Fall 2017

Advancing Intersection Management by Utilizing Cost Effective Intelligent Vehicle Concepts and Vehicle-to-Infrastructure Communication Techniques

Bernard O. Ibru

ADVANCING INTERSECTION MANAGEMENT BY UTILIZING COST EFFECTIVE INTELLIGENT VEHICLE CONCEPTS AND VEHICLE-TO-INFRASTRUCTURE COMMUNICATION TECHNIQUES

by

BERNARD IBRU

(Under the direction of Valentin Soloiu)

ABSTRACT

Intelligent Transport Systems (ITS) is a growing field of research which focuses on the alleviation of traffic congestion and road accidents caused by miscommunication or confusion of human drivers. Intelligent Intersection Management is a subdivision of ITS which focuses on the seamless management of vehicles arriving at, traversing and exiting intersections to prevent congestion and collision within or around the intersection. This research sought to develop a cost effective method of implementing wireless Vehicle-to-Infrastructure (V2I) communication based Intelligent Intersection Management, by employing the use of 1:4.5 scale version autonomous vehicle prototypes, on a similarly scaled four-way intersection. This was accomplished by employing Robot Operating System (ROS) on a single board computer platform which communicated comma-separated integers via Zigbee XBee radio transceivers to prioritize navigation of vehicles arriving at the intersection based on arrival time and the vehicles' projected paths.

INDEX WORDS: Intelligent, Autonomous, Vehicles, Self-driving cars, Intelligent Transport Systems, Wireless communication, Vehicle-to-Infrastructure communication, V2I, Robotic navigation

ADVANCING INTERSECTION MANAGEMENT BY UTILIZING COST EFFECTIVE INTELLIGENT VEHICLE CONCEPTS AND VEHICLE-TO-INFRASTRUCTURE COMMUNICATION TECHNIQUES

by

BERNARD IBRU

B.S., Georgia Institute of Technology, 2012

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial

Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

ADVANCING INTERSECTION MANAGEMENT BY UTILIZING COST EFFECTIVE INTELLIGENT VEHICLE CONCEPTS AND VEHICLE-TO-INFRASTRUCTURE COMMUNICATION TECHNIQUES

by

BERNARD IBRU

Major Professor:	Valentin Soloiu
Committee:	Biswanath Samanta
	Anoop Desai

Electronic Version Approved:
December 2017

DEDICATION

To my mother, late father and my siblings for all the support, moral and otherwise over the years.

To my girlfriend, Anne for her support throughout my time in higher education.

ACKNOWLEDGEMENTS

A thesis cannot be completed without extraordinary support. The author would like to acknowledge the efforts of all the professors that have taken their time to clarify complex scenarios that proved difficult for the author to decipher. The writer would also like to give a special thanks to Professor Soloiu and the colleagues in the Engines lab for providing support and ideas necessary for this thesis.

Appreciation is particularly extended to Tyler Naes for teaching the basics of ROS programming, Charvi Popat for helping with data collection and brainstorming ideas, Dr Vlcek for instructing the author on proper formatting styles and data collection methodology, and also to the members of TMAE 7530, especially Thomas Beyerl and Emile Maroha who provided moral support when needed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 The Need for Intelligent Intersection Management

In recent years, automakers have begun to focus research efforts into various forms or intelligent vehicles, thereby breaking the over hundred-year primitive interface between driver and car. This novel and expensive area of research was not ventured into on a whim, or on some need to be unnecessarily futuristic, but because technology has advanced enough to improve the standard of living; particularly in vehicle-operator interface.

Each year over 9,520 accident related fatalities occur in intersections within the United States (Subramanian and Lombardo 2007) and some of these accidents occur in intersections when confused, distracted or under-the-influence vehicle operators proceed through the intersection when they do not have the right of way (Peden et al. 2006). A number of these accident are ‘innocent’, and are caused by visual obstructions in the intersection design or when the intersection design is complex and unfamiliar to a new vehicle operator (Yousef and Shatnawi 2010). In some cases however, even with the implementation of traffic control systems, fatalities still occur at said intersections as presented in Figure 1.1.

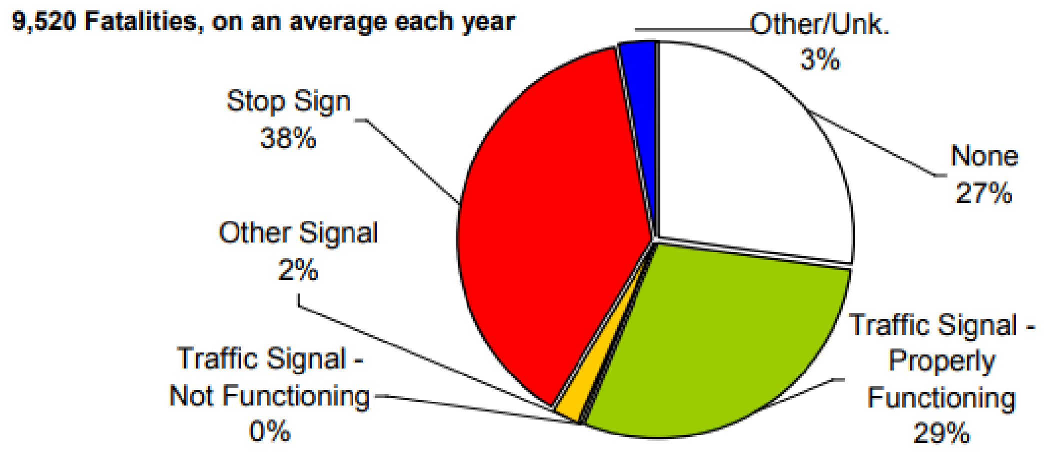


Figure 1.1 Fatal crash statistics based on method of intersection management (Subramanian and Lombardo 2007)

Intelligent vehicle technology, in conjunction with Intelligent Transportation Systems, provide the opportunity to offload some or all of the tasks associated with operating a vehicle for consumer, industrial, and military applications (Banjanovic-Mehmedovic et al. 2016). Aside from the more obvious need to save lives, improving the quality of living by reducing road congestion is another important spur for research into improving Intelligent Transport Systems (Zhu and Ukkusuri 2015). The need for these intelligent systems is glaringly exhibited in the fact that, as shown in Figure 1.1, having no intersection management systems involved less fatal accidents than both widely used conventional intersection management systems (stop signs and traffic lights).

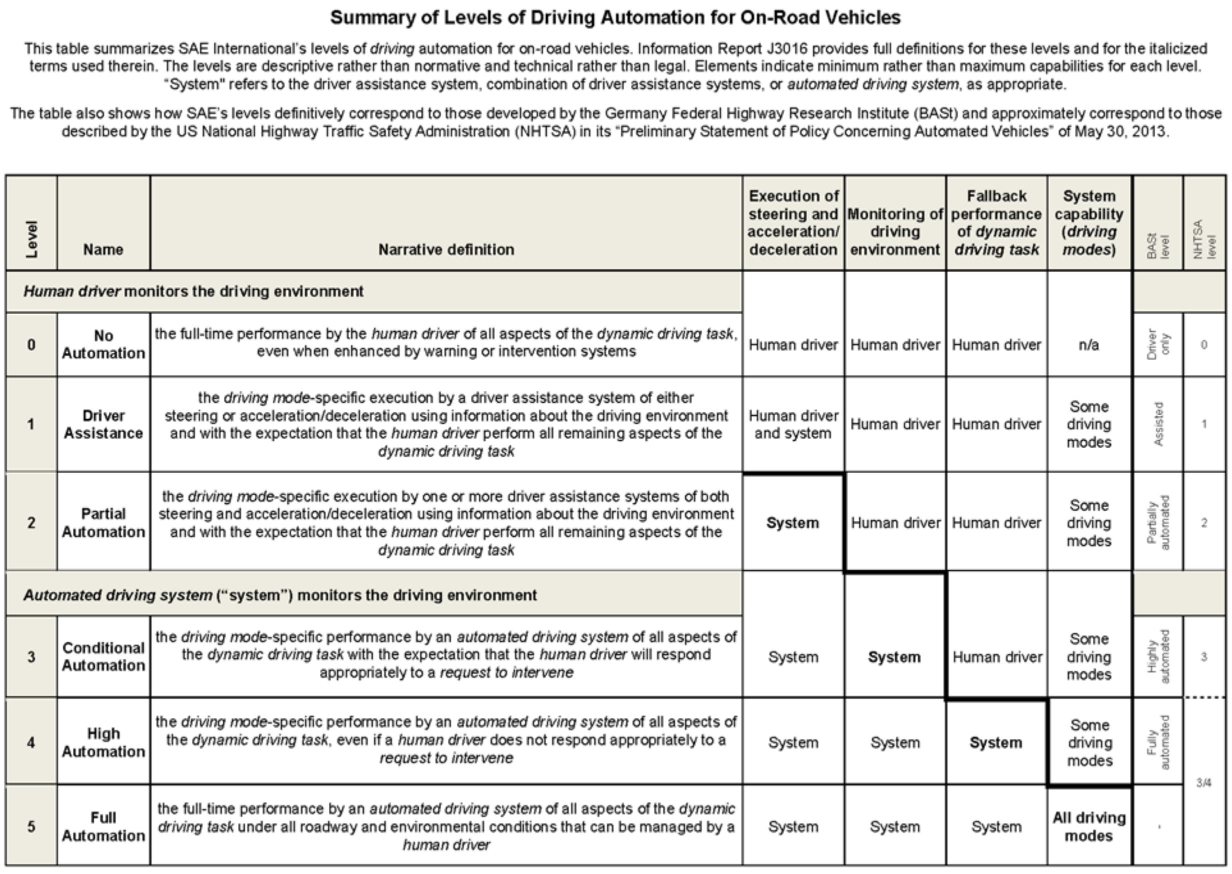
The existing methods for traffic management, surveillance and control are also not adequately efficient in terms of the performance, cost, and the effort needed for maintenance and support. For example, The 2007 Urban Mobility Report estimates total annual cost of congestion for U.S. urban areas at 89.6 billion dollars, the value of 4.5 billion hours of delay and 6.9 billion gallons of excess fuel consumed (Yousef and Shatnawi 2010). In an independent research effort, the traffic engineering department in Jordan estimated that the total cost due to congestion in the year 2007 was around 150 million US$ (Greater Amman Municipality 2007), proving this is a worldwide problem.

As such, numerous methods have been envisioned and developed to alleviate traffic congestion and ensure fast, smooth and safe traffic flow (Barker 1960; Bayraktaroglu 1990; Kamal et al. 2013). Such systems provide the potential to reduce accidents caused by operator neglect or inattention, as well as reduce congestion and delays related to route planning and execution (Yousef and Shatnawi 2010).

Most current traffic light control systems use one of three control approaches: fixed-time, actuated, or adaptive (Zhou et al. 2010). These systems prove to be highly expensive and still pose the issue of poor traffic management during periods of high traffic flow (Dresner 2009). By implementing one of many versions of intelligent intersection routing, which can significantly reduce wait times, these optimizations can be improved at a fraction of the cost.

1.2 SAE Levels of Automation

The term ‘intelligent vehicle’ encompasses any form of driving automation incorporated in a road going vehicle. Levels of automation were suggested by the Society of Automotive Engineers (SAE) to characterize how much of the driving responsibilities are undertaken by the automation system and the human drivers respectively. These levels of automation are presented in Figure 1.2.



Summary of Levels of Driving Automation for On-Road Vehicles

This table summarizes SAE International's levels of *driving* automation for on-road vehicles. Information Report J3016 provides full definitions for these levels and for the italicized terms used therein. The levels are descriptive rather than normative and technical rather than legal. Elements indicate minimum rather than maximum capabilities for each level. "System" refers to the driver assistance system, combination of driver assistance systems, or *automated driving system*, as appropriate.

The table also shows how SAE's levels definitively correspond to those developed by the Germany Federal Highway Research Institute (BASt) and approximately correspond to those described by the US National Highway Traffic Safety Administration (NHTSA) in its "Preliminary Statement of Policy Concerning Automated Vehicles" of May 30, 2013.

Level	Name	Narrative definition	Execution of steering and acceleration/ deceleration	Monitoring of driving environment	Fallback performance of *dynamic driving task*	System capability (*driving modes*)	BASt level	NHTSA level
***Human driver* monitors the driving environment**								
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a	Driver only	0
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes	Assisted	1
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes	Partially automated	2
***Automated driving system* ("system") monitors the driving environment**								
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task* with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes	Highly automated	3
4	High Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes	Fully automated	3/4
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**	-	

Figure 1.2 Summary of levels of on-road vehicle driving automation (Beyerl 2017)

As shown in Figure 1.2, the determination of levels is dependent on whether the human driver or the vehicle’s system is responsible for execution of steering and acceleration/deceleration, monitoring of driving environment, and fallback performance of dynamic driving tasks. Level 0 automation involves a vehicle which is entirely controlled by the human driver without any interference from any of the vehicle’s systems. This level of automation is what had been widespread before the introduction of vehicle autonomy and has proven to be unhelpful to human drivers as they are prone to fatigue, distractions and the occasional bad decision.

Level 1 automation has been present in many vehicles produced in the past few decades and involves having the vehicles' system handle some of the tasks previously left to the driver such as maintaining a desired speed. This feature commonly called cruise control has proven to reduce fatigue in human drivers undertaking multiple hour long drives (Ioannou and Chien 1993). This system however has its drawbacks as the vehicle is still reliant on the human operator's constant monitoring of the driving environment, and in some cases have led to catastrophic results in cases where the driver succumbed to fatigue in especially long drives (Sagberg 1999). To curb this problem, in recent years, vehicle manufacturers have slowly introduced active safety features such as back up sensing, blind spot warning and lane departure warning, which monitor the environment but required the human driver to take action to prevent avoidable accidents. These systems shift into Level 3 autonomy and can begin to be referred to as intelligent vehicles.

Further levels of autonomy include Level 4 and 5 autonomy, which include total or semi total control of the vehicle's driving, as well as monitoring the driving environment and dynamically responding to the pre-recognized environments, adjust for unforeseen circumstances. These vehicles are what can truly be referred to as autonomous vehicles and these are the types of vehicles under development in the Intelligent Vehicle Laboratory as shown in Figure 1.3.

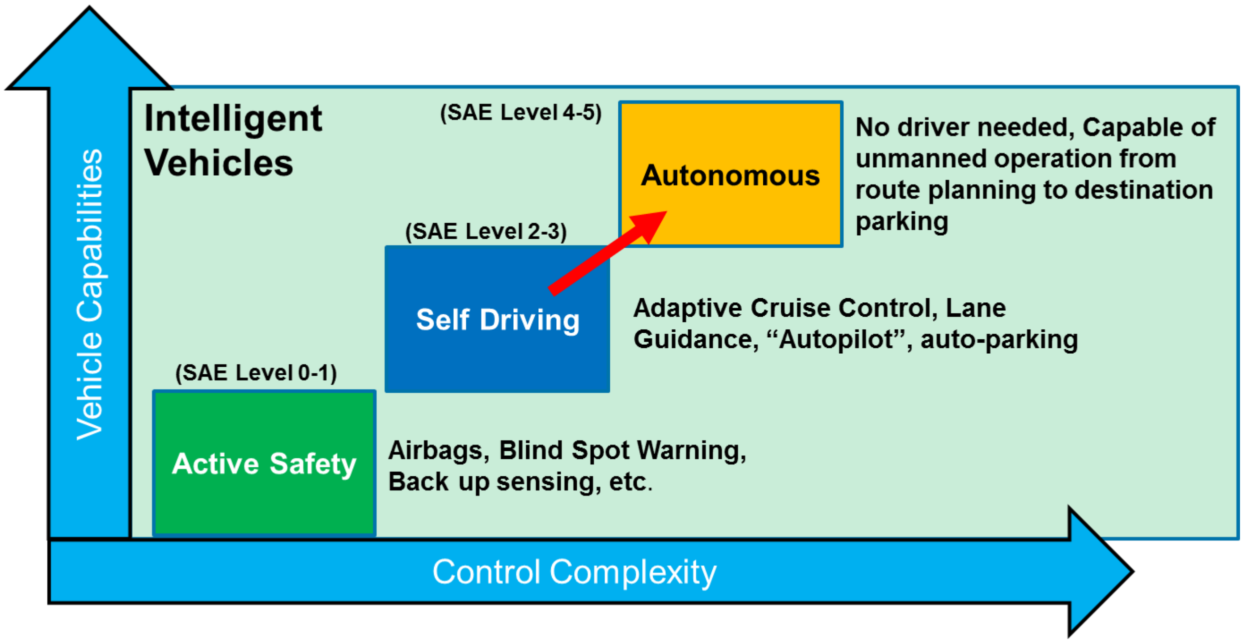

Vehicle Capabilities
Intelligent Vehicles
(SAE Level 4-5)
Autonomous
No driver needed, Capable of unmanned operation from route planning to destination parking
(SAE Level 2-3)
Self Driving
Adaptive Cruise Control, Lane Guidance, "Autopilot", auto-parking
(SAE Level 0-1)
Active Safety
Airbags, Blind Spot Warning, Back up sensing, etc.
Control Complexity

Figure 1.3 Levels of autonomy showing the goals of the intelligent vehicle laboratory (B. Ibru et al. 2016)

Achieving the goal of developing an effective autonomous vehicle system involves two major categories; path planning, which involves developing a preprogrammed path, and path following which involves effectively navigating said preprogrammed path.

1.3 Path Planning and Path Following

Effective Path Planning, which is the backbone for proper autonomous routing of any form, is achieved by utilizing environmental sensors to establish the world around the vehicle. This also includes, vehicle-to-vehicle communication to determine threats and/or relevant information not recognized by the onboard environmental sensors, and in advanced systems, an accurate map in which the aforementioned information would be overlaid to compute an infallible construction of the vehicles local and global world.

Path Following is then incorporated after effectively determining the pre-planned path, and is achieved by monitoring the vehicles' movement using telemetry sensors, wheel encoders and inertial measurement units to properly ascertain the trajectory and pose of the vehicle as well as to correct for external changes to the vehicle which could change its preplanned trajectory. This field of research is being conducted by the second branch of the Intelligent Vehicle Laboratory and will only be touched on lightly in this body of work, as this study is focused on the Path Planning field of research.

1.4 The Research Problem

As previously mentioned, the goal of the Intelligent Vehicle Laboratory is to aid the human driver by offsetting confusing driving scenarios normally unfamiliar to the typical driver. These scenarios include unforeseen obstacles on the driveway, unfamiliar roads, unfamiliar and non-standard intersections, as well as inclement weather or low-light conditions. To demonstrate the advantages of the aforementioned concepts, a prime example of an unfamiliar/non-standard intersection present in Downtown Statesboro which connects S Main St, Brannen St, W Brannen St, Statesboro Rd, Fair Rd and Central St, as shown in Figure 1.4, was chosen.

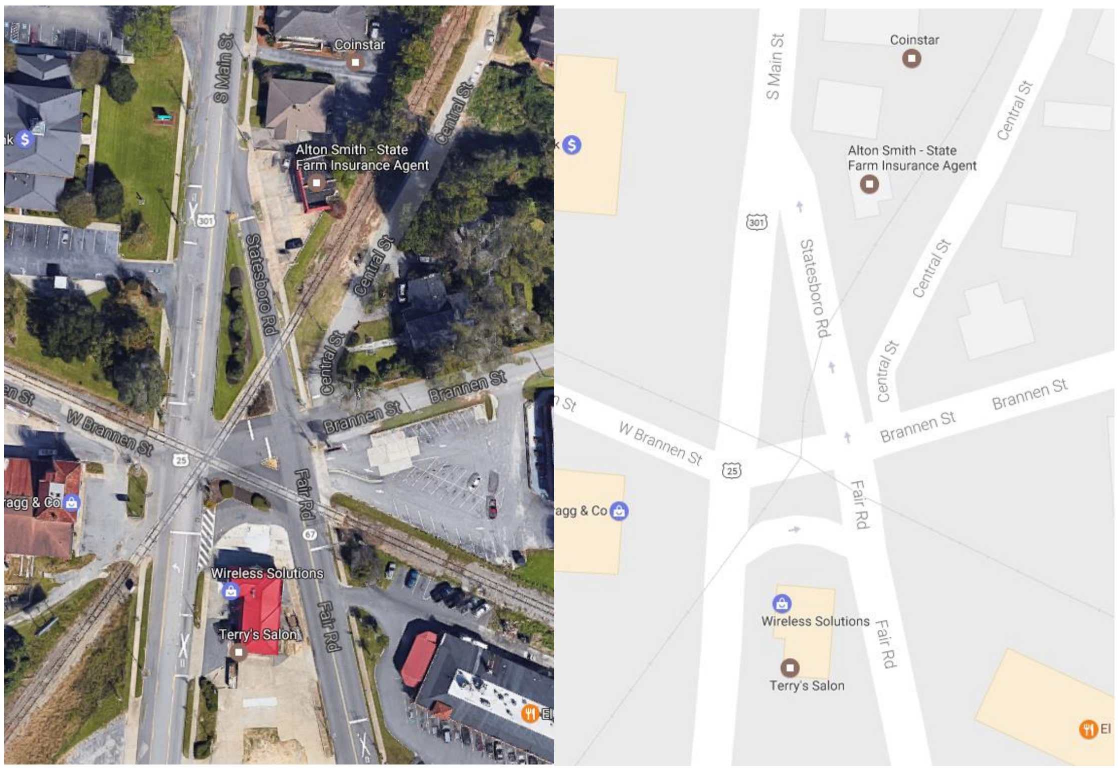


Figure 1.4. Satellite and map view of nonstandard intersection in Downtown Statesboro (Google Maps 2017)

This intersection is composed of five major roads connecting at irregular angles, two railroads and five stop signs. Unlike a more standard intersection, there are no stop signs on S Main St and hence, all other commuters must yield to vehicles navigating this road while recognizing the stop signs and keeping track of road name changes. Navigating the intersection has proven to be difficult for first time travelers, even with the help of a GPS Navigation System.

To compound the problems associated with this intersection, there are areas of the intersection which make it extremely difficult to yield to oncoming traffic. This is due to obstructed visibility issues caused by the shrubbery and acute angles present in some of the turns as shown in Figure 1.5

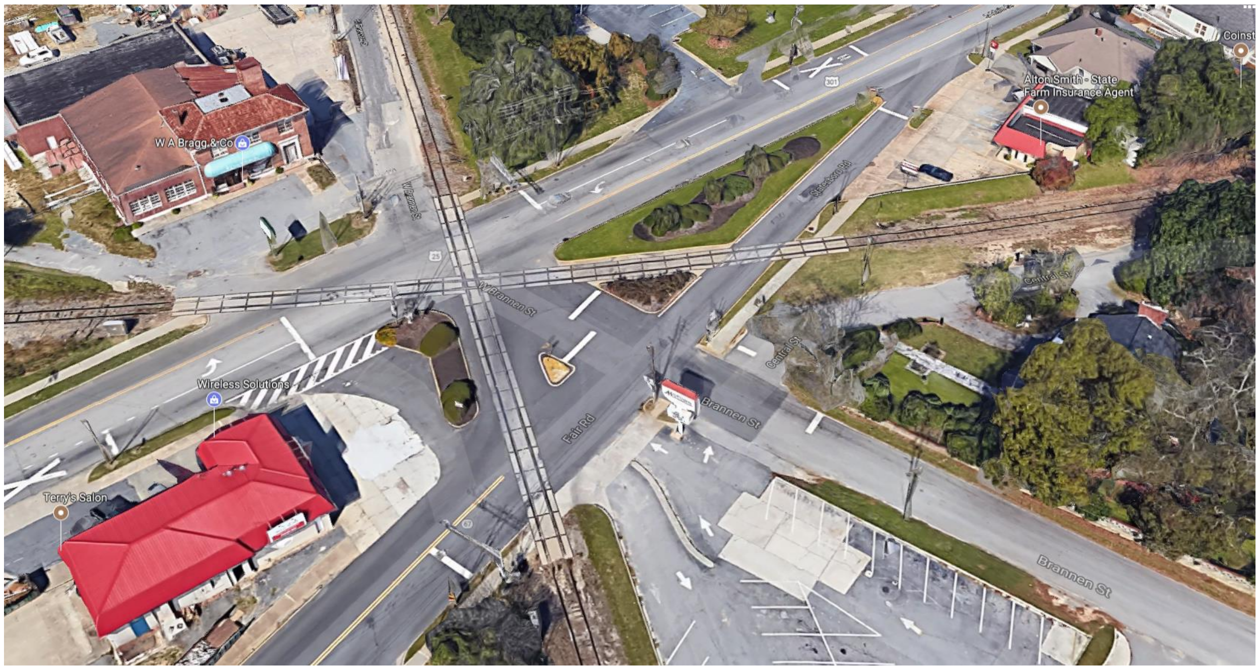


Figure 1.5 Satellite imagery of nonstandard intersection showing multiple blind spots (Google Maps 2017)

Using this intersection as motivation, the author decided that a better means of intersection traffic control needed to be implemented here without incurring the costs usually associated with the implementation of traffic signals. A research hypothesis was developed to try to accomplish this task.

1.5 Hypothesis

If a central means of communication at an intersection is established using cost effective radio transmitters, a number of intelligent vehicles approaching from different directions may seamlessly navigate the intersection, no matter how complex it is, without the need for visual referencing and human intervention.

1.6 Criteria for Success

The goal of this thesis is to develop a communication system using cost effective radio transceivers and a controller, in the form of a single board computer platform, to establish a communication network that would help autonomous vehicles determine order and priority when approaching the intersection. For the purpose of this thesis. The goals will involve three parts:

- communicating with a fully autonomous vehicle equipped with a preprogrammed map of the intersection, capable of navigating the turns using external and internal sensors,
- developing a human interface module for communicating with human drivers of vehicles not equipped with advanced driver assistance systems (ADAS), and
- testing all possible combinations of scenarios possible at a scaled intersection to prove reliability of the system.

The thesis will be deemed successful as long as the central network can effectively communicate with both the autonomous vehicles and the user interface effectively, while assigning priority correctly to all approaching vehicles, based on their time of arrival and projected path.

CHAPTER 2

LITERATURE REVIEW

2.1 Conventional Intersection Management Methods

As stated in the previous chapter, the problem of cost effective intersection management, especially in remote and/or none standard intersections have few, currently implemented solutions. Other aspects are also considered when implementing such solutions, such as congestion history, neighborhood and number vehicles that traverse the intersection. The two most common methods are the stop-signs and traffic lights

2.1.1 Stop Sign Intersection Management

Stop signs are currently the most cost effective intersection management method for intersection management involving immoveable hexagonal markers at the corners of each of the legs of the intersection, signaling for the human driver to come to a complete stop and not proceed until the driver has the right-of-way. The advantages of this method include the ease of installation and the relatively cheap cost of installation which cost about $350 for parts and about $75/hr. overtime for labor amounting to up to $500 per sign in total labor costs lasting a lifetime of 7 years (EmedCo 2017). The main disadvantage of this management method is the fact that decisions of who has the right-of-way rests solely on the human drivers, and their non-verbal agreement, as to who approached the traffic sign first. This leads to occasional miscommunication leading to 63,552 accidents per year costing $420 million in damages (Choi 2010; Federal Highway Administration 2009). Most of these crashes arise from angle type collisions in which one vehicle collides into the side of another vehicle already in the intersection causing a more severe accident (WSDOT 2017).

2.1.2 Traffic Signal Intersection Management

Taking into account the issues realized when implementing standards stop sign intersections, such as the severity of collisions and miscommunication between the human drivers, counties elected to incorporate an automated device which would alternate the right of way interchangeably across the intersections, effectively taking the decision-making away from the vehicle operator. First implemented in the early 20th century, when the increasing number of road vehicle necessitated a need for an automated system, traffic signals have proved to be invaluable and is the current go-to management system, especially at busy intersections (Federal Highway Administration 2009).

The advantage of these systems include accident reduction (accidents that do occur tend to involve rear end collisions as opposed to more sever angle type collision (WSDOT 2017)). A disadvantage of this system however is the cost related in the hardware, implementation and maintenance of the traffic signal. The implementation of traffic costs tax payers between $250,000 and $500,000 to purchase and install a traffic signal. Additionally, electric bills and routine maintenance amount to about $8,000 a year (WSDOT 2017).

Another major disadvantage of this style of intersection management is time. Early traffic signals operated on a time sequence; allowing each branch of the intersection an allotted time for right-of-way. The most notable problem with this system is that there will be instances where a traffic laden branch of the intersection would be given the right-of-way in favor of another branch with no vehicles. This leads to unnecessary increased fuel cost, time delays, and accidents (WSDOT 2017). A more pertinent issue is the fact that when the aforementioned scenario occurs, vehicle operators become tempted to ‘run’ the traffic signal; proceeding when not their right-of-way, thereby having them break the law. In the city of Alpharetta, Georgia, where traffic-control monitoring devices were implemented to reduce the incidence of operators ‘running’ red light, 9,210 violations were recorded at just 7 locations where the devices were implemented, leading to the issuance of 6,758 traffic citations amounting to $342,298 in monetary penalties paid (Osborne 2009).

2.2 Optimization of Intersection Management

To alleviate some of the problems associated with inefficient traffic signal timing, traffic signal conditioning has been envisioned/implemented in some aspect. These optimization efforts range from coordinating the traffic signal timings to match the usual traffic in the intersection as well as timings between neighboring intersections, to reduce the amount of time spent idling during a trip through several intersections.

2.2.1 Optimizing Traffic Signal Timing

To optimize the amount of time each leg of an intersection has the right-of-way, traffic signal coordination is the most commonly used method. Traffic signal coordination occurs when a group of two or more traffic signals work together, so that vehicles moving through the group would make the least number of stops possible. In order for this to happen, each traffic signal in the group must allow a green light for all directions of travel during a fixed time period. In addition, that fixed time period must be the same for each traffic signal in the group. Since each traffic signal in a group runs through all its directions in the same time period, it then becomes possible to "line up" the green lights for one direction. The way the green lights "line up" depends on the distance between traffic signals and the speed of the traffic (WSDOT 2017).

In order to properly implement this concept, several factors must be considered, such as pedestrian crossings; for safety, enough time must be allowed for pedestrians to cross the street from curb to curb walking at a pace of four feet per second. This rule of thumb is referred to as the pedestrian interval and is variable depending on population of elderly pedestrians and/or railroad preemption (WSDOT 2017). Other important factors that must be considered when implementing this level of coordination are cross traffic and left turn signals. Like pedestrian crossing, enough time should be allocated to clear the waiting traffic on the cross street. The heavier the cross traffic, such as experienced near schools, businesses, and other heavy traffic generators, the more time needed to clear them through the intersection, and the less time available for the green light in the "coordinated" direction (WSDOT 2017). Where left-turning traffic is especially heavy and/or the amount of opposing traffic is so heavy that there are not enough gaps in the traffic to safely complete a left-turn; left-turn signals are usually installed. The amount of time for left-turning traffic also limits the time permitted for the "through" traffic flow in the opposite direction

(WSDOT 2017). Owing to these limitations, several methods have been researched to circumvent the method of preprogrammed fixed timing.

2.3 Currently Employed Supplementary Systems to Aid Traffic Signal Timing

In an attempt to alleviate the issues related to traffic signal timing, several technologies have been developed to reduce the amount of unnecessarily wasted time at said intersections. Two relevant example of these detectors are discussed.

2.3.1 In-pavement Detectors

One such method currently in use, is using in-pavement detectors in or under the roadways. Inductive detector loops are the most common type and they consist of sensors buried in the road to detect the presence of traffic waiting at the signal, and thus reduce the amount of right-of-way time given to the empty leg of the intersection (MarshProducts 2000). A timer is frequently used as a default setting in times of low or no traffic, such as late at night. The loop detectors work in a similar fashion to metal detectors and hence smaller vehicles or vehicles with low metal contents may fail to be detected by this system causing them to wait indefinitely unless there is also a default timer incorporated as part of the control system. Examples of such loop detectors are presented in Figure 2.1 (MarshProducts 2000).

These loop detectors are buried directly in the traffic lane and is a continuous run of wire that enters and exits from the same point. The two ends of the loop wire are connected to the loop extension cable, which in turn connects to the vehicle detector. The loop is powered by a detector which causes a magnetic field to form in the loop area. The loop resonates at a constant frequency that the detector monitors. A base frequency is established when there is no vehicle over the loop.

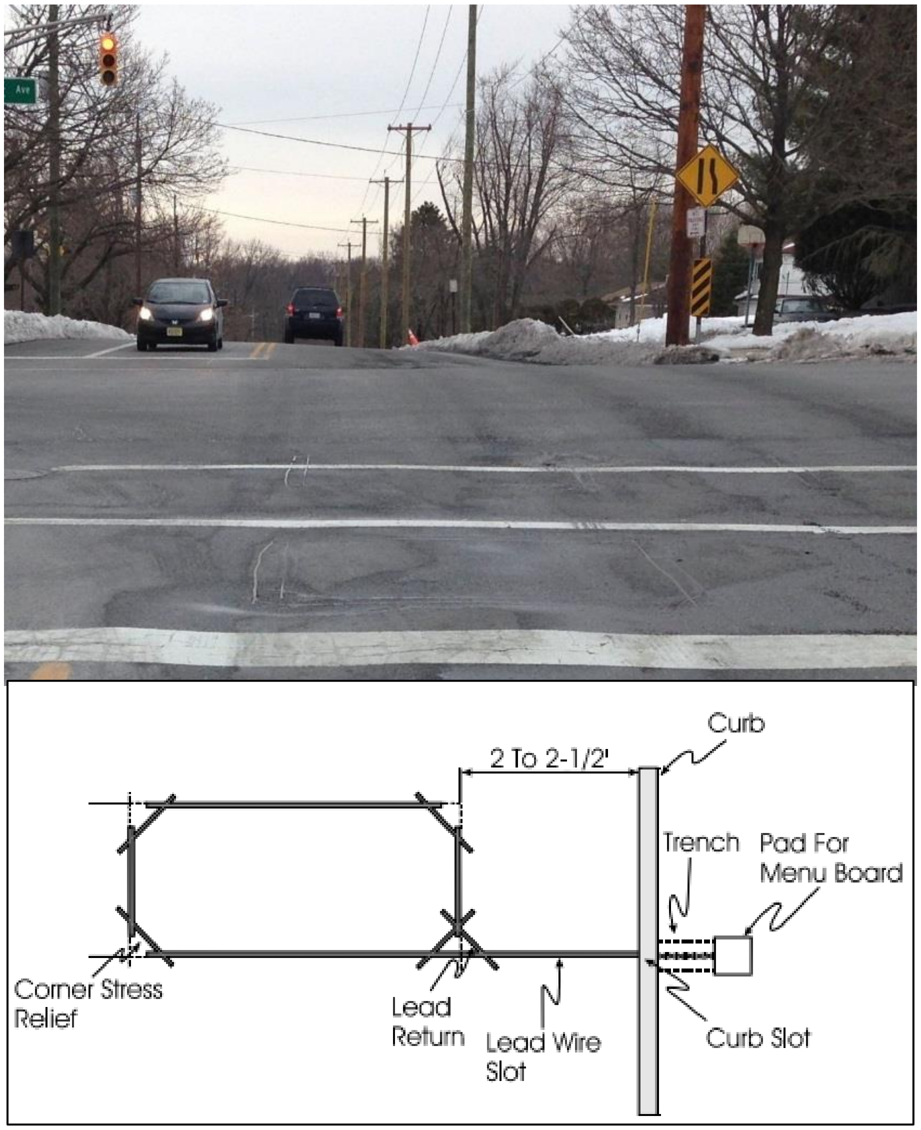


Figure 2.1 Top; Saw cut loop detectors for vehicle detection buried in the pavement at an intersection. Bottom; Schematic of the loop saw cut loop (MarshProducts 2000)

When a large metal object, such as a vehicle, moves over the loop, the resulting resonate frequency increases. This increase in frequency is sensed and, depending on the design of the detector, forces a normally open relay to close. The relay will remain closed until the vehicle leaves the loop and the frequency returns to the base level. The relay can trigger any number of devices such as an audio intercom system, a gate, or relevant to this case, a traffic light (MarshProducts 2000). The frequency change registered when a vehicle disrupts the magnetic field of the loop is shown in Figure 2.2. The complete configuration of the loop detector enabled traffic signal and supporting systems are presented in Figure 2.3.

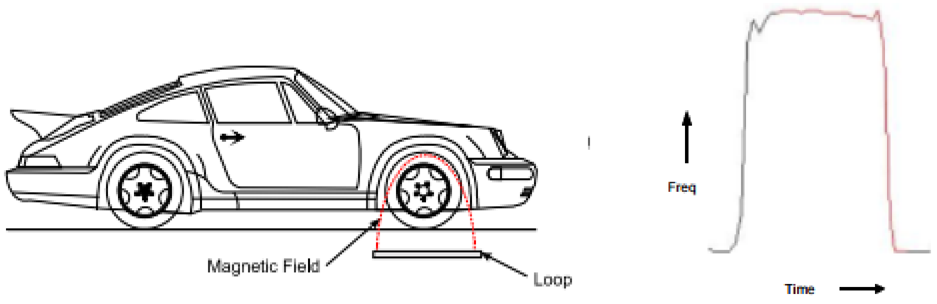


Figure 2.2 Graphical representation of the change in frequency register when a vehicle disrupts the magnetic field of a loop (MarshProducts 2000)

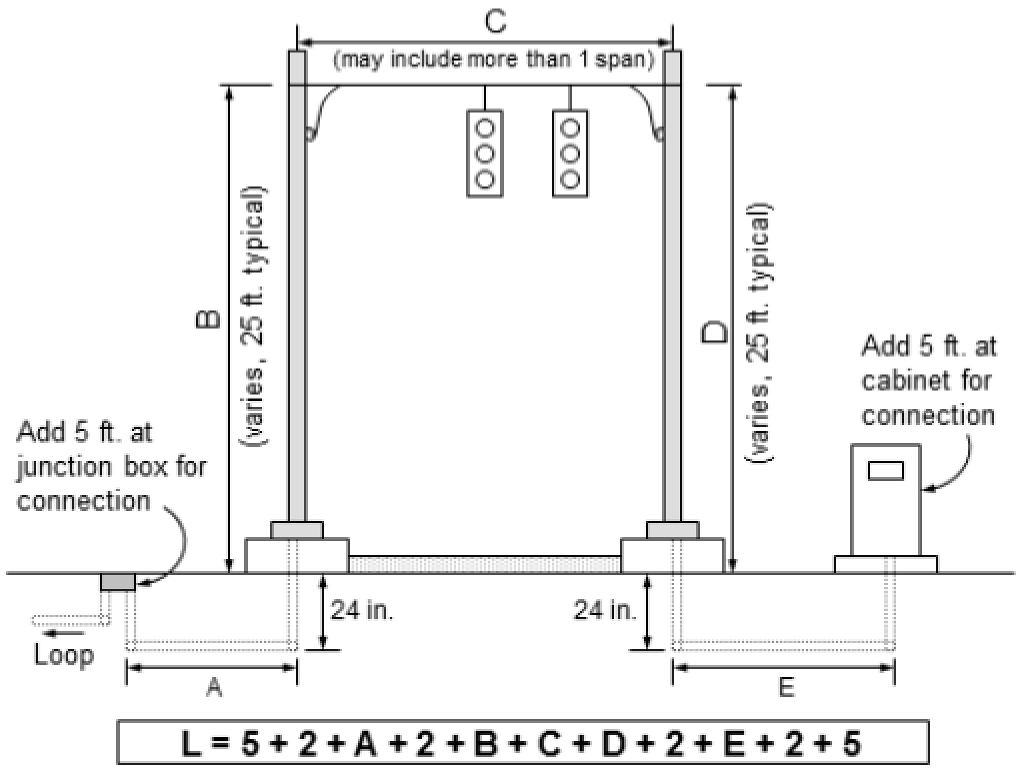

C
(may include more than 1 span)
B
(varies, 25 ft. typical)
D
(varies, 25 ft. typical)
Add 5 ft. at junction box for connection
Add 5 ft. at cabinet for connection
24 in.
24 in.
Loop
A
E
L = 5 + 2 + A + 2 + B + C + D + 2 + E + 2 + 5

Figure 2.3 Configuration of the loop detector-enabled traffic signal configuration (Sullivan et al. 2015)

Another older method of recognizing vehicles at an intersection is the treadle switch which is a permanent in-pavement roadway traffic sensor for sensing roadway traffic comprising an extruded conductive elastomeric housing having an elongated cavity in the lower side. This elongated cavity comprises of an upper wall and a pair of spaced lower walls, a flat multiconductor ribbon Teflon™ cable having at least some of the conductors mounted in the cavity with a small space from the upper wall (Tyburski 2002). A visual representation of one such device is presented in Figure 2.4.

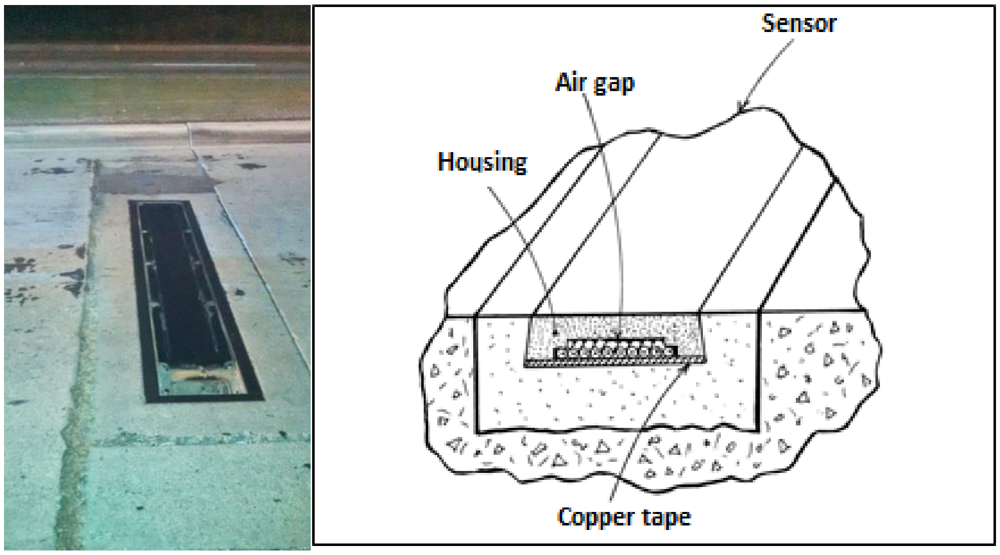


Figure 2.4 Visual representation of an unresurfaced treadle switch with schematic showing its comprising parts (Tyburski 2002)

The treadle switch works by utilizing the circuit presented in Figure 2.5. The piezoelectric effect caused by the difference in resistance under pressure is amplified, using the aforementioned circuit to amplify minute voltage differences caused by a vehicle's weight pushing on the plate. Newer sensors have been manufactured which proved to be sensitive enough to detect tire pressure even under the required three inches of resurfaced concrete. This precision is obtained by implementing a novel approach to piezoelectricity. When the weighted tire of a vehicle traverses the sensor and makes contact on top of grooves, the air gap is distorted by the collapse of the conductive elastomeric material (second electrode) causing the residual charge within the sensor element (first electrode/first dielectric) to change resulting in the generation of electric signal on the sensors first electrode (conductor). A rubber-insulated transmitting wire electrically bonded to the sensors conductor on one end and on the other end via cables connected to the analyzing equipment (Tyburski 2002). The signal, now amplified by the circuit in Figure 2.5, is connected to a logic controller which in turn regulates the traffic light timing.

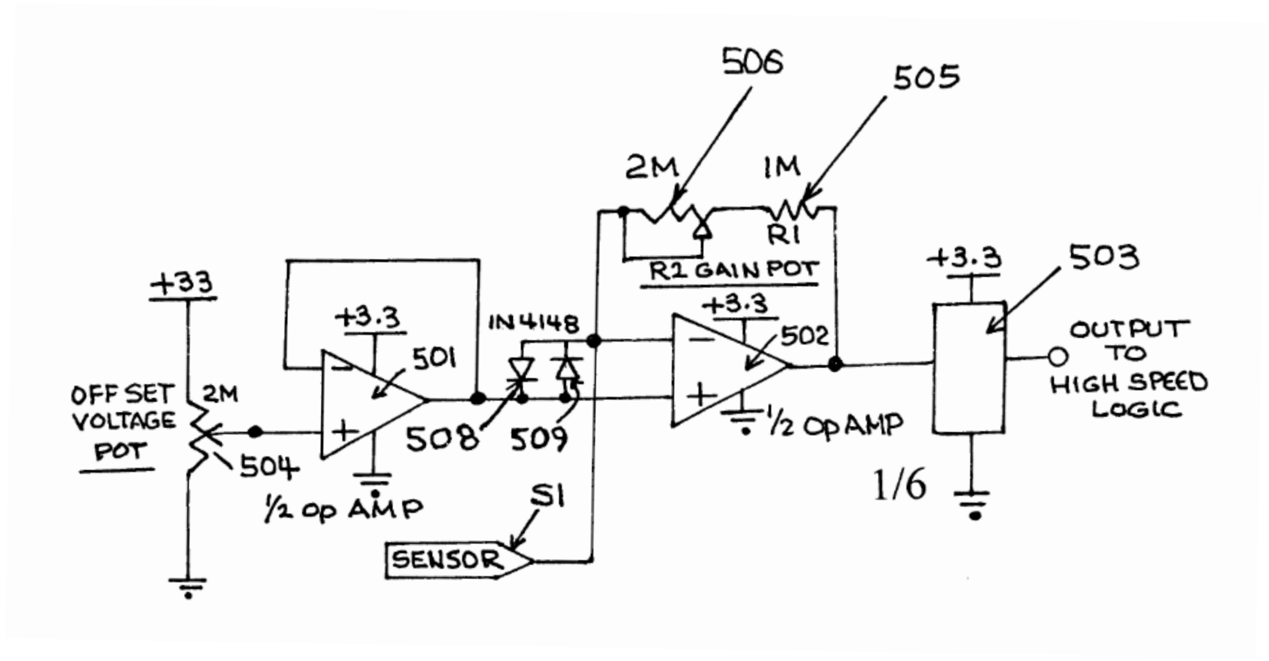


Figure 2.5 Circuit diagram of a residual charge-effect amplifier utilized in the in-pavement traffic sensor system (Tyburski 2002)

Attempts have been made to limit the amount of modification to the road paving necessary to achieve vehicle detection at intersections. The slow shift from pressure sensors and loop detectors to less intrusive detection systems have been accompanied by innovations which utilize several electromagnetic wave spectrums. One of the earlier iterations of such a device is the Traffic Safety Monitoring Apparatus (Schweitzer, Bodenhelmer, and Ben David 1989).

This system comprised of apparatus for establishing a pair of precisely spaced radiation beams in association with a thoroughfare, whereby passage of a vehicle along the thoroughfare interrupts the radiation beams, apparatus for sensing interruption of the radiation beams and providing output indications of vehicle speed and separation between adjacent vehicles (headway) and apparatus for photographing vehicles fulfilling predetermined criteria including photography trigger apparatus which is responsive to the sensed vehicle speed of the vehicle being photographed for providing a consistently positioned photographic record of the vehicle, irrespective of vehicle speed (Schweitzer, Bodenhelmer, and Ben David 1989). A visual representation is included in Figure 2.6.

