab_Name)
        'End If

 Tab_Name = "c:\Dist_1\Route_No" & dtRoutes.Rows(RC)("ID").ToString & ".txt"
        'MsgBox("File Name is " & Tab_Name)

        Dim Q_String, B_Long, S_Long, B_Lat, S_Lat, SRcount As String
        Dim Br_Record(,) As String
        RecordCount = CInt(txtRecordCnt.Text)
   'Dim myTrans As OleDb.OleDbTransaction = Br_Con.BeginTransaction()

        RCount = 0
        ReDim Br_Record(100, 5)
        For Q = 0 To ((RecordCount * 10) - 2)

            If LongLat(Q, 0) >= LongLat(Q + 1, 0) Then
                B_Long = LongLat(Q, 0)
                S_Long = LongLat(Q + 1, 0)
                long_zone = 100
            Else
                B_Long = LongLat(Q + 1, 0)
                S_Long = LongLat(Q, 0)
                long_zone = 10
            End If
            If LongLat(Q, 1) >= LongLat(Q + 1, 1) Then
                B_Lat = LongLat(Q, 1)
                S_Lat = LongLat(Q + 1, 1)
                lat_zone = 100
            Else
                B_Lat = LongLat(Q + 1, 1)
                S_Lat = LongLat(Q, 1)
                lat_zone = 10
            End If
            Q_String = "Select BR_NUM,NO_SPANS,DECK_WIDTH,LATITUDE,LONGITUDE,VERT_1 from
Bridge where LONGITUDE>=" & S_Long & " and LONGITUDE<=" & B_Long & "and LATITUDE>=" & S_Lat
& "and LATITUDE<=" & B_Lat & "and VERT_1>' '"
        da_GetBridges = New OleDb.OleDbDataAdapter(Q_String, Br_Con)
        cb_GetBridges = New OleDb.OleDbCommandBuilder(da_GetBridges)
```

```
'da_GetBridges.UpdateCommand = New OleDb.OleDbCommand("Update Tab_Name")

                da_GetBridges.Fill(dt_GetBridges)
                CURBridge_TxtBx.Text = dt_GetBridges.Rows.Count
                Cur_Bridge_Ct = CInt(dt_GetBridges.Rows.Count)
                Sum_Br_Count = Sum_Br_Count + Cur_Bridge_Ct
                Tot_Bridge_Ct.Text = Sum_Br_Count
                Me.Refresh()
                'For W = 1 To 1000000

                'Next

                If dt_GetBridges.Rows.Count > 0 Then
                   For BD = 0 To (dt_GetBridges.Rows.Count - 1)
                      RCount = RCount + 1
 Br_Record(RCount, 0) = dt_GetBridges.Rows(BD)("BR_NUM").ToString
'Br_Record(RCount, 1) = dt_GetBridges.Rows(BD)("NO_SPANS").ToString
'Br_Record(RCount, 2) = dt_GetBridges.Rows(BD)("DECK_WIDTH").ToString
 Br_Record(RCount, 3) = dt_GetBridges.Rows(BD)("LATITUDE").ToString
 Br_Record(RCount, 4) = dt_GetBridges.Rows(BD)("LONGITUDE").ToString
'Br_Record(RCount, 5) = dt_GetBridges.Rows(BD)("VERT_1").ToString
 SRcount = CStr(RCount)
    'MsgBox("Current Number of Bridges Found is " & SRcount)
 StLatDegMin = Microsoft.VisualBasic.Left(Br_Record(RCount, 3), 4)
 StLongDegMin = Microsoft.VisualBasic.Left(Br_Record(RCount, 4), 4)
 StLatSec = Microsoft.VisualBasic.Right(Br_Record(RCount, 3), 2)
 StLongSec = Microsoft.VisualBasic.Right(Br_Record(RCount, 4), 2)
                      DecLatMin = CDec(StLatSec)
                      DecLongMin = CDec(StLongSec)