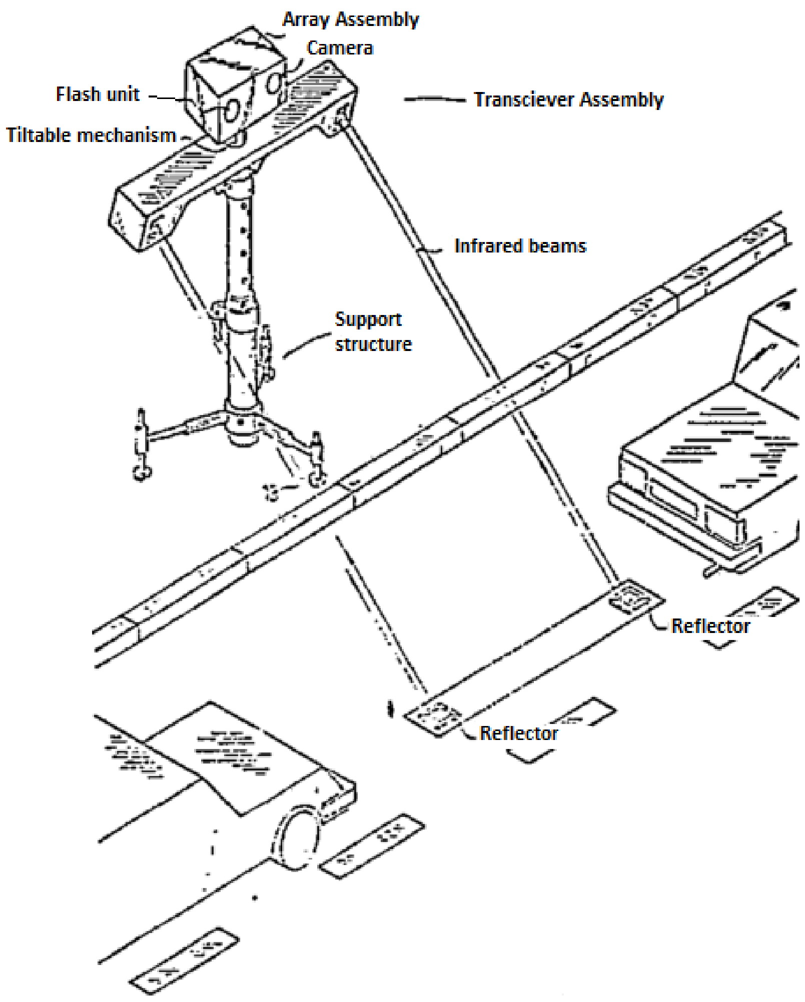
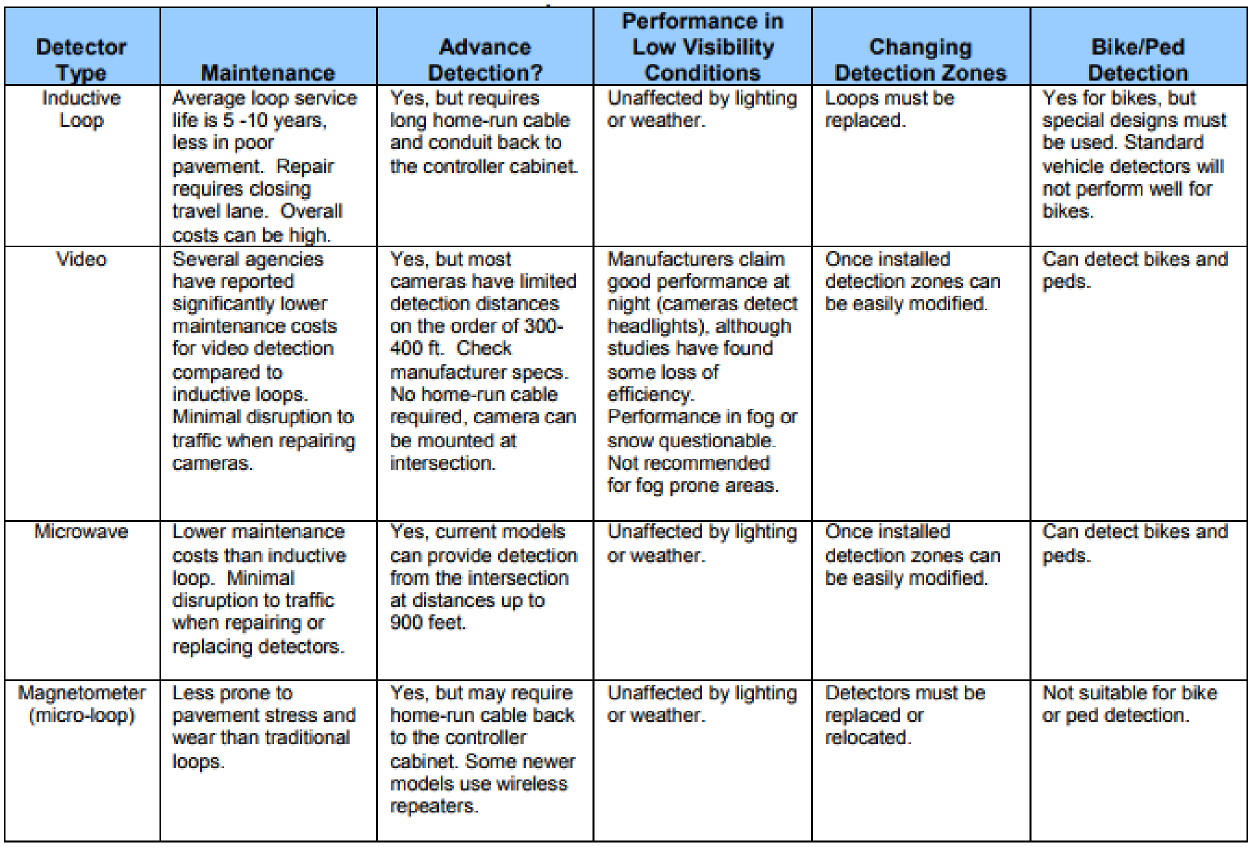


Figure 2.6 Visual representation of the Traffic Safety Monitoring Apparatus (Schweitzer, Bodenhelmer, and Ben David 1989)

2.3.2 Non-Intrusive Detectors

In an attempt to limit the physical corrections made to the road, over-roadway sensors have been developed in place of inductive loops. These technologies include video image processors, sensors that use electromagnetic waves, or acoustic sensors to detect the presence of vehicles at the intersection waiting for the right of way (Sullivan et al. 2015). These off-road sensors are more favorable than in-roadway sensors because they are immune to the natural degradation associated with paved wear and tear, competitively priced to install in terms of monetary and labor cost and danger to installation personnel, and have the capacity to act as real-time traffic management devices (Sullivan et al. 2015). They also act as multi-lane detectors and collect pertinent data types not available from in-roadway sensors.



Detector Type	Maintenance	Advance Detection?	Performance in Low Visibility Conditions	Changing Detection Zones	Bike/Ped Detection
Inductive Loop	Average loop service life is 5 -10 years, less in poor pavement. Repair requires closing travel lane. Overall costs can be high.	Yes, but requires long home-run cable and conduit back to the controller cabinet.	Unaffected by lighting or weather.	Loops must be replaced.	Yes for bikes, but special designs must be used. Standard vehicle detectors will not perform well for bikes.
Video	Several agencies have reported significantly lower maintenance costs for video detection compared to inductive loops. Minimal disruption to traffic when repairing cameras.	Yes, but most cameras have limited detection distances on the order of 300-400 ft. Check manufacturer specs. No home-run cable required, camera can be mounted at intersection.	Manufacturers claim good performance at night (cameras detect headlights), although studies have found some loss of efficiency. Performance in fog or snow questionable. Not recommended for fog prone areas.	Once installed detection zones can be easily modified.	Can detect bikes and peds.
Microwave	Lower maintenance costs than inductive loop. Minimal disruption to traffic when repairing or replacing detectors.	Yes, current models can provide detection from the intersection at distances up to 900 feet.	Unaffected by lighting or weather.	Once installed detection zones can be easily modified.	Can detect bikes and peds.
Magnetometer (micro-loop)	Less prone to pavement stress and wear than traditional loops.	Yes, but may require home-run cable back to the controller cabinet. Some newer models use wireless repeaters.	Unaffected by lighting or weather.	Detectors must be replaced or relocated.	Not suitable for bike or ped detection.

Figure 2.7 Comparison of detection methods for vehicle detection at intersections (Sullivan et al. 2015)

2.3.3 Wireless Sensor Networks (WSN)

As previously mentioned, most current traffic light control systems use one of three control approaches: fixed-time, actuated, or adaptive. In each case the overriding goal is the same, to maximize safety, speed, and energy efficiency or minimize waiting time, number of vehicle stops (Zhou et al. 2010). This is usually accomplished using wired connections between the sensors and the control units as presented in Figure 2.3. The shift toward non-intrusive sensors which use elements that do not require installations such as radars, lasers and video camera have become more common. This shift has also required the collected information to be transmitted quickly and reliably or else the information will not be useful in alleviated traffic problems (Fernández-Lozano et al. 2015). This need for reliability has led to the development of Wireless Sensor Networks (WSN). These sensor networks can involve a central controller communicating with the sensors to determine the order of the traffic signals deployed. Some of these controllers have been known to perform with fuzzy logic and other means (Gong and Zhang 2014). With further advancements however, controllers have been programmed to not just garner information from strategically located sensors on the intersection/roadways, but from the vehicle traversing the vehicle/roadways themselves. A schematic of a Wireless Sensor Network is presented in Figure 2.8.

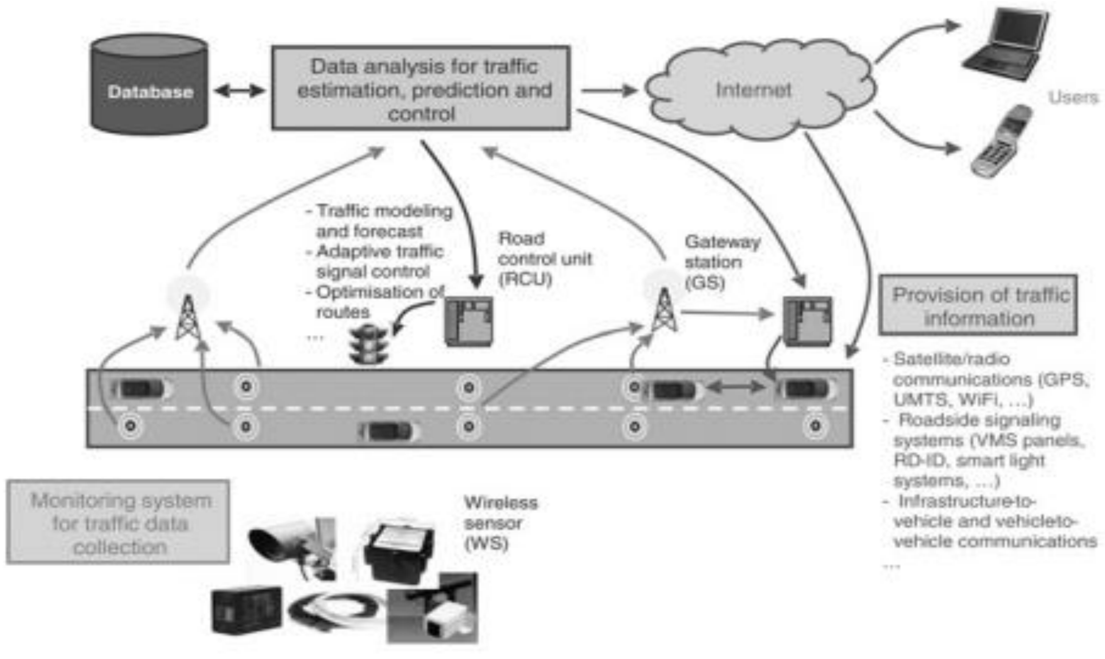

Database
Data analysis for traffic estimation, prediction and control
Internet
Users
- Traffic modeling and forecast
- Adaptive traffic signal control
- Optimisation of routes
...
Road control unit (RCU)
Gateway station (GS)
Provision of traffic information
- Satellite/radio communications (GPS, UMTS, WiFi, ...)
- Roadside signaling systems (VMS panels, RD-ID, smart light systems, ...)
- Infrastructure-to-vehicle and vehicle-to-vehicle communications
...
Monitoring system for traffic data collection
Wireless sensor (WS)

Figure 2.8 Logical scheme of functionalities of an intelligent transport system based on WSN (Pascale et al. 2012)

2.4 Vehicle-to-Vehicle and Vehicle-to-Infrastructure Technology

The idea of communicating traffic relevant information over several spectrums of electromagnetic wave proved to be an ideal method for reliable information transfer, particularly at short ranges. More research was done into incorporating this style of communications directly into vehicles, so as to establish communication not only between vehicles, but also between vehicles and traffic infrastructure (Dey et al. 2016). This was the premise behind the establishment of DSRC.

2.4.1 Dedicated Short Range Communication (DSRC)

Dedicated Short Range Communication or DSRC is a one-way or two-way short-range to medium-range wireless communication channel specifically designed for automotive use (Palma and Lindsey 2011) and is governed by a set of protocols and standards. DSRC is designed to support a variety of applications based on vehicular communication and is under active development in the United States and in other countries (Kenney 2011). The primary motivation was to enable collision prevention applications which depended on frequent data exchanges among vehicles, and between vehicles and roadside infrastructure. This information transfer occurs with a range exceed 100m (Miucic, Popovic, and Mahmud 2009). The U.S. Department of Transportation (DOT) has estimated that vehicle-to-vehicle (V2V) communication based on DSRC can address 82% of all crashes in the United States involving unimpaired drivers, potentially saving thousands of lives and billions of dollars (Resendes 2010). The National Highway Traffic Safety Administration (NHTSA) within the U.S. DOT decided in 2013 to use regulations to encourage deployment of DSRC equipment in new vehicles in the U.S. (Resendes 2010; Commission 2006). A visual representation of this system of communication is represented in Figure 2.9.

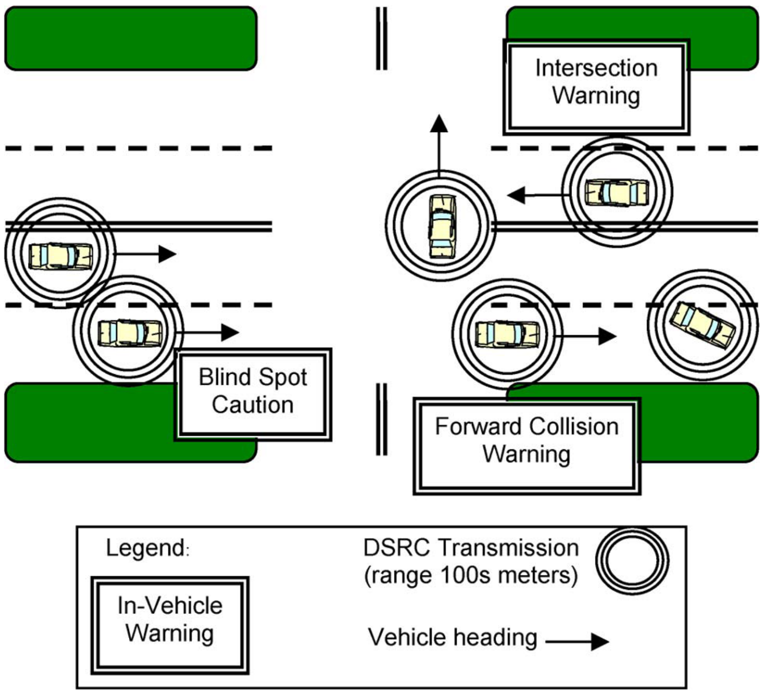


Figure 2.9 Vehicles sending safety messages, displaying in-vehicle warning (Kenney 2011)

As shown in Figure 2.9, each DSRC-equipped vehicle broadcasts its basic state information, including location, speed and acceleration, several times per second over a range of a few hundred meters. Each vehicle also receives these safety messages from DSRC-equipped neighbors and then uses these messages to compute trajectory of each neighbor, comparing it with their own predicted path, and determining if any of the neighbors poses a collision threat (Kenney 2011). In addition to V2V communications, DSRC equipped vehicles may also communicate with DSRC roadside units (RSUs) using safety messages and other types of messages, such as the geometry of the approaching intersection, the state of the signals at the intersection, and the existence of a hazard (such as a disabled vehicle or emergency vehicle) (Kenney 2011).

There are currently governing standards set forth by the respective governments dictating the interoperability between vehicles and devices on the same network. The intricacies of these standards will not be discussed in this thesis as it is not covered in the scope of developing cost effective intersection management, however it is important to understand the term “Dedicated” refers to the fact that the U.S. Federal Communications Commission has allocated 75 MHz of licensed spectrum in the 5.9 GHz band (5.850 GHz – 5.925 GHz) for DSRC communication (Kenney 2011). This spectrum is divided into different channels in which V2V messages are expected to be exchanged on Channel 172, a specific channel designated for safety (Commission 2006; Jiang et al. 2006). The term “Short Range” in DSRC is meant to convey that the communication takes place over hundreds of meters, a shorter distance than cellular and WiMAX services typically support (Kenney 2011).

2.4.2 Current Uses of DSRC

DSRC, being the forefront and most promising prospect for Vehicle to Vehicle/Infrastructure (V2X), is currently implemented in the real world, particularly in toll collecting such as the E-ZPass system (Palma and Lindsey 2011). It is a means of Automated Vehicle Identification (AVI) and is a member of the class of tag & beacon systems. The system works with these utilization of antennas mounted on overhead gantries which communicate with tags or transponders on vehicles as they pass by (Palma and Lindsey

2011). Like Automated Number Plate Recognition (ANPR), a technology which uses digital cameras and optical character recognition (OCR) software to record an image of a vehicle and its license plate (Kwaśnicka and Wawrzyniak 2002), DSRC technology can be used for all three basic functions: road usage measurement, data communication, and enforcement. It can also be used in conjunction with on-board units, to operate a zonal tolling system by activating a vehicle’s on-board unit when it crosses into the zone, and deactivating it when the vehicle leaves the zone (Iseki and Demisch 2012).

DSRC technology operates in the radio frequency or microwave range of the electromagnetic spectrum. Some tolling systems communicate in the infrared range. DSRC and infrared tolling systems differ in their susceptibility to interference but have similar functionalities (Virginia Department of Transportation 2008). Most tolled facilities in the US use DSRC technology with E-ZPass being the most widely used system. Furthermore, the governing body responsible for E-ZPass; the Inter-Agency Group, has succeeded in establish DSRC as a standard.

DSRC is simply a mode of communication. To effectively manage intersections, a method or algorithm for categorizing vehicle communicating with DSRC has to be developed. One such method coined the Cooperative Vehicle Intersection Control (CVIC) system. This system would enable cooperation between vehicles and infrastructure for effective intersection operations and management when all vehicles are fully automated. Assuming such a CVIC environment existed, this algorithm would not require a traffic signal. The CVIC algorithm was designed to manipulate individual vehicles’ maneuvers so that vehicles could safely cross the intersection without colliding with other vehicles. By eliminating the potential overlaps of vehicular trajectories coming from all conflicting approaches at the intersection, the CVIC algorithm seeks a safe maneuver for every vehicle approaching the intersection and manipulates each of them (Lee and Park 2012).

The algorithm worked emulating a yield-sign controlled intersection setup in which vehicle operators could proceed through the intersection depending on the distance and speed of the approaching vehicle. This collision prevention technique would be continuously calculated on board the intelligent vehicle's processor. A visual time-space representation of the projected trajectories A and B of the approaching vehicles is presented in Figure 2.10 and Figure 2.9 which show insufficient gap spaces, which could possibly lead to a collision, and possible sufficient gap spaces which do not pose collision threats (Lee and Park 2012).

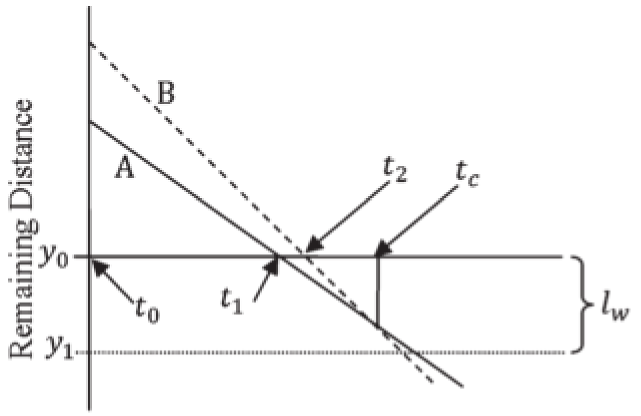


Figure 2.10 Insufficient gap case by vehicle trajectories (Lee and Park 2012)

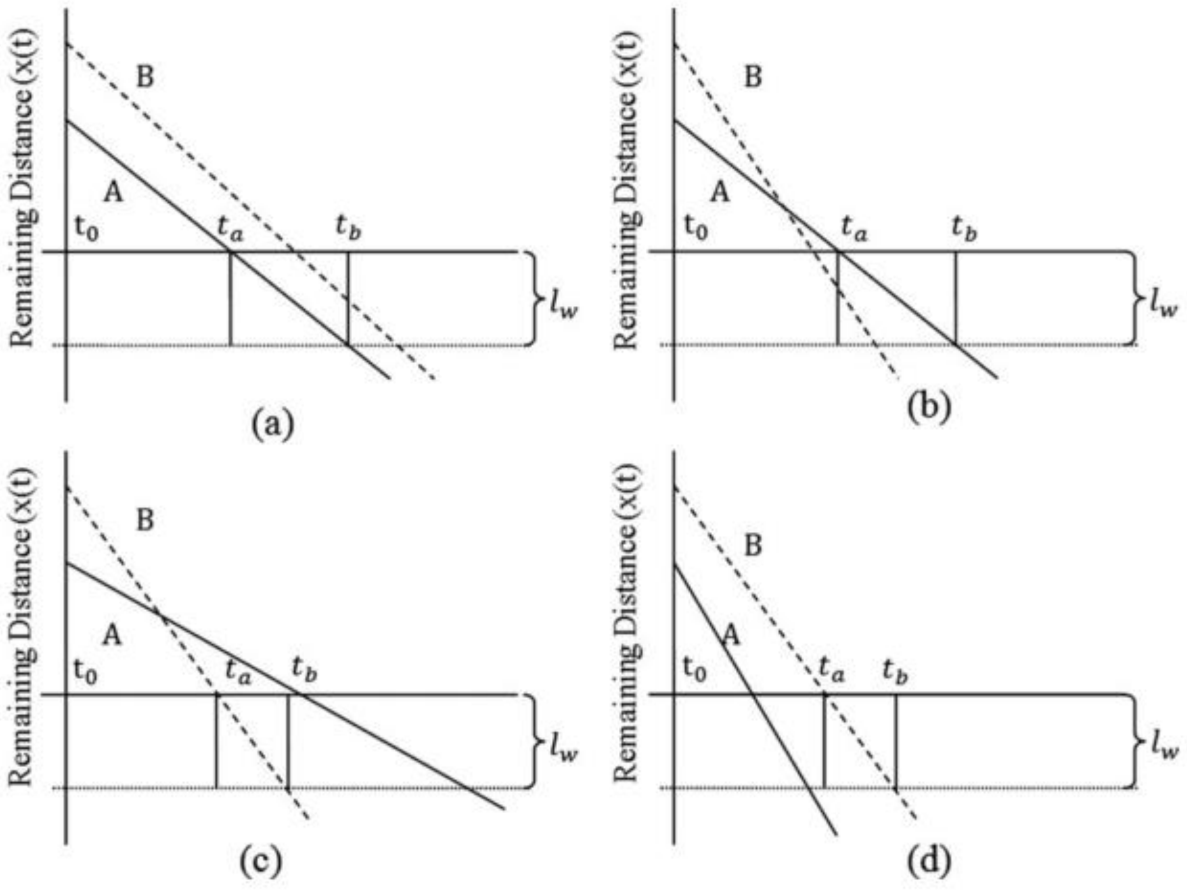


Figure 2.11 Possible sufficient gap combinations (Lee and Park 2012)

Here, the beginning and end of the intersection are denoted as y_0 and y_1 respectively, l_w denotes the intersection length t_0, t_1 and t_2 denote the starting time and intersection-entering times, and t_c denotes the time of collision, if a collision occurs. Using this method of establishing and estimating intersection vehicle trajectories and the data being continuously broadcasted from the vehicles approaching the intersection to the processing unit at the intersection, a representation similar the one presented in Figure 2.12 can be compiled for all vehicles approaching the intersection (Lee and Park 2012).

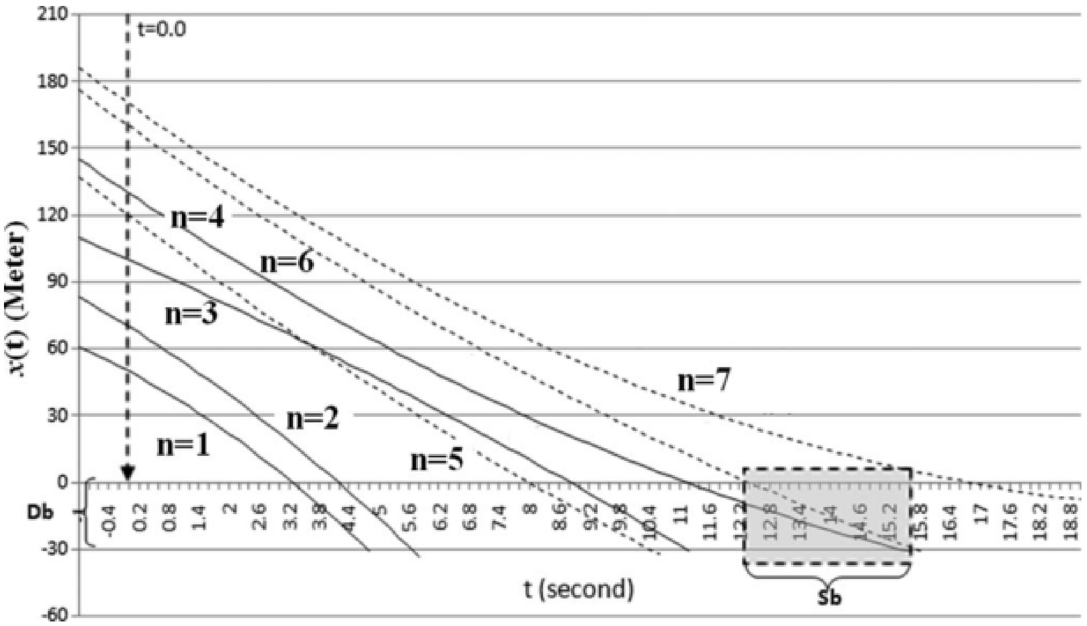


Figure 2.12 Visual representation of the trajectories of seven approaching vehicles (Lee and Park 2012)

Here, trajectories were calculated using Equation 1.

$$x_n(t) = x_n(0) - 0.5a_n t^2 - v_n t \qquad \text{(eq. 1)}$$

$x_n(t)$, $x_n(0)$ = predicted and current remaining distance to the intersection of vehicle n (m)
a_n = acceleration or deceleration rates of vehicle n (m/s^2)
v_n = current speed of vehicle n (m/s)
t = time (s)

Equation 1 Position calculation bases on the acquired acceleration distances and current speed

The shaded area, which is denoted as S_b depicts a situation in which two conflicting vehicles from each street are crossing the intersection at the same moment, thereby resulting in an overlap, which must be avoided. Equation 2 was used to define the curves of both vehicles' trajectories in the shaded box.

If $a \neq 0$

$$l = \int_{p}^{q} (1 + x'(t)^2)dt \qquad \text{(eq. 2a)}$$

Otherwise

$$l = \sqrt{(q-p)^2 + (l_w - x(p))^2)} \qquad \text{(eq. 2b)}$$

p = arrival time at the beginning of intersection (s)
q = departure time at the end of intersection (s)
l_w = intersection length (m)

Equation 2 Predicted trajectory definition of navigating vehicles (Lee and Park 2012)

By estimating the collision possibility in every combination of vehicles approaching the intersections, suggested changes to the vehicles' accelerations can be broadcasted to each vehicle to avoid said possible collisions. A simulation-based case study implemented on a hypothetical four-way single-lane approach intersection under varying congestion conditions as shown in Figure 2.13 showed that the CVIC algorithm, which employed the Active Set Method (ASM) based on sequential quadratic programming and the Interior Point Method (IPM) as analytical techniques which in turn were based on the calculation of the Karush–Khun–Tucker conditions, significantly improved intersection performance compared with conventional actuated intersection control: 99% and 33% of stop delay and total travel time reductions, respectively, were achieved. In addition, the CVIC algorithm significantly improved air quality and energy savings: 44% reductions of CO2 and 44% savings of fuel consumption (Lee and Park 2012). This system and similar ones have been proven to be capable of not only communicating with the vehicles navigating them but other close by intersections making for the possibility of implementing a network of Autonomous Intersection Management that could span an entire city (Hausknecht, Au, and Stone 2011).

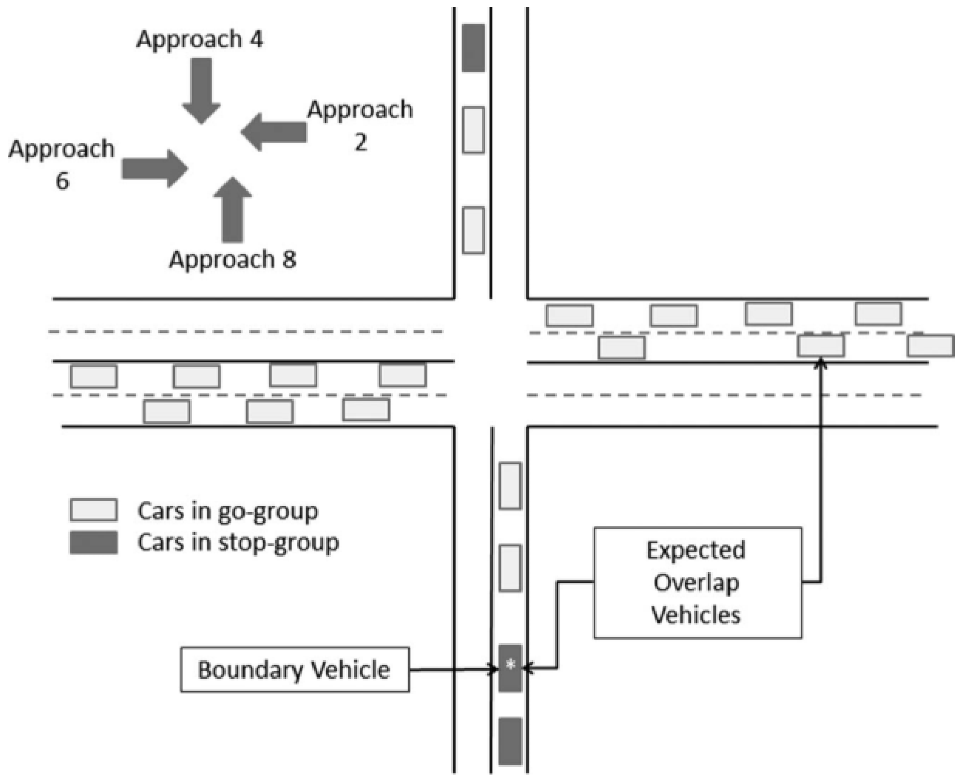
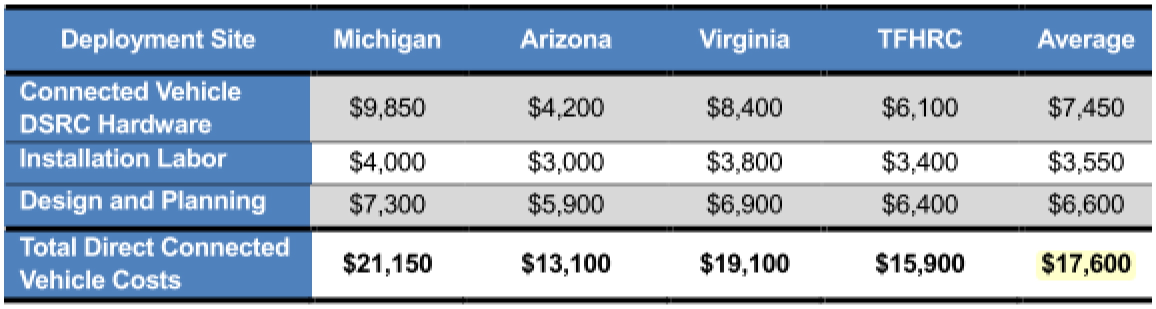


Figure 2.13 Example of vehicle navigation using CVIC algorithm

As robust as this system proved to be, several contingencies were required for its implementation. 100% of vehicles navigating the intersection would have to be fully automated which is not feasible within the next decade (Andersen 2017). Communication performances are assumed to be perfect resulting in no packet drops or any packet transmission delays. All vehicles would have to be travelling on a level terrain, resulting in no gravity acceleration effects while accelerating and decelerating (Lee and Park 2012). These limitations made for the need for possible alternatives to this system that would improve its viability.

2.4.3 Alternative Methods for V2V and V21 Communication

These enormous benefits of using DSRC system comes with its pitfalls. Currently, DSRC sites have a total average cost of $17,600 (Wright 2014). This cost includes equipment costs, installation costs and planning and design costs as shown in Figure 2.14.



Deployment Site	Michigan	Arizona	Virginia	TFHRC	Average
Connected Vehicle DSRC Hardware	$9,850	$4,200	$8,400	$6,100	$7,450
Installation Labor	$4,000	$3,000	$3,800	$3,400	$3,550
Design and Planning	$7,300	$5,900	$6,900	$6,400	$6,600
Total Direct Connected Vehicle Costs	**$21,150**	**$13,100**	**$19,100**	**$15,900**	**$17,600**

Figure 2.14 Average total costs per DSRC site (Wright 2014)

To alleviate the high cost and complexity of this system other alternatives have been developed and tested in the hopes of establishing more cost effective yet reliable vehicle communication. Several methods of achieving this goal were researched. Most of the researched material were involved in the field of computer science which relied heavily on computer aided theoretical algorithms and simulations rather than actual physical implementation and testing. Owing to this fact, of the numerous material perused, only the materials that had facets that would eventually by utilized in this thesis are presented. The works involved the recurring use of XBee radio transceiver by engineering researchers in efforts to establish some degree of intelligent transport system control.

2.4.3.1 Ubiquitous Communication for V2V and V2I for Thailand Intelligent Transportation System

Researchers in Thailand envisioned implementation of V2V and V2I for use in road traffic calculation and accident warning, and more importantly to help in better traffic management for Intelligent Transportation System. (Keeratiwintakorn, Thepnorarat, and Russameesawang 2009). Here, prototypes of road side equipment (RSE) and on board equipment (OBE), were made for communication between vehicles and roadside stations based on IEEE 802.15.4 and Zigbee wireless technologies. The Zigbee standard (which the XBee utilizes) was created as an extension for wireless personal area network (WPAN) under IEEE 802.15.4 standard for low speed, low power, and self-organized networks (Li et al. 2006). These standards have been proposed to be implemented in sensor nodes for data collection from vehicles to said stations (Chen et al. 2006).

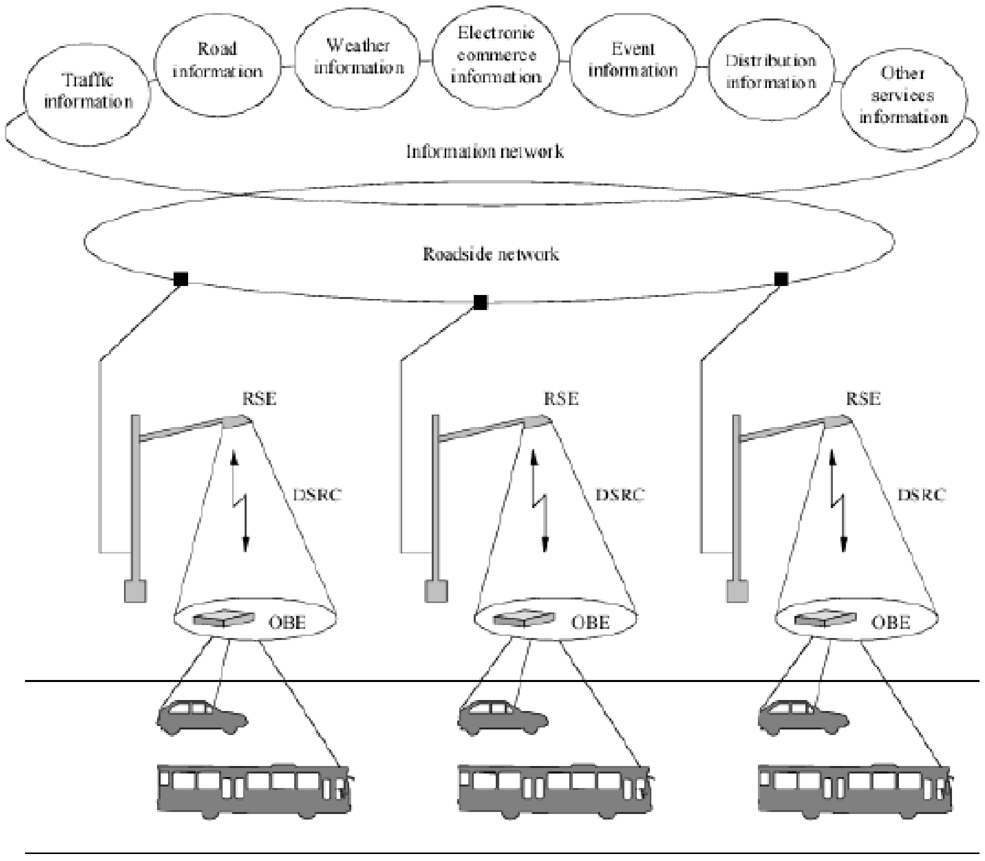


Figure 2.15 Proposed communication infrastructure for road networks (Keeratiwintakorn, Thepnorarat, and Russameesawang 2009)

This system was implemented in vehicles by OBEs presented in Figure 2.16 while very similar RSEs and presented in Figure 2.17, were developed which included GPS modules to not only let the vehicles know which module they are in communication with but also to let the vehicles no their GPS locations without any need for more advanced GPS tracking systems.

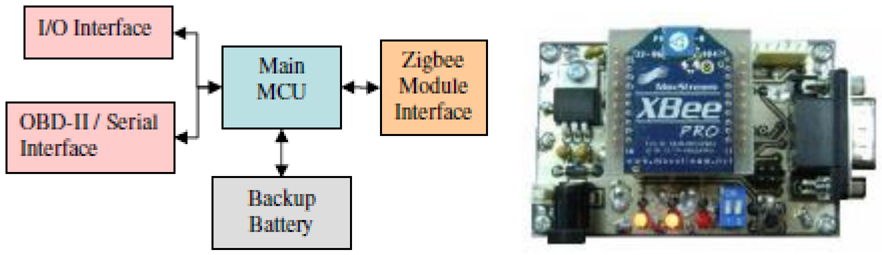


Figure 2.16 Block diagram and prototype of XBee Pro On Board Equipment (OBE) (Keeratiwintakorn, Thepnorarat, and Russameesawang 2009)

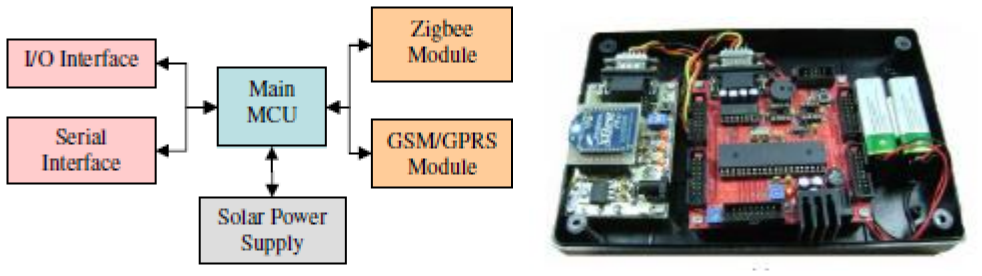


Figure 2.17 Block diagram and prototype of XBee Pro Roadside Equipment (RSE) (Keeratiwintakorn, Thepnorarat, and Russameesawang 2009)

The experiments conducted using these modules proved the usefulness and limitation of XBee modules as V2V and V2I communication system infrastructure. As shown in Figure 2.18, Xbees proved to have a most effective range of 120 m or 400ft and still viably functional at speed of 100 km/h or 62 mph, well within the confines needed for reliable intersection management (Keeratiwintakorn, Thepnorarat, and Russameesawang 2009).

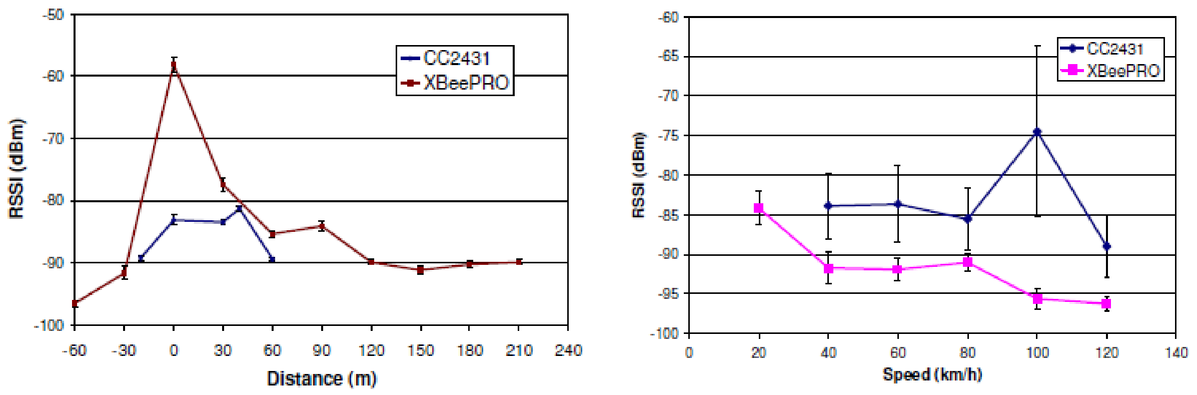


Figure 2.18 Zigbee signal strength as function of distance and wheel speed (Keeratiwintakorn, Thepnorarat, and Russameesawang 2009)

2.4.3.2 Monitoring of Cooperation between Autonomous Vehicles in Roundabout Environment

Other researchers have also utilized the XBee for various Intelligent Transportation Systems (ITS) particularly in Intersection management. In one such example a miniaturized version of the envisioned intersection management system was established using small scale robotic vehicles navigating a scaled version of a roundabout (Banjanovic-Mehmedovic et al. 2016). In this situation, roundabout intersection was chosen because of the increase in roundabout popularity due to the reduced number of conflict points normally rampant in classic intersections. This style of intersection tends to reduces driving speeds and increase driver attention further reducing the possibility of collision (Lipar and Kostanjsek 2017). The test environment was composed of a coordinator PC communicating wirelesses, through XBees, with several small Boebot robots, configured to follow black tape which simulated roadways, using infrared sensors (Banjanovic-Mehmedovic et al. 2016). The coordinator PC and robots are presented in Figure 2.19 and the simulated roadway is presented in Figure 2.20.

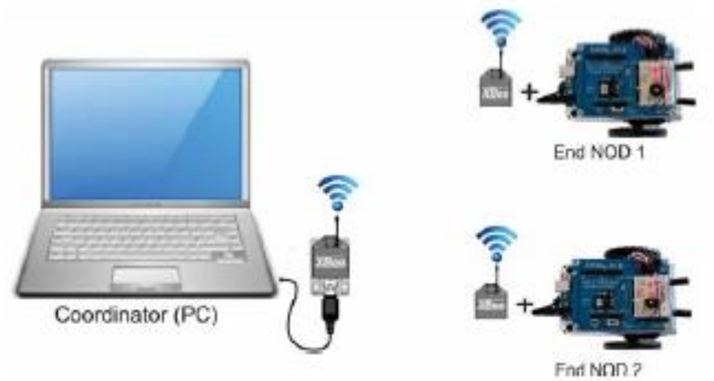


Figure 2.19 Experimental test bed for monitoring of the small scale communication protocol (Banjanovic-Mehmedovic et al. 2016)

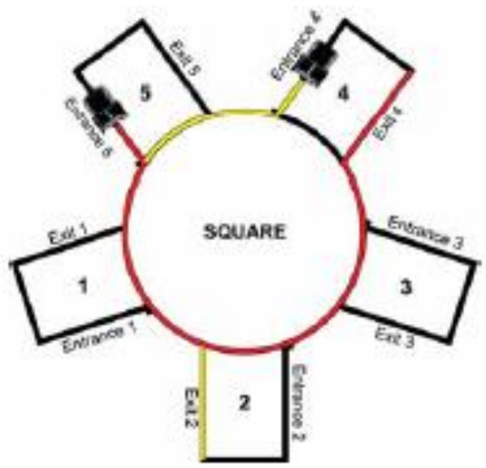


Figure 2.20 Roundabout scenario used to test the XBee enabled communication system (Banjanovic-Mehmedovic et al. 2016)

The researchers successfully controlled the vehicles' path while navigating the intersection and prevented collisions that could have arisen for uncontrolled navigation. The presence of the vehicles inside and outside the intersection was cleverly presented in Figure 2.21. Here the reader can visualize the behavior of the vehicle through the intersection. The author determined that this was a suitable means to present the results of the algorithms in this thesis.

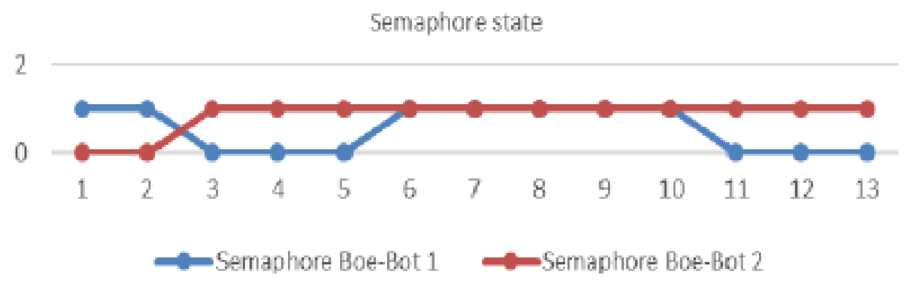


Figure 2.21 Visual representation of the autonomous vehicles' behavior through the intersection (Banjanovic-Mehmedovic et al. 2016)

CHAPTER 3

METHODOLOGY

3.1 Design and Development of Path Planning of Small Scale Prototype Vehicle

To effectively test ideas and algorithms envisioned to prove the hypothesis correct, a reasonable method had to be established for indoor data collection. This would make it easier to implement corrections in a safe and controlled environment that would not require closure of a public roadway. Several small scale test autonomous vehicle platforms were designed which would traverse a scaled model of the chosen intersection while performing the task of a level 5 autonomous vehicle.

3.1.1 Robotic Platform Base

The base of the small scale test autonomous vehicle was a circular Arlo Robotic Platform System with a 45 cm diameter and consisting of two differentially steered wheels, two supporting caster wheels, a power distribution board, ultrasonic sensors for short range obstacle detection, and quadrature encoders for wheel position monitoring and accurate distance measurement. Several other important components were then installed on this base including the Inertial Measurement Unit (IMU), Global Positioning System (GPS) Sensors, Image Sensors for lane recognition and stop sign recognition and an in house built LiDAR sensor for long range obstacle detection.

The schematic for the small scale vehicle platform that was used in all subsequent testing is presented in Figure 3.1. All wiring connections to the two microcontrollers are included as well as the names of the parts in Table 3.1

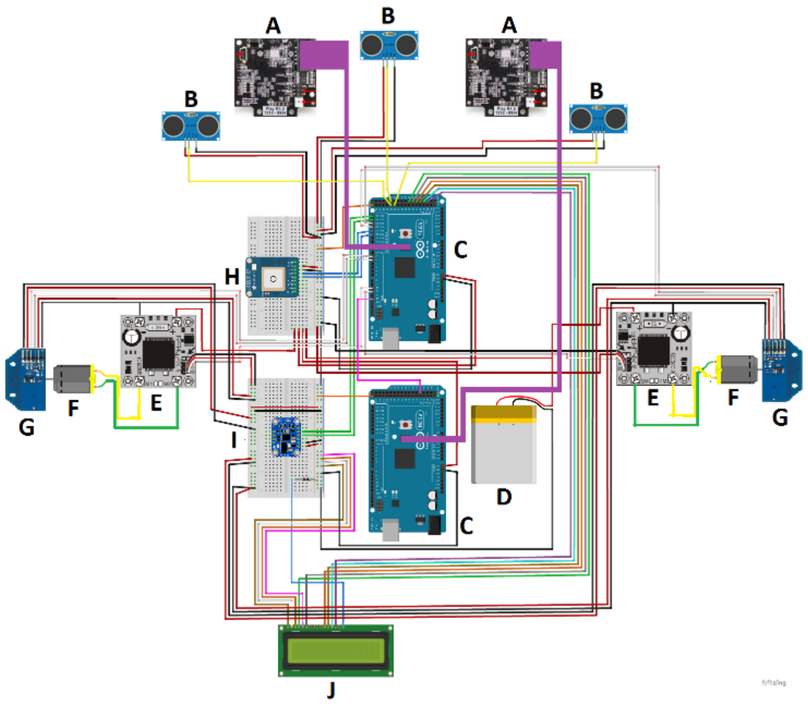


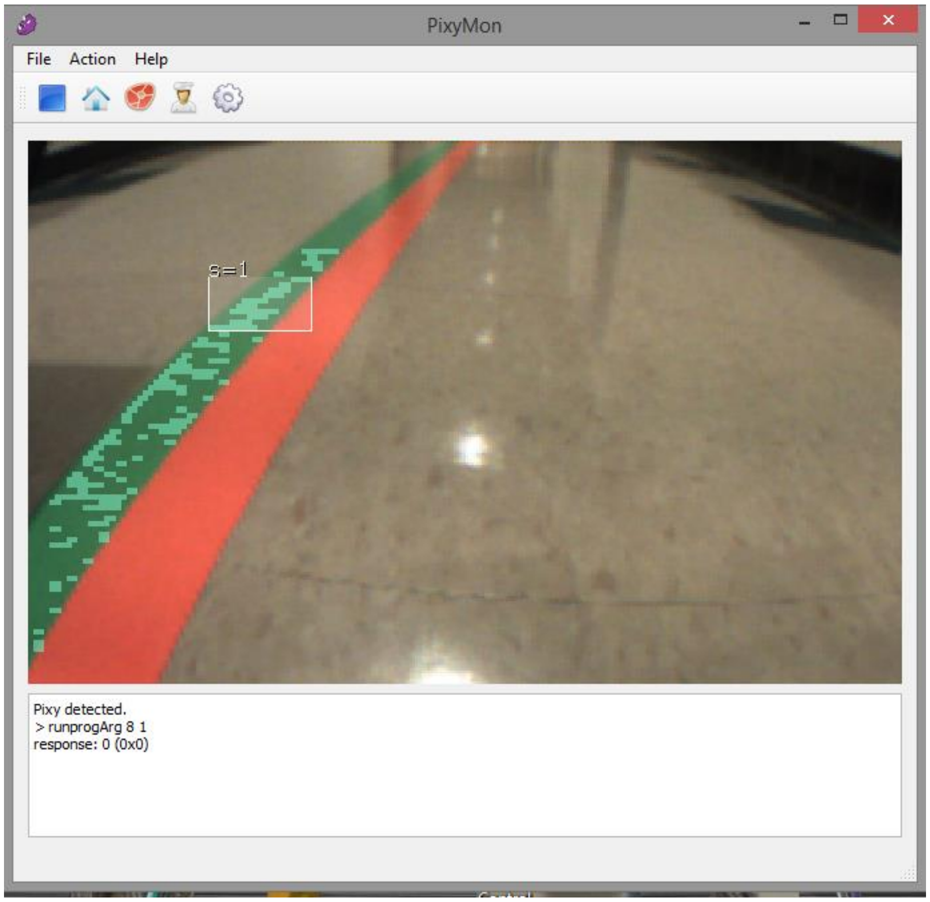
Figure 3.1 Schematic of the small scale test vehicle

Table 3.1 Component of the small scale test vehicle

	Component
A	Image Sensor
B	Ultrasonic Sensor
C	Arduino MEGA 2560 Microcontroller
D	Lithium Polymer Battery
E	HB-25 Motor Controller
F	12V DC Brushed Motor
G	Quadrature Encoder
H	GPS Sensor (Not used in this paper)
I	IMU Sensor
J	LCD Screen

3.1.2 Lane Following and Traffic Sign Recognition

An important aspect of any intelligent vehicle is optical recognition of the vehicle's world. This includes vital information necessary for proper functionality such as lane following and traffic sign recognition. Conventional autonomous vehicles utilize high level cameras and expensive image processors for edge detection and color pattern recognition (Bernard Ibru et al. 2017). As effective as these methods of optically defining the world around the vehicle is, for the small scale autonomous vehicle platforms, a more cost effective camera was required. This was achieved by using the PixyCamCMU Image Sensor which comes equipped with an in built image processor. The PixyCam worked by recognizing preprogrammed color pigments, grouping them together, and if they are larger than a predetermined pixel size, the processor draws a block around the pigment. The image sensor proceeds to output the Cartesian coordinates of the box as well as the length and breadth of the block as shown in Figure 3.2.



```
Pixy detected.
> runprogArg 8 1
response: 0 (0x0)
```

Figure 3.2 Visual Representation of the Output of the PixyCamCMU Image Sensor (Bernard Ibru et al. 2017)

Utilizing this information obtained from the image sensors, a program was written to follow the lane. By recognizing that the left line of the lane should be in a particular coordinate position in relation to the field of view of the small scale autonomous vehicle, the left and right wheels of the vehicle could be manipulated to maintain the correct position in relation to the designated left line of the lane while simultaneously moving forward. A flow chart of this program is presented in Figure 3.3.

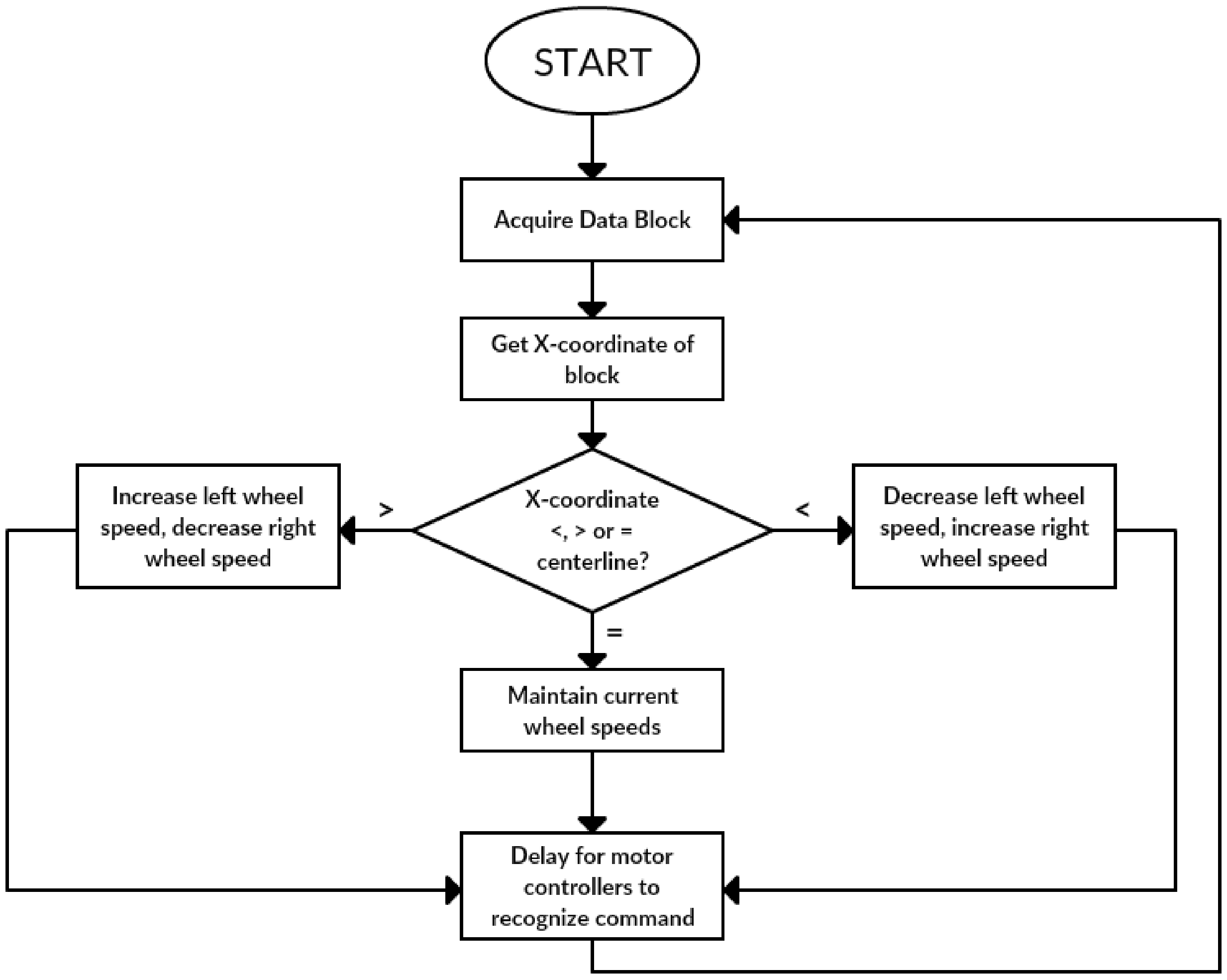

START
Acquire Data Block
Get X-coordinate of block
X-coordinate <, > or = centerline?
>
<
=
Increase left wheel speed, decrease right wheel speed
Decrease left wheel speed, increase right wheel speed
Maintain current wheel speeds
Delay for motor controllers to recognize command

Figure 3.3 Flow Chart Showing Premise for Initial Lane Following Method (Bernard Ibru et al. 2017)

After performing some experimental runs, it was discovered that the vehicle performed optimally in near perfect lighting (above 75 lux) but proved to present with sudden and erratic jerks at lower lighting. This was due to the fact that the camera was very selective to the particular shade of the preprogrammed colors in its database and slight various, particular those due to light and shade on the tracked color. This variation causes the camera to ‘see’ multiple smaller blocks rather than one large block as shown in Figure 3.2.

3.1.3 Pixel Data Optimization in Low Light

After establishing the fact that the image sensor performed poorly in low light condition, the degree of imprecision had to be quantified. To accomplish this, and also to determine if some sort of delay between readings is more advantageous in reducing said imprecision, an experiment was carried out in which a stationary object of the preprogrammed color was placed in front of the image sensor to determine the exact degree of imprecision encountered at varying reading intervals.

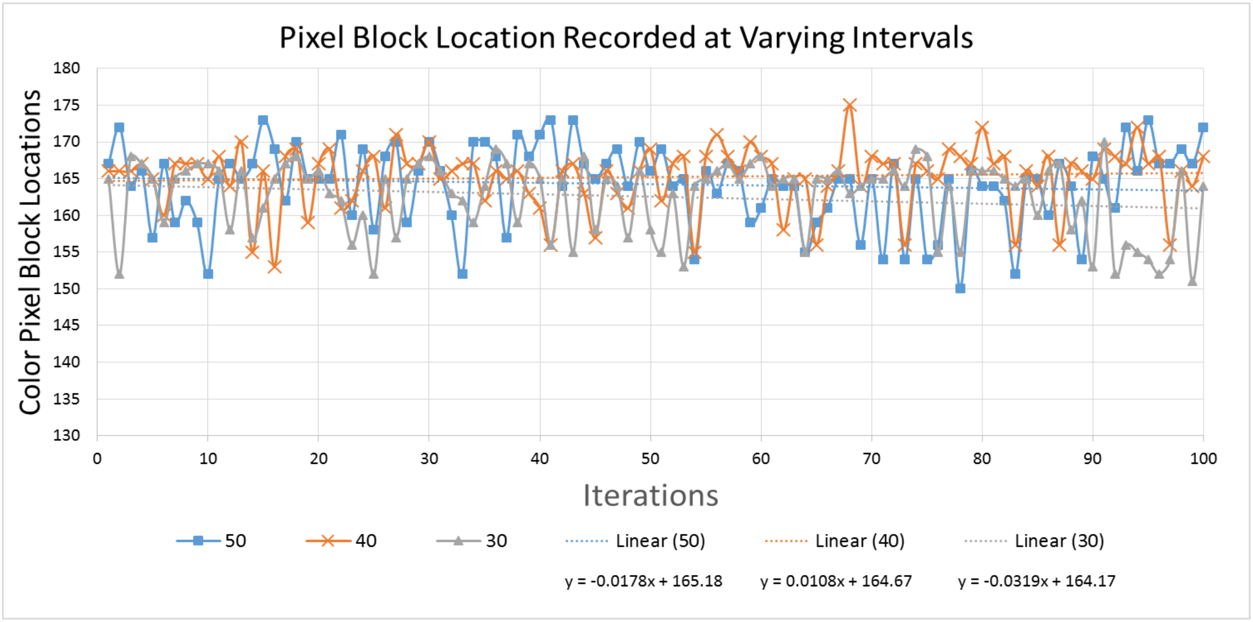


Figure 3.4 100 block pixel locations taken at intervals of 50, 40, and 30 readings respectively at a stationary target in pixel location 165 (Bernard Ibru et al. 2017)

The object was first placed on a relative pixel location of 165 and 100 reading were recorded at intervals of 50, 40 and 30 readings respectively. The results of this experiments are presented in Figure 3.4. as shown in the graphs, there were no significant variations in precision and hence the object was moved in such a way as to reduce the lighten and also observe any discrepancies in precision. The object was then positioned in pixel location 148 in front of the light source. Readings were then recorded ay intervals of 20, 10, 5 and 1 respectively to determine if this would affect the image sensors' precision in any way. These new readings were pressented in Figure 3.5.

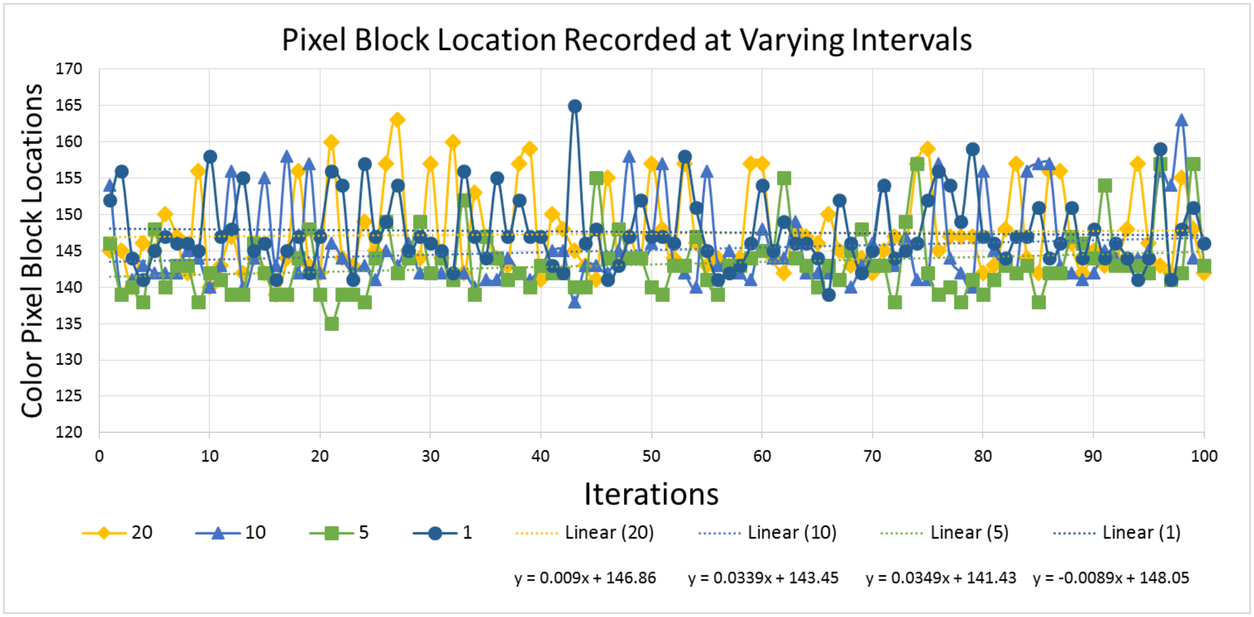


Figure 3.5 100 block pixel locations taken at intervals of 20, 10, 5, and 1 reading(s) respectively at a stationary target in pixel location 148 (Bernard Ibru et al. 2017)

After conducting the experiments, it was dicovered that a fluctuation of about $\pm$10 pixels was present in the readings of each set, which when compared to pixel scale of reference frame – 320 pixels, amounts to a $\pm$6% error (Bernard Ibru et al. 2017). This seemingly small discrepancy in the data, however, can lead to erroneous overcorrection and could result in noticeable jerks in movement of the test vehicle. It was also concluded that changes in the intervals of the readings did not positively or negatively affect the readings of image sensor and hence a new method to improve the precision of the pixel block readings was required.

After much deliberation, a statistical method of applying rolling averages to the image sensor data was devised. The rolling average method included taking information from several data blocks and averaging them to get a reading that would then be input to the motor controllers for appropriate corrections. Employing rolling average versus average allowed for historical data to be present in the memory, thereby ensuring that any singular anomaly/ random error/ deviation would not affect the correction of movement significantly (Bernard Ibru et al. 2017). The idea was first verified by graphically comparing the data obtained from Figure 3.4 and Figure 3.5 to averaged values obtained by finding the mean of the last 10 readings. This was carried out on every interval reading and intervals of every 50 readings. The results are presented in Figure 3.6.

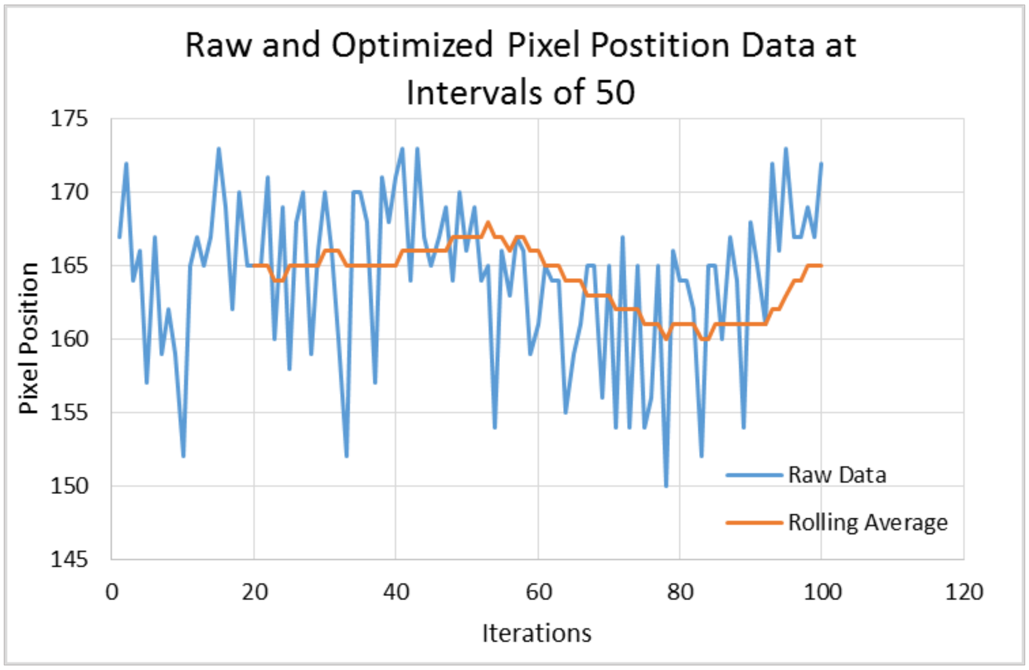


Figure 3.6 Raw and Optimized Pixel Block Reading after every 50 readings with Stationary Target Position at 165 (Bernard Ibru et al. 2017)

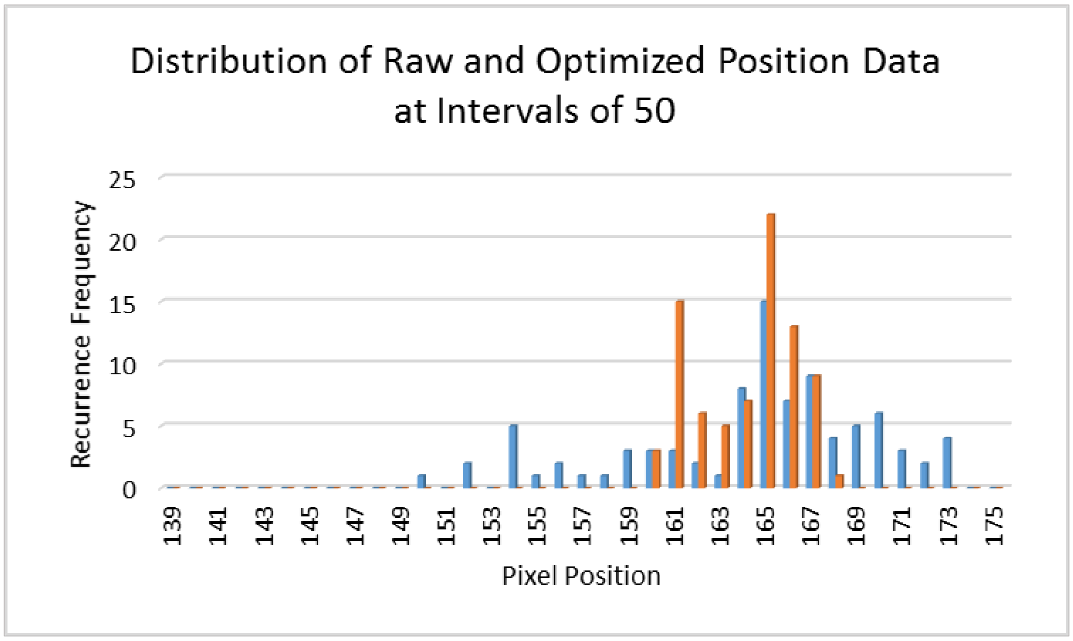


Figure 3.7 Distribution of Raw and Optimized Pixel Block Reading after every 50 reading with Stationary Target Position at 165 (Bernard Ibru et al. 2017)

When producing a distribution map of these readings, as shown in Figure 3.7, it was determined that even at intervals of 50 readings, the spread of the data points was much closer with the averaging method than with the raw data. The last step of the experiment was then to compare the data of the raw and averaged data without taking intervals between readings and then comparing this with the data acquired with an interval of 50 between readings.

As shown in Figure 3.8, the precision of the derived average data was vastly increases when compared to that of the raw data. The spread of the perceived readings was also proved to be reduced as shown in Figure 3.9.

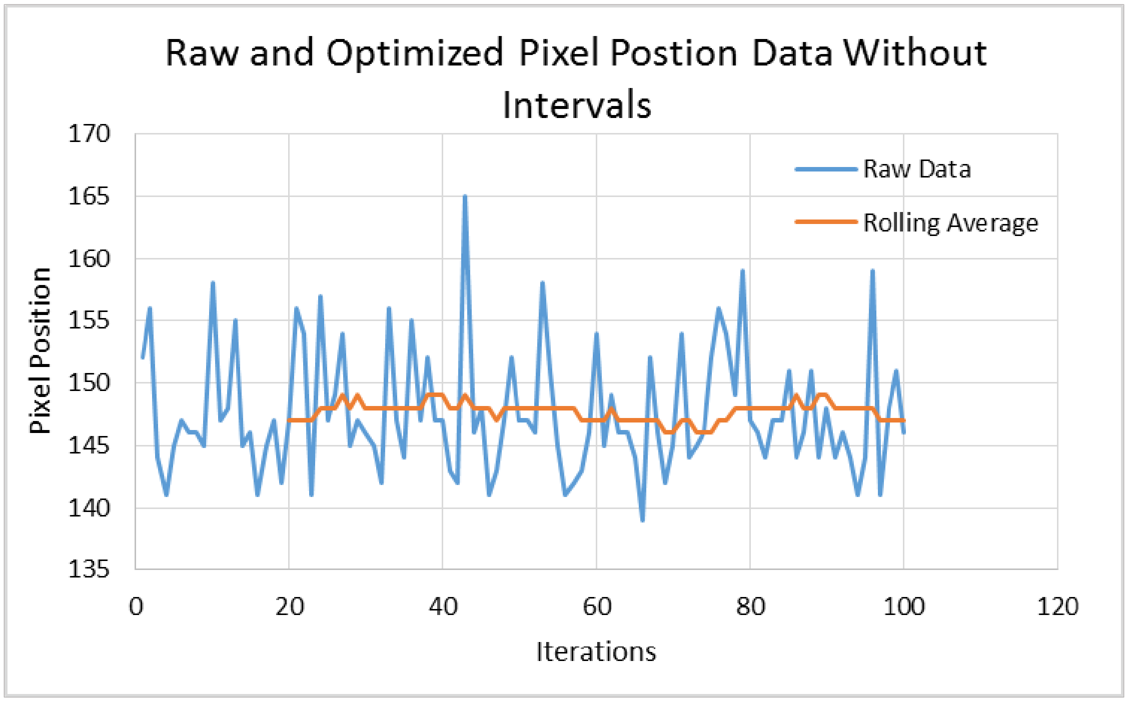


Figure 3.8 Raw and Optimized Pixel Block Reading at every reading with Stationary Target Position at 148 (Bernard Ibru et al. 2017)

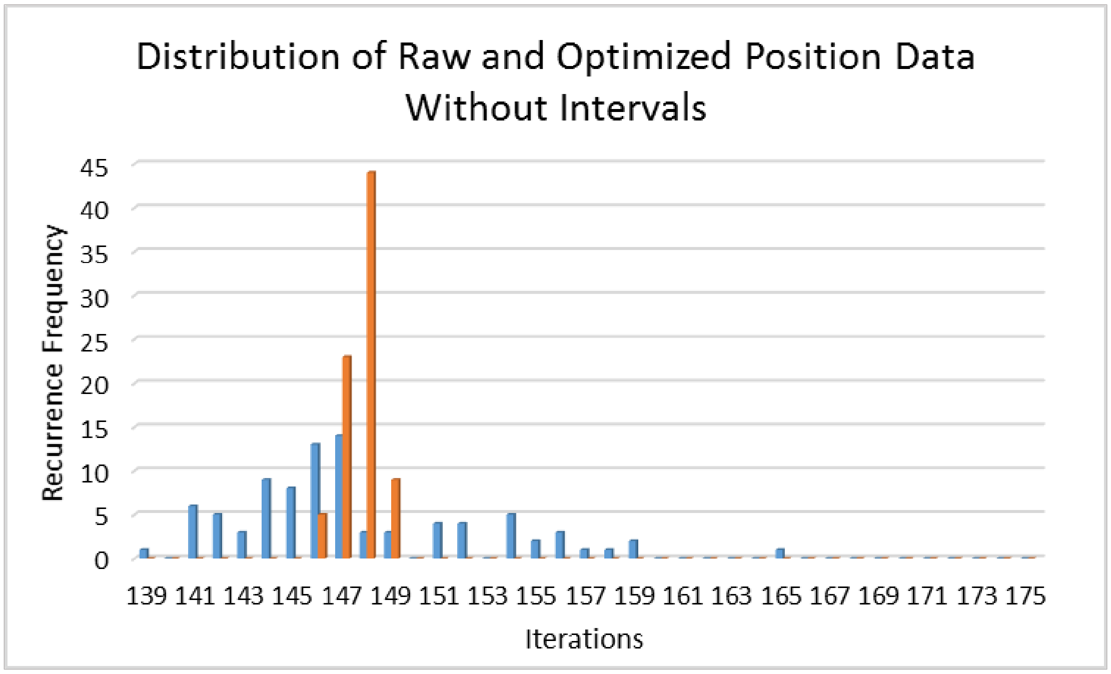


Figure 3.9 Distribution of Raw and Optimized Pixel Block Reading at every reading with Stationary Target Position at 148 (Bernard Ibru et al. 2017)

It was decided to proceed with this averaging method of obtaining visual data readings for the image sensor for navigation the lanes of the road for the purpose of this thesis because the roads being tested are straight lines and do not experience large discrepancies.

For the purpose of traffic sign recognition, this method will not be utilized. Due to the fact that the image sensor is not robust enough to recognize shapes and edges, the method for determining the presence of a traffic sign was to look for a particular color signature (Stop Sign Red) in a particular area being viewed by the right side mounted image sensor. The presence of this color signature in that area can be construed as the presence of a stop sign and the small scale autonomous test vehicle registers a stop sign and carries out all protocol linked with this finding. This was the method used throughout scaled testing of the chosen intersection.

3.1.3 Obstacle Detection and Recognition

Another important path planning application any autonomous vehicle is the ability to detect and recognize obstacles which could be a potential hazard if the vehicle collided with it. It is important for the vehicle to not only recognize that there is an obstacle in the vehicle's world but also to determine the course of action to mitigate a possible collision as well as warn other incoming vehicles of the threat and the course of action taken.

As displayed in Figure 3.10, all of the small scale vehicles were fitted with three Parallax PING Sensors; commonly used ultrasonic sensors which are known to be reliable for indoor test purposes. Ultrasonic sensors utilize sound and the time difference between pings or sound generation and echo or sound return to determine the distance between the sensor and the obstacle. This sensor has the advantage of having a wide lateral range of detection but they have a limited range of detection as shown in Figure 3.11.

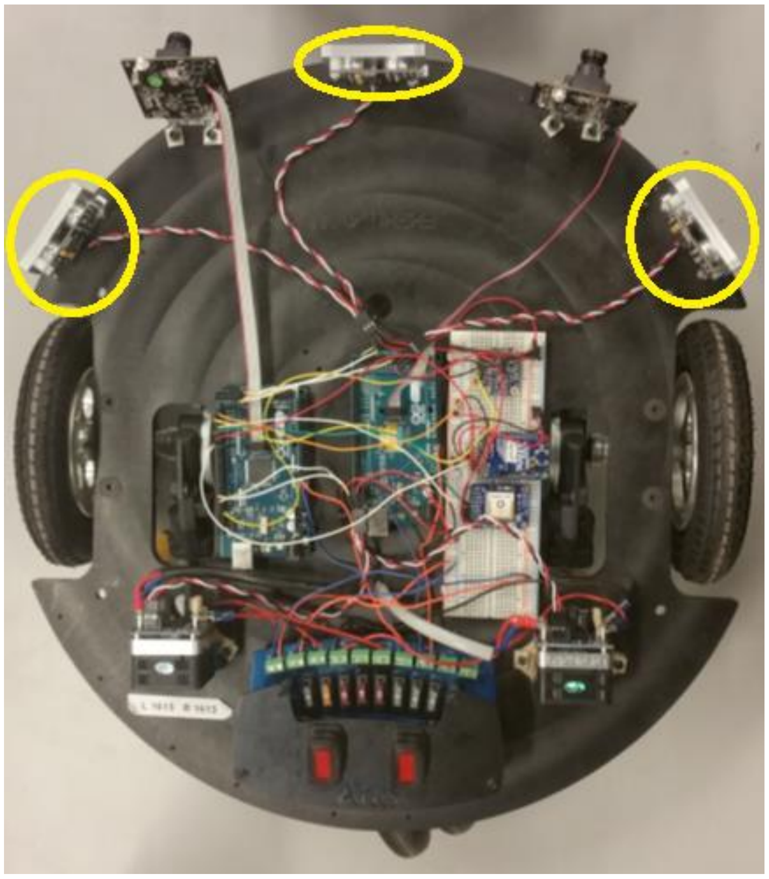
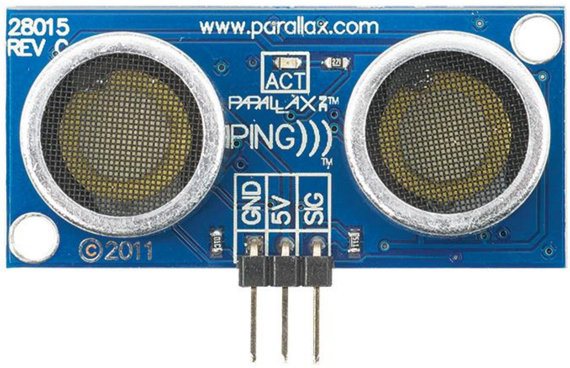


Figure 3.10 Small scale autonomous vehicle prototype showing PING sensors



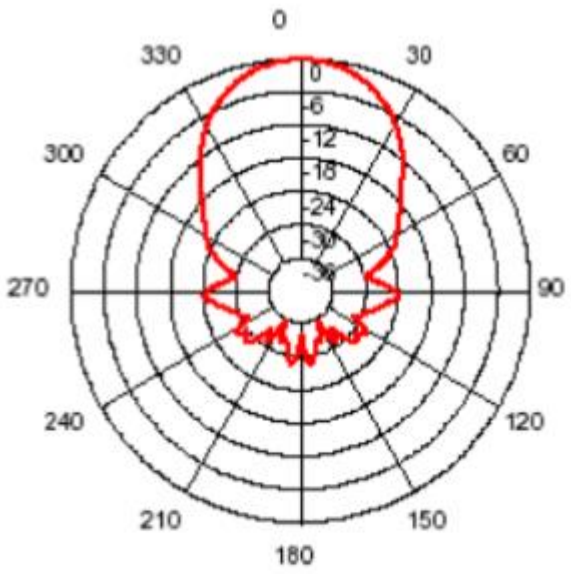


Figure 3.11 Left: Ultrasonic Ping Sensor Right: Detection angle and range of the ultrasonic sensor (Rios-Gutierrez 2017)

The PING sensors were first proven to be adequate for use in the small scale vehicle platforms by establishing calibration curve presented in Figure 3.12.

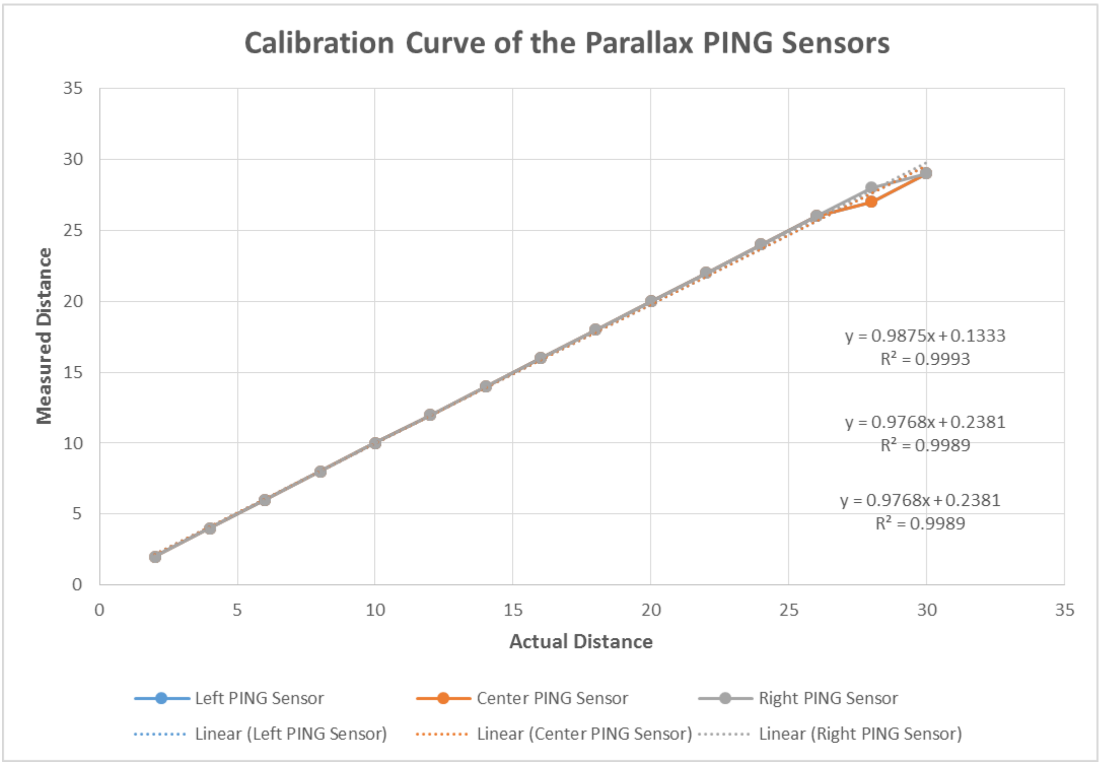


Figure 3.12 Characterization of the Parallax PING Sensors

3.2 Design and Development of Small Scale Test Vehicle Platforms (Path Following)

The aforementioned sensors, particularly the pixy image sensors are useful for environmental recognition and guidance motions such as lain following as well as traffic sign recognition. These sensors are however rendered unimportant when these environmental cues are not present or when the vehicle is required to ignore them so as to achieve a particular section of its navigation algorithm, such as intersection navigation. At an intersection, an autonomous vehicle must ignore the lines on the road so as not to accidentally follow the wrong routes. This begs the question of how these vehicles can effectively navigate from the entry leg of the intersection to the desired leg of the intersection optimally and reliably.

Path following algorithms are the only reliable method for autonomous intersection management, as a planned path can be programmed into the vehicle before it even reaches the desired intersection in question because the intersection is set and its distance parameters will never change save for a remodeling of the intersection. The concept seemed common sense, however the method for achieving accurate and reliable path navigation without the aid of environmental cues had to be established in the small scale intelligent vehicle platform.

The hurdles linked with blind intersection navigation in the small scale vehicle was achieved using an Inertial Measurement System (IMU) linked with quadrature wheel encoders to establish sensor fusion for reliable instantaneous control of the vehicle's movement and orientation. A co-researcher proved that a non-environmental sensor crucial method could be utilized to successfully and reliably navigating any intersection (Beyerl 2017). This was proven by taking several sections of the problem intersection and programming the scaled values for the resulting lead in, turning angle, turning radius, and lead out necessary to complete the navigation as shown in Figure 3.13.

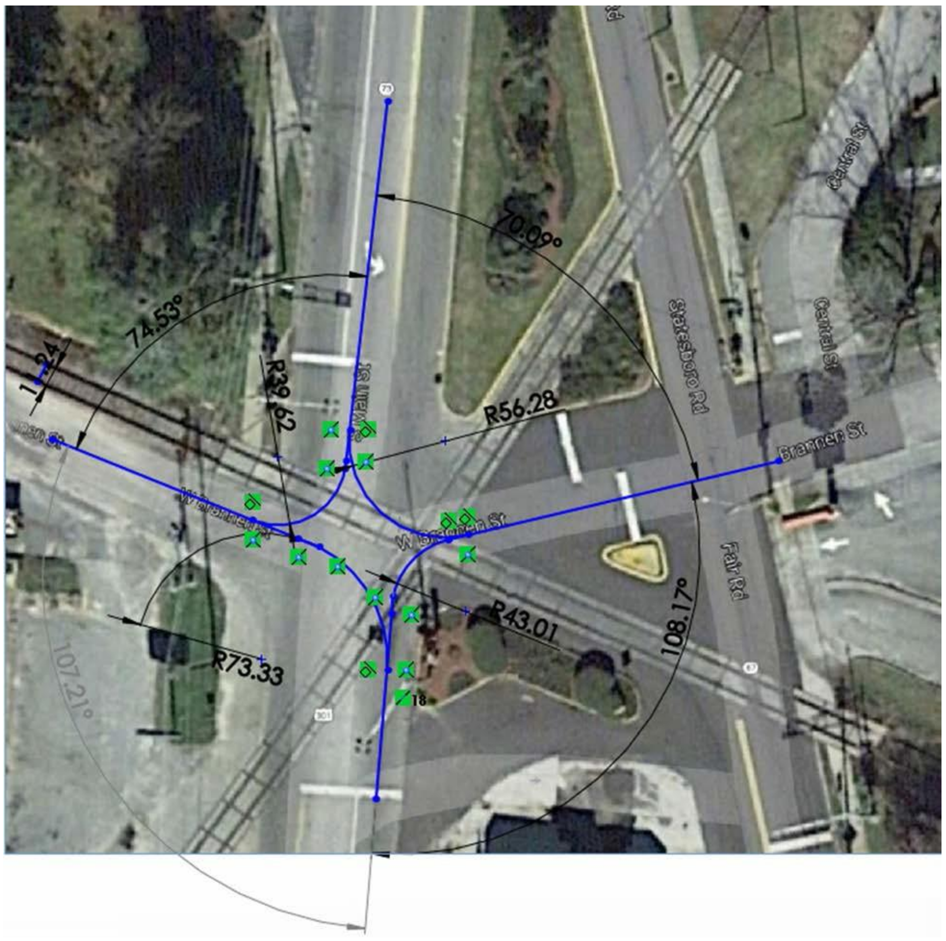
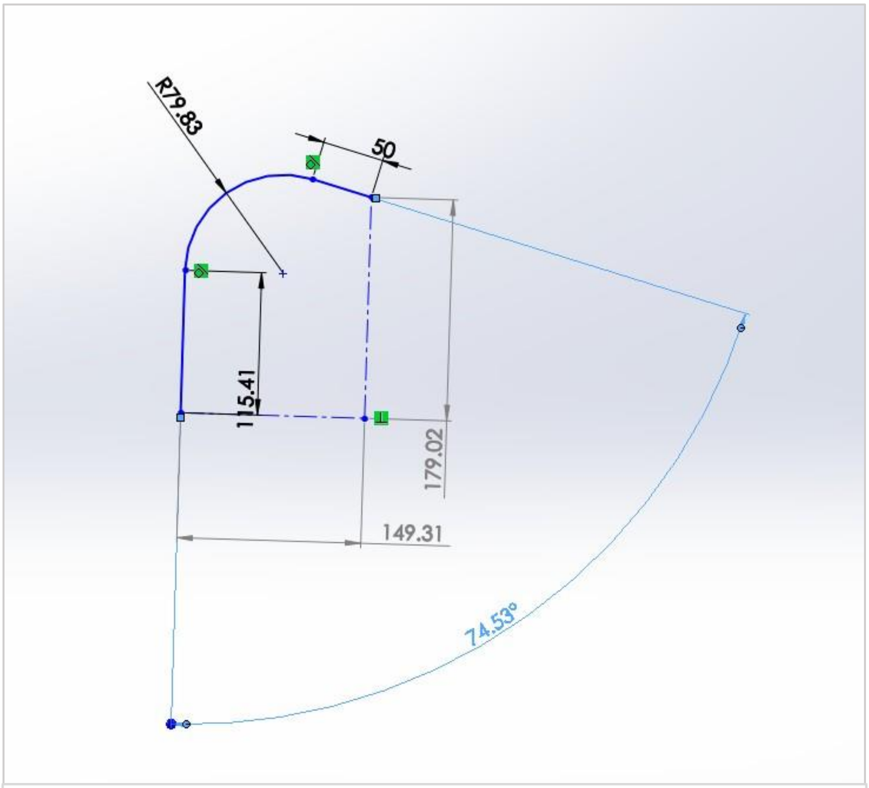


Figure 3.13 Tested navigation paths overlaid on the complex intersection (Beyerl 2017)

These paths where programmed into the test vehicles using various methods including implementing a spiro-circular arc, which would reduce jerk when the vehicle shifted from the straight line drive to the arc drive, and a more simplified transfer system (Beyerl 2017). After testing of the navigation algorithms, it was proven that by utilizing an IMU and wheel encoders the desired path of the vehicle can be accurate to a third of the vehicles width tolerance to a high degree of precision as shown in Figure 3.14 (Beyerl 2017).



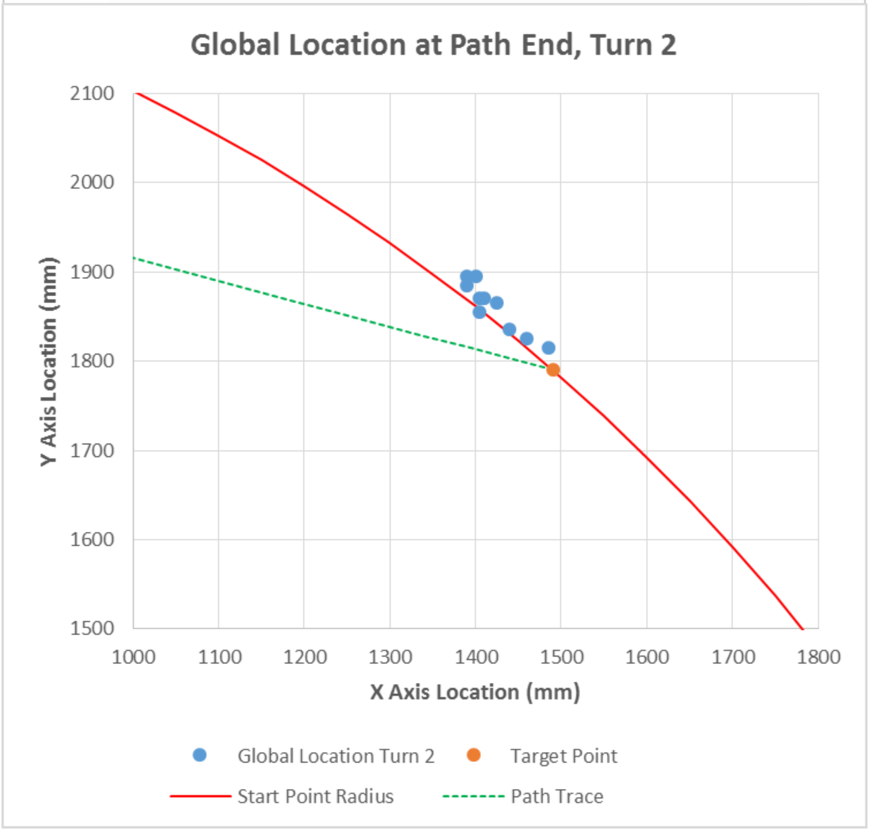


Figure 3.14 Left: Graphical representation of tested turning path showing pertinent parameter. Right: Marked endpoint showing target point, preferred path traced and start point radius (Beyerl 2017)

3.3 Establishing Vehicle-to-Infrastructure Communication

It was decided that to efficiently control autonomous vehicular navigation within an intersection, all the vehicles approaching the intersection would be made to wirelessly communicate with a base, which would be the sole control system collecting request from the vehicles and assigning priority based on the time of arrival of the vehicle and the clear paths present in the intersection. The task of establishing a communication protocol was divided into; i. Establishing communication vectors, which are particular to the vehicles and intersection being navigated and can be understood by both vehicle and base without any possibility of misreading/misunderstanding the data, ii. Programming the vehicle to not only navigate the intersection reliably and repeatably but also to communicate with the base, and iii. Making a base station which receives requests from all vehicles arriving at the intersection, determined priority, broadcasts right-of-way instructions, and receives completion communicated for each vehicle that leaves the intersection.