                      DecLatMin = (DecLatMin * 100) / 60
                      DecLongMin = (DecLongMin * 100) / 60
                      Tr_DMIN_LONG = Math.Truncate(DecLongMin)
                      Tr_DMIN_Lat = Math.Truncate(DecLatMin)
                      If Tr_DMIN_LONG < 10 Then
                         StLongDecMin = Z & CStr(Tr_DMIN_LONG)
                      Else : StLongDecMin = CStr(Tr_DMIN_LONG)
                      End If
                      If Tr_DMIN_Lat < 10 Then
                         StLatDecMin = Z & CStr(Tr_DMIN_Lat)
                      Else : StLatDecMin = CStr(Tr_DMIN_Lat)
                      End If
                      'StLongDecMin = CStr(Tr_DMIN_LONG)
                      Br_Record(RCount, 3) = StLatDegMin & StLatDecMin
                     Br_Record(RCount, 4) = StLongDegMin & StLongDecMin

                   Next

                End If
                dt_GetBridges.Clear()
                da_GetBridges.Dispose()
                cb_GetBridges.Dispose()

             Next
```

```
            'Br_ID, Br_Long, Br_Lat, Br_Vert, Br_DeWidth, Br_NoSpans
            Dim BrFile_R As New System.IO.StreamWriter(Tab_Name, True)
            BrFile_R.Write(Fst_MP_Lat)
            BrFile_R.Write(Fst_MP_Long)
            For B = 1 To RCount
               Br_ID = Br_Record(B, 0)
               'Br_NoSpans = Br_Record(B, 1)
               'Br_DeWidth = Br_Record(B, 2)
               Br_Long = Br_Record(B, 3)
               Br_Lat = Br_Record(B, 4)
               'Br_Vert = Br_Record(B, 5)
               BrFile_R.Write("B")
               'BrFile_R.Write(Br_ID)
               'BrFile_R.Write("LL")
               'BrFile_R.WriteLine(Br_NoSpans)
               'BrFile_R.WriteLine(Br_DeWidth)

               BrFile_R.Write(Br_Long)
               BrFile_R.Write(Br_Lat)
               'BrFile_R.WriteLine(Br_Vert)
               'BrFile_R.WriteLine(" ")
            Next
            BrFile_R.Write("X")
            BrFile_R.Write(Lst_MP_Lat)
            BrFile_R.WriteLine(Lst_MP_Long)
            BrFile_R.Close()
            BrFile_R.Dispose()
            'End Using ' Data File

            'End If


            dtMilepost.Clear()

            daMilepost.Dispose()
            cbMilePost.Dispose()
         Next       ' This is end of Plow_Routes For (RC) -- Next Set
      End Using ' Database Connection
      Br_Con.Close()
      Br_Con.Dispose()
MsgBox("District Routes Completed -- Enter New District Route Count to Continue")
   End Sub

Private Sub btnQuit_Click_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnQuit.Click
      Me.Close()
   End Sub
```

Conversion Module:

```
Module Decimal_to_DMS
   Public Function DectoDMS(ByVal stDecDegIn As Integer) As String
      Dim Dec_Deg, Fract_Deg, Fract_Min, Min, Sec As Decimal
      Dim Degree As Int32
      Dim Minset, secset, D_Deg_Trunc, Min_Trunc, Sec_Trunc As Integer
      Dim stDecDegTrunc, stMinTrunc, stSecTrunc, StDMS, zero As String
```

```
        'This Module Converts Longitude or Latitude for Decimal Degrees to
        'Degree:Minute:Second format
        Minset = 0
        secset = 0
        zero = CStr(Minset)
        Degree = CInt(stDecDegIn)
        Dec_Deg = Degree / 10000
        D_Deg_Trunc = Math.Truncate(Dec_Deg)
        Fract_Deg = Dec_Deg - D_Deg_Trunc
        Min = Fract_Deg * 60.0
        Min_Trunc = Math.Truncate(Min)
        If Min_Trunc < 10 Then
           stMinTrunc = zero & CStr(Min_Trunc)
        Else : stMinTrunc = CStr(Min_Trunc)
        End If
        Fract_Min = Min - Min_Trunc
        Sec = Fract_Min * 60 '
        Sec_Trunc = Math.Truncate(Sec)
        If Sec_Trunc < 10 Then
           stSecTrunc = zero & CStr(Sec_Trunc)
        Else : stSecTrunc = CStr(Sec_Trunc)
        End If
        stDecDegTrunc = CStr(D_Deg_Trunc)