3.3.1 XBee Request/Confirm Communication Vectors

As previously mentioned, XBee radio transmitters are the most cost effective and user friendly components for V2V and V2I communication in short distances. These transceivers were incorporated into each of the vehicles and the vehicle were programmed to “listen” for and transmit pertinent information to the established base. The major drawback of utilizing this transmitter-receiver network is that communication in the XBee network cannot easily be assigned from one transmitter to one desired receiver. In other words, communication cannot be easily localized and all transceivers in the same network “hear” all data transmitted similar to the network shown in Figure 3.15.

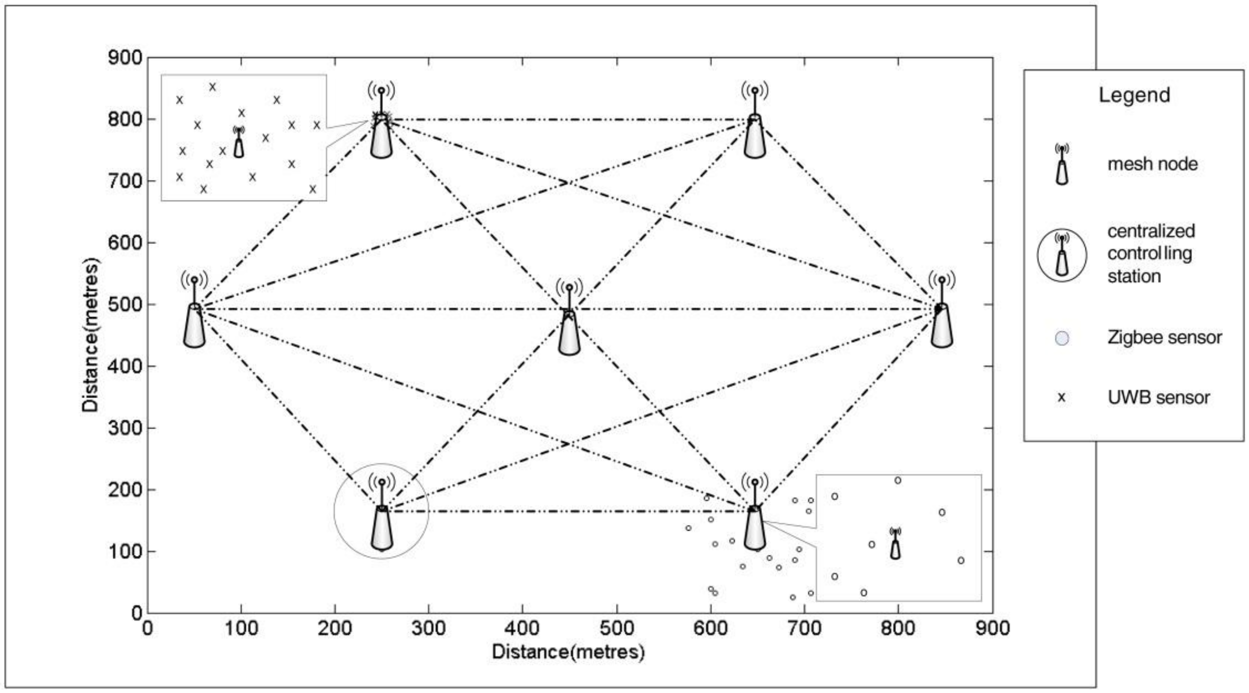


Figure 3.15 Zigbee Wireless Sensor Architecture (Online User Doe. (Techwomen.co) 2017)

The inability to easily have localized communication between the base and incoming vehicles proved to be troublesome as priority must be assigned to the vehicles and each vehicle must be able to decipher if the incoming message is meant for it particularly. This was achieved by limiting all communication to strings of comma separated integers in which the integer's value and position would determine what the message meant. The breakdown of this string is presented in Table 3.2.

Table 3.2 Request string communication definitions

Variable Location (5x1 array)	Definition
0	Destination integer (0: for robots, 1: for base)
1	Vehicle ID number (Assigned to each vehicle)
2	Starting location (Entry leg of intersection)
3	Desired location (Exit leg of the intersection)
4	Completion Integer (0: waiting or travelling, 1: completed navigation)

Position (0) of the array was established to determine if the information received should be registered by the base station. This only became necessary when a human operated computer was included to the system to send "Start command" (to be explained in the proceeding subsection). This avoided miscommunication arising from the base inaccurately assuming the start command was a vehicle at the intersection. Position (1) of the array was used to establish the Vehicle ID of all approaching and exiting vehicles. This was necessary for the base station to identify and categorize messages from each of the vehicles. All right of way messages with that vehicle's ID number included would be ignored by all other vehicles and acknowledged by the intended vehicle. Position (2) of the array was used to communicate what leg of the intersection the vehicle is at the time the message was sent. Position (3) of the array was used to communicate what leg of the intersection the vehicle intended to exit from. The previous two integers were required by the base to plot a path based on the vehicles entry and exit to determine if that path was open and it was safe to let the vehicle through. Finally, Position (4) was used as the indicator to let the base station know that the vehicle has completed its navigation protocol.

This system of introducing an array rather than a single message proved to be very useful and hence, it was decided that it would be used for another important string of information that would make conducting experiments faster and more reliable.

3.3.2 XBee User Command Communication Vectors

When writing the C-based Arduino programs for each of the vehicles, it was decided that all vehicles would have identical code save for one line; the Vehicle ID number. This was done to emulate real world settings where every autonomous vehicle would be programmed identically and every possible path to be taken at an intersection would already be decided and established before the vehicle reached the intersection; similar to how human drivers use Google maps to operate conventional vehicles.

The drawback of this decision would be that every vehicle would have to navigate the intersection by following the same paths, or, every vehicle would have to have all the possible paths of the intersection already programmed into the vehicle at the start of the experiment. The decision was made to use the later method and then assign intersection navigation paths to the vehicles at the start of the intersection using the same XBee network. At the start of each experiment, a "Command Vector", a string array of variable size depending on how many vehicles will be involved in the test, would be sent out to all vehicle detailing which path through the intersection each vehicle should take. The definition of the integers of the Command Vector is presented in Table 3.3.

Table 3.3 “Command Vector” definition for an experiment setup consisting of three vehicles

Variable Location (5x1 array)	Definition
0	Destination integer (Always 0 as it is only meant for the vehicles)
1	Turn value (Left, right or straight)
2	Vehicle [1] designator (0 or 1: tells vehicle with ID 1 if this message is meant for it)
3	Vehicle [2] designator (0 or 1: tells vehicle with ID 2 if this message is meant for it)
4	Vehicle [3] designator (0 or 1: tells vehicle with ID 3 if this message is meant for it)

3.3.3 Experimental Setup

As mention in Chapter 1, the experimental setup was envisioned to be based on the intersection presented in Figure 1.5. Recreating a scaled version of the intersection in an indoor controlled environment however, proved to be tasking. Lack of open spaces with consistent lighting and no obstacles, located indoors were a few of the difficulties encountered during experimental area setup. To alleviate this problem, the decision was made to simplify the intersection into a more standard intersection with for legs intersecting at 90^{o} angles as presented in Figure 3.16.

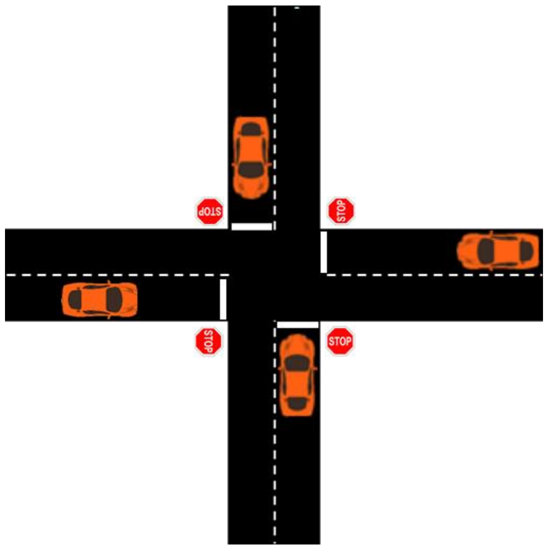


Figure 3.16 Simplified four-way intersection concept

Before proceeding with this modification in the original plan however, the transferability of the envisioned concept in a four-way intersection to the original intersection had to be fully investigated. After much deliberation, a strategy to characterize the four-way intersection which would be transferable to the original intersection as well as any other intersection was envisioned. The first step was to number each leg of the intersection starting on the eastern-most leg and rotating clockwise as shown in Figure 3.17.

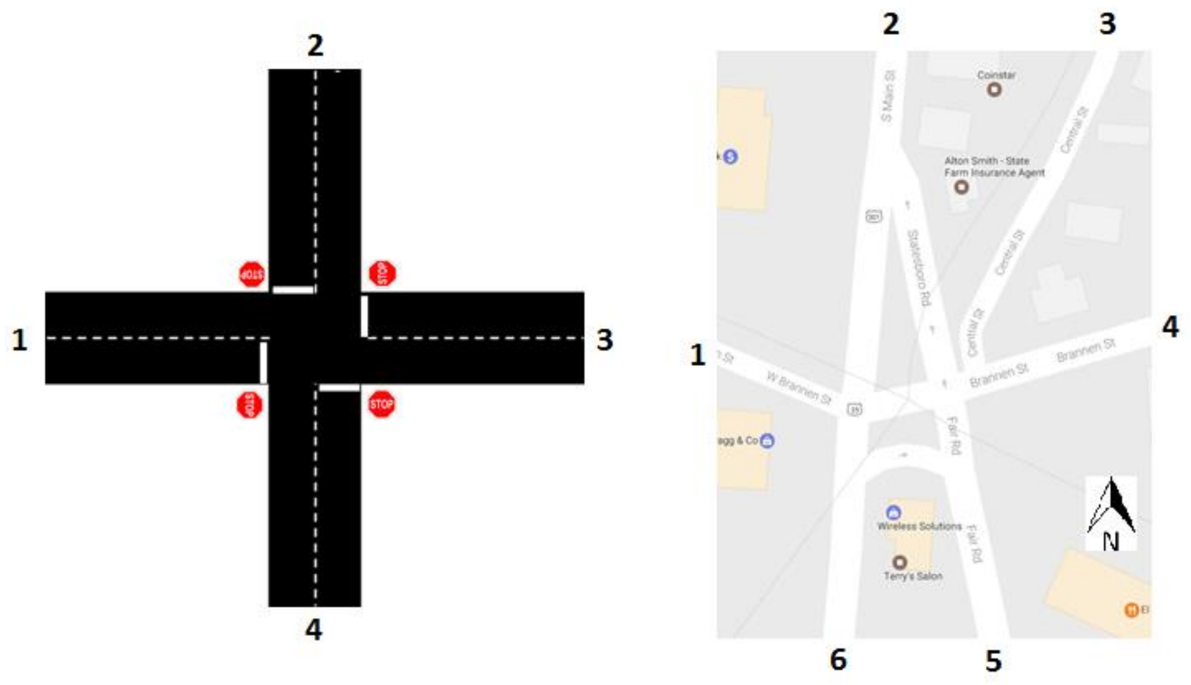


Figure 3.17 Numbered leg strategy used on simplified setup and original intersection

The numbered shown in Figure 3.17 indicate the numbers that would be transmitted during the experiment to represent the legs of the intersection. The request vector sent from the vehicle to the base would specify this entry number and the exit number. The exit number would be obtained by adding the turn value to the entry vector. For example, in the four-way intersection, a left turn will be defined as a “1”, a straight through path will be defined as a “2” and a right turn will be defined as a “3”. This would make a vehicle entering at leg 2, and turning left, to exit at leg “2 + 1 = 3”. In some situations however, a conditional statement had to be implemented to avoid situation of the vehicle appearing to turn to an exit leg that does not exit. This was achieved by including an if-statement into path calculation logic as presented in Equation 3.

Equation 3 Logic required to calculate exit path from request vector

$$ex = en + tn \qquad \text{(eq. 3a)}$$

$$if\ ex > tot; ex = ex - tot \qquad \text{(eq. 3b)}$$

ex = exit leg of the vehicle
en = entry leg of the vehicle
tn = turn value of the vehicle's path
tot = total number of intersection legs

These equations would prove applicable to any kind of intersection as there will be a permanent, global definition of all intersection legs and, as the number of turn values can be increased, each turn would be easily programmable into the small scale vehicle,

Acknowledging the transferability of the envisioned logic between the four-way intersection and any other kind of intersection, the four-way setup was used as the base model for all further testing in this thesis.

3.4 Vehicle and Base-station Programming Logic

3.4.1 C based Vehicle Control Algorithm

As discussed in Sub-chapter 3.3, the small scale vehicle was reliably and repeatably able to navigate the intersection, making it the ideal subject to test the intersection management communication. A program was written and uploaded which would have the vehicle start with a given path, follow the lane marker until a stop sign is recognized, recognize the stop sign, request priority by communicating with the base station, wait for an approval vector, and finally navigate the intersection based on the initially programmed path (left, right or straight through). The vehicle code is presented as a block diagram in Figure 3.18.

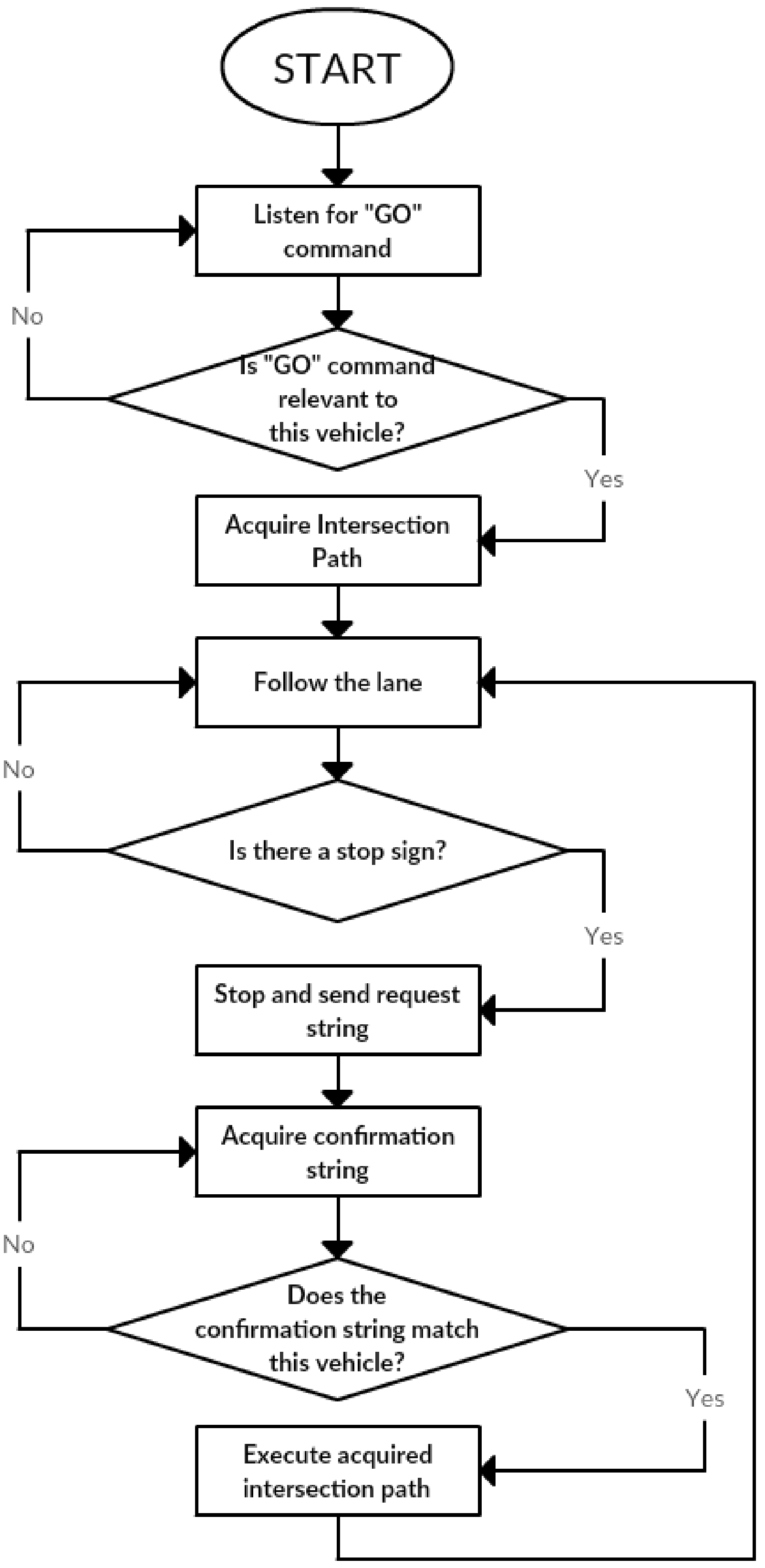

START
Listen for "GO" command
Is "GO" command relevant to this vehicle?
No
Yes
Acquire Intersection Path
Follow the lane
Is there a stop sign?
No
Yes
Stop and send request string
Acquire confirmation string
Does the confirmation string match this vehicle?
No
Yes
Execute acquired intersection path

Figure 3.18 Logic Diagram of the vehicle control program

3.4.2 C++ based Programming of the Base Station

The initial plan for the programming of the base station involved using a microcontroller similar to the one controlling the small scale vehicles to receive the requests from the small scale vehicle, process priority, and broadcast approval string to the waiting vehicles. This proved to be difficult for a microcontroller running C-based programming logic to accomplish due to the procedural nature of the C programing language. This style of programming, albeit involving numerous subroutines and other advance programming techniques, still had the drawback of being only able to execute one line of code at a time; either receiving new requests, or processing received requests and broadcasting approval strings. This would mean that a new incoming vehicle could possibly have its requesting string ignored simply because the base station was processing another request.

Even though the likelihood of two vehicles arriving so close to each other, as to overlap a request signal and request processing, was low for the experimental setup, in periods of high traffic flow in the real world this will most certainly happen. To alleviate this problem, a new method of receiving and processing requests had to be implements to essentially make the base station accomplish two tasks concurrently. To accomplish this, C++; a widely used object oriented programming language which diverts from procedural coding, needed to be implemented base station. This however required some kind of an interface that would link systems running under both aforementioned languages. Robot Operating System (ROS) proved to the solution to this problem.

The Robot Operating System (ROS) is a flexible framework for writing robot software. It comprises of a collection of tool, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide range of robotic platforms (ROS.org 2017). It is essentially a user friendly method of implementing complex programming and communication between differing mechatronic systems (in the case of this these C-based and C++ based systems). The system proved to be complex to learn but owing to the open source documentation available for the platform, an object oriented programming logic was developed which would successfully accomplish the task of the base station.

3.4.3 Complete Programming Framework of the Base Station

As previously mentioned, multiple tasks had to be undertaken simultaneously by the base station in order to make for an effective system for autonomous intersection navigation. To accomplish this using ROS, a system of nodes (subprograms capable of running simultaneously while communicating with each other using topics) and topics (named buses over which nodes exchange information) was utilized as shown in Figure 3.19.

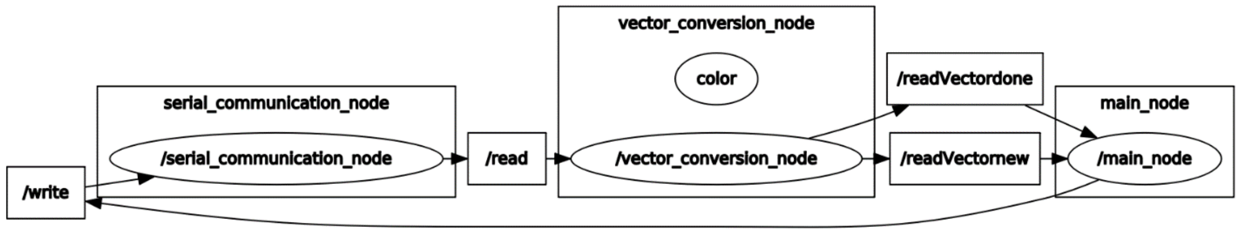


Figure 3.19 Residual Quadtree (RQT) displaying complete ROS base process

Each of these nodes have their own specific purpose within the communication protocol. The serial communication node, presented in Figure 3.20, opens up a serial port between the base processor and the XBee module connected to the base

3.4.3.1 Serial Communication Node

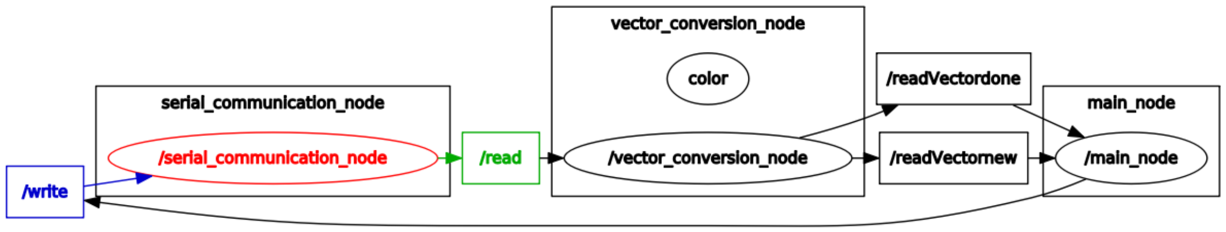


Figure 3.20 Residual Quadtree (RQT) emphasizing the serial communication node

This node established a baud rate of 57600 a set port of “/dev/ttyUSB0”. This baud rate or number of bits transmitted per second, was chosen as the system baud rate based on a careful balance of speed of data transfer and reliability. Owing to the fact that the vehicular micro-controllers have an upper limit of processing speed and hence risk the errors in the communicated data on the receiving end (Jimb0 2010), the aforementioned baud rate was chosen over the more standard but slower 9600. The speed of data transfer was necessary to improve the responsiveness of the test vehicles and the base, as less time will be spent during serial communication.

The “serial communication node” read all incoming data string and published them in the “read” topic which would be used by other codes in the program for prioritizing. This node was also subscribed to the “write” topic which was constantly updated by the other nodes in the program and any new changes in the “write” topic were instantaneously sent through the XBee serial port which in turn broadcasted the data string as a wireless message. This logic is presented in Figure 3.21.

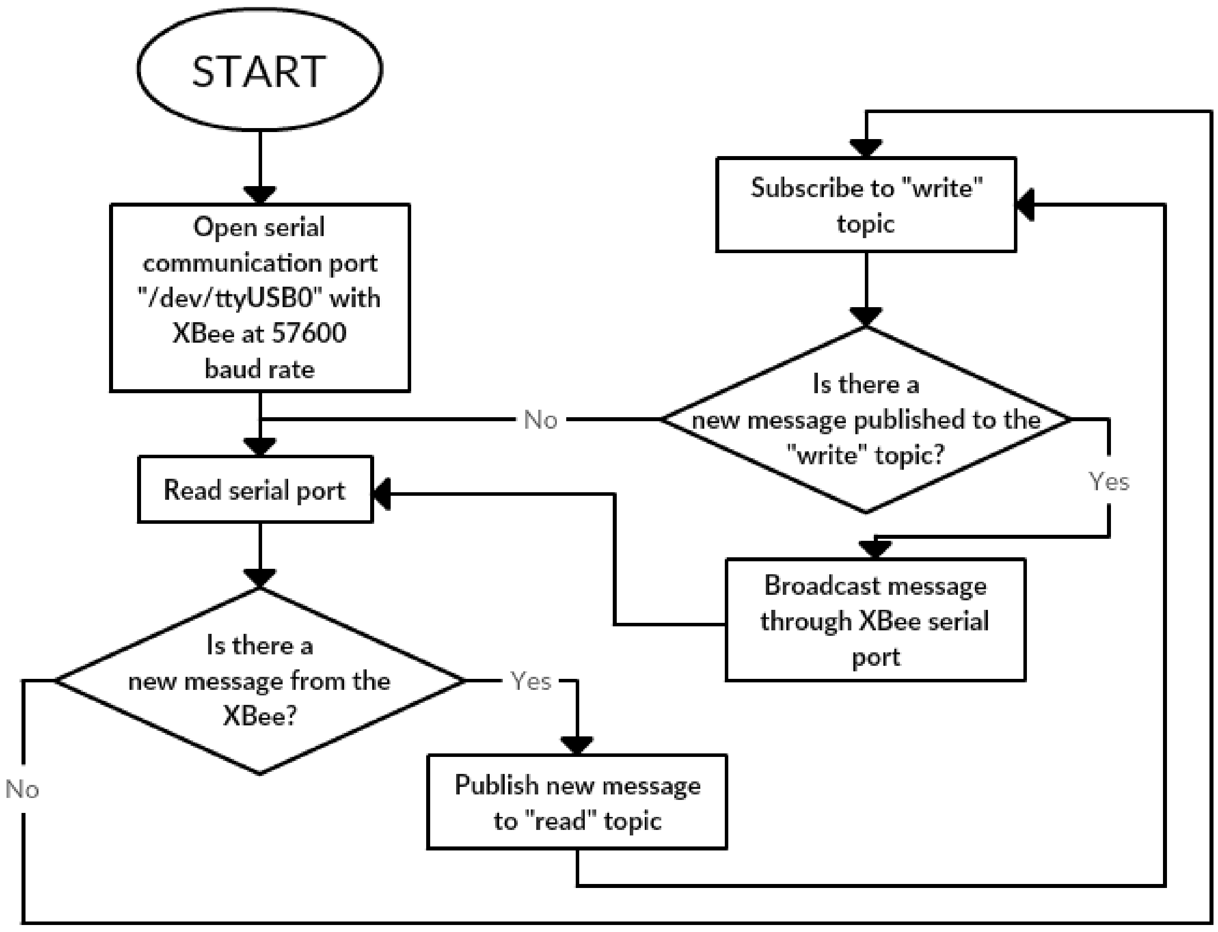

START
Open serial communication port "/dev/ttyUSB0" with XBee at 57600 baud rate
Read serial port
Is there a new message from the XBee?
Yes
No
Publish new message to "read" topic
Subscribe to "write" topic
Is there a new message published to the "write" topic?
No
Yes
Broadcast message through XBee serial port

Figure 3.21 Logic Diagram of Serial Communication Node

The aforementioned “read” topic consists of strings of comma separated integers which were received from the vehicles as the intersection requesting priority. The string format was used because the vehicle microcontrollers communicated using these strings and the “parseInt” feature. The consisting integers as mentioned in Table 3.2 are immensely significant towards assigning priority and hence each integer had to be parsed and analyzed by the main part of the program. The problem with this premise is that ROS cannot easily accomplish this task on a string message type.

3.4.3.2 Vector Conversion Node

To circumvent this problem, a “vector conversion node” was subscribed to the “read” topic which analyzed all the communications published there and converted the sting into a vector consisting of each of the comma separated integers. After performing the conversion, the first integer which determined if the string was meant to be read by the base station (Table 3.2), as well as the last digit which determined if the message was a request or a confirmation, were analyzed to determine the message’s relevance in the ensuing program. If it was determined by the node that the message was indeed meant for the base station, it was organized into the “readVectornew” and “readVectordone” topics, as shown in Figure 3.22, which would contain vehicle requests and approvals respectively. This logic is presented in Figure 3.23.

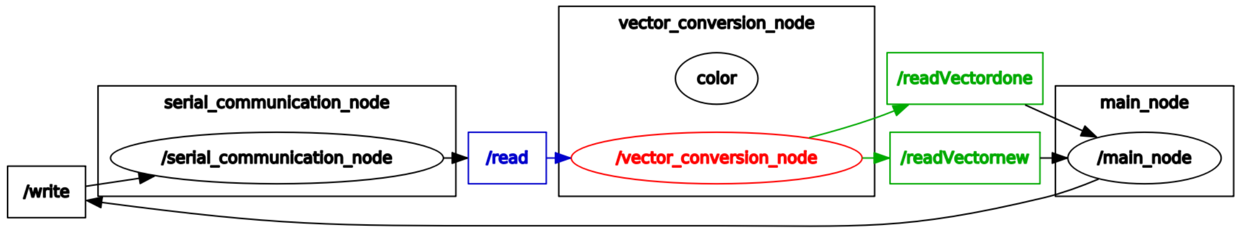


Figure 3.22 Residual Quadtree (RQT) emphasizing the vector conversion node

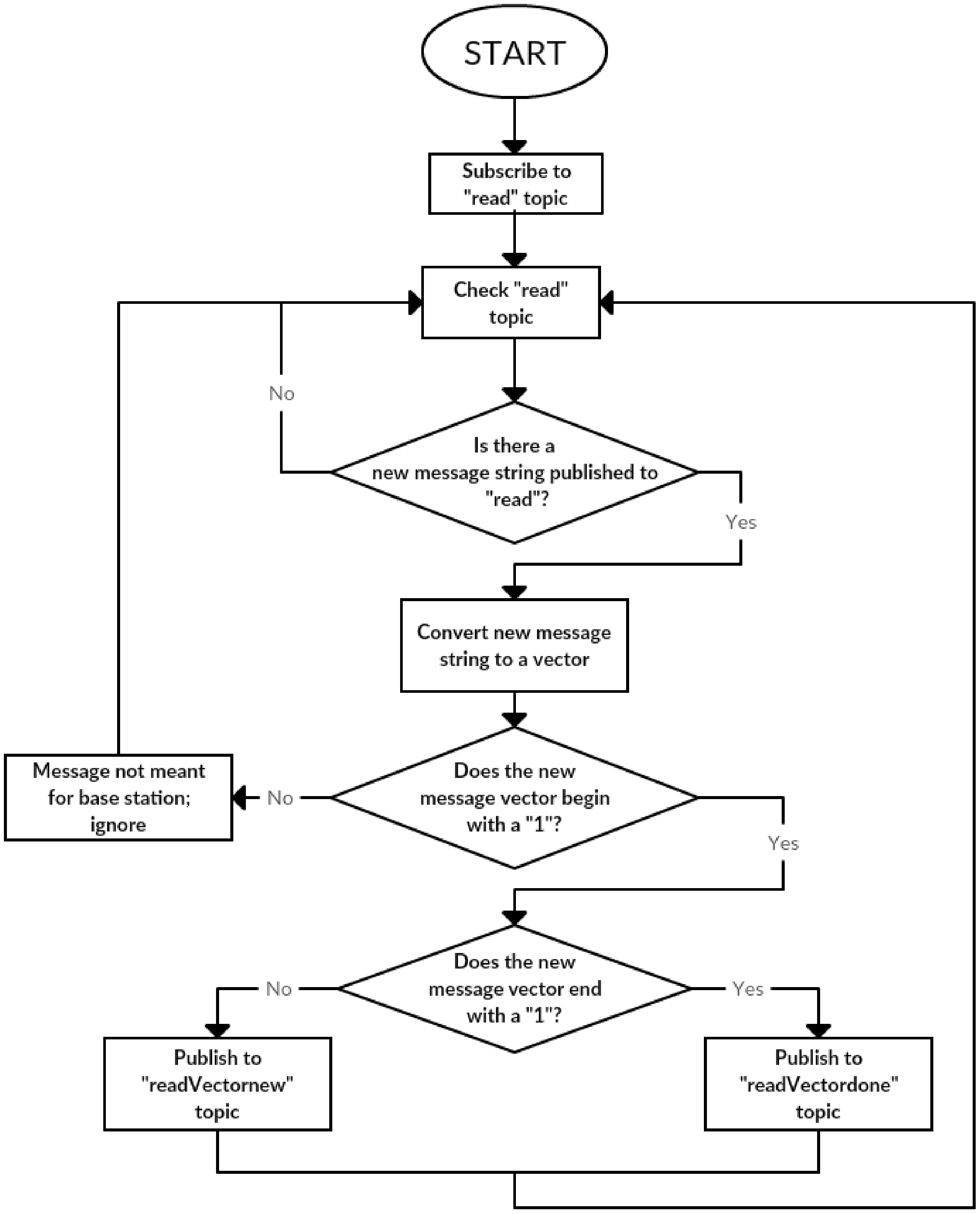

START
Subscribe to "read" topic
Check "read" topic
No
Is there a new message string published to "read"?
Yes
Convert new message string to a vector
Message not meant for base station; ignore
No
Does the new message vector begin with a "1"?
Yes
No
Does the new message vector end with a "1"?
Yes
Publish to "readVectornew" topic
Publish to "readVectordone" topic

Figure 3.23 Logic Diagram of the Vector Conversion Node

3.4.3.3 The Main Node

The creation of the “readVectornew” and “readVectordone” topics were important because, as shown in Figure 3.24, the “main node” was subscribed to both of them and consisted of two call-back functioned which were activated depending on if the read vector message was received from a new vehicle arriving at the intersection, or from a vehicle confirming the completion of its preplanned path respectively.

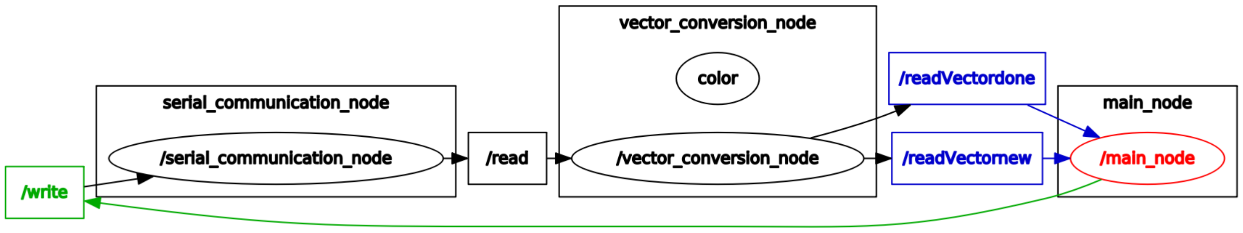


Figure 3.24 Residual Quadtree (RQT) emphasizing the main node

The new vehicle call back function in the main node was programmed to create a new vehicle object and propagated the objects relevant information, such as vehicle ID, paths, arrival intersection leg and destination leg, based on the vector message received from the “readVectornew” topic. The vehicles’ priority would then be assigned the first vehicle and every other vehicle arriving at the intersection. After calling this function, the main node proceeds to publish a request approval string to the “write” topic which in turn is broadcasted by the serial communication node to the highest priority vehicle. The main node was also programmed to close all of the roads and paths that may be affected by the navigation of this first arriving vehicle. This would prevent a collision between in the intersection as well as allow the navigation of multiple vehicles concurrently provided that their paths do not interfere or intersect.

The done vehicle call-back function in the main node was programmed to recognize when a vehicle has successfully navigated the intersection and then open all paths that were closed by the initial vehicles’ navigation. This will let the next waiting vehicle with the highest priority navigate the intersection. It should be noted that vehicle priority is based on time of arrival as well as that vehicles’ path as it compares to the “open” roads and path at that particular time in the intersection. This allows for vehicles to navigate the intersection concurrently in instances where both vehicles’ paths do not intersect such as in the case of two opposing vehicles navigating straight, or any combination of vehicle turning right as shown in Figure 3.25. This logic for the main is presented in Figure 3.26.

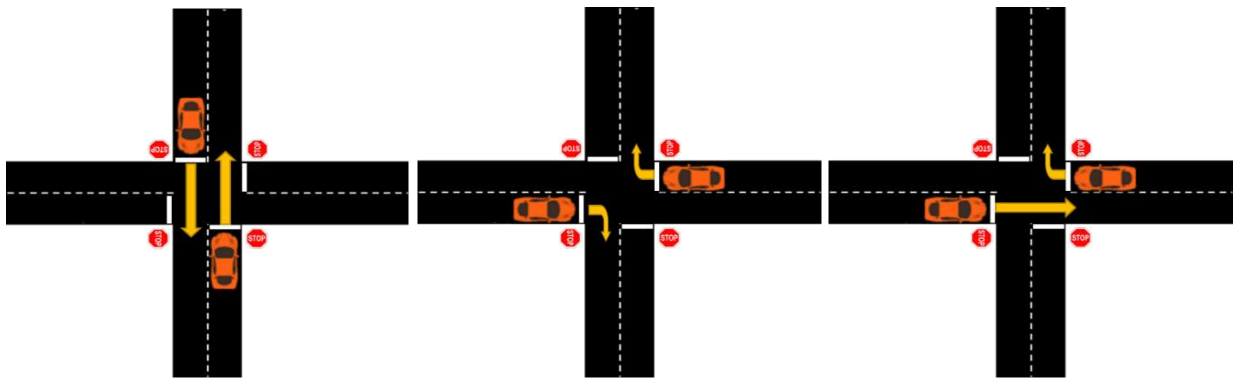


Figure 3.25 Situations for Accepted Concurrent Vehicle Navigation

An example of what path would be closed for any arriving vehicle while navigating is presented in Figure 3.27 for a vehicle entering the intersection in Leg 1 and exiting in Leg 2. This constitutes a left turn and would require the most number of paths closed. This concept is implemented in the main node when a vehicle is allowed to navigate the intersection. A list of all closed path combinations for every requested vehicle path is presented in Table 3.4

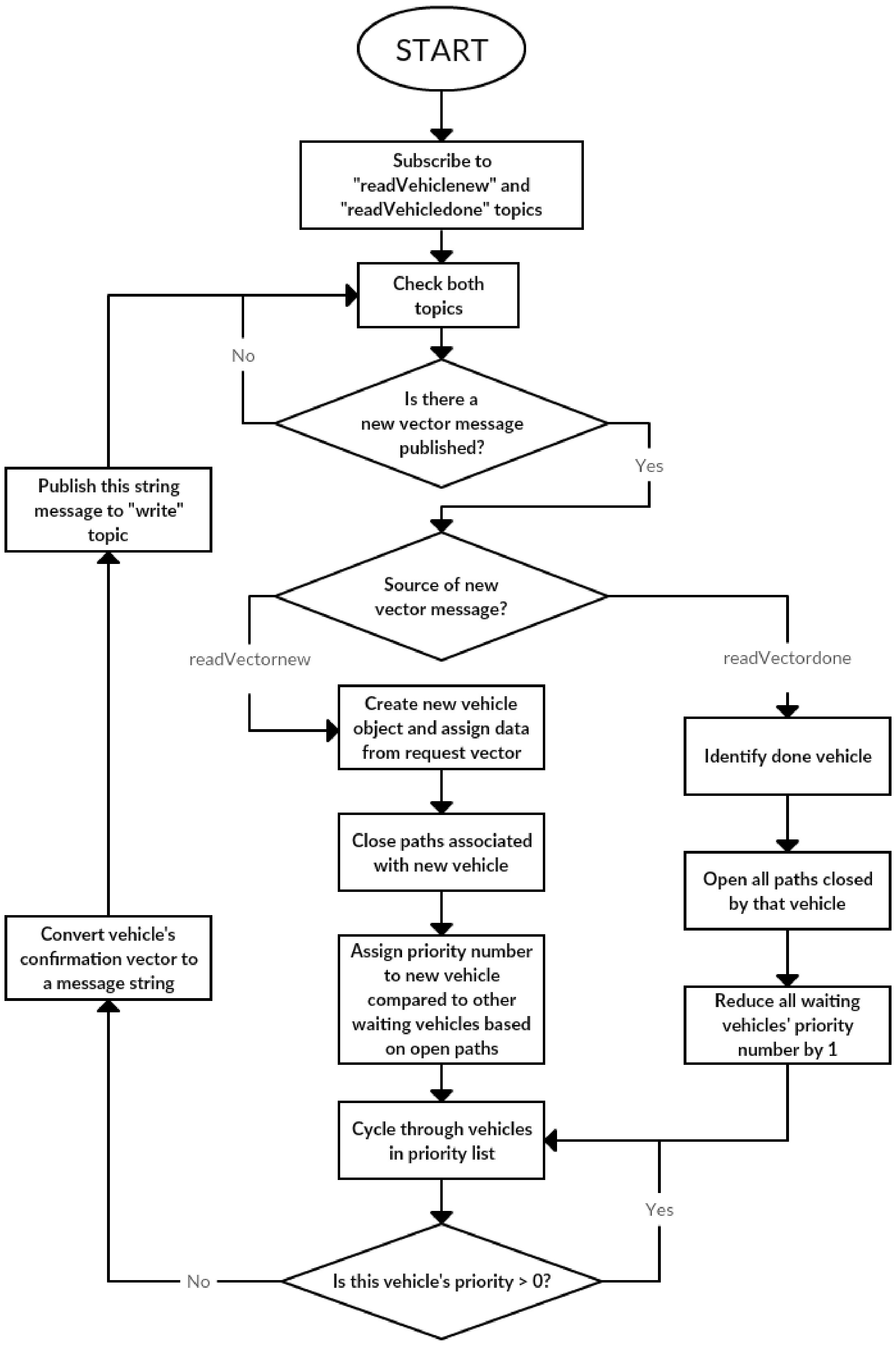

START
Subscribe to "readVehiclenew" and "readVehicledone" topics
Check both topics
Is there a new vector message published?
No
Yes
Source of new vector message?
readVectornew
readVectordone
Create new vehicle object and assign data from request vector
Close paths associated with new vehicle
Assign priority number to new vehicle compared to other waiting vehicles based on open paths
Identify done vehicle
Open all paths closed by that vehicle
Reduce all waiting vehicles' priority number by 1
Cycle through vehicles in priority list
Is this vehicle's priority > 0?
Yes
No
Convert vehicle's confirmation vector to a message string
Publish this string message to "write" topic

Figure 3.26 Logic Diagram for the Main Node

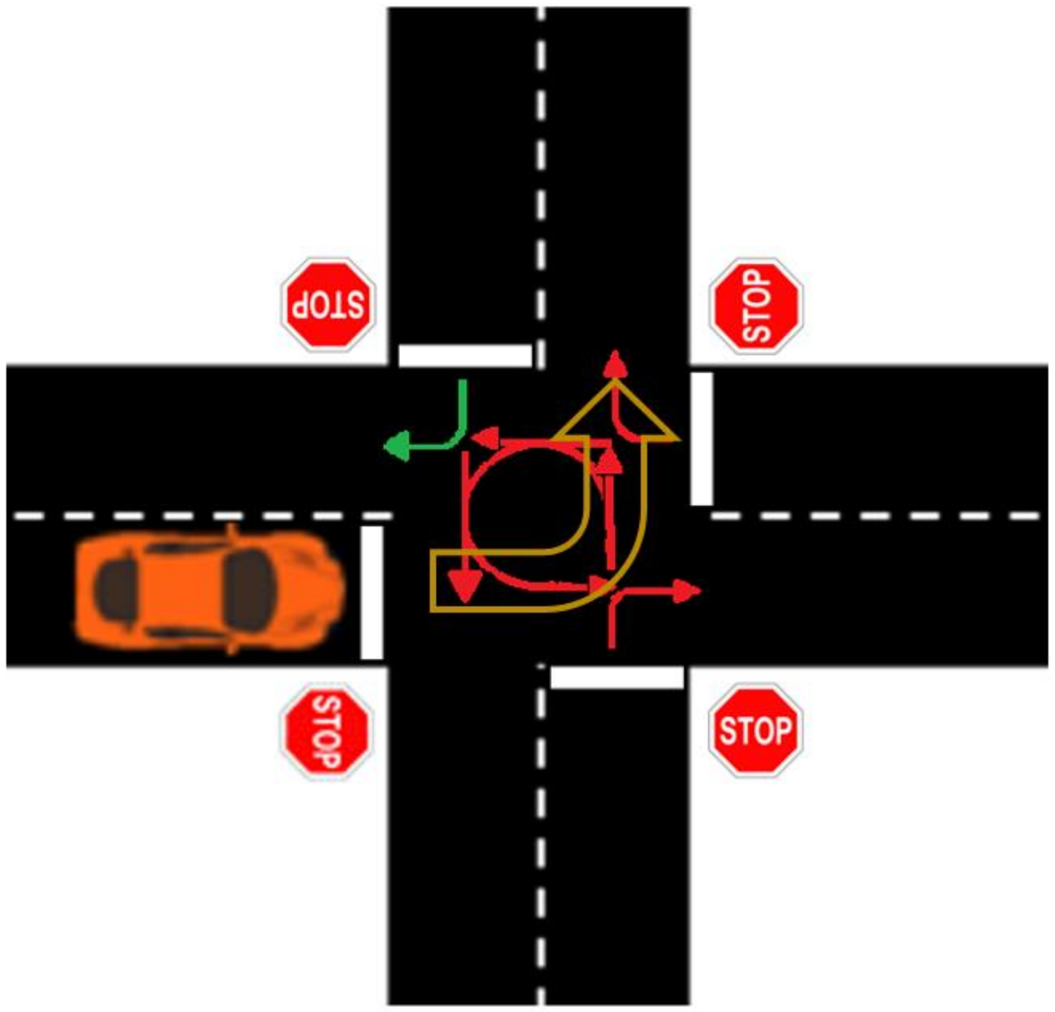


Figure 3.27 Requesting vehicle's path (gold) and subsequently closed paths (red) and open paths (green)

Table 3.4 Depiction of the affected path closed for every requested path navigation

		1-2	1-3	1-4	2-1	2-3	2-4	3-1	3-2	3-4	4-1	4-2	4-3
1-2	Left	1	0	0	0	1	1	1	1	1	1	1	1
1-3	Straight	0	1	0	0	1	1	0	0	1	1	1	1
1-4	Right	0	0	1	0	1	1	0	0	1	0	0	0
2-1	Right	0	0	0	1	0	0	1	0	1	1	0	0
2-3	Left	1	1	1	0	1	0	1	0	1	1	1	1
2-4	Straight	1	1	1	0	0	1	1	0	1	1	0	0
3-1	Straight	1	0	0	1	1	1	1	0	0	1	1	0
3-2	Right	1	0	0	0	0	0	0	1	0	1	1	0
3-4	Left	1	1	1	1	1	1	0	0	1	1	1	0
4-1	Left	1	1	0	1	1	1	1	1	1	1	0	0
4-2	Straight	1	1	0	0	1	0	1	1	1	0	1	0
4-3	Right	1	1	0	0	1	0	0	0	0	0	0	1

In Table 3.4 the left column represents the path requested by the approaching vehicle. The top row depicts the possible paths that could be closed based on the request of the incoming vehicle. The 0s and 1s determine if that particular path is closed (denoted as a 1 to signify the presence of a vehicle) or open (denoted as a 0 to signify the absence of a vehicle). As shown in the figure, left turn requests cause for the most closed paths; 9 out of 12, right turn requests, cause for the least closed paths; 4 out of 12, and the straight path causes for the closure of 7out of 12 path. These values were used to create vectors which would be tied to the vehicles requesting navigation. While the vehicles await approval in the priority list, the main node constantly compare closed road vectors to determine if any of the awaiting vehicles can begin navigation despite the fact that there is a vehicle currently navigating the intersection. This allows for a drastically shorter wait time than a system which waits for each vehicle to finish navigating before approving the next waiting vehicle.

3.4.4 Autonomous Communication Module

As appealing as the idea of Autonomous Intersection Management is, it will not be implementable to its full capacity without total level 5 autonomy on every vehicle attempting to navigate such an enabled intersection. The average vehicle today would have to rely on conventional intersection management tools to avoid congestions or accidents at said intersections. To circumvent this problem, aftermarket equipment can be obtained and installed in a non-intelligent vehicle which could connect to the vehicle's data bus, send and receive Basic Safety Messages (BSM), and provide advisories/warning. These equipment currently range from $252 to $291 depending on the number manufactured (Harding et al. 2014).