StDMS = """ & Microsoft.VisualBasic.Left(stDecDegTrunc, 2) & Microsoft.VisualBasic.Left(stMinTrunc, 2) &
Microsoft.VisualBasic.Left(stSecTrunc, 2) & """
        Return (StDMS)
    End Function

End Module
```

Appendix B: Dynamic C Listing of the IBIAS Software that is Employed in the On-Board Impact Avoidance Controller

Code “StLouis21Nov.c”

```
#class auto

#use "gps.lib"

#memmap xmem

#define DIGOUTCONFIG            0x02

#use "gltouchscreen.lib"                                        //

#use "tsCustKeyboard.lib" // custom keyset

#use "Terminal12.lib"                   // Good size font for the text entry box

#use "Terminal9.lib"

#define MAX_SENTENCE 100

#define CINBUFSIZE 127

#define COUTBUFSIZE 127

#ximport "samples\xmem\SL_Route_1.txt" t01

#define lDist 0.4

#define rDist 0.2

char buffer[1024], fin[1024];

int final[1024]; //, mov;

unsigned  long userX;

unsigned  long userY;

unsigned  long vkbX;

GPSPosition current_pos, tempvar, ar[2];

////////////////////////////////////////////////////////////////////////////////

long relate(int ip)

{

        fontInfo fi14x16,fi10x12,fi;

  glXFontInit(&fi, 17, 35, 32, 127, Font17x35);

   switch(ip)
```

```
  {
      case 1 :
       printf("\n in the case\n");
       return t01;
       break;
  }
}
////////////////////////////////////////////////////////////////////////////
void msDelay(unsigned int delay)
{
        auto unsigned long done_time;
        done_time = MS_TIMER + delay;
        while( (long) (MS_TIMER - done_time) < 0 );
}
////////////////////////////////////////////////////////////////////////////
GPSPosition moving(void)
{
        char sentence[MAX_SENTENCE];
        int input_char, count;
        int string_pos;
        char dir_string[2];
        float distance;
        //calculate distance from known coordinates
        GPSPosition arr;
        serCopen(4800);
        string_pos = 0;
        dir_string[1] = 0;
        count =0;
```

```
        //receive and parse GPS data
while(1)
//for(count = 0; count< 2; count++)
        {
                input_char = serCgetc();
        if(input_char == '\r' || input_char == '\n')
                {
                        sentence[string_pos] = 0; //add null
                        if(gps_get_position(&current_pos, sentence) == 0)
                        {
                                dir_string[0] = current_pos.lat_direction;
         //                     printf("Latitude%d: %d %f' %s\n",count,
         //                             current_pos.lat_degrees, current_pos.lat_minutes,
         //                             dir_string);
                                dir_string[0] = current_pos.lon_direction;
         //                     printf("Longitude%d: %d %f' %s\n",count,
         //                             current_pos.lon_degrees, current_pos.lon_minutes,
         //                             dir_string);
          arr.lat_degrees = current_pos.lat_degrees;
          arr.lat_minutes = current_pos.lat_minutes;
          arr.lon_degrees = current_pos.lon_degrees;
          arr.lon_minutes = current_pos.lon_minutes;
        //if(count==0)
          //msDelay(1000);
         count++;
                        }
                        string_pos = 0;
                }
```

```
                else if(input_char > 0)
                {
                        sentence[string_pos] = input_char;
                        string_pos++;
                        if(string_pos == MAX_SENTENCE)
                                string_pos = 0;     //reset string if too large
                }
        //printf("count = %d", count);
    if(count == 1)
        break;
    //msDelay(3000);
  }
        return(arr);
}
////////////////////////////////////////////////////////////////////////////
int changeInDist()
{
        float dist;
  int retn;
  retn =0;
        ar[0] = moving();
        msDelay(1000);
 /*
        printf("\n ar[0] lat deg = %d\n", ar[0].lat_degrees);
        printf("\n ar[0] lat min = %f\n", ar[0].lat_minutes);
        printf("\n ar[0] lon deg = %d\n", ar[0].lon_degrees);
        printf("\n ar[0] lon min = %f\n", ar[0].lon_minutes);
 */
```