The same predicament would apply to the system developed for this thesis. If any vehicle approaching the vehicle is not XBee/intelligent enabled, it could disrupt the operations in the intersections and possibly cause confusion or an accident. It was decided that a separate XBee enabled module with its own microprocessor capable of sending and receiving messages would be developed. This module would be pluggable into a conventional vehicle and contain pushbuttons which would request priority in the same format as the small scale intelligent vehicle prototypes. When the vehicle receives approval from them base, the vehicle would then navigate the intersection and relay a completion message, in the same format at an intelligent vehicle, to open up the intersection for other vehicles. The vehicle wiring diagram and vehicle logic for this module are presented in Figure 3.28

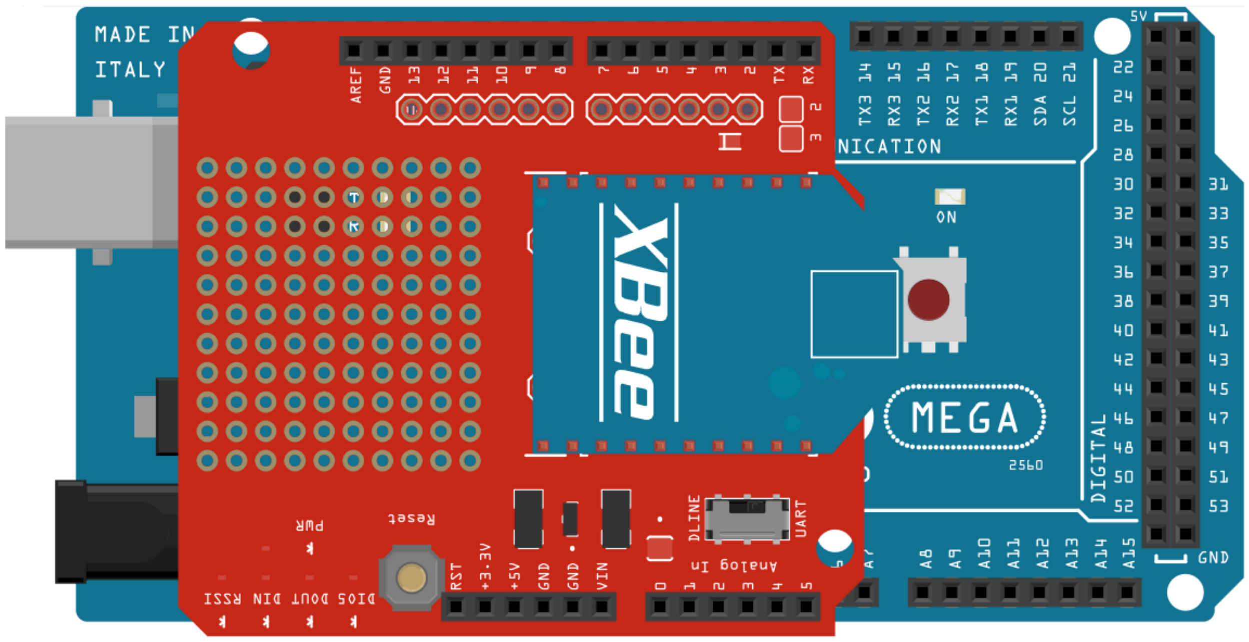


Figure 3.28 Visual representation of the autonomous communication module

CHAPTER 4

RESULTS

To evaluate the effectiveness of the completed base station, quantitative data was taking on every possible combination of vehicles arriving at the intersection and requesting priority in a controlled environment to establish the viability of the system. This data indicates the efficacy of V2I communication using comma-separated variables over an XBee radio transceiver.

4.1 Determining Test Runs for Data Collection

`The 1:4.1 scale vehicle prototypes were placed in each leg of the intersection and the start commands presented in Table 4.1 were broadcasted at the start of each test to all the test vehicles.

Table 4.1 Start commands broadcasted at the start of each test run

Test No.	Vehicle 0	Vehicle 1	Vehicle 2	Start Commands
1	Left	Left	Left	0,1,1,1,1
2	Left	Left	Straight	0,1,1,1,0 / 0,2,0,0,1
3	Left	Left	Right	0,1,1,1,0 / 0,3,0,0,1
4	Left	Straight	Left	0,1,1,0,1 / 0,2,0,1,0
5	Left	Straight	Straight	0,1,1,0,0 / 0,2,0,1,1
6	Left	Straight	Right	0,1,1,0,0 / 0,2,0,1,0 / 0,3,0,0,1
7	Left	Right	Left	0,1,1,0,1 / 0,2,0,1,0
8	Left	Right	Straight	0,1,1,0,0 / 0,3,0,1,0 / 0,2,0,0,1
9	Left	Right	Right	0,1,1,0,0 / 0,3,0,1,1
10	Straight	Left	Left	0,2,1,0,0 / 0,1,0,1,1
11	Straight	Left	Straight	0,2,1,0,1 / 0,1,0,1,0
12	Straight	Left	Right	0,2,1,0,0 / 0,1,0,1,0 / 0,3,0,0,1
13	Straight	Straight	Left	0,2,1,1,0 / 0,1,0,0,1
14	Straight	Straight	Straight	0,2,1,1,1
15	Straight	Straight	Right	0,2,1,1,0 / 0,3,0,0,1
16	Straight	Right	Left	0,2,1,0,0 / 0,3,0,1,0 / 0,1,0,0,1
17	Straight	Right	Straight	0,2,1,0,1 / 0,3,0,1,0
18	Straight	Right	Right	0,2,1,0,0 / 0,3,0,1,1
19	Right	Left	Left	0,3,1,0,0 / 0,1,0,1,1
20	Right	Left	Straight	0,3,1,0,0 / 0,1,0,1,0 / 0,2,0,0,1
21	Right	Left	Right	0,3,1,0,1 / 0,1,0,1,0
22	Right	Straight	Left	0,3,1,0,0 / 0,2,0,1,0 / 0,1,0,0,1
23	Right	Straight	Straight	0,3,1,0,0 / 0,2,0,1,1
24	Right	Straight	Right	0,3,1,0,1 / 0,2,0,1,0
25	Right	Right	Left	0,3,1,1,0 / 0,1,0,0,1
26	Right	Right	Straight	0,3,1,1,0 / 0,2,0,0,1
27	Right	Right	Right	0,3,1,1,1

This made for 27 separate test which encompassed every possibility that would arise at a similar intersection in the real word. It was originally envisioned to carry out multiple runs of the same test number to determine if the order of the time of arrival would affect priority and the output of approval commands from the base station. This proved to be redundant, as the vehicles, programmed to always reach the intersection in the order of their Vehicle ID would enact all the turn scenarios in which other turn combination would be reminiscent of the same initial test combination with a different arrival time for each vehicle. Taking for example Tests 6, 12 and 22 from Table 4.1. All these runs feature the same combination of Left, Right and Straight but in varying order. By having the vehicles always approaching the intersection in the order of their priority number but completing Tests 6, 12 and 22, for all intents and purposes the base will recognize the same combination but with varying times of arrival there by testing multiple temporal possibilities concurrently without the need for redundant and unnecessary testing.

4.2 Results of Intersection Management by the Developed System

The data collected from all the runs presented in Table 4.1 was recorded on video and presented as a graph for the purpose of this thesis for visual representation in keeping with presentation styles of other researcher mentioned in the Literature Review. A brief description of the results and graph is presented for each test run. In the graphs, ‘0’ in the place of Navigation progress represents that the vehicle is either approaching or exiting the intersection, ‘1’ represents that the vehicle is waiting at the stop sign and ‘2’ represents that the vehicle is navigating the intersection (turning left, right or proceeding straight).

4.2.1 Test 1, All Vehicles Turn Left

In this test, all approaching vehicles were programmed to turn left after receiving approval from the base station to navigate the intersection. The results of the test are presented in Figure 4.1 through Figure 4.27. The vehicle behaviors are presented graphically in Figure 4.1.

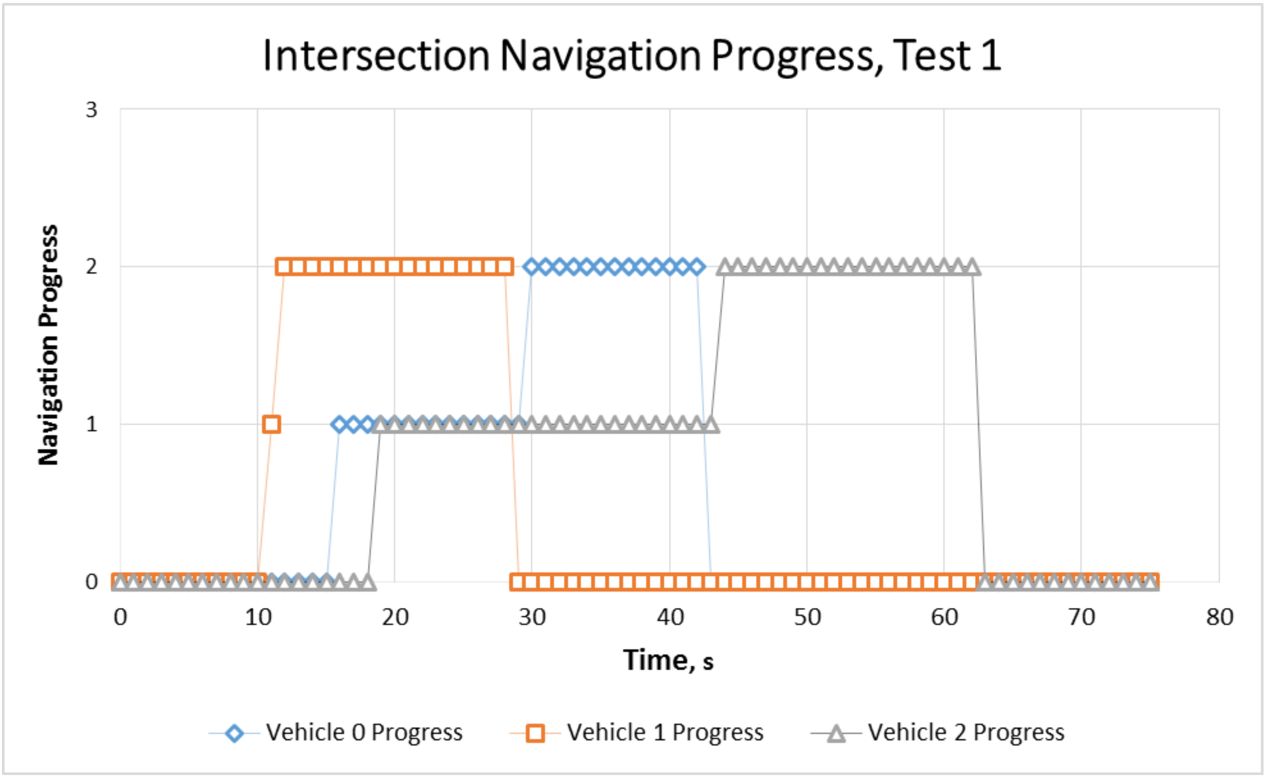


Figure 4.1 Visual representation of the vehicles navigating in Test 1

In this test, Vehicle 1 arrived at the intersection before Vehicle 0, requesting to turn left, and had a wait time of 1s before being approved by the base to navigate the intersection. Vehicle 0 arrived shortly after and was approved to proceed only with its left turn only after Vehicle 1 had completed its planned intersection navigation path which made for a wait time of 14s. The third vehicle, Vehicle 2, was then allowed to navigate the intersection after recording a wait time of 25s.

4.2.2 Test 2, Vehicles 0 and 1 turn left; Vehicle 2 proceeds straight

The second test was implemented in which the combination of start commands which had Vehicle 0 and 1 turn left, and Vehicle 2 proceed straight through the intersection was broadcasted to all the test vehicles. The vehicle behaviors are presented graphically in Figure 4.2.

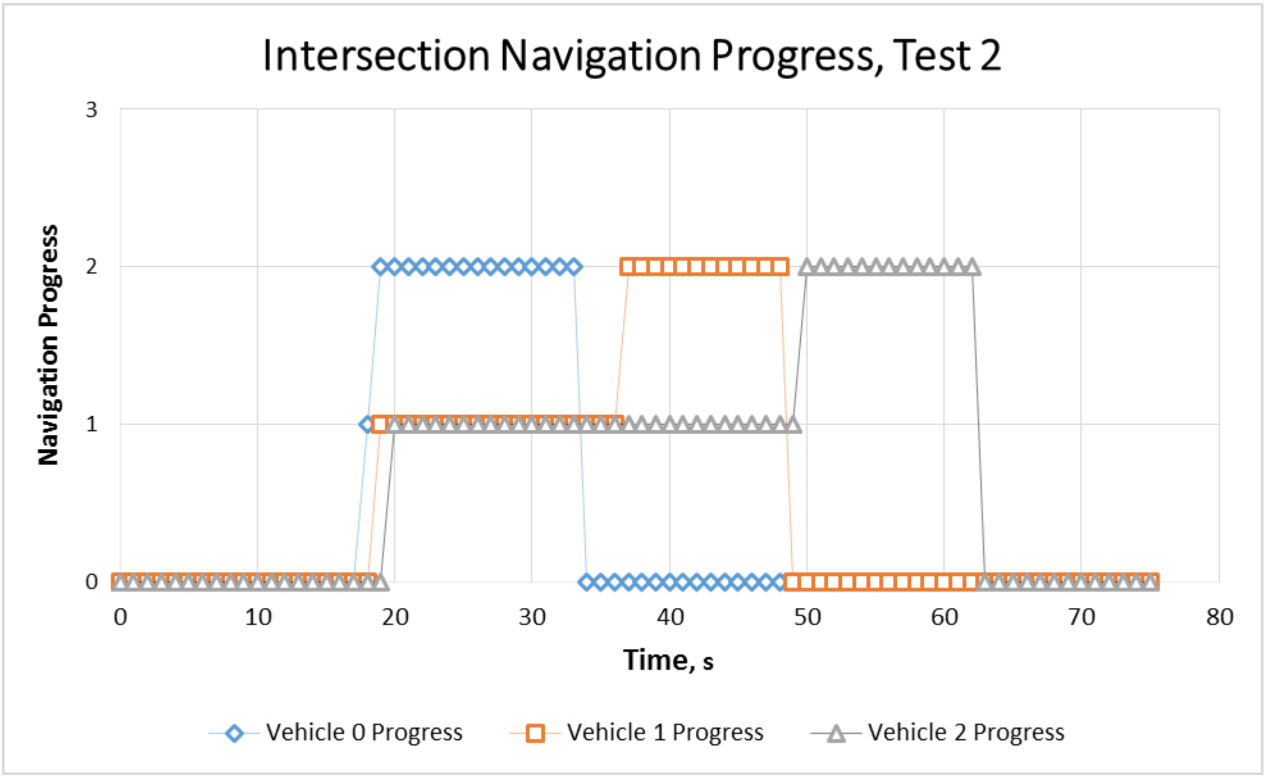


Figure 4.2 Visual representation of the vehicles navigating in Test 2

In this test, Vehicle 0 arrived at the intersection virtually the same time as Vehicle 1 and Vehicle 2, however, due to the fact that the base station recorded Vehicle 0's request string first, it was allowed to proceed first with only a 1s wait time. Vehicle 1 proceeded next with a 18s wait time and after it had completed its navigation, Vehicle 2 was allowed to proceed with a wait time of 30s.

4.2.3 Test 3, Vehicles 0 and 1 turn left; Vehicle 2 turns right

The third test was implemented in which the combination of start commands which had Vehicle 0 and 1 turn left, and Vehicle 2 turn right when navigating the intersection was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.3.

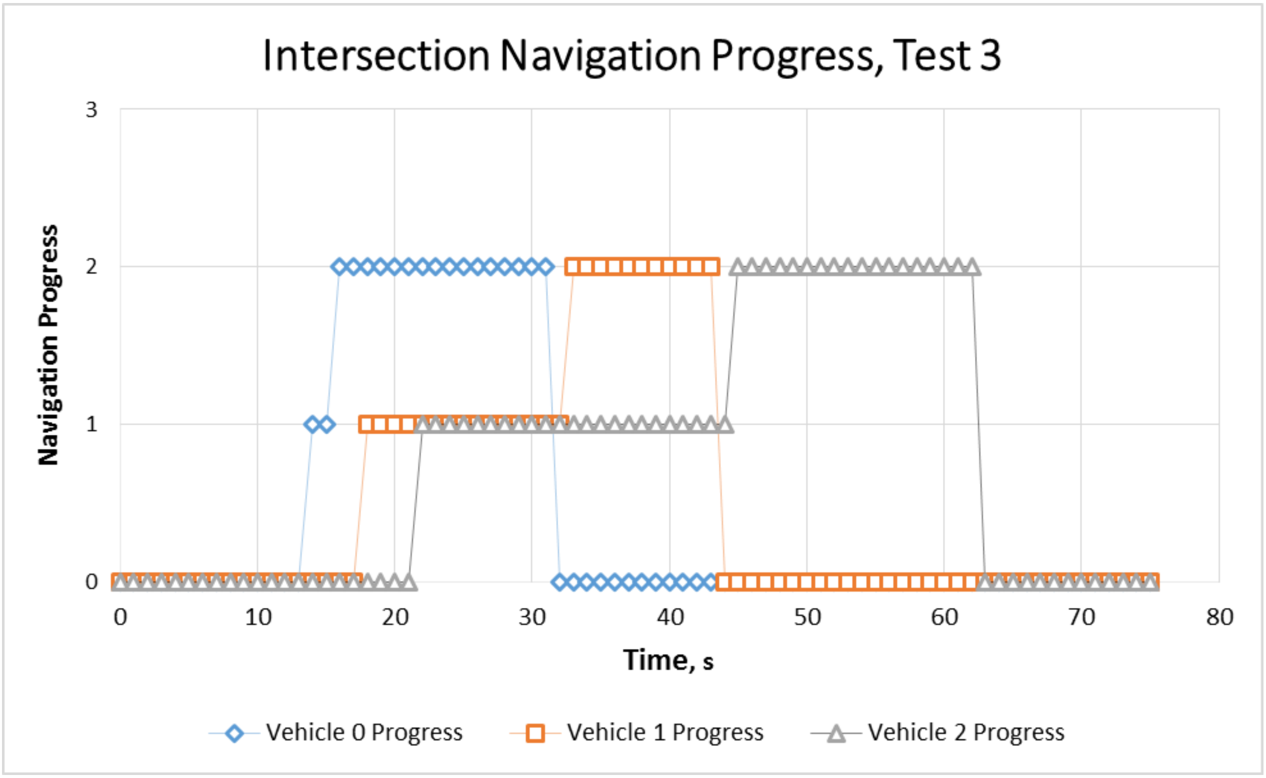


Figure 4.3 Visual representation of the vehicles navigating in Test 3

In this test, Vehicle 0 arrived first and recorded a wait time of 2s before receiving approval to navigate the intersection. While it was navigating, Vehicle 1 and Vehicle 2 arrived at the intersection and broadcasted their request strings, awaiting approval. After Vehicle 0 completed its navigation protocol, Vehicle 1 and then Vehicle 2 navigated the intersection one after the other, after receiving their respective approval strings from the base station.

4.2.4 Test 4, Vehicles 0 and 2 turn left; Vehicle 1 proceeds straight

The fourth test was implemented in which the combination of start commands which had Vehicle 0 and Vehicle 2 turn left, Vehicle 1 proceed straight through the intersection was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.4.

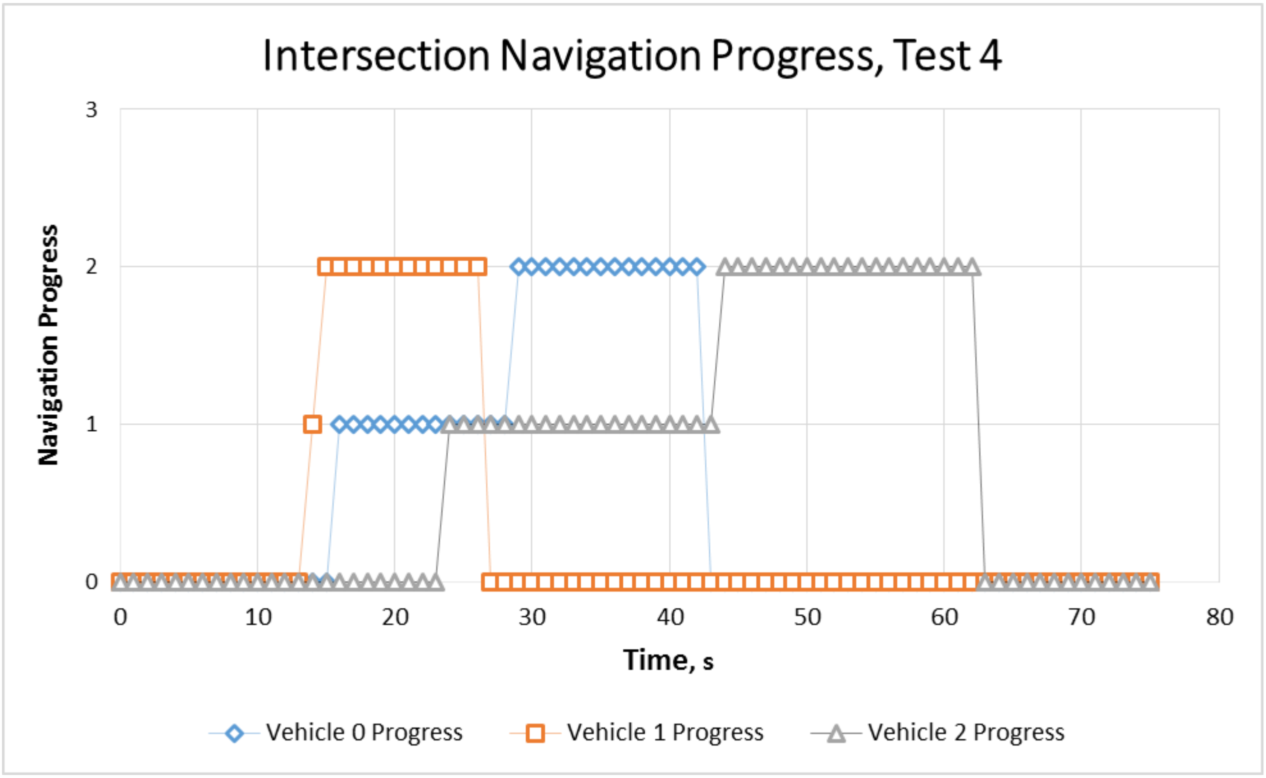


Figure 4.4 Visual representation of the vehicles navigating in Test 4

In this test, Vehicle 0 arrived first at the intersection and proceeded to complete its navigation protocol after recording a 1s wait time. Vehicle 1 arrived at the intersection shortly after and recorded a wait time of 13s before receiving its approval string from the base. Vehicle 2 was the last to proceed with its navigations protocol after recording a wait time of 20s.

4.2.5 Test 5, Vehicles 0 turns left; Vehicle 1 and 2 proceed straight

The fifth test was implemented in which the combination of start commands which had Vehicle 0 turn left, Vehicle 1 and Vehicle 2 proceed straight through the intersection was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.5.

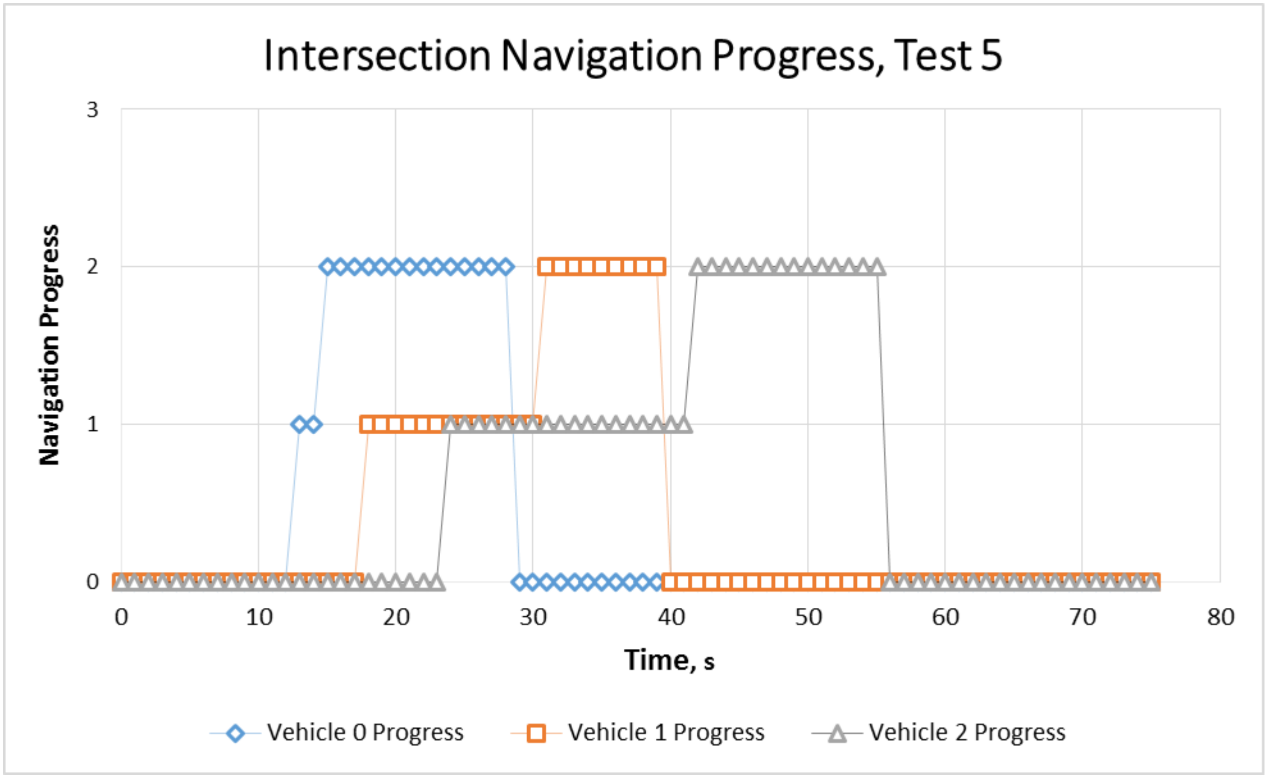


Figure 4.5 Visual representation of the vehicles navigating in Test 5

In this test, Vehicle 0 arrived first at the intersection and recorded a wait time of 2s, before receiving its approval string and proceeding through the intersection. Vehicle 1 arrived shortly afterwards and recorded a wait time of 13s before receiving its approval string and proceeding with its programmed navigation protocol. Vehicle 2 arrived while Vehicle 0 was still completing its navigation protocol but did not receive its approval string until Vehicle 1 had completed its navigation protocol, recording a wait time of 18s.

4.2.6 Test 6, Vehicle 0 turns left, Vehicle 1 proceeds straight, Vehicle 2 turns right

The sixth test was implemented in which the series of start commands which had Vehicle 0 turn left, Vehicle 1 proceed straight through the intersection and Vehicle 2 turn left in the intersection was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented in Figure 4.6.

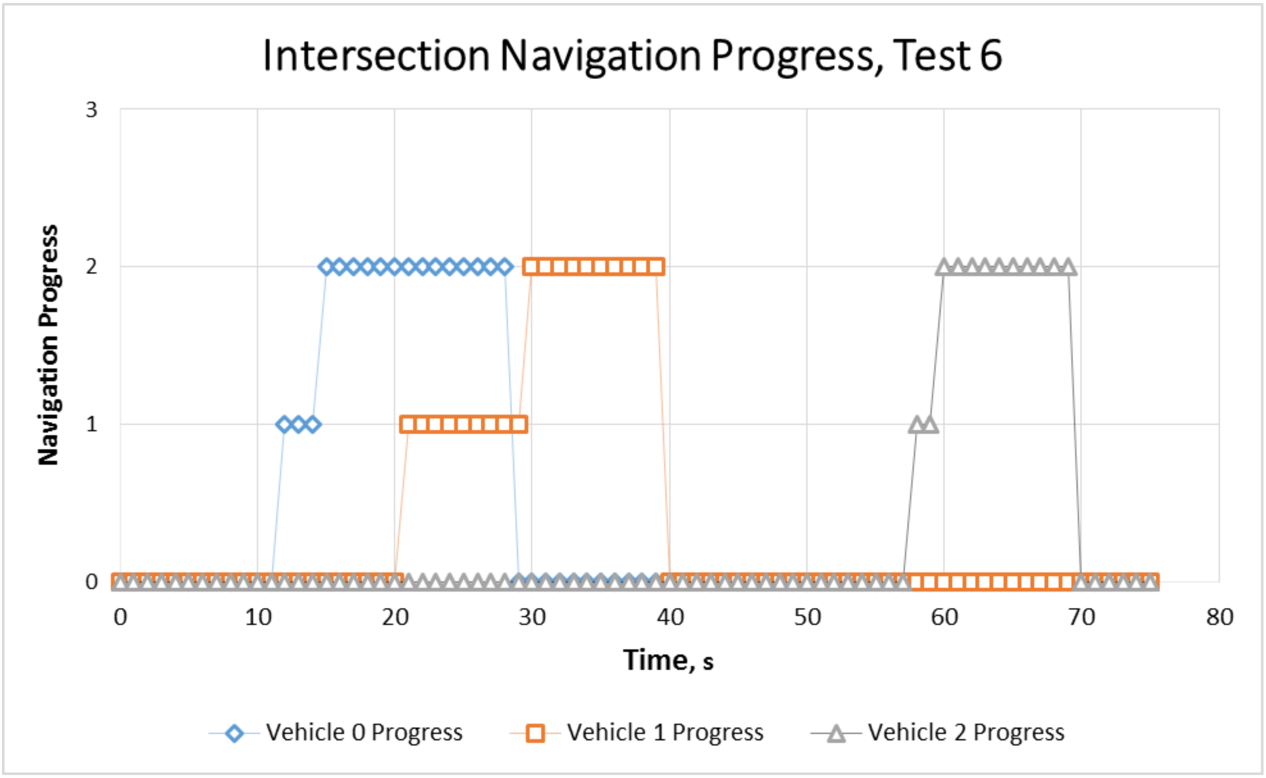


Figure 4.6 Visual representation of the vehicles navigating in Test 6

In this test, the first vehicle, Vehicle 0, received its approval string and proceeded through the intersection after recording a wait time of three seconds. Vehicle 1 arrived while Vehicle 0 was still navigating and did not receive its approval string until after Vehicle 0 had completed its navigation protocol. As presented in Figure 4.6, Vehicle 2 did not arrive at the intersection until well after Vehicle 1 had completed its navigation protocol but only had to wait 2s to receive its approval string from the base, hereby proving the effectiveness of the base at reducing wait times if there are no conflicting paths present within the intersection.

4.2.7 Test 7, Vehicles 0 and 2 turn left, Vehicle 1 turns right

The seventh test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 2 turn left, and Vehicle 1 turn right in the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.7.

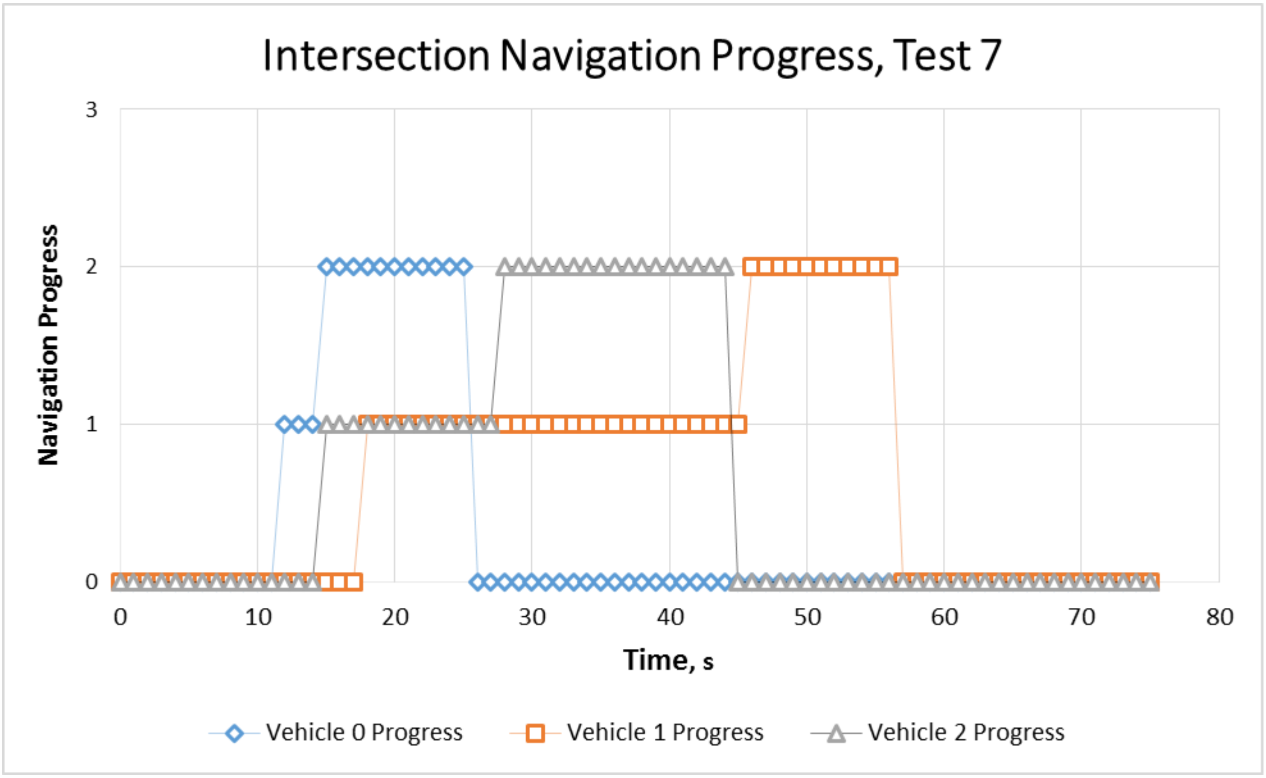


Figure 4.7 Visual representation of the vehicles navigating in Test 7

In this test, Vehicle 0 arrived at the intersection and received its approval string after recording a wait time of 3s. Vehicle 1 arrived and broadcasted its request string just as Vehicle 0 began its navigation protocol but did not receive its approvals string from the base until after Vehicle 0 had completed its navigation through the intersection. Vehicle 1 recorded a wait time of 13s. Vehicle 2 arrived at the intersection while Vehicle 0 was still navigating through the intersection but did not receive its approval string until after Vehicle 1 had broadcasted its confirmation string, letting the base broadcast Vehicle 2's approval string. Vehicle 2's wait time was 27s.

4.2.8 Test 8, Vehicle 0 turns left, Vehicle 1 turns right, Vehicle 2 proceeds straight

The eighth test was implemented in which the series of start commands, which had Vehicle 0 and turn left, Vehicle 1 turn right in the intersection, and Vehicle 2 proceed straight through the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.8

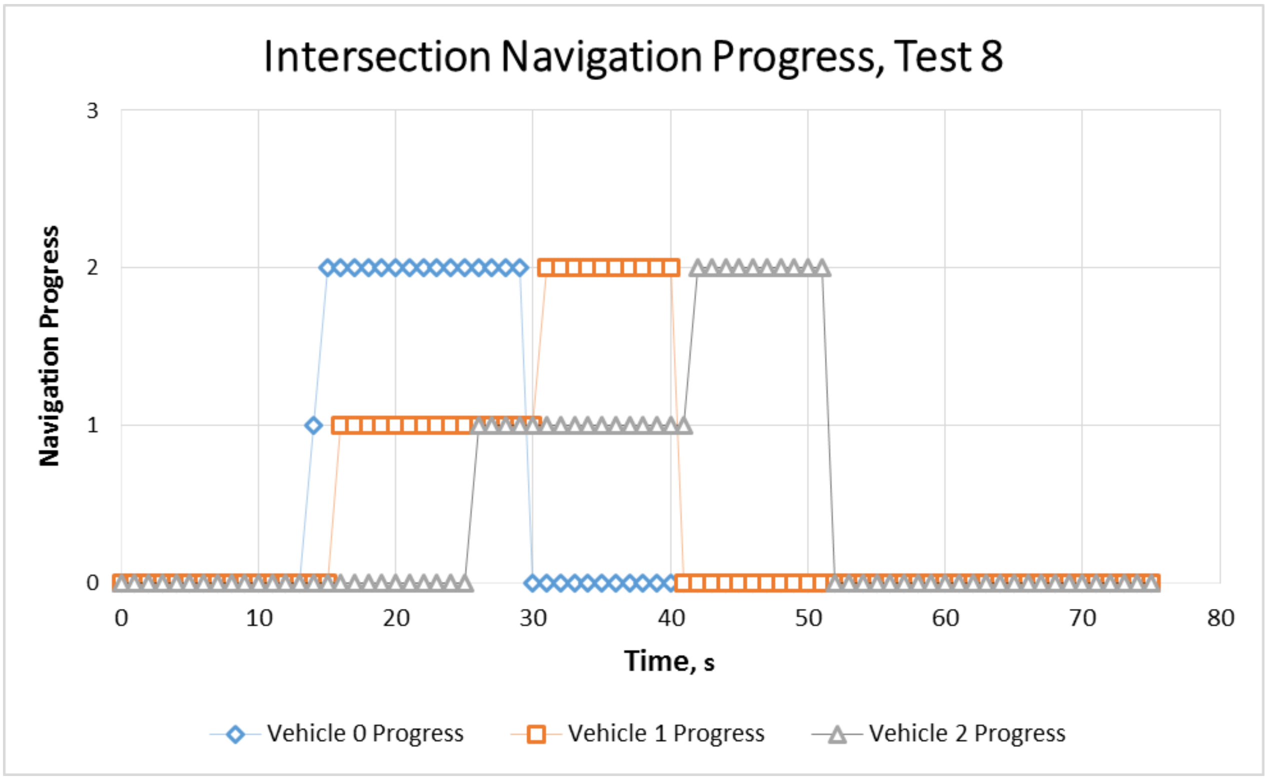


Figure 4.8 Visual representation of the vehicles navigating in Test 8

In this test, Vehicle 0 proceed through the intersection having recorded a wait time of 1s. Vehicle 1 arrived shortly after Vehicle 0 began its navigation protocol but was not given an approval sting until Vehicle 1 had completed its navigation protocol. Vehicle 1 recorded a wait time of 15s. Vehicle 2 arrived at the intersection while Vehicle 0 was still navigating the intersection but after Vehicle 1 arrived at the intersection and hence was not given an approval string from the base until after Vehicle 1 had completed its navigation protocol. Vehicle 2 recorded a wait time of 16s.

4.2.9 Test 9, Vehicle 0 turns left, Vehicles 1 and 2 turn right

The ninth test was implemented in which the series of start commands, which had Vehicle 0 turn left, and Vehicle 1 and Vehicle 2 turn right while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.9.

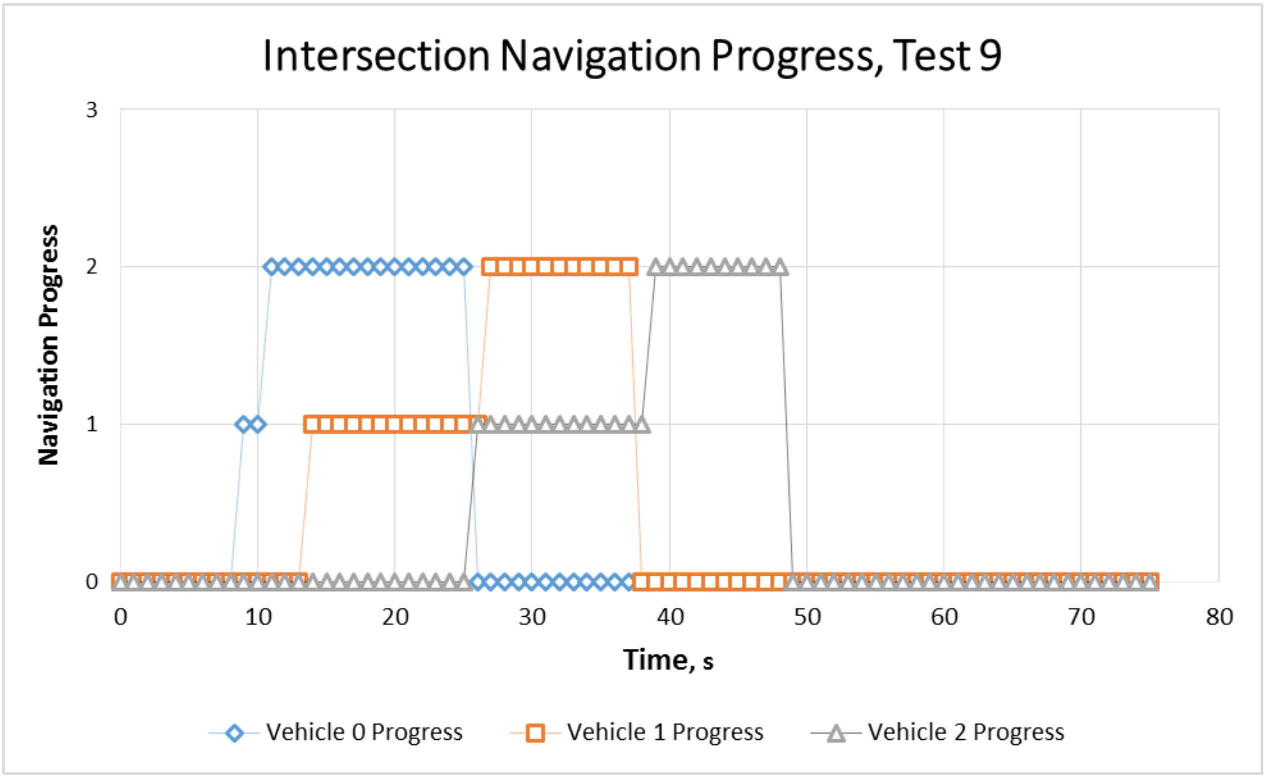


Figure 4.9 Visual representation of the vehicles navigating in Test 9

In this test, Vehicle 0 received its approval string after recording a wait time of 2s. Vehicle 1 arrived at the intersection while Vehicle 0 was still completing its navigation protocol but did not receive its approval string until after Vehicle 0 completed its navigation through the intersection. Vehicle 1 recorded a wait time of 13s. Vehicle 2 arrived at the intersection just as Vehicle 0 was exiting the intersection but did not receive an approval string until Vehicle 1 had completed its navigation protocol and broadcasted its confirmation string to the base. Vehicle 2 recorded a wait time of 13s.

4.2.10 Test 10, Vehicle 0 proceeds straight, Vehicles 1 and 2 turn left

The tenth test was implemented in which the series of start commands, which had Vehicle 0 proceed straight through the intersection, and Vehicle 1 and Vehicle 2 turn left while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.10.

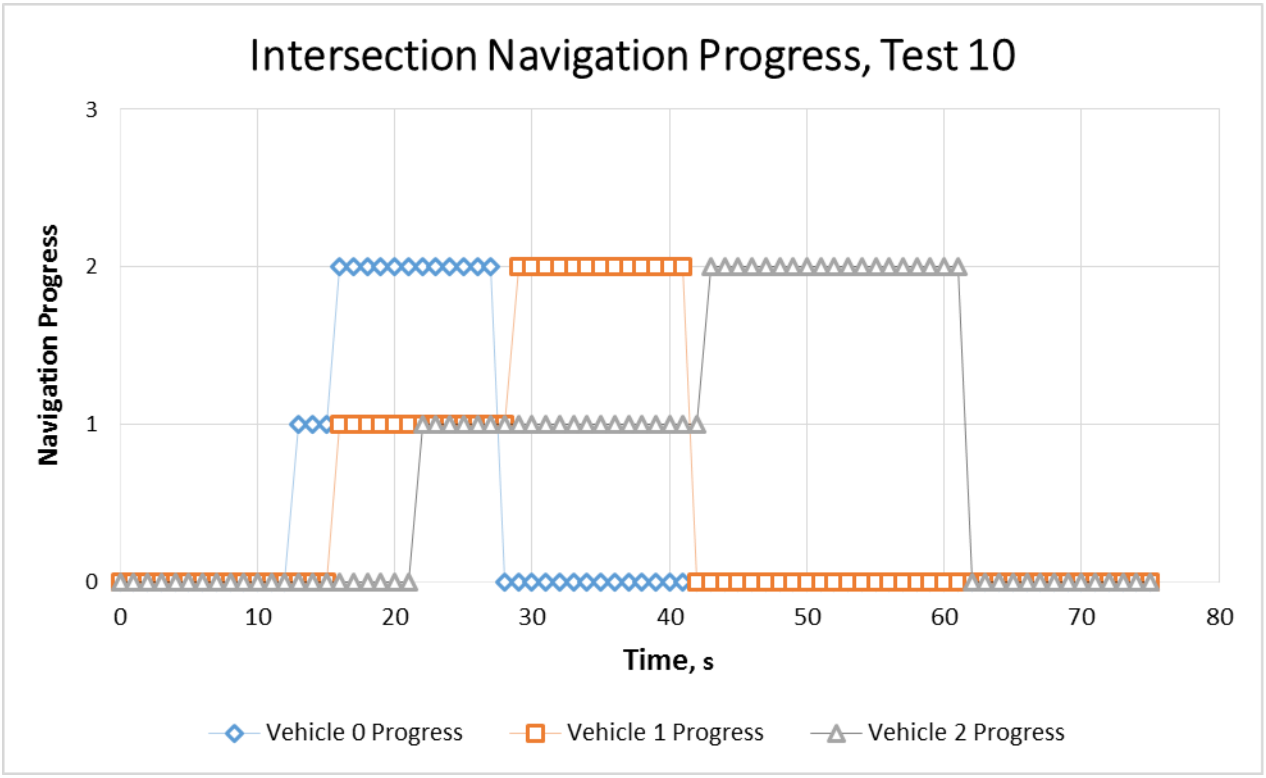


Figure 4.10 Visual representation of the vehicles navigating in Test 10

In this test, Vehicle 0 arrived and was first to navigate the intersection after recording a wait time of 3s between the broadcast of its request string to the base station and its receipt of the approval string from the base. Vehicle 1 arrived at the base just as Vehicle 0 began its navigation protocol but did not receive an approval string from the base until Vehicle 0 had completed its navigation protocol. Vehicle 1 recorded a wait time of 13s. Vehicle 2 arrived at the intersection while Vehicle 0 was still navigating the intersection but did not receive its approval string till Vehicle 1 had completed navigation hence, Vehicle 2 recorded a wait time of 21s.

4.2.11 Test 11, Vehicles 0 and 2 proceed straight, Vehicle 1 turns left

The eleventh test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 2 proceed straight through the intersection, and Vehicle 1 turn left while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.11.