```
            ar[1] = moving();
 /*
            printf("\n ar[1] lat deg = %d\n", ar[1].lat_degrees);


            printf("\n ar[1] lat min = %f\n", ar[1].lat_minutes);
            printf("\n ar[1] lon deg = %d\n", ar[1].lon_degrees);
            printf("\n ar[1] lon min = %f\n", ar[1].lon_minutes);
 */
            dist = gps_ground_distance(&ar[0],&ar[1]);
            //printf("\n\ndistance = %f\n\n", dist);
            if(fabs(dist) > 0.002)
        retn = 1;
  /// cHANGES MADE FOR STATIONARY RUN. PLEASE CHANGE THIS B4 ACTUAL RUN - RAVI 9/19/07
                        //printf("\nretn = %d", retn);
   retn = 1;
   return(retn);
}
////////////////////////////////////////////////////////////////////////////////
void lowerbox(void)
{
  int i, delay,j, mov;
  fontInfo fi14x16,fi10x12,fi;
  brdInit();
  digOutConfig(DIGOUTCONFIG);
  //printf("In side lowerbox 1");
 //glInit();
// Turn on the Backlight, setup the contrast.
            glBackLight(1);
```

```
        // Set the Contrast
        glSetContrast(255);
        // Initialize the fonts
glXFontInit ( &fi14x16,14,16,0x20,0x7E,Terminal12 );
glXFontInit ( &fi10x12,10,12,0x20,0x7E,Terminal9 );
glXFontInit(&fi, 17, 35, 32, 127, Font17x35);
glBlock(50,15,225,100);
//mov = 1;
        mov = changeInDist();
//printf("value of mov = %d", mov);
if(mov == 1)
{
do
{
  digOut(1,0);
  costate
        {
   //     printf("In lowerbox costate");
        glSetBrushType(PIXXOR);
    glPrintf(100,25,&fi,"Lowering");
    glPrintf(135,65,&fi,"Box");
    waitfor (DelayMs(500));
    for(i=0; i < 3; i++)
    {
      for(delay = 0; delay < 8000; delay++);
       buzzer(1);
                                        for(delay = 0; delay < 10000; delay++);
      buzzer(0);
```

```
        //msDelay(100);
                                }
                        }
   j = digIn(12);
   //printf("input 12 = %d", j);
   }while(digIn(12)==1);
   }digOut(1,1);
}
////////////////////////////////////////////////////////////////////////////
void raisebox(void)
{
   int i, delay, mov, x, co;
   fontInfo fi14x16,fi10x12,fi;
   brdInit();
   digOutConfig(DIGOUTCONFIG);
            // Turn on the Backlight, setup the contrast.
            glBackLight(1);
            // Set the Contrast
            glSetContrast(255);
            // Initialize the fonts
            glXFontInit ( &fi14x16,14,16,0x20,0x7E,Terminal12 );
            glXFontInit ( &fi10x12,10,12,0x20,0x7E,Terminal9 );
   glXFontInit(&fi, 17, 35, 32, 127, Font17x35);
   glBlock(50,15,225,100);
   //mov = 1;
   mov = changeInDist();
   //printf("in raise");
   if (mov == 1)
```

```
{
x = 1;
co = 0;
do
{
        digOut(2,0);
  costate
        {
        glSetBrushType(PIXXOR);
    glPrintf(100,25,&fi,"Raising");
    glPrintf(135,65,&fi,"Box");
    waitfor (DelayMs(500));
    printf("\n in rasiebox function");
    for(i=0; i < 3; i++)
    {
        for(delay = 0; delay < 2000; delay++);
        buzzer(1);
        for(delay = 0; delay < 3000; delay++);
        buzzer(0);
    //msDelay(100);
                                }
    if(digIn == 1 || co >5)
    {
        x = 0;
    }
  co++;
  printf("\n count = %d", co);
                    }
```