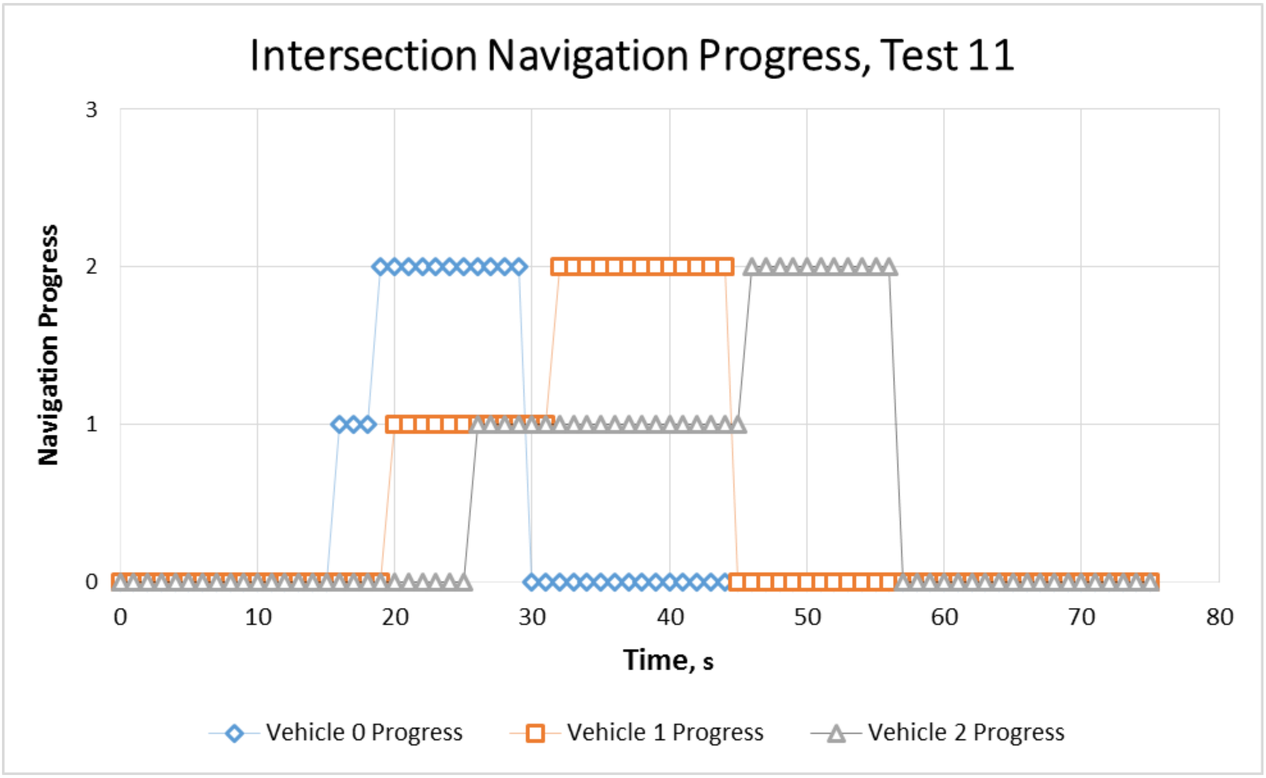


Figure 4.11 Visual representation of the vehicles navigating in Test 11

In this test, Vehicle 0 recorded a wait time of 3s. Vehicles 1 and 2 arrived at the intersection while Vehicle 0 was still navigating, however, because Vehicle 1 arrive first, it received its approval string from the base first, leaving Vehicle 2 to begin its intersection navigation protocol upon completion of Vehicle 1's protocol. The wait time for Vehicles 1 and 2 were recorded as 12s and 20s respectively.

4.2.12 Test 12, Vehicle 0 proceeds straight, Vehicle 1 turns left, Vehicle 2 turns right

The twelfth test was implemented in which the series of start commands, which had Vehicle 0 proceed straight through the intersection, Vehicle 1 turn left and Vehicle 2 turn right while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.12.

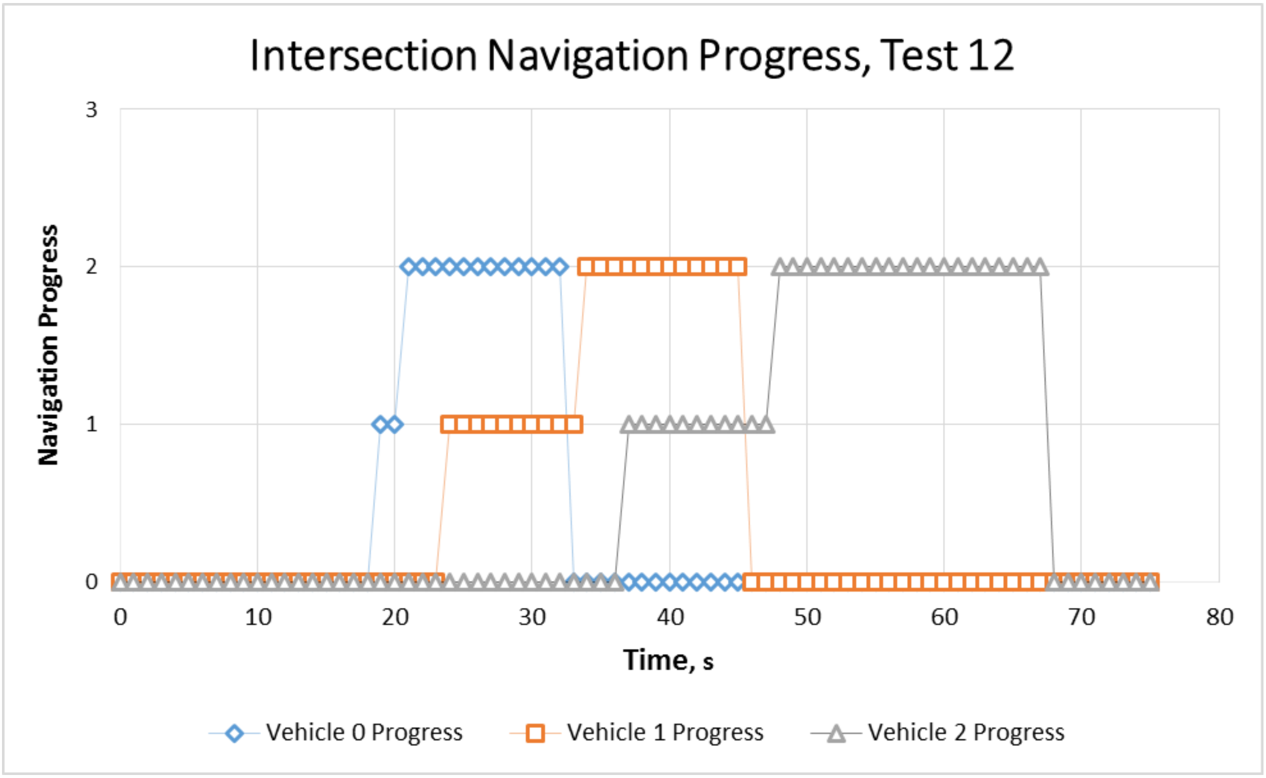


Figure 4.12 Visual representation of the vehicles navigating in Test 12

In this test, Vehicle 0 recorded a wait time of 2s before receiving its approval string from the base. Vehicle 1 arrived at the intersection while Vehicle 0 was still navigated and recorded a wait time of 10s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection while Vehicle 1 was navigating the intersection and recorded a wait time of 11s before receiving its approval string to navigate the intersection.

4.2.13 Test 13, Vehicles 0 and 1 proceed straight, Vehicle 2 turns left

The thirteenth test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 1 proceed straight through the intersection, and Vehicle 2 turn left while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.13.

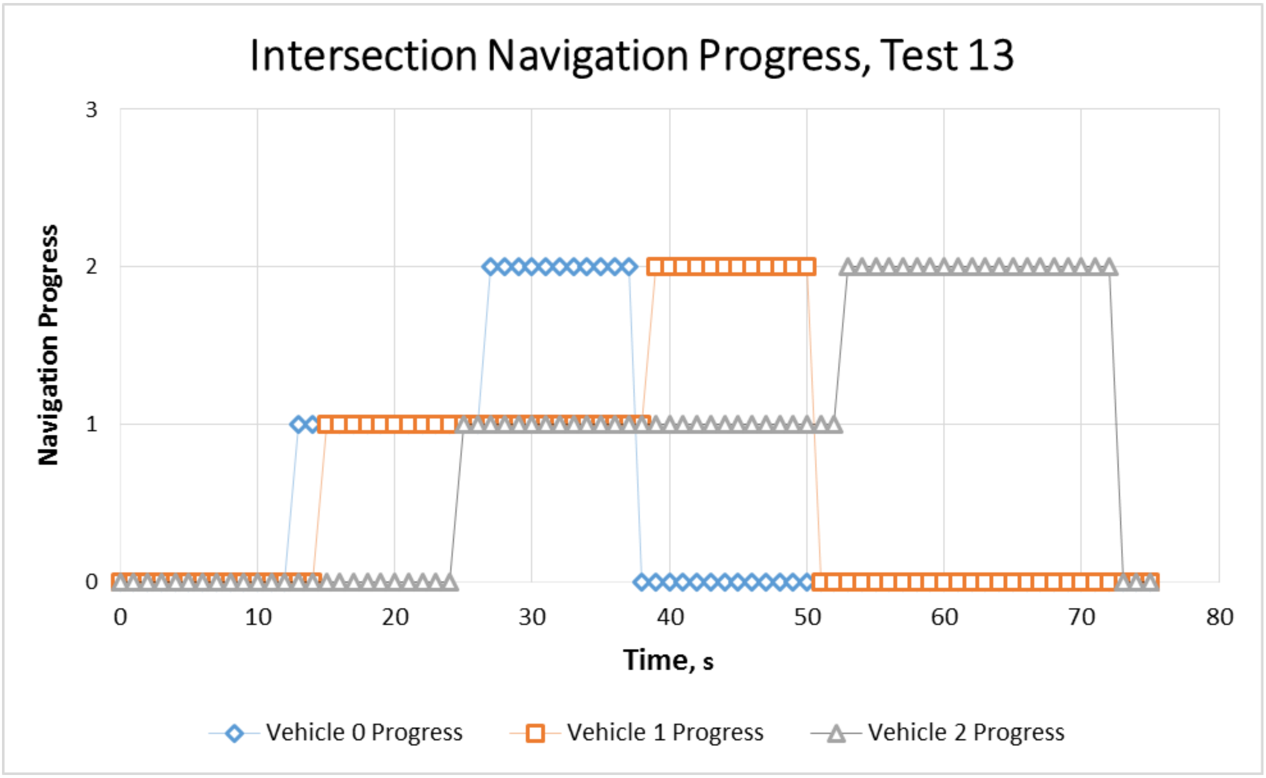


Figure 4.13 Visual representation of the vehicles navigating in Test 13

In this test, Vehicle 0 recorded a wait time of 13s; considered an outlier compared to all other first vehicle wait times, before receiving its approval string from the base. Vehicle 1 arrived at the intersection while Vehicle 0 was still waiting for an approval string and recorded a wait time of 24s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection while Vehicle 1 was just started navigating the intersection and recorded a wait time of 28s before receiving its approval string to navigate the intersection. This test proved that in the event of a malfunction, the base station would resume normal prioritizing as soon as the malfunction on the part of the vehicles was overcome.

4.2.14 Test 14, All Vehicles proceed straight

The fourteenth test was implemented in which the series of start commands, which had all Vehicles proceed straight through the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.14.

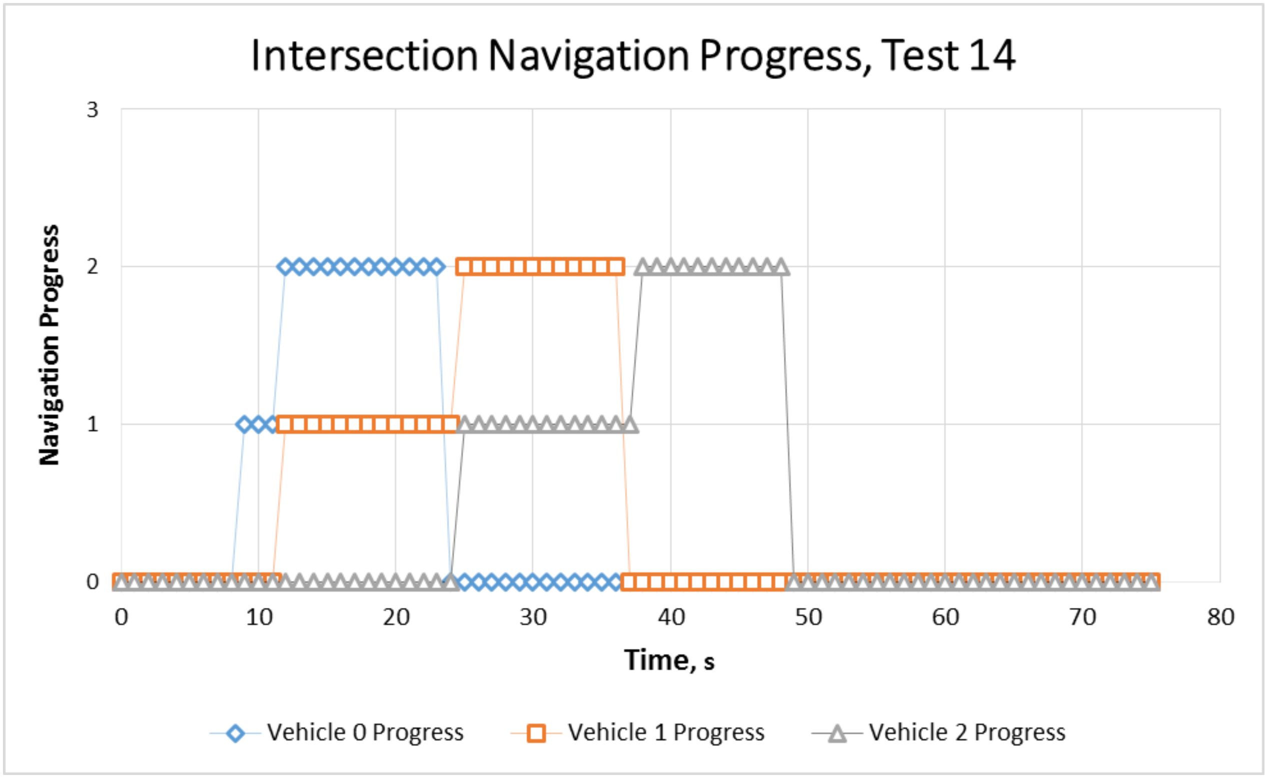


Figure 4.14 Visual representation of the vehicles navigating in Test 14

In this test, Vehicle 0 recorded a wait time of 3s before receiving its approval string from the base. Vehicle 1 arrived at the intersection when Vehicle 0 was beginning its navigation protocol and recorded a wait time of 13s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection as Vehicle 1 began navigating the intersection and recorded a wait time of 13s before receiving its approval string to navigate the intersection.

4.2.15 Test 15, Vehicles 0 and 1 proceed straight, Vehicle 2 turns right

The fifteenth test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 1 proceed straight through the intersection, and Vehicle 2 turn right while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.15.

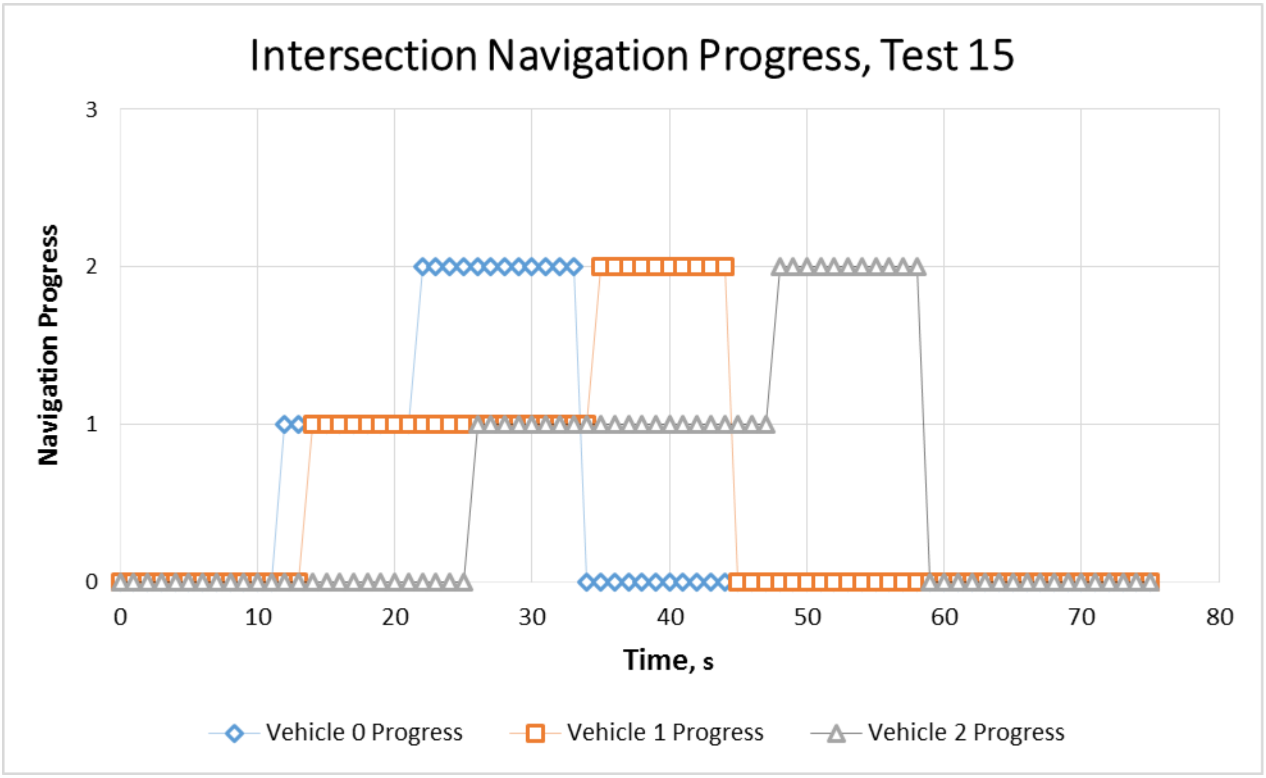


Figure 4.15 Visual representation of the vehicles navigating in Test 15

In this test, Vehicle 0 recorded yet another outlier wait time of 10s before receiving its approval string from the base. Vehicle 1 arrived at the intersection when Vehicle 0 was still awaiting its approval string and eventually recorded a wait time of 21s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection while Vehicle 0 was navigating the intersection and recorded a wait time of 22s before receiving its approval string to navigate the intersection.

4.2.16 Test 16, Vehicle 0 proceeds straight, Vehicle 1 turns right, Vehicle 2 turns left

The sixteenth test was implemented in which the series of start commands, which had Vehicle 0 proceed straight through the intersection, Vehicle 1 turn right, and Vehicle 2 turn left while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.16.

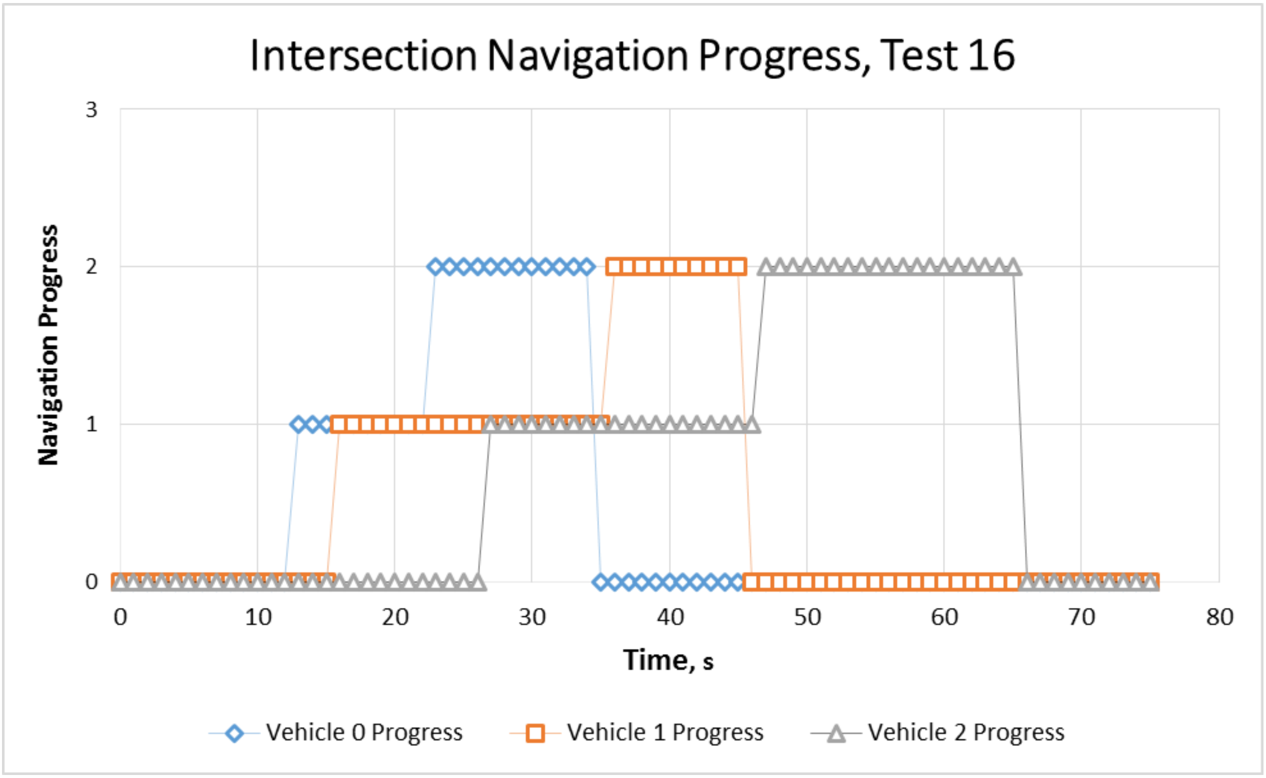


Figure 4.16 Visual representation of the vehicles navigating in Test 16

In this test, Vehicle 0 recorded yet another outlier wait time of 10s before receiving its approval string from the base. Vehicle 1 arrived at the intersection when Vehicle 0 was still awaiting its approval string and eventually recorded a wait time of 20s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection while Vehicle 0 was navigating the intersection and recorded a wait time of 20s before receiving its approval string to navigate the intersection.

4.2.17 Test 17, Vehicles 0 and 2 proceed straight, Vehicle 1 turns right

The seventeenth test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 2 proceed straight through the intersection, and Vehicle 1 turn right while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.17.

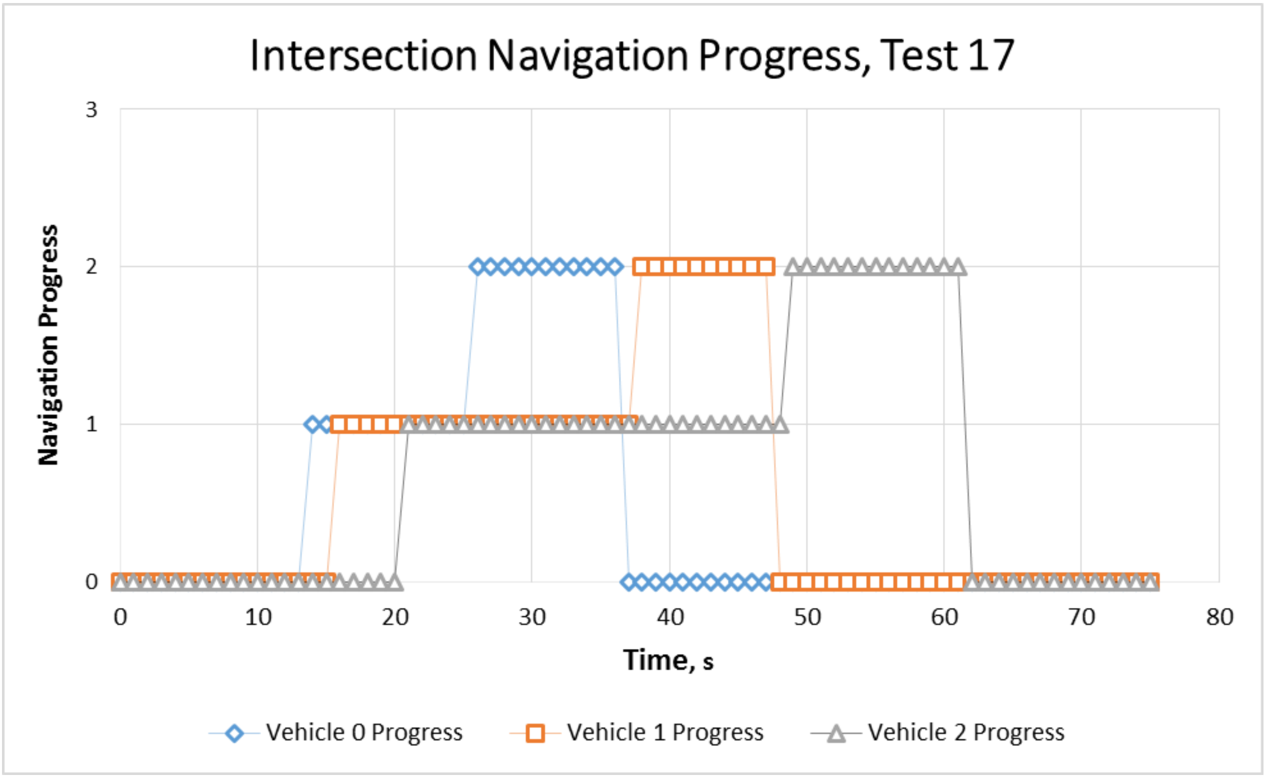


Figure 4.17 Visual representation of the vehicles navigating in Test 17

In this test, Vehicle 0 recorded the last recorded outlier wait time of 12s before receiving its approval string from the base. Vehicle 1 arrived at the intersection when Vehicle 0 was still awaiting its approval string and eventually recorded a wait time of 22s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection while Vehicle 0 was navigating the intersection and recorded a wait time of 28s before receiving its approval string to navigate the intersection. No more outlier wait times were recorded in the rest of the tests.

4.2.18 Test 18, Vehicle 0 proceeds straight, Vehicle 1 and 2 turn right

The eighteenth test was implemented in which the series of start commands, which had Vehicle 0 proceed straight through the intersection, and Vehicle 1 and Vehicle 2 turn right while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.18.

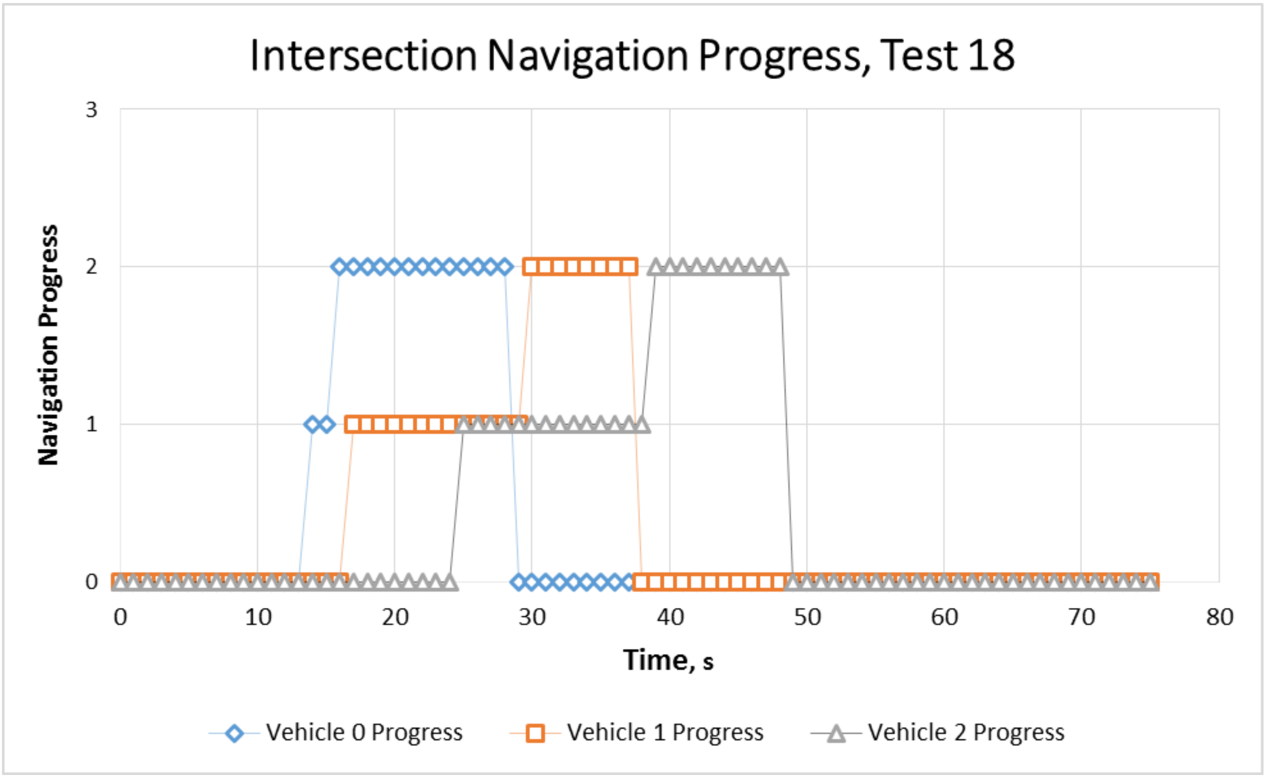


Figure 4.18 Visual representation of the vehicles navigating in Test 18

In this test, Vehicle 0 recorded a reasonable wait time of 2s before receiving its approval string from the base. Vehicle 1 arrived at the intersection when Vehicle 0 was performing its navigation protocol and recorded a wait time of 13s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection while Vehicle 0 was still navigating the intersection and recorded a wait time of 14s before receiving its approval string to navigate the intersection.

4.2.19 Test 19, Vehicle 0 turns right, Vehicles 1 and 2 turns left

The nineteenth test was implemented in which the series of start commands, which had Vehicle 0 turn right while navigating the intersection, and Vehicle 1 and Vehicle 2 turn left while navigating the intersection, was broadcasted to all the test vehicles. The vehicles’ navigational patterns are presented graphically in Figure 4.19.

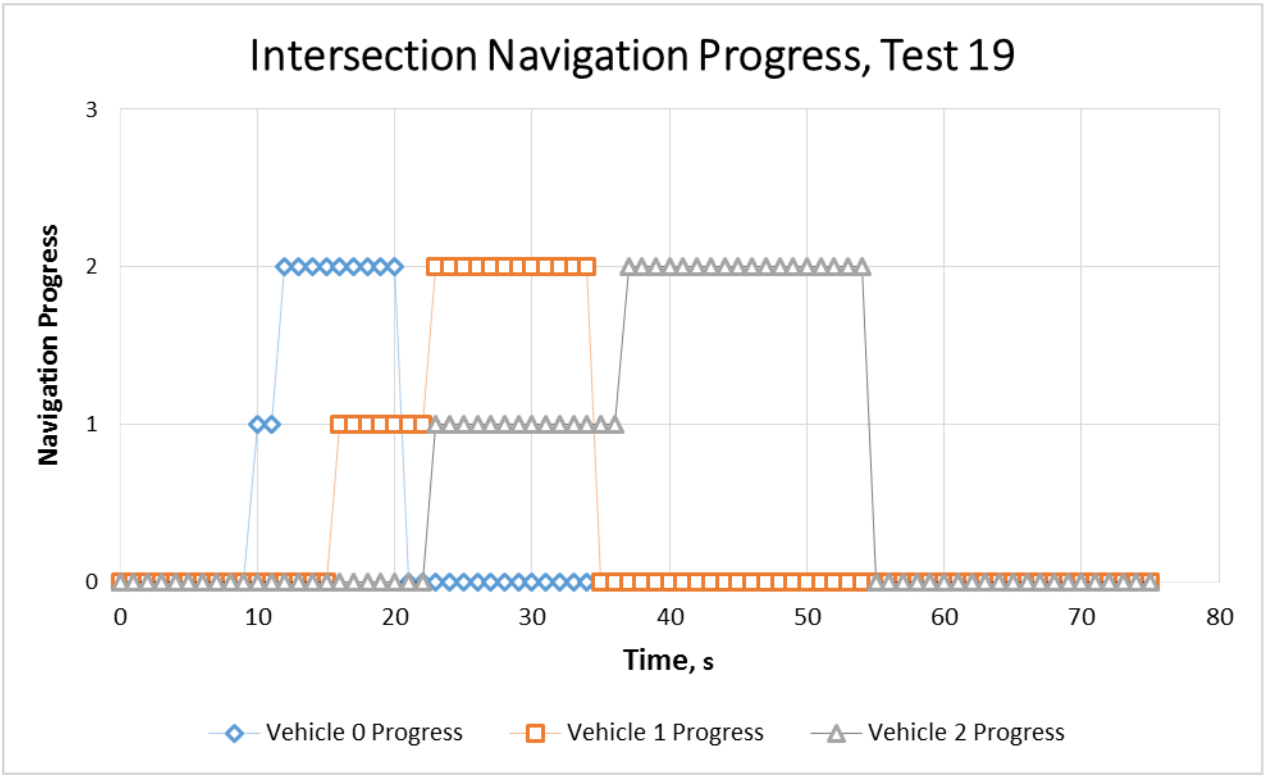


Figure 4.19 Visual representation of the vehicles navigating in Test 19

In this test, Vehicle 0 recorded a wait time of 2s before receiving its approval string from the base. Vehicle 1 arrived at the intersection when Vehicle 0 was performing its navigation protocol and recorded a wait time of 7s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection just as Vehicle 1 began navigating the intersection and recorded a wait time of 14s before receiving its approval string to navigate the intersection.

4.2.20 Test 20, Vehicle 0 turns right, Vehicle 1 turns left, Vehicle 2 proceeds straight

The twentieth test was implemented in which the series of start commands, which had Vehicle 0 turn right while navigating the intersection, Vehicle 1 turn left and Vehicle 2 proceed straight through the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.20.

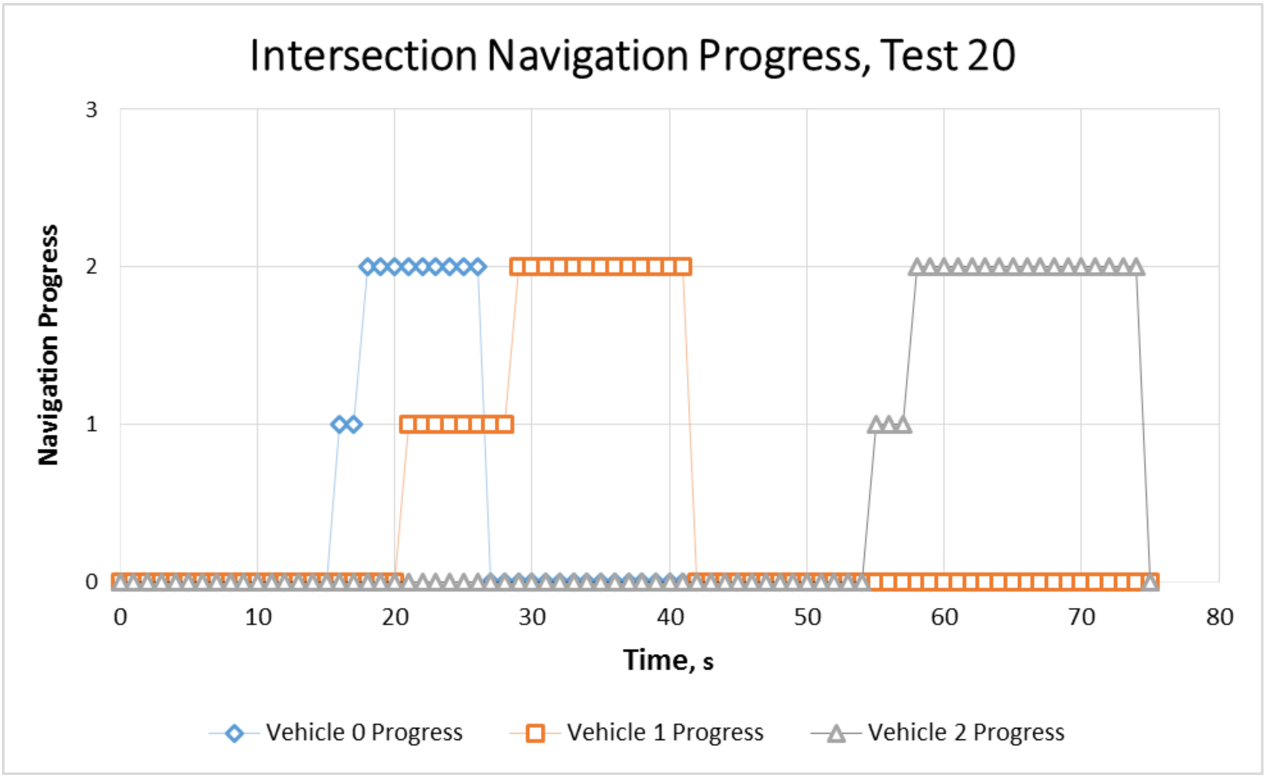


Figure 4.20 Visual representation of the vehicles navigating in Test 20

In this test, Vehicle 0 recorded a wait time of 2s before receiving its approval string from the base. Vehicle 1 arrived at the intersection when Vehicle 0 was performing its navigation protocol and recorded a wait time of 8s before receiving the approval string to navigate the intersection. Vehicle 2 arrived at the intersection long after Vehicle 1 completed its navigation protocol and recorded a wait time of just 3s before receiving its approval string to navigate the intersection since there were no other vehicles in the intersection or waiting to navigate the intersection.

4.2.21 Test 21, Vehicles 0 and 2 turn right, Vehicle 1 turns left

The twenty-first test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 2 turn right while navigating the intersection, and Vehicle 1 turn left while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.21.

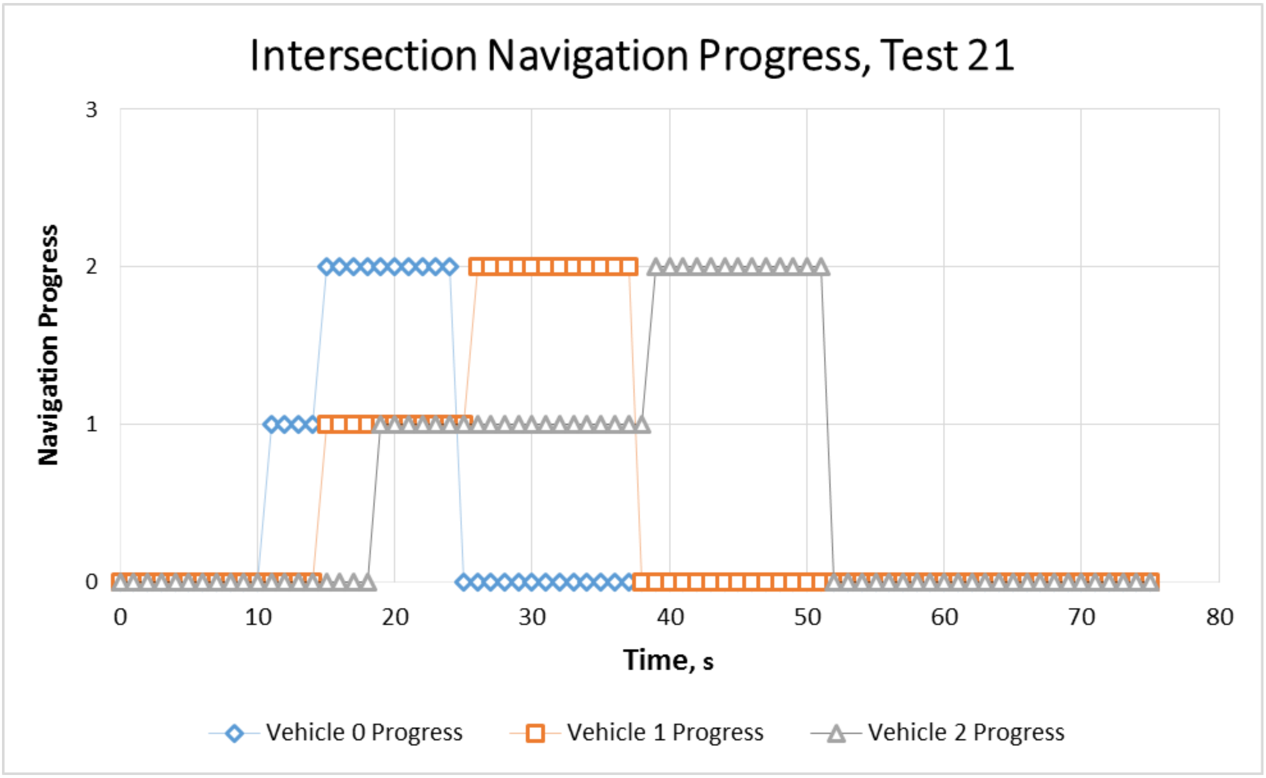


Figure 4.21 Visual representation of the vehicles navigating in Test 21

In this test, Vehicle 0 recorded a wait time of 4s before receiving its approval string from the base. Vehicle 1 arrived at the intersection just as Vehicle 0 began its navigation protocol and recorded a wait time of 11s before receiving its approval string to navigate the intersection. Vehicle 2 arrived at the intersection before Vehicle 0 completed its navigation protocol and recorded a wait time of 20s before receiving its approval string to navigate the intersection.

4.2.22 Test 22, Vehicle 0 turns right, Vehicle 1 proceeds straight, Vehicle 2 turns left

The twenty-second test was implemented in which the series of start commands, which had Vehicle 0 turn right while navigating the intersection, Vehicle 1 proceed straight through and Vehicle 2 turn left while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.22.

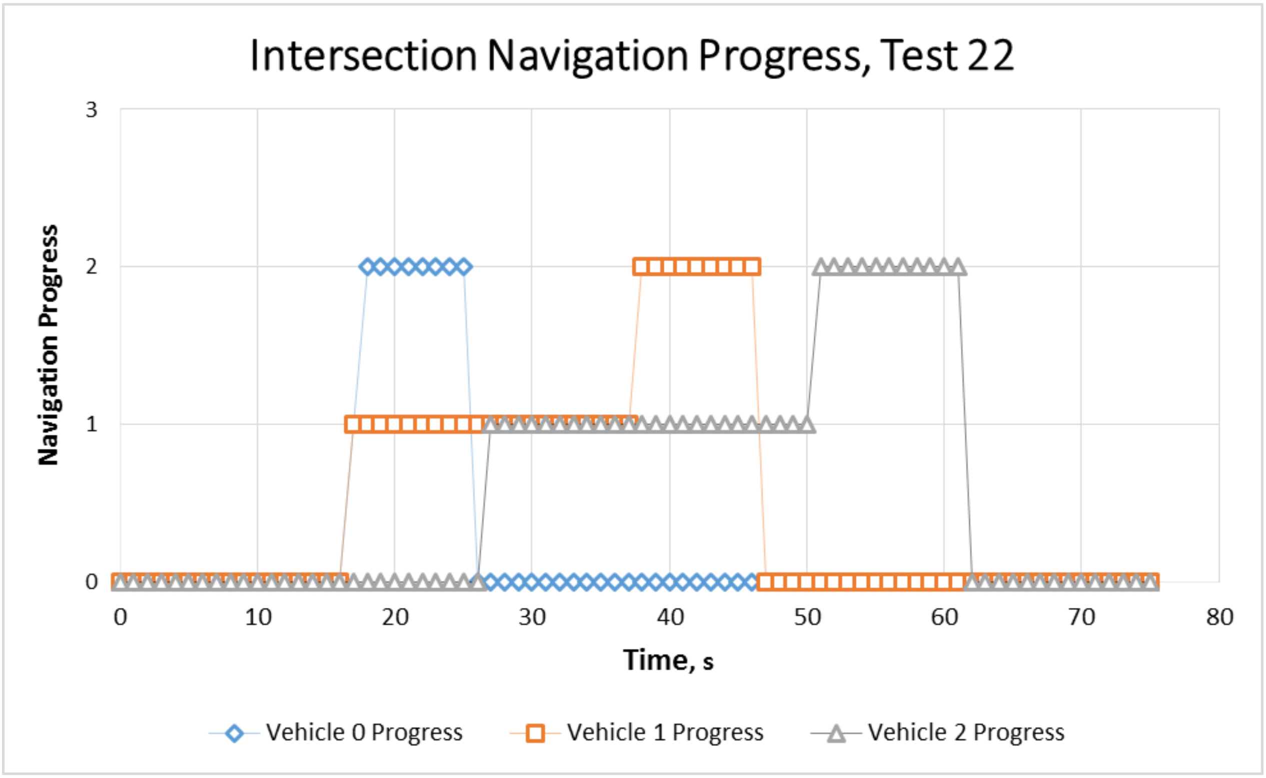


Figure 4.22 Visual representation of the vehicles navigating in Test 22

In this test, Vehicle 0 recorded a wait time of 1s before receiving its approval string from the base. Vehicle 1 arrived at the intersection at the same time Vehicle 0 but took a long time to register its approval string thereby making it navigate long after Vehicle 0 completed its navigation protocol. Vehicle 1 recorded a wait time of 21s before receiving its approval string to navigate the intersection. Vehicle 2 arrived at the intersection just after Vehicle 0 completed its navigation protocol but since it arrived after Vehicle 1 had to wait for it to complete its navigation protocol and hence recorded a wait time of 24s before receiving its approval string to navigate the intersection.

4.2.23 Test 23, Vehicle 0 turns right, Vehicle 1 and 2 proceed straight

The twenty-third test was implemented in which the series of start commands, which had Vehicle 0 turn right while navigating the intersection, and Vehicle 1 and Vehicle 2 proceed straight through the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.23.

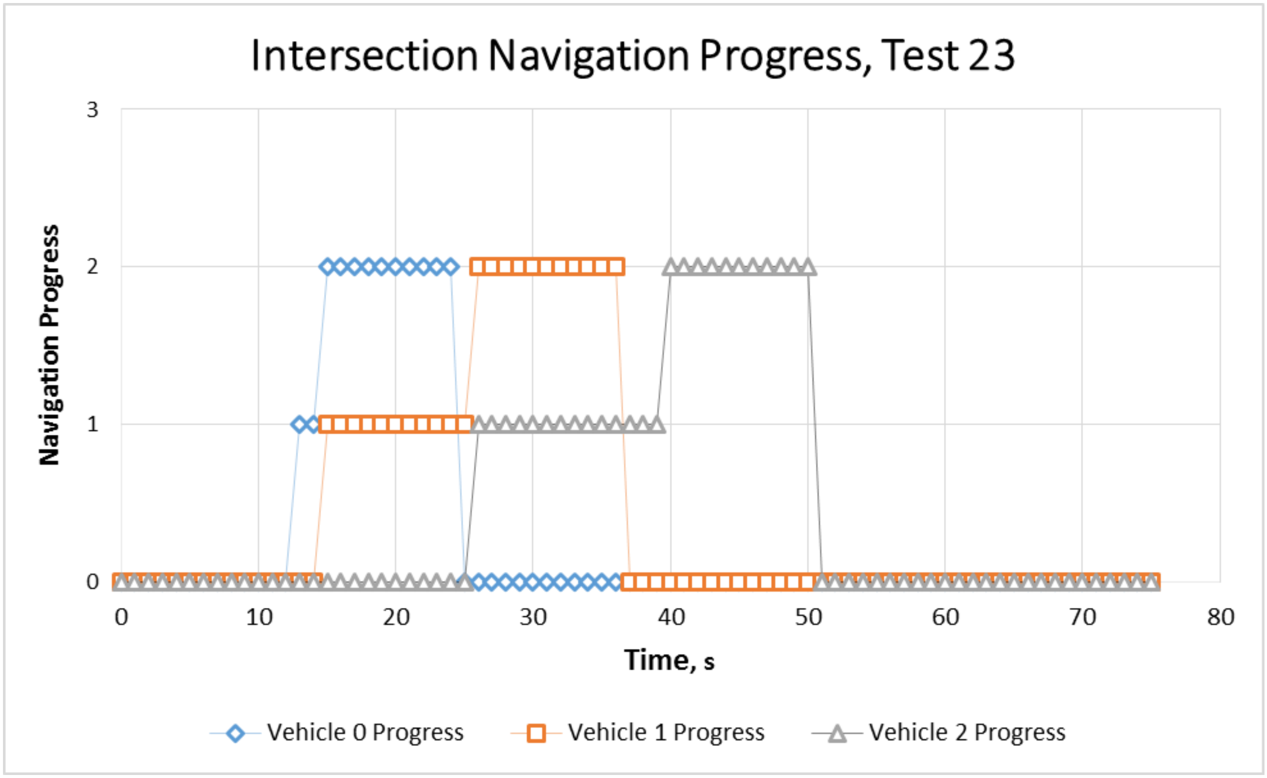


Figure 4.23 Visual representation of the vehicles navigating in Test 23

In this test, Vehicle 0 recorded a wait time of 2s before receiving its approval string from the base. Vehicle 1 arrived at the intersection just as Vehicle 0 began its navigation protocol and recorded a wait time of 11s before receiving its approval string to navigate the intersection. Vehicle 2 arrived at the intersection just as Vehicle 1 began its navigation protocol and recorded a wait time of 14s before receiving its approval string to navigate the intersection.