```
   }while(x);
  } digOut(2,1);
}
////////////////////////////////////////////////////////////////////////////
int getGPS()
{
        char sentence[MAX_SENTENCE];
        int input_char, count, retn;
        int string_pos;
        char dir_string[2];
        float distance;
        serCopen(4800);
        string_pos = 0;
        dir_string[1] = 0;
        retn =0;
        //receive and parse GPS data
        input_char = serCgetc();
        //printf("input = %d",input_char);
         if(input_char == '\r' || input_char == '\n')
                {
                        sentence[string_pos] = 0; //add null
                  //    printf("%s\n", sentence);
                        if(gps_get_position(&current_pos, sentence) == 0)
                        {
        dir_string[0] = current_pos.lat_direction;
                printf("Latitude: %d %f' %s\n",current_pos.lat_degrees, current_pos.lat_minutes,dir_string);
                dir_string[0] = current_pos.lon_direction;
                //    printf("Longitude: %d %f' %s\n",
```

```
                        //          current_pos.lon_degrees, current_pos.lon_minutes,
                        //          dir_string);
        retn =0;
            return (retn);
                        }
string_pos = 0;
                        }
            else
      retn = 1;
 return(retn);
}
////////////////////////////////////////////////////////////////////////////
GPSPosition getposition()
{
            char sentence[MAX_SENTENCE];
            int input_char, count;
            int string_pos;
            char dir_string[2];
   float distance;
   serCopen(4800);
   string_pos = 0;
            dir_string[1] = 0;
            //receive and parse GPS data
            while(1)
            {
                        input_char = serCgetc();
     //printf("input = %d",input_char);
     if(input_char == '\r' || input_char == '\n')
```

```
                {
                        sentence[string_pos] = 0; //add null
                 //     printf("%s\n", sentence);
                        if(gps_get_position(&current_pos, sentence) == 0)
                        {
        dir_string[0] = current_pos.lat_direction;
                         //printf("Latitude: %d %f' %s\n",
                        //      current_pos.lat_degrees, current_pos.lat_minutes,
                        //      dir_string);
                        dir_string[0] = current_pos.lon_direction;
                         //     printf("Longitude: %d %f' %s\n",
                          //    current_pos.lon_degrees, current_pos.lon_minutes,
                         //     dir_string);
        return (current_pos);
                        }
                        string_pos = 0;
                }
                else if(input_char > 0)
                {
                        sentence[string_pos] = input_char;
                        string_pos++;
                        if(string_pos == MAX_SENTENCE)
                                string_pos = 0;     //reset string if too large
                }
  }
}
```

```
////////////////////////////////////////////////////////////////////////////

GPSPosition extract(int off)

{

  char array[12];

  int i, cnt;

  cnt = 1;

  for(i = 0; i<12; i++)

  {

   array[i] = buffer[off+cnt]-'0';

   cnt++;

  }

            tempvar.lat_degrees = array[0]*10 + array[1];

           tempvar.lat_minutes = (array[2]*10 + array[3]) + (array[4]*10 + array[5])*0.01;

           tempvar.lon_degrees = array[6]*10 + array[7];

           tempvar.lon_minutes = array[8]*10 + array[9] + (array[10]*10 + array[11])*0.01;

           tempvar.lat_direction = 'N';

            tempvar.lon_direction = 'W';

            return(tempvar);

}

////////////////////////////////////////////////////////////////////////////

void kprin(GPSPosition loc)

{

           int lat, lon,lat1, lon1;

           float flat, flon;

            lat = loc.lat_degrees;

            lon = loc.lon_degrees;

           flat = loc.lat_minutes;

            flon = loc.lon_minutes;
```

```
        flat = flat *100;
        flon = flon*100;
        lat1 = (int)flat;
        lon1 = (int)flon;
  if(lat < 10)
        printf("0%d",lat);
  else
        printf("%d", lat);
  if(lat1 <1000)
        printf("0%d",lat1);
  else
        printf("%d", lat1);
  if(lon < 10)
        printf("0%d", lon);
  else
        printf("%d", lon);
  if(lon1 <1000)
        printf("0%d",lon1);
  else
        printf("%d", lon1);
}
////////////////////////////////////////////////////////////////////////////////
void main()
{
// Setup some variables for extracting data out of the keyboard.
  GPSPosition last,start, currLoc, currLoc1,nextBrg, newBrg, currBrg, cbr;
  GPSPosition rc1;
        int btn,iVal,x, nextPrn;
```

```
  int counter, endLoc, endLat, endLon;
  int final, length;
  int offse, loop, inLoop, outLoop;
  //int mov;
  int manualFlag, raiseFlag, noff, i;
  int gpsret,delay;
  long fptr;
        float fVal;
  float distance, distance1,newDist, dist, nd, y;
  int offPass, ofx, ignore;
  ////////////new dist calc var///////////
  float latDiff, lonDiff;
        char sVal[100],pVal[100];
  unsigned wKey;
  int keyflag, keypad_active, done, tmp, alen;
  int newD;
  //char buffer[512];
        fontInfo fi14x16,fi10x12,fi;
  brdInit();
        glInit();
  keyInit();
  keypadDef();
 // Turn on the Backlight, setup the contrast.
glBackLight(1);
        // Set the Contrast
glSetContrast(255);
        // Initialize the fonts
glXFontInit ( &fi14x16,14,16,0x20,0x7E,Terminal12 );
```

```
glXFontInit ( &fi10x12,10,12,0x20,0x7E,Terminal9 );
        // Initialize the gltouchscreen.lib library
userX = btnInit( (int)100 );
        //  Initialize the Virtual keyboard keyset.
vkbX = tscVKBInit( (int)62, &fi14x16,&fi10x12);
manualFlag = 0;
offse = 12;
//mov = moving();
loop =1;
 currLoc.lat_degrees = 0;
  //gpsret = 0;
        //  Set the Button Attributes (since the function is non-blocking,
        //  it will need to be repeatedly called until a 1 is returned).
        while (!tscVKBAttributes(vkbX,1,500,100,1));
                // Create 5 buttons to be displayed and used on the LCD
        btnCreateText(userX,1,40,10,230,80,1,1,&fi10x12,"ENTER A\nROUTE NUMBER");
  btnCreateText(userX,2,40,150,230,80,1,1,&fi10x12,"ENTER THE \n NEW BRIDGE");
        btnAttributes(userX,1,0,0,0,1);
        btnAttributes(userX,2,0,0,0,1);
  btnMsgBox(0,0,320,240,NULL,"",1,0);     // will display a border
  while (!btnDisplayLevel(userX,1)); // Will display all buttons of Level 1
 // btnDisplayLevel(userX,1);
  fVal = 0;
        //lVal = 0;
        iVal = 0;
        sprintf(sVal,"");
        sprintf(pVal,"");
```

```
final = 1;
while (final)
{
        costate
                {
                // Wait for a button to be pressed
                 waitfor ( (          btn = btnGet(userX) ) >= 0 );
    //btn = btnGet(userX);
                        switch (btn)
                        {
                case 1:
        // Display the Virtual keyboard for use with ints
        waitfor ( tscVKBGetInt(vkbX,&iVal,-32000,32000,6,&fi14x16,&fi10x12,
                          "ENTER ROUTE") );
                printf ( "Route Number = %d\n",iVal );
         final =0;
         break;
    /*
      case 2:
         waitfor ( tscVKBGetInt(vkbX,&iVal,-32000,32000,6,&fi14x16,&fi10x12,        "ENTER ROUTE") );
         printf("yes");
         break;
         */
    }
                }
}
fptr = relate(iVal);
xmem2root(&length,fptr,sizeof(long));
```

```
  //if(length>512)  // Preventive measure to conrol buffer overflow
            //          length=512;
  xmem2root(buffer,fptr+4,(int)length);
/* To check the contents of Buffer
  for(x=0;x<length;x++)
  {
                  printf("%c",buffer[(int)x]);
  }
*/
  counter =0;
//   printf("the length = %d", length);
  for(x=0;x<length; x++)
  {
          if(buffer[x] == 'B')
          counter++;
  }
  printf("\nnum of bridges = %d", counter);
  start = extract(-1);
  for(x=0;x<length; x++)
  {
          if(buffer[x] == 'X')
          endLoc = x;
  }
  last = extract(endLoc);
  alen = 0;
  tmp =12;
 //  printf(" actual length = %d", alen);
 ///  for(x =0; x< alen; x++)
```