4.2.24 Test 24, Vehicles 0 and 2 turn right, Vehicle 1 proceeds straight

The twenty-fourth test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 2 turn right while navigating the intersection, and Vehicle 1 proceed straight through the intersection, was broadcasted to all the test vehicles. The vehicles’ navigational patterns are presented graphically in Figure 4.24.

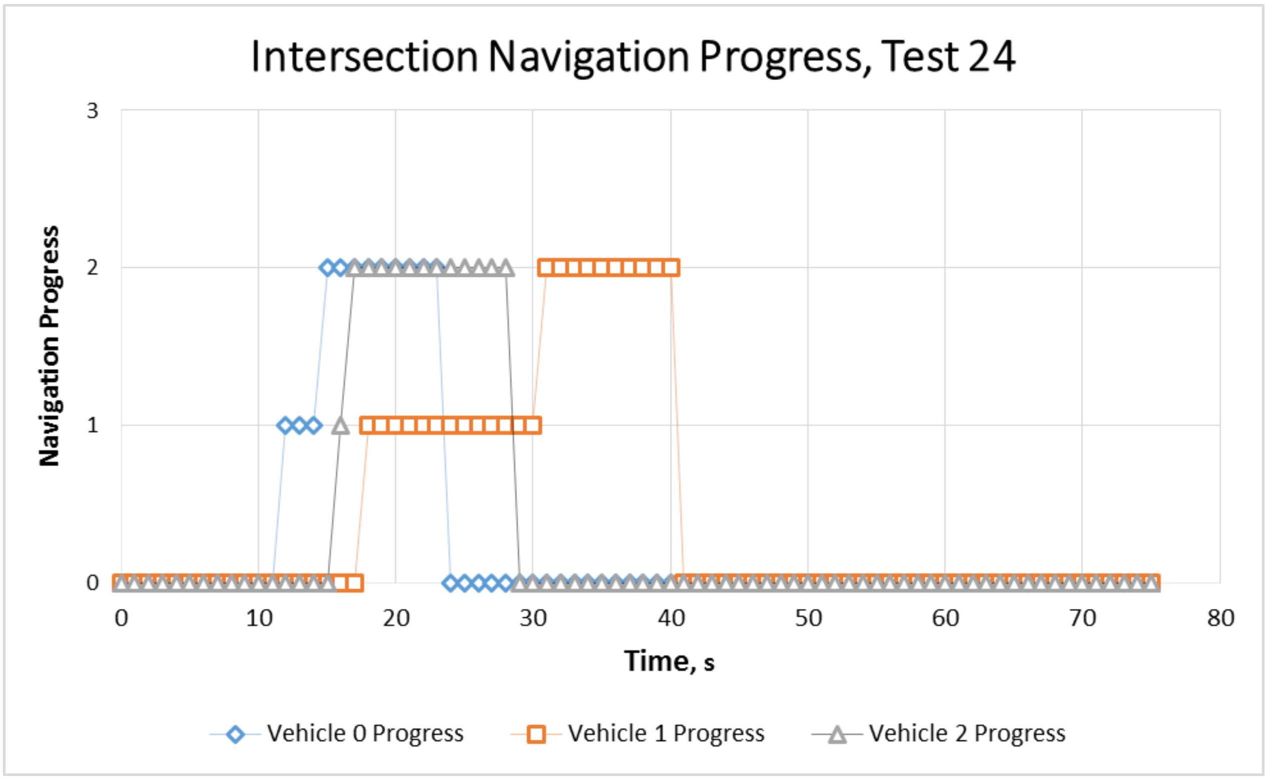


Figure 4.24 Visual representation of the vehicles navigating in Test 24

In this test, Vehicle 0 arrived first at the intersection and recorded a wait time of 3s before beginning its navigation protocol. Peculiarly, Vehicle 2 arrived at the intersection before Vehicle 1 and since its path did not conflict with the paths blocked off by Vehicle 0 was approved to navigate the intersection concurrently with Vehicle 0 having only recorded a wait time of 1s. Vehicle 1 arrived at the intersection while Vehicle 0 and Vehicle 2 were navigated and hence recorded a wait time of 13s before being approved to navigate the intersection. This test proved that the base station had the capability of sending approval strings to multiple vehicles for them to navigate concurrently provided that none of their blocked paths intersected.

4.2.25 Test 25, Vehicles 0 and 1 turn right, Vehicle 2 turns left

The twenty-fifth test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 1 turn right while navigating the intersection, and Vehicle 2 turn left while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.25.

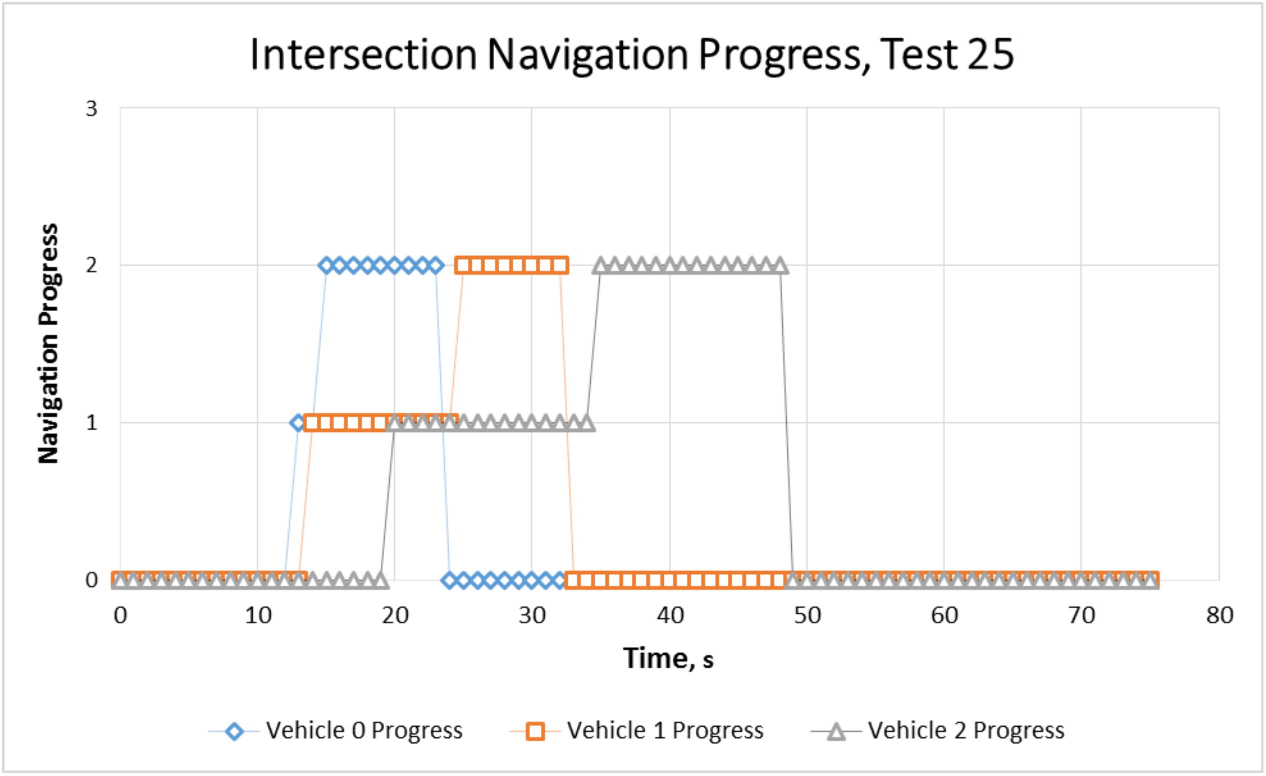


Figure 4.25 Visual representation of the vehicles navigating in Test 25

In this test, Vehicle 0 recorded a wait time of 2s before receiving its approval string from the base. Vehicle 1 arrived at the intersection just before Vehicle 0 began its navigation protocol and recorded a wait time of 11s before receiving its approval string to navigate the intersection. Vehicle 2 arrived at the intersection while Vehicle 0 was still completing its navigation protocol and recorded a wait time of 15s before receiving its approval string to navigate the intersection.

4.2.26 Test 26, Vehicles 0 and 1 turn right, Vehicle 2 proceeds straight

The twenty-sixth test was implemented in which the series of start commands, which had Vehicle 0 and Vehicle 1 turn right while navigating the intersection, and Vehicle 2 proceeds straight through the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.26.

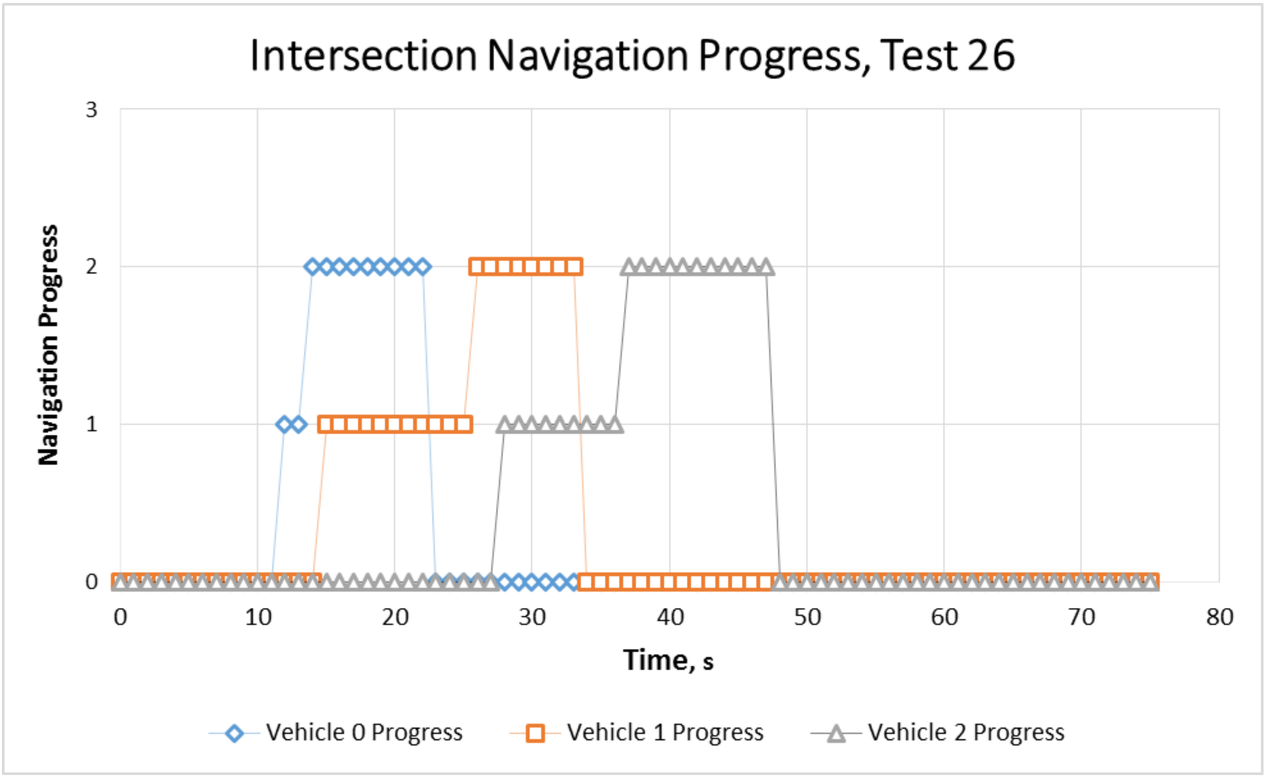


Figure 4.26 Visual representation of the vehicles navigating in Test 26

In this test, Vehicle 0 recorded a wait time of 2s before receiving its approval string from the base. Vehicle 1 arrived at the intersection while Vehicle 0 was completing its navigation protocol and recorded a wait time of 11s before receiving its approval string to navigate the intersection. Vehicle 2 arrived at the intersection after Vehicle 1 began its navigation protocol and recorded a wait time of 9s before receiving its approval string to navigate the intersection.

4.2.27 Test 27, All Vehicles turn right

The twenty-seventh test was implemented in which the series of start commands, which had all vehicles turn right while navigating the intersection, was broadcasted to all the test vehicles. The vehicles' navigational patterns are presented graphically in Figure 4.27.

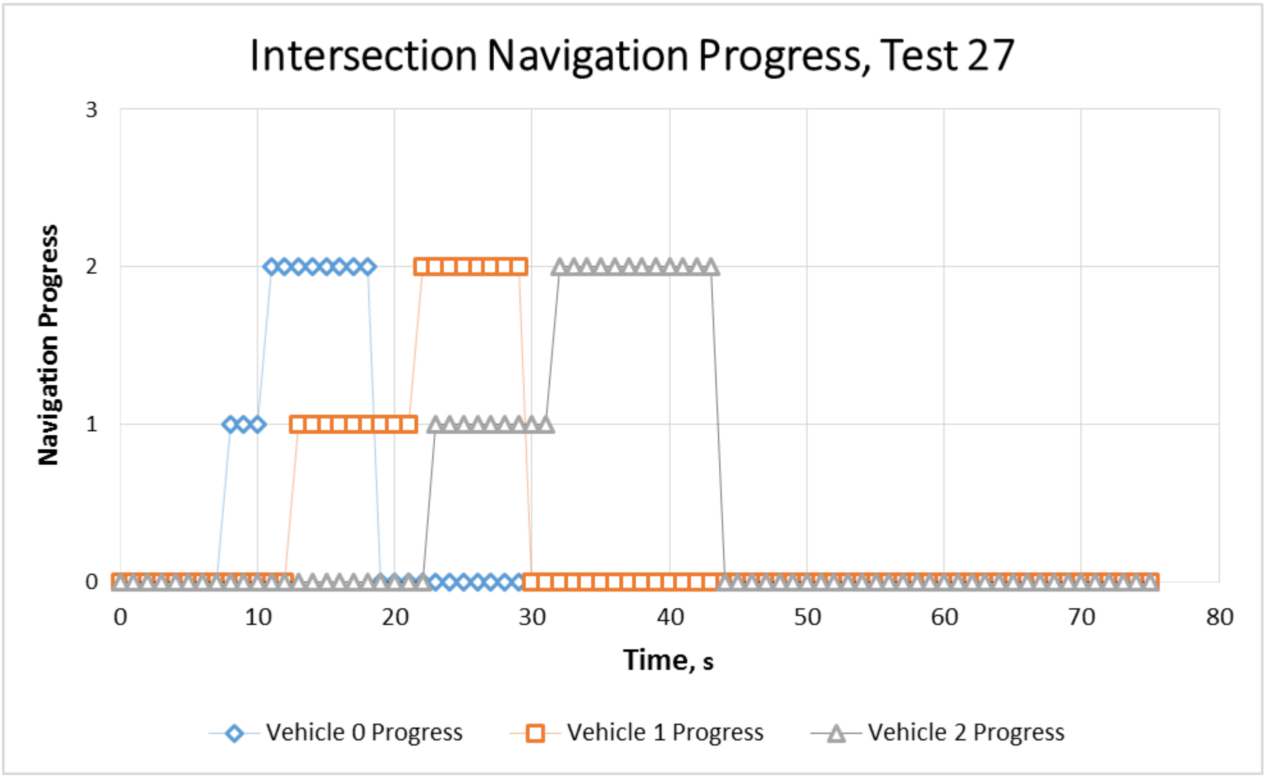


Figure 4.27 Visual representation of the vehicles navigating in Test 27

In this test, Vehicle 0 recorded a wait time of 3s before receiving its approval string from the base. Vehicle 1 arrived at the intersection while Vehicle 0 was completing its navigation protocol and recorded a wait time of 9s before receiving its approval string to navigate the intersection. Vehicle 2 arrived at the intersection after Vehicle 1 had begun its navigation protocol and recorded a wait time of 9s before receiving its approval string to navigate the intersection.

4.2.28 Test 28, Inclusion of a Disabled Vehicle

The twenty-eighth test was implemented with turning values similar to Test 24. This test included a new parameter which was the addition of a disabled vehicle in front of Vehicle 1 thereby blocking off the leg of the intersection. This was done to determine if the envisioned system would work even with the presence of an obstacle blocking one of the intersection legs. The vehicles' navigational patterns are presented graphically in Figure 4.28.

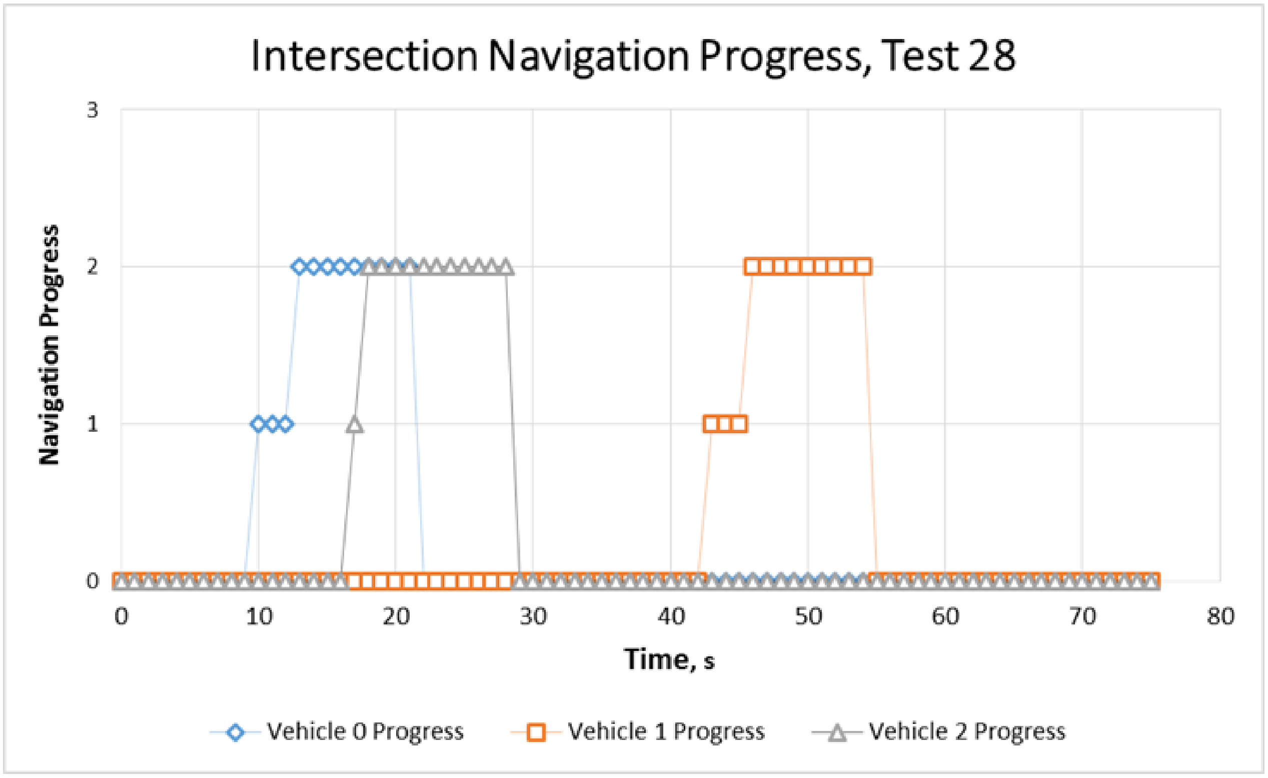


Figure 4.28 Visual representation of the vehicles navigating in Test 28

In this test, Vehicle 0 arrived first at the intersection and recorded a wait time of 3s before beginning its navigation protocol. Peculiarly, Vehicle 2 arrived at the intersection before Vehicle 1 and since its path did not conflict with the paths blocked off by Vehicle 0 was approved to navigate the intersection concurrently with Vehicle 0 having only recorded a wait time of 1s. Vehicle 1 detected an obstacle (disabled vehicle) with its ultrasonic sensors right at the stop sign and even though it recognized the stop sign, did not send a request signal to the base. This was because, in the event of a confirmation, the vehicle would be unable to traverse the intersection. After the obstacle was removed by the test taker however, normal operations resumed at the intersection showing the envisioned systems' ability to operate even with the presence of obstacles.

4.3 Statistical Analyses of Wait Times

As presented in the preceding graphs, the wait-times of each vehicle that arrived at the intersection were logged for future perusal. The wait times of all the vehicles arriving first in the intersection are presented in Figure 4.29.

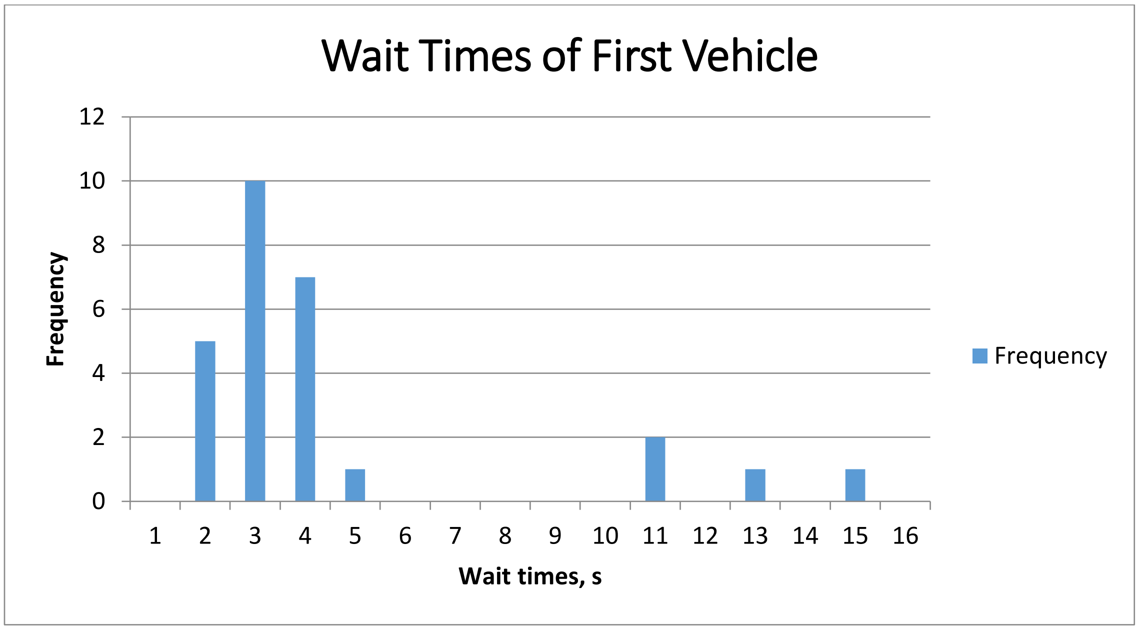


Figure 4.29 Visual representation of the frequency of the wait times of the vehicles arriving first in the intersection

It was found that the average wait time of vehicles arriving at the intersection first was 3.57s. This was higher than most recorded values due to the presence of outliers. The standard deviation of the wait-times of the first arriving vehicles was found to be 3.46s indicating that 68.2% of all future readings would be predicted to fall between 0.11s and 7.03s. The outliers were caused by the vehicles awaiting their confirmation strings, misinterpreting the strings and hence delaying their navigation protocol. This was caused by the propensity of the microcontrollers of other vehicles to send strings with slightly staggered integers, blending a request string to a confirmation string from the base thereby confusing the intended vehicle.

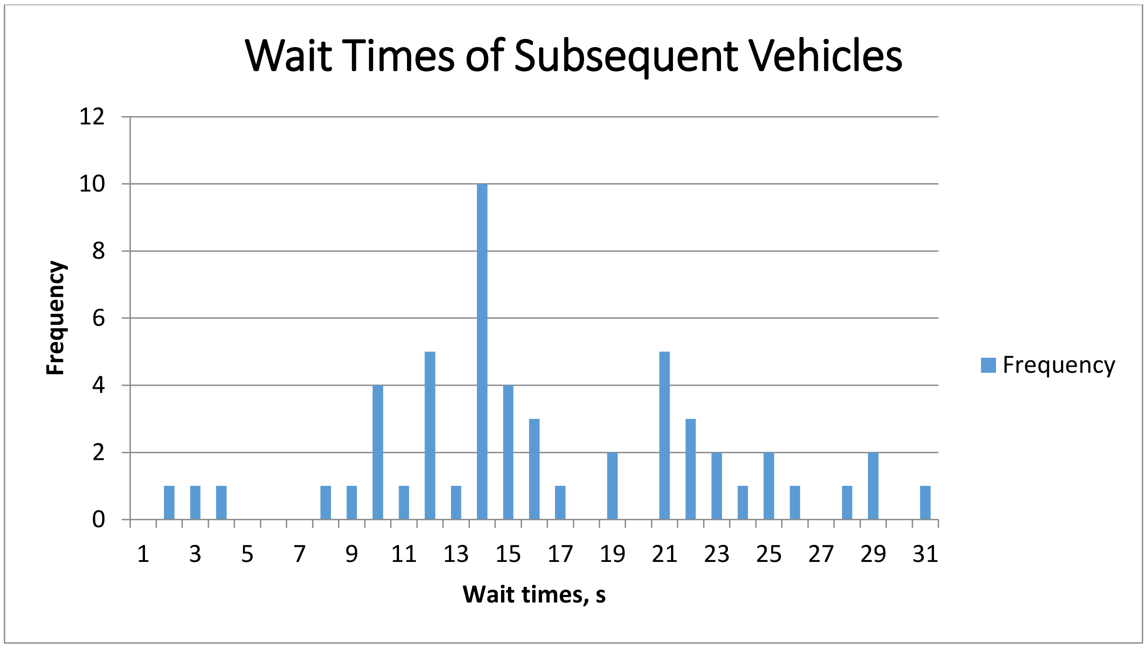


Figure 4.30 Visual representation of the frequency of the wait times of the vehicles arriving after the first vehicle at the intersection

The common wait times of the second and third vehicle was lumped together due to the nature of these wait times. In both cases, the vehicle's request string has to go through the full program of the base station which tests the base stations' proficiency in handling multiple vehicles. It was found that the average wait time of vehicles arriving second or third at the intersection was 15.43s. The standard deviation of the wait-times of the second and third vehicles arriving at the intersection were found to be 6.58s indicating that 68.2% of all future readings would be predicted to fall between 8.85s and 22.01s. These times compared favorable to the average traffic signal cycle time of 60s (Wu et al. 2015).

It should be noted that this cycle time does not imply that all vehicles arriving at the traffic signal would have to wait 60s but that the average wait time of several monitored vehicles in the study exhibited this average. This means the wait time could range from as little as half a second for the first vehicles in the queue arriving just as the light changes to as many as 300s for a vehicle arriving in the end of a long queue and needed several cycles to proceed.

Knowing this, it should also be noted that the envisioned intersection management system was designed for areas with moderate traffic density such as the intersection presented from Statesboro, GA.

This would mean that it should be compared with traffic signals with similarly moderate traffic flows. The traffic signal cycle time of 60s accounts for the averaged wait time of vehicles at all traffic signals at all times, meaning that the average should be comparable to the wait times of the envisioned system with the constrained traffic flow condition.

4.4 Cost Benefit Analysis

In keeping with the title of this thesis, the author had to prove that the designed system was indeed cost effective, and hence could be implemented at a competitive price to other intersection management styles, currently implemented in the real world. Cost comparison analyses were performed and presented in Table 4.2 through Table 4.7 ranging from 5% to 10% respectively. Here, four intersection management styles were compared to the author’s system. These four systems were; a conventional four way stop sign only intersection, a four-way intersection fitted with traffic signal and inductor loops to help reduce wait times, a newer style transit signal priority detection system currently envisioned for modes of mass transit transportation, and a Dedicated Short Range Communication (DSRC) system in the process of being developed and geared toward Intelligent Transport System management of Intelligent Vehicles.

Table 4.2 Cost Comparison Analysis with an inflation rate of 5% (ITSJPO 2017; Wright 2014; EmedCo 2017; Crowe 2012; Rosenlund 2017; Gustafson et al. 2014)

Instersection Management Systems - Cost Comparison Analysis							
Stop Sign Intersection Costs	**Initial Cost (US$)**	**Year 1 (US$)**	**Year 2 (US$)**	**Year 3 (US$)**	**Year 4 (US$)**	**Year 5 (US$)**	**NPV for 5 years (US$)**
Hardware (stop signs/poles) (4)	-800.00						-800.00
Hardware (in ground stanchion) (4)	-600.00						-600.00
Installation Labor ($73/hr)	-1,600.00						-1,600.00
Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-376.66
Total Cost for 5 yrs							**-3,376.66**
Conventional Traffic Light Costs							
Traffic signals/Installation (4)	-191,200.00						-191,200.00
O & M for traffic lights		-4,000.00	-4,000.00	-4,000.00	-4,000.00	-4,000.00	-17,317.91
Loop Detector (4)	-3,000.00						-3,000.00
Signal Preemption Reciever (2)	-3,500.00	-115.00	-115.00	-115.00	-115.00	-115.00	-3,997.89
Power Backup	-1,200.00	-100.00	-100.00	-100.00	-100.00	-100.00	-1,632.95
Linked Signal System LAN	-30,500.00	-350.00	-350.00	-350.00	-350.00	-350.00	-32,015.32
Signal Controller	-10,000.00						-10,000.00
Signal Retiming (every 4 yrs)					-3,000.00		-2,468.11
Total Cost for 5 yrs							**-261,632.17**
Transit Signal Priority Detection Systems Cost							
Optical Emitters (4)	-9,000.00						-9,000.00
Wayside Reader	-30,000.00						-30,000.00
Loop detector(4)	-3,000.00						-3,000.00
"Smart" Loops (added on to loop detector) (4)	-2,500.00						-2,500.00
Vehicle mouted GPS	-8,000.00						-8,000.00
Wireless comunication system	-10,000.00						-10,000.00
Power		-1,971.00	-1,971.00	-1,971.00	-1,971.00	-1,971.00	-8,533.40
Cellular Data		-420.00	-420.00	-420.00	-420.00	-420.00	-1,818.38
Total Cost for 5 yrs							**-72,851.78**
Dedicated Short Range Comunication (DSRC)							
Hardware	-7,450.00						-7,450.00
Instalation labor	-3,500.00						-3,500.00
Design and Planning	-6,600.00						-6,600.00
On board equipment	-4,150.00						-4,150.00
Power		-100.00	-100.00	-100.00	-100.00	-100.00	-432.95
Traditional Maintenance		-500.00	-500.00	-500.00	-500.00	-500.00	-2,164.74
Licence/Maintenance Agreements		-200.00	-200.00	-200.00	-200.00	-200.00	-865.90
SCMS Certificate Licence		-50.00	-50.00	-50.00	-50.00	-50.00	-216.47
Total Cost for 5 yrs							**-25,380.06**
Envisioned V2I Communication System Cost							
Pcduino Mini Computer	-59.99						-59.99
Xbee Wireless Transmitter/USB connector	-61.95						-61.95
Solar Panel	-24.99						-24.99
Rechargeable battery	-17.99	-17.99	-17.99	-17.99	-17.99	-17.99	-95.88
Hardware (stop signs/poles)	-800.00						-800.00
Hardware (in ground stanchion)	-600.00						-600.00
Stop Sign Installation Labor ($73/hr)	-1,600.00						-1,600.00
Stop Sign Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-376.66
Total Cost for 5 yrs							**-3,619.47**

Table 4.3 Cost Comparison Analysis with an inflation rate of 6% (Wright 2014; EmedCo 2017; ITSJPO 2017; Crowe 2012; Rosenlund 2017; Gustafson et al. 2014)

Instersection Management Systems - Cost Comparison Analysis							
Stop Sign Intersection Costs	**Initial Cost (US$**	**Year 1 (US$**	**Year 2 (US$**	**Year 3 (US$**	**Year 4 (US$**	**Year 5 (US$**	**NPV for 5 years (US$)**
Hardware (stop signs/poles) (4)	-800.00						-800.00
Hardware (in ground stanchion) (4)	-600.00						-600.00
Installation Labor ($73/hr)	-1,600.00						-1,600.00
Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-366.48
Total Cost for 5 yrs							**-3,366.48**
Conventional Traffic Light Costs							
Traffic signals/Installation (4)	-191,200.00						-191,200.00
O & M for traffic lights		-4,000.00	-4,000.00	-4,000.00	-4,000.00	-4,000.00	-16,849.46
Loop Detector (4)	-3,000.00						-3,000.00
Signal Preemption Reciever (2)	-3,500.00	-115.00	-115.00	-115.00	-115.00	-115.00	-3,984.42
Power Backup	-1,200.00	-100.00	-100.00	-100.00	-100.00	-100.00	-1,621.24
Linked Signal System LAN	-30,500.00	-350.00	-350.00	-350.00	-350.00	-350.00	-31,974.33
Signal Controller	-10,000.00						-10,000.00
Signal Retiming (every 4 yrs)					-3,000.00		-2,376.28
Total Cost for 5 yrs							**-261,005.72**
Transit Signal Priority Detection Systems Cost							
Optical Emitters (4)	-9,000.00						-9,000.00
Wayside Reader	-30,000.00						-30,000.00
Loop detector(4)	-3,000.00						-3,000.00
"Smart" Loops (added on to loop detector) (4)	-2,500.00						-2,500.00
Vehicle mouted GPS	-8,000.00						-8,000.00
Wireless comunication system	-10,000.00						-10,000.00
Power		-1,971.00	-1,971.00	-1,971.00	-1,971.00	-1,971.00	-8,302.57
Cellular Data		-420.00	-420.00	-420.00	-420.00	-420.00	-1,769.19
Total Cost for 5 yrs							**-72,571.76**
Dedicated Short Range Comunication (DSRC) Cost							
Hardware	-7,450.00						-7,450.00
Instalation labor	-3,500.00						-3,500.00
Design and Planning	-6,600.00						-6,600.00
On board equipment	-4,150.00						-4,150.00
Power		-100.00	-100.00	-100.00	-100.00	-100.00	-421.24
Traditional Maintenance		-500.00	-500.00	-500.00	-500.00	-500.00	-2,106.18
Licence/Maintenance Agreements		-200.00	-200.00	-200.00	-200.00	-200.00	-842.47
SCMS Certificate Licence		-50.00	-50.00	-50.00	-50.00	-50.00	-210.62
Total Cost for 5 yrs							**-25,280.51**
Envisioned V2I Communication System Cost							
Pcduino Mini Computer	-59.99						-59.99
Xbee Wireless Transmitter/USB connector	-61.95						-61.95
Solar Panel	-24.99						-24.99
Rechargeable battery	-17.99	-17.99	-17.99	-17.99	-17.99	-17.99	-93.77
Hardware (stop signs/poles)	-800.00						-800.00
Hardware (in ground stanchion)	-600.00						-600.00
Stop Sign Installation Labor ($73/hr)	-1,600.00						-1,600.00
Stop Sign Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-366.48
Total Cost for 5 yrs							**-3,607.18**

Table 4.4 Cost Comparison Analysis with an inflation rate of 7% (Wright 2014; EmedCo 2017; ITSJPO 2017; Crowe 2012; Rosenlund 2017; Gustafson et al. 2014)

Instersection Management Systems - Cost Comparison Analysis							
Stop Sign Intersection Costs	**Initial Cost (US$**	**Year 1 (US$**	**Year 2 (US$**	**Year 3 (US$**	**Year 4 (US$**	**Year 5 (US$**	**NPV for 5 years (US$)**
Hardware (stop signs/poles) (4)	-800.00						-800.00
Hardware (in ground stanchion) (4)	-600.00						-600.00
Installation Labor ($73/hr)	-1,600.00						-1,600.00
Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-356.72
Total Cost for 5 yrs							**-3,356.72**
Conventional Traffic Light Costs							
Traffic signals/Installation (4)	-191,200.00						-191,200.00
O & M for traffic lights		-4,000.00	-4,000.00	-4,000.00	-4,000.00	-4,000.00	-16,400.79
Loop Detector (4)	-3,000.00						-3,000.00
Signal Preemption Reciever (2)	-3,500.00	-115.00	-115.00	-115.00	-115.00	-115.00	-3,971.52
Power Backup	-1,200.00	-100.00	-100.00	-100.00	-100.00	-100.00	-1,610.02
Linked Signal System LAN	-30,500.00	-350.00	-350.00	-350.00	-350.00	-350.00	-31,935.07
Signal Controller	-10,000.00						-10,000.00
Signal Retiming (every 4 yrs)					-3,000.00		-2,288.69
Total Cost for 5 yrs							**-260,406.09**
Transit Signal Priority Detection Systems Cost							
Optical Emitters (4)	-9,000.00						-9,000.00
Wayside Reader	-30,000.00						-30,000.00
Loop detector(4)	-3,000.00						-3,000.00
"Smart" Loops (added on to loop detector) (4)	-2,500.00						-2,500.00
Vehicle mouted GPS	-8,000.00						-8,000.00
Wireless comunication system	-10,000.00						-10,000.00
Power		-1,971.00	-1,971.00	-1,971.00	-1,971.00	-1,971.00	-8,081.49
Cellular Data		-420.00	-420.00	-420.00	-420.00	-420.00	-1,722.08
Total Cost for 5 yrs							**-72,303.57**
Dedicated Short Range Comunication (DSRC) Cost							
Hardware	-7,450.00						-7,450.00
Instalation labor	-3,500.00						-3,500.00
Design and Planning	-6,600.00						-6,600.00
On board equipment	-4,150.00						-4,150.00
Power		-100.00	-100.00	-100.00	-100.00	-100.00	-410.02
Traditional Maintenance		-500.00	-500.00	-500.00	-500.00	-500.00	-2,050.10
Licence/Maintenance Agreements		-200.00	-200.00	-200.00	-200.00	-200.00	-820.04
SCMS Certificate Licence		-50.00	-50.00	-50.00	-50.00	-50.00	-205.01
Total Cost for 5 yrs							**-25,185.17**
Envisioned V2I Communication System Cost							
Pcduino Mini Computer	-59.99						-59.99
Xbee Wireless Transmitter/USB connector	-61.95						-61.95
Solar Panel	-24.99						-24.99
Rechargeable battery	-17.99	-17.99	-17.99	-17.99	-17.99	-17.99	-91.75
Hardware (stop signs/poles)	-800.00						-800.00
Hardware (in ground stanchion)	-600.00						-600.00
Stop Sign Installation Labor ($73/hr)	-1,600.00						-1,600.00
Stop Sign Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-356.72
Total Cost for 5 yrs							**-3,595.40**

Table 4.5 Cost Comparison Analysis with an inflation rate of 8% (Wright 2014; EmedCo 2017; ITSJPO 2017; Crowe 2012; Rosenlund 2017; Gustafson et al. 2014)

Instersection Management Systems - Cost Comparison Analysis							
Stop Sign Intersection Costs	**Initial Cost (US$**	**Year 1 (US$**	**Year 2 (US$**	**Year 3 (US$**	**Year 4 (US$**	**Year 5 (US$**	**NPV for 5 years (US$)**
Hardware (stop signs/poles) (4)	-800.00						-800.00
Hardware (in ground stanchion) (4)	-600.00						-600.00
Installation Labor ($73/hr)	-1,600.00						-1,600.00
Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-347.37
Total Cost for 5 yrs							**-3,347.37**
Conventional Traffic Light Costs							
Traffic signals/Installation (4)	-191,200.00						-191,200.00
O & M for traffic lights		-4,000.00	-4,000.00	-4,000.00	-4,000.00	-4,000.00	-15,970.84
Loop Detector (4)	-3,000.00						-3,000.00
Signal Preemption Reciever (2)	-3,500.00	-115.00	-115.00	-115.00	-115.00	-115.00	-3,959.16
Power Backup	-1,200.00	-100.00	-100.00	-100.00	-100.00	-100.00	-1,599.27
Linked Signal System LAN	-30,500.00	-350.00	-350.00	-350.00	-350.00	-350.00	-31,897.45
Signal Controller	-10,000.00						-10,000.00
Signal Retiming (every 4 yrs)					-3,000.00		-2,205.09
Total Cost for 5 yrs							**-259,831.81**
Transit Signal Priority Detection Systems Cost							
Optical Emitters (4)	-9,000.00						-9,000.00
Wayside Reader	-30,000.00						-30,000.00
Loop detector(4)	-3,000.00						-3,000.00
"Smart" Loops (added on to loop detector) (4)	-2,500.00						-2,500.00
Vehicle mouted GPS	-8,000.00						-8,000.00
Wireless comunication system	-10,000.00						-10,000.00
Power		-1,971.00	-1,971.00	-1,971.00	-1,971.00	-1,971.00	-7,869.63
Cellular Data		-420.00	-420.00	-420.00	-420.00	-420.00	-1,676.94
Total Cost for 5 yrs							**-72,046.57**
Dedicated Short Range Comunication (DSRC) Cost							
Hardware	-7,450.00						-7,450.00
Instalation labor	-3,500.00						-3,500.00
Design and Planning	-6,600.00						-6,600.00
On board equipment	-4,150.00						-4,150.00
Power		-100.00	-100.00	-100.00	-100.00	-100.00	-399.27
Traditional Maintenance		-500.00	-500.00	-500.00	-500.00	-500.00	-1,996.36
Licence/Maintenance Agreements		-200.00	-200.00	-200.00	-200.00	-200.00	-798.54
SCMS Certificate Licence		-50.00	-50.00	-50.00	-50.00	-50.00	-199.64
Total Cost for 5 yrs							**-25,093.80**
Envisioned V2I Communication System Cost							
Pcduino Mini Computer	-59.99						-59.99
Xbee Wireless Transmitter/USB connector	-61.95						-61.95
Solar Panel	-24.99						-24.99
Rechargeable battery	-17.99	-17.99	-17.99	-17.99	-17.99	-17.99	-89.82
Hardware (stop signs/poles)	-800.00						-800.00
Hardware (in ground stanchion)	-600.00						-600.00
Stop Sign Installation Labor ($73/hr)	-1,600.00						-1,600.00
Stop Sign Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-347.37
Total Cost for 5 yrs							**-3,584.11**

Table 4.6 Cost Comparison Analysis with an inflation rate of 9% (Wright 2014; EmedCo 2017; ITSJPO 2017; Crowe 2012; Rosenlund 2017; Gustafson et al. 2014)

Instersection Management Systems - Cost Comparison Analysis							
Stop Sign Intersection Costs	**Initial Cost (US$)**	**Year 1 (US$)**	**Year 2 (US$)**	**Year 3 (US$)**	**Year 4 (US$)**	**Year 5 (US$)**	**NPV for 5 years (US$)**
Hardware (stop signs/poles) (4)	-800.00						-800.00
Hardware (in ground stanchion) (4)	-600.00						-600.00
Installation Labor ($73/hr)	-1,600.00						-1,600.00
Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-338.40
Total Cost for 5 yrs							**-3,338.40**
Conventional Traffic Light Costs							
Traffic signals/Installation (4)	-191,200.00						-191,200.00
O & M for traffic lights		-4,000.00	-4,000.00	-4,000.00	-4,000.00	-4,000.00	-15,558.61
Loop Detector (4)	-3,000.00						-3,000.00
Signal Preemption Reciever (2)	-3,500.00	-115.00	-115.00	-115.00	-115.00	-115.00	-3,947.31
Power Backup	-1,200.00	-100.00	-100.00	-100.00	-100.00	-100.00	-1,588.97
Linked Signal System LAN	-30,500.00	-350.00	-350.00	-350.00	-350.00	-350.00	-31,861.38
Signal Controller	-10,000.00						-10,000.00
Signal Retiming (every 4 yrs)					-3,000.00		-2,125.28
Total Cost for 5 yrs							**-259,281.53**
Transit Signal Priority Detection Systems Cost							
Optical Emitters (4)	-9,000.00						-9,000.00
Wayside Reader	-30,000.00						-30,000.00
Loop detector(4)	-3,000.00						-3,000.00
"Smart" Loops (added on to loop detector) (4)	-2,500.00						-2,500.00
Vehicle mouted GPS	-8,000.00						-8,000.00
Wireless comunication system	-10,000.00						-10,000.00
Power		-1,971.00	-1,971.00	-1,971.00	-1,971.00	-1,971.00	-7,666.50
Cellular Data		-420.00	-420.00	-420.00	-420.00	-420.00	-1,633.65
Total Cost for 5 yrs							**-71,800.16**
Dedicated Short Range Comunication (DSRC)							
Hardware	-7,450.00						-7,450.00
Instalation labor	-3,500.00						-3,500.00
Design and Planning	-6,600.00						-6,600.00
On board equipment	-4,150.00						-4,150.00
Power		-100.00	-100.00	-100.00	-100.00	-100.00	-388.97
Traditional Maintenance		-500.00	-500.00	-500.00	-500.00	-500.00	-1,944.83
Licence/Maintenance Agreements		-200.00	-200.00	-200.00	-200.00	-200.00	-777.93
SCMS Certificate Licence		-50.00	-50.00	-50.00	-50.00	-50.00	-194.48
Total Cost for 5 yrs							**-25,006.20**
Envisioned V2I Communication System Cost							
Pcduino Mini Computer	-59.99						-59.99
Xbee Wireless Transmitter/USB connector	-61.95						-61.95
Solar Panel	-24.99						-24.99
Rechargeable battery	-17.99	-17.99	-17.99	-17.99	-17.99	-17.99	-87.96
Hardware (stop signs/poles)	-800.00						-800.00
Hardware (in ground stanchion)	-600.00						-600.00
Stop Sign Installation Labor ($73/hr)	-1,600.00						-1,600.00
Stop Sign Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-338.40
Total Cost for 5 yrs							**-3,573.29**

Table 4.7 Cost Comparison Analysis with an inflation rate of 10% (Wright 2014; EmedCo 2017; ITSJPO 2017; Crowe 2012; Rosenlund 2017; Gustafson et al. 2014)

Instersection Management Systems - Cost Comparison Analysis							
Stop Sign Intersection Costs	**Initial Cost (US$)**	**Year 1 (US$)**	**Year 2 (US$)**	**Year 3 (US$)**	**Year 4 (US$)**	**Year 5 (US$)**	**NPV for 5 years (US$)**
Hardware (stop signs/poles) (4)	-800.00						-800.00
Hardware (in ground stanchion) (4)	-600.00						-600.00
Installation Labor ($73/hr)	-1,600.00						-1,600.00
Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-329.80
Total Cost for 5 yrs							**-3,329.80**
Conventional Traffic Light Costs							
Traffic signals/Installation (4)	-191,200.00						-191,200.00
O & M for traffic lights		-4,000.00	-4,000.00	-4,000.00	-4,000.00	-4,000.00	-15,163.15
Loop Detector (4)	-3,000.00						-3,000.00
Signal Preemption Reciever (2)	-3,500.00	-115.00	-115.00	-115.00	-115.00	-115.00	-3,935.94
Power Backup	-1,200.00	-100.00	-100.00	-100.00	-100.00	-100.00	-1,579.08
Linked Signal System LAN	-30,500.00	-350.00	-350.00	-350.00	-350.00	-350.00	-31,826.78
Signal Controller	-10,000.00						-10,000.00
Signal Retiming (every 4 yrs)					-3,000.00		-2,049.04
Total Cost for 5 yrs							**-258,753.98**
Transit Signal Priority Detection Systems Cost							
Optical Emitters (4)	-9,000.00						-9,000.00
Wayside Reader	-30,000.00						-30,000.00
Loop detector(4)	-3,000.00						-3,000.00
"Smart" Loops (added on to loop detector) (4)	-2,500.00						-2,500.00
Vehicle mouted GPS	-8,000.00						-8,000.00
Wireless comunication system	-10,000.00						-10,000.00
Power		-1,971.00	-1,971.00	-1,971.00	-1,971.00	-1,971.00	-7,471.64
Cellular Data		-420.00	-420.00	-420.00	-420.00	-420.00	-1,592.13
Total Cost for 5 yrs							**-71,563.77**
Dedicated Short Range Comunication (DSRC)							
Hardware	-7,450.00						-7,450.00
Instalation labor	-3,500.00						-3,500.00
Design and Planning	-6,600.00						-6,600.00
On board equipment	-4,150.00						-4,150.00
Power		-100.00	-100.00	-100.00	-100.00	-100.00	-379.08
Traditional Maintenance		-500.00	-500.00	-500.00	-500.00	-500.00	-1,895.39
Licence/Maintenance Agreements		-200.00	-200.00	-200.00	-200.00	-200.00	-758.16
SCMS Certificate Licence		-50.00	-50.00	-50.00	-50.00	-50.00	-189.54
Total Cost for 5 yrs							**-24,922.17**
Envisioned V2I Communication System Cost							
Pcduino Mini Computer	-59.99						-59.99
Xbee Wireless Transmitter/USB connector	-61.95						-61.95
Solar Panel	-24.99						-24.99
Rechargeable battery	-17.99	-17.99	-17.99	-17.99	-17.99	-17.99	-86.19
Hardware (stop signs/poles)	-800.00						-800.00
Hardware (in ground stanchion)	-600.00						-600.00
Stop Sign Installation Labor ($73/hr)	-1,600.00						-1,600.00
Stop Sign Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-329.80
Total Cost for 5 yrs							**-3,562.91**

As presented in Table 4.2 through Table 4.7, the most expensive Intersection management system by far was the tradition traffic signal system with loop detectors with a stop sign only system, being two orders of magnitude less expensive to implement. Even though the DSRC system was much less expensive than the traffic signal system, it was still an order of magnitude more expensive than the author's system which was the closest version of technology to it. This proves that the author’s V2I communication system for intersection management was not only efficient but cost effective as well.