```
//          printf("%d", fin[x]);
  //printf("\nlast mile post starts at = %d", endLoc);
  //printf("buffer [endLoc+1] =%d ", buffer[endLoc+1]-'0');
/*
  printf("\nbridge lat deg = %d\n", start.lat_degrees);
  printf("bridge lat min = %f\n", start.lat_minutes);
  printf("bridge lon deg = %d\n", start.lon_degrees);
  printf("bridge lon min = %f\n", start.lon_minutes);
  */
  //printf("Wait for a button to be pressed");
          currLoc.lat_degrees = 0;
          currLoc.lat_minutes = 0.0;
 // cbr.lat_degrees = 0;
 //          cbr.lat_minutes = 0.0;
 //mov = 1;
 offPass = 0;
 raiseFlag = 0;
 ignore = 0;
          while(loop)
          {
          currLoc = getposition();

          //offse = 12;
          costate
          {
                    keyProcess();
                    waitfor(DelayMs(10));
          }
```

```
costate
{
//printf("Stage 2 begins");
keypadDef();
keyflag = 0;
done = FALSE;
while(!done)
{
        waitfor(wKey = keyGet());
kprin(currLoc);
        done = TRUE;
}
}
costate
{
if(digIn(12) == 1)
        {
offse = 12;
//currLoc1 = getposition();
for(i = 0; i < counter; i++)
                {
   if(offse == offPass)
   {
        offse = offse + 13;
     i = i + 1;
   }
   currBrg = extract(offse);
/*  printf("\n");
```

```
        kprin(currLoc);
        printf("\n");
        kprin(currBrg);
     */
    //    printf("offPass = %d", offPass);
                        distance  = gps_ground_distance(&currLoc, &currBrg);
                    //    printf("distance = %f\n", distance);
                        if(distance < 0.4)// original value 0.4
                        {
        //   printf("\nloc  = %d", offse);
           offPass = offse;
           cbr.lat_degrees = currBrg.lat_degrees;
           cbr.lat_minutes = currBrg.lat_minutes;
           cbr.lon_degrees = currBrg.lon_degrees;
           cbr.lon_minutes = currBrg.lon_minutes;
           lowerbox();
                        break;
                        }
                offse = offse + 13;
    }
     ignore = 0;
 }
if(digIn(13) == 0)
{
    offse = 12;
  // while(offse < endLoc)
  //currLoc1 = getposition();
  // {
```

```
 for(i=0; i< counter; i++)
 {
    distance1 = 0.0;
            cbr = extract(offPass);
    //distance1  = gps_ground_distance(&currLoc, &currBrg);
    distance1 =   gps_ground_distance(&cbr, &currLoc);
 //    printf("\nd = %f ",distance1);
            printf("\n");
    kprin(currLoc);
//   printf("\n--->");
  //   kprin(cbr);
    if(distance1 < 0.1 ) // original value 0.1
                        {
                                raiseFlag = 1;
                        }
    printf("\nraiseflag = %d", raiseFlag);
                        if(distance1 > 0.1 && raiseFlag == 1)
                        {
                        x = 12;
          y = 100.00;
          for(i =0; i< counter; i++)
          {
              if( x != offPass)
             {
              rc1 = extract(x);
               nd =      gps_ground_distance(&rc1, &currLoc);
               if(nd< y)
               {
```

```
                    y = nd;
                ofx = x;
              }
            }
            x = x+13;
          }
          if(y < 0.3)
          {
           offPass = ofx;
           printf("new bridge in Sight");
          }
          if( y > 0.3)
          {
            printf("Should raise now");
            printf("Ignore = %d", ignore);
            if(ignore != 1)
            {
             raisebox();
              ignore = 1;
              raiseFlag = 0;
             break;
            }
          }
                    }
      offse = offse + 13;
                    }
// }
            }
```

```
        }
        }
}
```