Cost Benefit Analyses were carried out to determine the plausibility of the implementation of said Intersection management system. Since there were no definitive records of crashes at the problem intersection, a generalized number had to be determined to assign cost of accident damages. Assuming one property-damage-only (PDO) crash was averted annually due to the implementation of a reliable control system, cost benefit analyses were completed for all the researched intersection management styles and presented in Table 4.8 through Table 4.12.

Table 4.8 Cost Benefit Analysis with convention stop sign intersection assuming one PDO crash annually (EmedCo 2017; Federal Highway Administration 2009; Crowe 2012; Rosenlund 2017)

	Intersection Management Systems - Cost Benefit Analysis						
Stop Sign Intersection Costs	**Initial Cost (US$)**	**Year 1 (US$)**	**Year 2 (US$)**	**Year 3 (US$)**	**Year 4 (US$)**	**Year 5 (US$)**	**NPV for 5 years (US$)**
Hardware (stop signs/poles) (4)	-800.00						-800.00
Hardware (in ground stanchion) (4)	-600.00						-600.00
Installation Labor ($73/hr)	-1,600.00						-1,600.00
Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-376.66
Total Cost for 5 yrs							**-3,376.66**
Benefits of reduced Accidents							
Property Damage Only Crashes		2390.05	2390.05	2390.05	2390.05	2390.05	10,347.67
Total cost savings due to reduced accidents							**10,347.67**
Benefit-Cost Ratio for 5 years							**3.06**

Table 4.9 Cost Benefit Analysis with convention traffic signal system assuming one PDO crash annually (ITSJPO 2017; Federal Highway Administration 2009)

	Intersection Management Systems - Cost Benefit Analysis						
Conventional Traffic Light Costs	**Initial Cost (US$)**	**Year 1 (US$)**	**Year 2 (US$)**	**Year 3 (US$)**	**Year 4 (US$)**	**Year 5 (US$)**	**NPV for 5 years (US$)**
Traffic signals/Installation (4)	-191,200.00						-191,200.00
O & M for traffic lights		-4,000.00	-4,000.00	-4,000.00	-4,000.00	-4,000.00	-17,317.91
Loop Detector (4)	0.00						0.00
Signal Preemption Reciever (2)	-3,500.00	-115.00	-115.00	-115.00	-115.00	-115.00	-3,997.89
Power Backup	-1,200.00	-100.00	-100.00	-100.00	-100.00	-100.00	-1,632.95
Linked Signal System LAN	-30,500.00	-350.00	-350.00	-350.00	-350.00	-350.00	-32,015.32
Signal Controller	-10,000.00						-10,000.00
Signal Retiming (every 4 yrs)					-3,000.00		-2,468.11
Total Cost for 5 yrs							**-258,632.17**
Benefits of reduced Accidents							
Property Damage Only Crashes		2390.05	2390.05	2390.05	2390.05	2390.05	10,347.67
Total cost savings due to reduced accidents							**10,347.67**
Benefit-Cost Ratio for 5 years							**0.04**

Table 4.10 Cost Benefit Analysis with Transit Signal Priority Detection system assuming one PDO crash annually (ITSJPO 2017; Federal Highway Administration 2009)

Intersection Management Systems - Cost Benefit Analysis							
Transit Signal Priority Detection Systems Cost	**Initial Cost (US$)**	**Year 1 (US$)**	**Year 2 (US$)**	**Year 3 (US$)**	**Year 4 (US$)**	**Year 5 (US$)**	**NPV for 5 years (US$)**
Optical Emitters (4)	-9,000.00						-9,000.00
Wayside Reader	-30,000.00						-30,000.00
Loop detector(4)	-3,000.00						-3,000.00
"Smart" Loops (added on to loop detector) (4)	-2,500.00						-2,500.00
Vehicle mouted GPS	-8,000.00						-8,000.00
Wireless comunication system	-10,000.00						-10,000.00
Power		-1,971.00	-1,971.00	-1,971.00	-1,971.00	-1,971.00	-8,533.40
Cellular Data		-420.00	-420.00	-420.00	-420.00	-420.00	-1,818.38
Total Cost for 5 yrs							**-72,851.78**
Benefits of reduced Accidents							
Property Damage Only Crashes		2390.05	2390.05	2390.05	2390.05	2390.05	10,347.67
Total cost savings due to reduced accidents							**10,347.67**
Benefit-Cost Ratio for 5 years							**0.14**

Table 4.11 Cost Benefit Analysis with DSRC system assuming one PDO crash annually (Wright 2014; ITSJPO 2017; Federal Highway Administration 2009)

Intersection Management Systems - Cost Benefit Analysis							
Dedicated Short Range Comunication (DSRC)	**Initial Cost (US$)**	**Year 1 (US$)**	**Year 2 (US$)**	**Year 3 (US$)**	**Year 4 (US$)**	**Year 5 (US$)**	**NPV for 5 years (US$)**
Hardware	-7,450.00						-7,450.00
Instalation labor	-3,500.00						-3,500.00
Design and Planning	-6,600.00						-6,600.00
On board equipment	-4,150.00						-4,150.00
Power		-100.00	-100.00	-100.00	-100.00	-100.00	-432.95
Traditional Maintenance		-500.00	-500.00	-500.00	-500.00	-500.00	-2,164.74
Licence/Maintenance Agreements		-200.00	-200.00	-200.00	-200.00	-200.00	-865.90
SCMS Certificate Licence		-50.00	-50.00	-50.00	-50.00	-50.00	-216.47
Total Cost for 5 yrs							**-25,380.06**
Benefits of reduced Accidents							
Property Damage Only Crashes		2390.05	2390.05	2390.05	2390.05	2390.05	10,347.67
Total cost savings due to reduced accidents							**10,347.67**
Benefit-Cost Ratio for 5 years							**0.41**

It was found that the five-year benefit of implementing a conventional traffic signal with loop detectors in a four-way intersection was $31,954 assuming a 5% inflation rate as shown in Table 4.9. Keeping in mind that this benefit is solely contingent on the assumption of the two aforementioned accidents occurring annually, it is no wonder why the problem intersection has not already been fitted with a conventional traffic light system. This case is similar to most versions of AIM, however, when compared with the benefits of the envisioned system, a different scenario can be unfolded.

Table 4.12 Cost Benefit Analysis with the envisioned system assuming one PDO crash annually (Federal Highway Administration 2009)

	Intersection Management Systems - Cost Benefit Analysis						
Envisioned V2I Communication System Cost	**Initial Cost (US$)**	**Year 1 (US$)**	**Year 2 (US$)**	**Year 3 (US$)**	**Year 4 (US$)**	**Year 5 (US$)**	**NPV for 5 years (US$)**
Pcduino Mini Computer	-59.99						-59.99
Xbee Wireless Transmitter/USB connector	-61.95						-61.95
Solar Panel	-24.99						-24.99
Rechargeable battery	-17.99	-17.99	-17.99	-17.99	-17.99	-17.99	-95.88
Hardware (stop signs/poles)	-800.00						-800.00
Hardware (in ground stanchion)	-600.00						-600.00
Stop Sign Installation Labor ($73/hr)	-1,600.00						-1,600.00
Stop Sign Yearly Maintenance Labor		-87.00	-87.00	-87.00	-87.00	-87.00	-376.66
Total Cost for 5 yrs							**-3,619.47**
Benefits of reduced Accidents							
Property Damage Only Crashes		2390.05	2390.05	2390.05	2390.05	2390.05	10,347.67
Total cost savings due to reduced accidents							**10,347.67**
Benefit-Cost Ratio for 5 years							**2.86**

It was found that the five-year benefit of implementing the envisioned system in a four-way intersection was $10,347 assuming a 5% inflation rate as shown in Table 4.8 through Table 4.12. The benefits of this study was elimination of one property-damage-only accident every year. As shown in Table 4.8 through Table 4.12, aside from in the conventional stop sign intersections and the envisioned system, the benefits do not outweigh the costs leading to many municipalities to not see the need for implementation of more advanced intersection management systems. An example of this is in the problem intersection chosen as motivation for this study.

Even though this cost-benefit analysis is solely based on the assumption of the occurrence of accidents that have not been common in the past, the possible benefits may very well outweigh the costs significantly. This is due to the fact that with the development rate of Statesboro, GA, the city will become a more urban setting in the next few decades, thereby increasing the likelihood of traffic accidents as they are directly proportional to the traffic density in an area.

CHAPTER 5

CONCLUSION

5.1 Validation of Hypothesis

The study indicated that a cost effective wireless communication system could be designed to assign priority to vehicles entering a pre-mapped intersection based on the vehicles' times of arrival and their projected paths through the intersection. The system was proven to allow, in certain cases where the paths of two arriving vehicles do not intersect, two vehicles to navigate the intersection concurrently without a possibility of collision.

The study proved the cost effectiveness of the proposed system by performing a cost comparison analysis, in which the components and maintenance costs of this system were compared to other conventional and novel concepts for Intelligent Intersection Management. The vehicles traversing the intersection in the study were programmed to turn off all environmental sensors while in the intersection as this would further prove the study's hypothesis that no visual referencing or human intervention was necessary to carry out this cost effective variation of Intelligent Intersection Management.

The study simplified the initially intended five-way intersection to a more conventional four way intersection in such a way as to prove transferability of the study's concepts to virtually any possible design of an intersection, provided that the paths through the intersection were preprogrammed. The method of preprogramming the paths into the tested vehicles could range from manual mapping of the intersection and coding of the vehicle's intersection navigation protocol as was performed in the study, to developing an algorithm which autonomously builds the information based on recording multiple paths as human drivers traversed the intersection.

5.2 Validation of Criteria for Success

The criteria for success involved developing a system of intersection management which included: communicating with a fully autonomous vehicle equipped with a preprogrammed map of the intersection, capability of navigating the turns using external and internal sensors, developing a human interface module for communicating with human drivers of vehicles not equipped with advanced drivers assistance systems (ADAS), and testing all possible combinations of scenarios possible at a scaled intersection to prove reliability of the system.

As presented in the study, a fully autonomous small scale prototype vehicle was developed with autonomous lane following and traffic sign recognition, as well as obstacle recognition and avoidance protocols, and fitted with an XBee radio transceiver for reliable wireless communication with an autonomous base station. An autonomous communication module was developed which would simulate the strings broadcasted by an autonomous vehicle requesting priority after arriving at the intersection. This module would be compatible with conventional vehicles which do not have Level 4 or higher intelligence capabilities. Finally, the system was tested in 27 different combination of vehicle turns to establish the system's robustness and proved capable of handling all combinations tested.

5.3 Recommendations for Future Research

Several concerns were recorded during the implementation of the study which suggest for a more thorough perusal. One such concern was the inability of the Arduino microcontrollers which served as the 'brain' of the small scale autonomous vehicle prototypes to distinguish the components of two different strings broadcasted simultaneously due to the microcontroller's relatively slow upload rate. A recommendation would be to replace the microcontrollers with single board computer platforms identical with the one employed in the base station. These platforms proved to be much faster and more reliable with wireless communication and able to perform multiple tasks simultaneously. They also proved to possess overall higher processing power.

The next recommendation for future research would be the implementation of the same system in, first a similar scale five-way intersection experimental setup modelled after the problem intersection, then in multiple human scaled autonomous vehicle prototypes, and finally implementing tests in the actual problem intersection. The latter part of this recommendation will involve finalizing the development of the meso-scaled and large scaled autonomous vehicle platforms currently in development, and implementing the necessary sensors and single board computer platforms in both of them.

Modules for pedestrian crossings on the intersection would also have to be developed to accommodate pedestrians crossing the intersections. This would be fairly easy to implement, as all it would require is a microcontroller/single board computer platform which broadcasts a particular string that closes one leg of the intersection temporarily to the base station, allowing other unaffected vehicles to continue their navigation protocols.

The final recommendation would be to implement several XBees equipped with a secure method of communication (not vulnerable to cyber-attacks) on each autonomous vehicle to allow for communication over separate designated networks which would be used for different kinds of messages similarly to the bandwidths used in Designated Short Range Communication (DSRC). This would allow for the use of wireless communication for much more than just intelligent intersection management. Perfecting this system through the collaboration with experts in Computer Science in the future would be necessary to develop a system truly immune to cyber-attacks.

BIBLIOGRAPHY

Andersen, Carl. 2017. “Probability of Implementing Level 5 Autonomy in Intelligent Vehicles.” *Interview by Author*. Detroit, Michigan.

Banjanovic-Mehmedovic, Lejla, Adnan Husejnovic, Ivan Bosanki, and Suad Kasapovic. 2016. “Monitoring of Cooperation between Autonomous Vehicles in Roundabout Environemnt.” In *International Confernece for NEW TECHNOLOGIES NT - Development and Application*. Mostar, Bosnia and Herzegovina: Society for Robotics of Bosnia and Herzegovina. https://www.researchgate.net/publication/304150944_Monitoring_of_cooperation_between_autonomous_vehicles_in_roundabout_environemnt.

Barker, John L. 1960. Vehicle Detector. US2965893 A, issued 1960. https://www.google.com/patents/US2965893.

Bayraktaroglu, Burhan. 1990. Traffic Light Control System and Method. 4,908,615, issued 1990. https://www.google.com/patents/US7689347.

Beyerl, Thomas. 2017. “Intelligent Vehicle Complex Environment Path Following Capabilities, Based on a Telemetry Sensor Fusion Approach Utilizing Open Loop Navigation.” Georgia Southern University.

Chen, Wenjie, Lifeng Chen, Zhanglong Chen, and Shiliang tu. 2006. “WITS: A Wireless Sensor Network for Intelligent Transportation System.” *First International Multi- Symposiums on Computer and Computational Sciences, IMSCCS’06*.

Choi, E-H. 2010. “Crash Factors in Intersection-Related Crashes: An on-Scene Perspective,” no. September: 37. doi:http://dx.doi.org/10.1037/e621942011-001.

Commission, U.S. Federal Communications. 2006. “Amendment of the Commission’s Rules Regarding Dedicated Short-Range Communication Services in the 5.850-5.925 GHz Band (5.9 GHz Band).” *FCC 06-110*. https://apps.fcc.gov/edocs_public/attachmatch/FCC-06-110A1.pdf.

Crowe, Aaron. 2012. “You Hit It, You Bought It.” *Carinsurance.com*. http://www.nasdaq.com/article/you-hit-it-you-bought-it-cm140671.

Dey, Kakan Chandra, Anjan Rayamajhi, Mashrur Chowdhury, Parth Bhavsar, and James Martin. 2016. “Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) Communication in a Heterogeneous Wireless Network - Performance Evaluation.” *Transportation Research Part C: Emerging Technologies* 68. Elsevier Ltd: 168–84. doi:10.1016/j.trc.2016.03.008.

Dresner, Kurt Mauro. 2009. “Autonomous Intersection Management.” https://repositories.lib.utexas.edu/handle/2152/ETD-UT-2009-12-689.

EmedCo. 2017. “Official Standard STOP Signs.” http://www.emedco.com/all-stop-signs-cstp1a.html?keycode=WB0106&campname=PC-03-TPS%7C%7CNB%7C%7CPPC%7C%7CSigns%7C%7CStopSigns%7C%7CAll-RB&gclid=CjwKCAjwqcHLBRAqEiwA-j4AyKoljv5fVjPSWKjZG2kvVhEMFar-Um5rAEYtUdNIXrCV3KvSHtJQ3BoCBUAQAvD_BwE.

Federal Highway Administration. 2009. “The National Intersection Safety Problem,” no. November: 2007–10.

Fernández-Lozano, J.J., Miguel Martín-guzmán, Juan Martín-ávila, and A. García-Cerezo. 2015. “A Wireless Sensor Network for Urban Traffic Characterization and Trend Monitoring.” *Sensors*, 26143–69. doi:10.3390/s151026143.

Gong, Yue-jiao, and Jun Zhang. 2014. "Real-Time Traffic Signal Control for Modern Roundabouts by Using Particle Swarm Optimization-Based Fuzzy Controller," no. August.

Google Maps. 2017. "Google Maps Views of Downtown Statesboro Nonstandard Intersection." https://www.google.com/maps/place/GA-21+%26+US-301+%26+Statesboro+Hwy,+Sylvania,+GA+30467/@32.4378484,-81.7842762,148a,35y,44.99t/data=!3m1!1e3!4m5!3m4!1s0x88fa28dc3e47f311:0x694683a9df27b9!8m2!3d32.7291446!4d-81.6542315.

Greater Amman Municipality. 2007. "Traffic Report Study 2007." Amman, Jordan.

Gustafson, Joe, Kyle Hartnett, Howard Preston, Michael Barry, K C Atkins, and C H M Hill. 2014. "Traffic Sign Maintenance / Management Handbook." St. Paul, Minnesota. http://www.dot.state.mn.us/stateaid/trafficsafety/retroreflectivity/mndot-traffic-sign-maint-man-large.pdf.

Harding, John, Gregory Powell, Rebecca Yoon, Joshua Fikentscher, Charlene Doyle, Dana Sade, Mike Lukuc, Jim Simons, and Jing Wang. 2014. "Vehicle-to-Vehicle Communications : Readiness of V2V Technology for Application," no. August: 327. doi:Report No DOT HS 812 014.

Hausknecht, Matthew, Tsz Chiu Au, and Peter Stone. 2011. "Autonomous Intersection Management: Multi-Intersection Optimization." *IEEE International Conference on Intelligent Robots and Systems*, no. September: 4581–86. doi:10.1109/IROS.2011.6048565.

Ibru, B., J. Williams, I. Augusma, V. Soloiu, and T. Beyerl. 2016. "Location Sensor Fusion and Error Correction in Intelligent Vehicles." In *ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE)*. Vol. 12. doi:10.1115/IMECE201667084.

Ibru, Bernard, Valentin Soloiu, Thomas Beyerl, Tyler Naes, Charvi Popat, Cassandra Sommer, and Brittany Williams. 2017. "Optimizing Color Detection with Robotic Vision Sensors for Lane Following and Traffic Sign Recognition in Small Scale Autonomous Test Vehicles." In *SAE World Congress*. Detroit, Michigan: SAE International. doi:10.4271/2017-01-0096.Copyright.

Ioannou, Petros A., and C. C. Chien. 1993. "Autonomous Intelligent Cruise Control." *IEEE Transactions on Vehicular Technology* 42 (4): 657–72. doi:10.1109/25.260745.

Iseki, Hiroyuki, and Alexander Demisch. 2012. "Examining the Linkages between Electronic Roadway Tolling Technologies and Road Pricing Policy Objectives." *Transportation Research Parc C* 36 (1): 121–32. doi:10.1016/j.retrec.2012.03.008.

ITSJPO. 2017. "Cost Database; Office of of the Assistant Secretary for Research and Technology, U.S. Department of Transportation." *Transportation Research Bureau*. http://www.itskrs.its.dot.gov/its/benecost.nsf/ID/2706A0BD21F048F585257B65005F20B0?OpenDocument&Query=Home.

Jiang, Daniel, Vikas Taliwali, Andreas Meier, and Wieland Holfelder. 2006. "Design of 5.9 GHz DSRC-Based Vehicular Safety Communication." *IEEE Wireless Communications*, no. October: 36–43.

Jimb0. 2010. "Serial Communication Course." *Sparkfun*. http://www.eeherald.com/section/design-guide/esmod7.html.

Kamal, M. A S, J. Imura, A. Ohata, T. Hayakawa, and K. Aihara. 2013. "Coordination of Automated Vehicles at a Traffic-Lightless Intersection." *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, no. Itsc: 922–27. doi:10.1109/ITSC.2013.6728350.

Keeratiwintakorn, Phongsak, Ekkabut Thepnorarat, and Apirak Russameesawang. 2009. "Ubiquitous

Communication for V2V and V2I for Thailand Intelligent Transportation System.” *European Telecommunications*, no. January: 81–85.

Kenney, John B. 2011. “Dedicated Short-Range Communications (DSRC) Standards in the United States.” *Proceedings of the IEEE* 99 (7): 1162–82. doi:10.1109/JPROC.2011.2132790.

Kwaśnicka, Halina, and Bartosz Wawrzyniak. 2002. “License Plate Localization and Recognition in Camera Pictures.” *3rd Symposium on Methods of Artificial Intelligence*, 243–46. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:License+plate+localization+and+recognition+in+camera+pictures#0.

Lee, Joyoung, and Byungkyu Park. 2012. “Development and Evaluation of a Cooperative Vehicle Intersection Control Algorithm under the Connected Vehicles Environment.” *IEEE Transactions on Intelligent Transportation Systems* 13 (1): 81–90. doi:10.1109/TITS.2011.2178836.

Li, Benliang, Houjun Wang, Bin Yan, and Chijun Zhang. 2006. “The Research of Applying Wireless Sensor Networks to Intelligent Transportation System(ITS) Based On IEEE 802.15.4.”

Lipar, Peter, and Jure Kostanjsek. 2017. “Traffic Calming in Slovenia.”

MarshProducts. 2000. “The Basics of Loop Vehicle Detection.” http://www.marshproducts.com/pdf/Inductive Loop Write up.pdf.

Miucic, Radovan, Zeljko Popovic, and Syed M. Mahmud. 2009. “Experimental Characterization of DSRC Signal Strength Drops.” *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 311–15. doi:10.1109/ITSC.2009.5309695.

Online User Doe. (Techwomen.co). 2017. “Zigbee Wireless Sensor Network Architecture.”

Osborne, Terri. 2009. “2009 Annual Traffic Report.” Alpharetta, GA. http://www.dot.ga.gov/DriveSmart/SafetyOperation/Documents/Red/2009AnnualRedLightReport.pdf.

Palma, André De, and Robin Lindsey. 2011. “Traffic Congestion Pricing Methodologies and Technologies.” *Transportation Research Part C* 19 (June 2011): 1377–99. doi:10.1016/j.trc.2011.02.010.

Pascale, A., M. Nicoli, F. Deflorio, B. Dalla Chiara, and U. Spagnolini. 2012. “Wireless Sensor Networks for Traffic Management and Road Safety.” *IET Intelligent Transport Systems* 6 (1): 67. doi:10.1049/iet-its.2010.0129.

Peden, Margie, Richard Scurfield, David Sleet, Dinesh Mohan, Adnan A. Hyder, Eva Jarawan, and Colin Mathers. 2006. “World Report on Road Traffic Injury Prevention.” *World Health Organization*, 47. doi:10.1016/j.puhe.2005.09.003.

Resendes, Ray. 2010. “Vehichle-to-Vehicle and Safety Pilot.” https://www.its.dot.gov/presentations/Safety_workshop2010/Vehicle-toVehicle and Safety Pilot -- R Resendes.pdf.

Rios-Gutierrez, Fernando (Georgia Southern University). 2017. “Distance Meters and Object Detectors.” Statesboro, GA.

ROS.org. 2017. “ROS Documantation.” *Open Source Robotics Foundation*. http://wiki.ros.org/.

Rosenlund, Joseph. 2017. “LED Traffic Signals.” *City of Yakima, Washington Department of Public Works Streets and Traffic Division*. https://www.yakimawa.gov/services/streets/led-traffic-signals/.

Sagberg, Fridulv. 1999. “Road Accidents Caused by Drivers Falling Asleep.” *Accident Analysis and Prevention* 31 (6): 639–49. doi:10.1016/S0001-4575(99)00023-8.

Schweitzer, Naftali, Joseph S. Bodenhelmer, and Gerald Ben David. 1989. Traffic Safety Monitoring Apparatus. EP0339988A2, issued 1989. https://www.google.com/patents/EP0339988A2.

Subramanian, Rajesh, and Louis Lombardo. 2007. “Analysis of Fatal Motor Vehicle Traffic Crashes and Fatalities at Intersections.” *Npo-121*, no. February: 3. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/810682.

Sullivan, Andrew, Pe L Steven Jones, Elsa Tedla Ehsan Doustmohammadi, Robert Blankenship, Aldot Maintenance Bureau Tim Barnett, Stuart Manson, and Gary Moore. 2015. “Traffic Signal Design Guide & Timing Manual Alabama Department of Transportation TRAFFIC SIGNAL DESIGN GUIDE & TIMING MANUAL,” no. June.

Tyburski, Robert. 2002. Permanent In-pavement Roadway Traffic Sensor System. US 6,417,785 B1, issued 2002. doi:10.1074/JBC.274.42.30033.(51).

Virginia Department of Transportation. 2008. “Tolling Facilities Report.” *Report to the General Assembly of Virginia* 3202 (January): 1–46.

Wright, James. 2014. “National Connected Vehicle Field Infrastructure Footprint Analysis Final Report.” *Final Report v1 — June 27, 2014*. Nationwide, United States. doi:FHWA-JPO-14-125.

WSDOT. 2017. “Traffic Signals and Signal Coordinations.” *Traffic Operations*. Accessed July 24. https://wsdot.wa.gov/Operations/Traffic/signals.htm#Coordination and Timing.

Wu, Yao, Jian Lu, Hong Chen, and Haifei Yang. 2015. “Development of an Optimization Traffic Signal Cycle Length Model for Signalized Intersections in China.” *Mathematical Problems in Engineering* 2015 (1). doi:10.1155/2015/954295.

Yousef, Khallil M. Ahmad, and Ali Mohammd Shatnawi. 2010. “Intelligent Traffic Light Flow Control System Using Wireless Sensors Networks.” *Journal of Information Science and Engineering* 768 (August 2015): 753–68.

Zhou, B, J Cao, X Zeng, and H Wu. 2010. “Adaptive Traffic Light Control in Wireless Sensor Network-Based Intelligent Transportation System.” *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, 1–5. doi:10.1109/VETECF.2010.5594435.

Zhu, Feng, and Satish V. Ukkusuri. 2015. “A Linear Programming Formulation for Autonomous Intersection Control within a Dynamic Traffic Assignment and Connected Vehicle Environment.” *Transportation Research Part C: Emerging Technologies* 55 (2015). Elsevier Ltd: 363–78. doi:10.1016/j.trc.2015.01.006.

APPENDICES

APPENDIX A – Vehicle Control Algorithm

This Arduino program controlled Vehicle 0 through 2 to carry out all autonomous function. This program was not modified in any way between tests. All intersection navigation turns were preprogrammed. The vehicle awaited an initial "GO" command to determine which of the turns would be executed.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
#include <LiquidCrystal.h>
#include <Pixy.h>
#include <Servo.h>
Pixy pixy;
int blockSize;

int StopPIN = 13;
int Turn1 = 52;
int whichTurn = 12;
int Stopval;
int Turnval1;
int whichTurnval;
int setSize = 300;
int tvariance;
int turnSpeed;
float centerline = 45;//55;
float delta;
long duration[] = {0, 0, 0, 0};
long inches[] = {0, 0, 0, 0};
long cm[] = {0, 0, 0, 0};
long meter[] = {0, 0, 0, 0};
long sensorValue[] = {0, 0, 0};

long microsecondsToInches(long microseconds) {
  return microseconds / 74 / 2;
}
long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}

#define BNO055_SAMPLERATE_DELAY_MS (100)

float setDir;   //set hold direction in degrees
float pos1;
float pos2;
float pos3;
int ObstMsg = 0;
int legTime = 5000;
float variance;
float deviation;
unsigned long previousMillis = 0;
unsigned long currentMillis = 0;
const long interval = 200;
float var;
int vID = 0;  //Vehicle Identification Number (only difference between programs uploaded on each of the three robots)
int sID = vID + 2;  //start Identification Number (because I shifted the start command over one digit)
int startcmd[5];
int requestatus[5];
int confirmstatus[5];
```

```
Adafruit_BNO055 bno = Adafruit_BNO055(55);

Servo leftWheel;
Servo rightWheel;
LiquidCrystal lcd(41, 43, 45, 47, 49, 51);

void displaySensorDetails(void)
{
  sensor_t sensor;
  bno.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" xxx");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" xxx");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" xxx");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

void displaySensorStatus(void)
{
  /* Get the system status values (mostly for debugging purposes) */
  uint8_t system_status, self_test_results, system_error;
  system_status = self_test_results = system_error = 0;
  bno.getSystemStatus(&system_status, &self_test_results, &system_error);

  /* Display the results in the Serial Monitor */
  Serial.println("");
  Serial.print("System Status: 0x");
  Serial.println(system_status, HEX);
  Serial.print("Self Test:     0x");
  Serial.println(self_test_results, HEX);
  Serial.print("System Error:  0x");
  Serial.println(system_error, HEX);
  Serial.println("");
  delay(500);
}

void displayCalStatus(void)
{
  /* Get the four calibration values (0..3) */
  /* Any sensor data reporting 0 should be ignored, */
  /* 3 means 'fully calibrated" */
  uint8_t system, gyro, accel, mag;
  system = gyro = accel = mag = 0;
  bno.getCalibration(&system, &gyro, &accel, &mag);

  /* The data should be ignored until the system calibration is > 0 */
  Serial.print("\t");
  if (!system)
  {
    Serial.print("! ");
  }

  /* Display the individual values */
  Serial.print("Sys:");
  Serial.print(system, DEC);
  Serial.print(" G:");
  Serial.print(gyro, DEC);
  Serial.print(" A:");
  Serial.print(accel, DEC);
  Serial.print(" M:");
  Serial.print(mag, DEC);
}
```

```
void setup()  //program initializer
{
  pinMode(A1, INPUT);
  Serial.begin(9600);  //sets baud rate for wired communication (debugging using serial monitor)
  Serial3.begin(57600);  //sets baud rate for wireless communication
  Serial.println("Orientation Sensor Test"); Serial.println("");
  Serial.println("Starting...\n");
  pinMode(StopPIN, INPUT);
  pinMode(Turn1, INPUT);
  pinMode(whichTurn, OUTPUT);
  pixy.init();
  leftWheel.attach(10);
  rightWheel.attach(11);
  if (!bno.begin())
  {
    /* There was a problem detecting the BNO055 ... check your connections */
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!");
    while (1);
  }

  delay(1000);

  /* Display some basic information on this sensor */
  displaySensorDetails();

  /* Optional: Display current status */
  displaySensorStatus();

  bno.setExtCrystalUse(true);


  lcd.begin (16, 2);

  /* Get a new sensor event */
  sensors_event_t event;
  bno.getEvent(&event);

  while (Serial3.available() < 5) {}  //wait for full command set
  for (int n = 0; n < 5; n++)
  {
    startcmd[n] = Serial3.parseInt();    //load command variable with turn instruction set
  }

  //for (int n = 0; n < 5; n++)      //test routine for confirmation
  //{
    //Serial3.println(startcmd[n]);
  //}

  while (startcmd[1] < 1 || startcmd[sID] != 1)  //wait for my start command
  {
    leftWheel.writeMicroseconds(1500);
    rightWheel.writeMicroseconds(1500);
    //Serial3.println("Test Stop Command");
    displayCalStatus();
    delay(500);

    if (Serial3.available() > 4)
    {
      for (int n = 0; n < 5; n++)
      {
        startcmd[n] = Serial3.parseInt();
      }

    }
  }
  //Serial3.print("Start Navigating recieved by");
  //Serial3.println(vID);

}
```

```
void sense()  //subroutine for obstacle detection using the three ultrasonic sensors
{
  for (int j = 0; j < 3; j++) {
    int pingPin = j + 30;
    pinMode(pingPin, OUTPUT);
    digitalWrite(pingPin, LOW);
    delayMicroseconds(2);
    digitalWrite(pingPin, HIGH);
    delayMicroseconds(5);
    digitalWrite(pingPin, LOW);
    pinMode(pingPin, INPUT);
    duration[j] = pulseIn(pingPin, HIGH);
    inches[j] = microsecondsToInches(duration[j]);
    cm[j] = microsecondsToCentimeters(duration[j]);
    meter[j] = cm[j] * 0.01;
  }
}

void executeTurn1() {

  leftWheel.writeMicroseconds(1500);
  rightWheel.writeMicroseconds(1500);
  delay(150);

  sense();
  if (inches[0] <= 6 || inches[1] <= 6 || inches[2] <= 6) {
    Serial.println("Obstacle");
    leftWheel.writeMicroseconds(1500);
    rightWheel.writeMicroseconds(1500);
    while (inches[0] <= 6 || inches[1] <= 6 || inches[2] <= 6 ) {
      sense();
    };

  }
  else {
    sensors_event_t event;
    bno.getEvent(&event);
    pos1 = event.orientation.x;
    delay(BNO055_SAMPLERATE_DELAY_MS);

    for (int cycles = 0; cycles < 85; cycles ++)
    {
      sensors_event_t event;
      bno.getEvent(&event);

      setDir = pos1;
      if (event.orientation.x - setDir > 180)
      {
        deviation = 360 - event.orientation.x + setDir;
      }
      else
      {
        deviation = setDir - event.orientation.x;
      }
      leftWheel.writeMicroseconds(1607 + (5 * deviation));       // corrected drive
      rightWheel.writeMicroseconds(1610 - (5 * deviation));
      delay(BNO055_SAMPLERATE_DELAY_MS);
    }


    pos2 = pos1 - 90;
    delay(100);

    for (setDir = pos1; setDir >= pos2; setDir -= 5) {
      sensors_event_t event;
      bno.getEvent(&event);

      /* Optional: Display calibration status */
      //displayCalStatus();
      /* Optional: Display sensor status (debug only) */
      //displaySensorStatus();

      /* New line for the next sample
        Serial.println(""); */

      if (event.orientation.x - setDir > 180)
      {
        deviation = 360 - event.orientation.x + setDir;
      }
      else
      {
        deviation = setDir - event.orientation.x;
      }

      leftWheel.writeMicroseconds(1607 + (2.5 * deviation));
      rightWheel.writeMicroseconds(1610 - (5 * deviation));


      /* Wait the specified delay before requesting nex data */
      delay(BNO055_SAMPLERATE_DELAY_MS);

    }


    for (int cycles = 0; cycles < 25; cycles ++)
    {
      sensors_event_t event;
      bno.getEvent(&event);
      setDir = pos2;
      if (event.orientation.x - setDir > 180)
      {
        deviation = 360 - event.orientation.x + setDir;
      }
      else
      {
        deviation = setDir - event.orientation.x;
      }
      leftWheel.writeMicroseconds(1607 + (5 * deviation));       // corrected drive
      rightWheel.writeMicroseconds(1610 - (5 * deviation));
      delay(BNO055_SAMPLERATE_DELAY_MS);
    }
  }
  ObstMsg = 1;
  confirmstatus[0] = 1;
  for (int n = 0; n < 4; n++)
  {
    Serial3.print(confirmstatus[n]);
    Serial3.print(",");
  }
  Serial3.print(confirmstatus[4]);
}

void executeTurn2() {

  leftWheel.writeMicroseconds(1500);
  rightWheel.writeMicroseconds(1500);
  delay(150);

  sense();
  if (inches[0] <= 6 || inches[1] <= 6 || inches[2] <= 6) {
    Serial.println("Obstacle");
    leftWheel.writeMicroseconds(1500);
    rightWheel.writeMicroseconds(1500);
    while (inches[0] <= 6 || inches[1] <= 6 || inches[2] <= 6 ) {
      sense();
    };

  }
  }
  else {
    sensors_event_t event;
    bno.getEvent(&event);
    pos1 = event.orientation.x;
    delay(BNO055_SAMPLERATE_DELAY_MS);

    for (int cycles = 0; cycles < 105; cycles ++)
    {
      sensors_event_t event;
      bno.getEvent(&event);

      setDir = pos1;
      if (event.orientation.x - setDir > 180)
      {
        deviation = 360 - event.orientation.x + setDir;
      }
      else
      {
        deviation = setDir - event.orientation.x;
      }
      leftWheel.writeMicroseconds(1607 + (5 * deviation));
      rightWheel.writeMicroseconds(1610 - (5 * deviation));

      delay(BNO055_SAMPLERATE_DELAY_MS);
    }
  }
  confirmstatus[0] = 1;
  for (int n = 0; n < 4; n++)
  {
    Serial3.print(confirmstatus[n]);
    Serial3.print(",");
  }
  Serial3.print(confirmstatus[4]);
}

void executeTurn3() {

  leftWheel.writeMicroseconds(1500);
  rightWheel.writeMicroseconds(1500);
  delay(150);

  sense();
  if (inches[0] <= 6 || inches[1] <= 6 || inches[2] <= 6) {
    Serial.println("Obstacle");
    leftWheel.writeMicroseconds(1500);
    rightWheel.writeMicroseconds(1500);
    while (inches[0] <= 6 || inches[1] <= 6 || inches[2] <= 6 ) {
      sense();
    };

  }
  else {
    sensors_event_t event;
    bno.getEvent(&event);
    pos1 = event.orientation.x;
    delay(BNO055_SAMPLERATE_DELAY_MS);

    for (int cycles = 0; cycles < 35; cycles ++)  //keep proceeding forward for predetermined number of cycles
    {
      sensors_event_t event;
      bno.getEvent(&event);
    setDir = pos1;
    if (event.orientation.x - setDir > 180)
    {
      deviation = 360 - event.orientation.x + setDir;
    }
    else
    {
      deviation = setDir - event.orientation.x;
    }
    leftWheel.writeMicroseconds(1607 + (5 * deviation));       // corrected drive
    rightWheel.writeMicroseconds(1610 - (5 * deviation));

    delay(BNO055_SAMPLERATE_DELAY_MS);
  }


  pos2 = pos1 + 90;  //makes the vehicle turn right 90 degrees
  delay(100);

  for (setDir = pos1; setDir <= pos2; setDir += 5)
  {
    sensors_event_t event;
    bno.getEvent(&event);

    /* Optional: Display calibration status */
    //displayCalStatus();

    /* Optional: Display sensor status (debug only) */
    //displaySensorStatus();

    /* New line for the next sample
      Serial.println(""); */

    if (event.orientation.x - setDir > 180)
    {
      deviation = 360 - event.orientation.x + setDir;
    }
    else
    {
      deviation = setDir - event.orientation.x;
    }

    leftWheel.writeMicroseconds(1607 + (2.5 * deviation));
    rightWheel.writeMicroseconds(1610 - (5 * deviation));


    /* Wait the specified delay before requesting nex data */
    delay(BNO055_SAMPLERATE_DELAY_MS);

  }


  for (int cycles = 0; cycles < 25; cycles ++) //keep proceeding forward for predetermined number of cycles
  {
    sensors_event_t event;
    bno.getEvent(&event);
    setDir = pos2;
    if (event.orientation.x - setDir > 180)
    {
      deviation = 360 - event.orientation.x + setDir;
    }
    else
    {
      deviation = setDir - event.orientation.x;
    }
```

```
      leftWheel.writeMicroseconds(1607 + (5 * deviation));       // corrected drive
      rightWheel.writeMicroseconds(1610 - (5 * deviation));
      delay(BNO055_SAMPLERATE_DELAY_MS);
    }
  }
  ObstMsg = 1;
  confirmstatus[0] = 1;
  for (int n = 0; n < 4; n++)
  {
    Serial3.print(confirmstatus[n]);
    Serial3.print(",");
  }
  Serial3.print(confirmstatus[4]); //sends message to base to confirm completion of intersection navigation
}

void waitTurn() {
  requestatus[0] = 1;   //if 0, then the base knows the message is not for it. if 1, then the base knows the message is meant for it.
  requestatus[1] = vID;  //first item in request string to let the base know which vehicle is requesting priority
  requestatus[2] = vID + 1;  //this is the starting position of the vehicle. mentioned in the start command
  requestatus[3] = vID + 1 + startcmd[1];  //assuming each vehicles start position remains unchanged, adding the turn direction will
  if (requestatus[3] > 4){
    requestatus[3] = requestatus[3] - 4;
  }
  requestatus[4] = 0;  //requesting priority is 0 if the vehicle is still waiting and 1 if it finished navigating
  for (int n = 0; n < 4; n++)
  {
    Serial3.print(requestatus[n]);
    Serial3.print(",");
  }
  Serial3.print(requestatus[4]);

  Serial.println("Request Sent");

  while (Serial3.available() < 5) {}
  for (int n = 0; n < 5; n++) // make note to check
  {
    confirmstatus[n] = Serial3.parseInt();
  }
  while (confirmstatus[1] != vID || confirmstatus[4] != 1)  //make a note to check
  {
    if (Serial3.available() > 4)
    {
      for (int n = 0; n < 5; n++)
      {
        confirmstatus[n] = Serial3.parseInt();
      }
    }
    else {};
  }

  Serial.println("Request awarded");
}

void loop() {
  Stopval = digitalRead(StopPIN);  //looks for presence of stop sign

  if (Stopval == LOW)  //routine followed when no stop sign detected
  {
    Serial.print("Go Staight  Stopval = ");
    Serial.println(Stopval);

    follow_median();
  }

  else if (Stopval == HIGH) //routine followed when stop sign detected
  {
    Serial.print("Turn1 is stop sign on? = ");
    Serial.println(Stopval);
    leftWheel.writeMicroseconds(1500);
    rightWheel.writeMicroseconds(1500);
    waitTurn();
    if (startcmd[1] == 1)  //routine followed when initial path involved turning left at intersection
    {
      //Serial3.println("Executing turn 1");
      executeTurn1();
    }
    if (startcmd[1] == 2)  //routine followed when initial path involved proceeding straight at intersection
    {
      //Serial3.println("Executing turn 2");
      executeTurn2();
    }
    if (startcmd[1] == 3)  //routine followed when initial path involved turning right at intersection
    {
      //Serial3.println("Executing turn 3");
      executeTurn3();
    }
    else {
      Serial.print("Confusing command for =");
      Serial.println(vID);
    }
  }
}

void follow_median()
{
  static int i = 0;
  int j;
  uint16_t blocks;
  char buf[32];

  blocks = pixy.getBlocks();

  if (blocks) {
    sense();
    if (inches[0] <= 6 || inches[1] <= 6 || inches[2] <= 6) {
      Serial.println("Obstacle");
      //Serial3.print(1);
      leftWheel.writeMicroseconds(1500);
      rightWheel.writeMicroseconds(1500);
      while (inches[0] <= 6 || inches[1] <= 6 || inches[2] <= 6 ) {
        sense();
        if (ObstMsg == 1) {
          //Serial3.print("1");
        }
        else {
          //Serial3.print("0");
        }
      }
    }
    else {
      i++;
      if (i % 50 == 0)
      {
        {}
      }

      delta = (centerline - pixy.blocks[0].x);

      leftWheel.writeMicroseconds(1600 - (1 * delta));
      rightWheel.writeMicroseconds(1615 + (0.25 * delta));
      Serial.println("Navigating");
      delay (100);
    }

  }

  else {
    Serial.println("Error Finding Line");
    leftWheel.writeMicroseconds(1500);
    rightWheel.writeMicroseconds(1500);
    delay (100);
  }
}
```

APPENDIX B – Main Node of Base Station and included Main.h header file

This node and header file were responsible for organizing incoming vehicles into priority lists and determining which vehicles had the right of way. The list was constantly updated when new vehicles arrived at the intersection and when previous vehicle had completed the intersection. It was also responsible for making intersecting paths unavailable to incoming vehicles based on the path of the currently navigating vehicle.

```
#include <main.h>

int main (int argc, char** argv)
{
   ros::init(argc, argv, "main_node");
   Main main;
   //main.roads_on();

   while(ros::ok())
   {
      ros::spin();
   }
   ros::shutdown();
   return 0;
}


#pragma once
#include <ros/ros.h>
#include <serial/serial.h>
#include <std_msgs/String.h>
#include <std_msgs/Empty.h>
#include <V2I_communication/read_vector.h>
#include <robot.h>
#include <vector>
#include <string>
#include <sstream>
#include <iterator>
#include <algorithm>
#include <iostream>
#include <stdio.h>

#ifndef MAIN_H
#define MAIN_H

using namespace std;

class Main
{
   public:
      Main(){
         robot_pub = nh.advertise<std_msgs::String>("write", 5);
         new_vector_sub = nh.subscribe<V2I_communication::read_vector>("readVectornew", 5,
&Main::new_vehicle_callback, this);
         done_vector_sub = nh.subscribe<V2I_communication::read_vector>("readVectordone", 5,
&Main::done_vehicle_callback, this);
```

```
    }

/*******************
NEW_VEHICLE_CALLBACK
*******************/

      //[1, vID, current_location, desired_location, status]

      void new_vehicle_callback(const V2I_communication::read_vector::ConstPtr& msg){
         //sleep(1);
         cout << "past main_callback" << endl;
         cout << "size of vehicle is: " << vehicle.size() << endl;
         if (vehicle.size() == 0){
            sleep(1);
            cout << "past main_callback if statement" << endl;
            vehicle.resize(1);
            cout << "test 1" << endl;
            vehicle[0].set_vehicle_id(msg->data[1]);
            vehicle[0].set_current_location(msg->data[2]);
            vehicle[0].set_desired_location(msg->data[3]);
            vehicle[0].set_status(msg->data[4]);
            vehicle[0].set_priority(0);
            vehicle[0].open_all();
            vehicle[0].close_roads();
            checkingvector.resize(12);
            checkingvector = vehicle[0].get_road_status();
            for(t = 0; t < 12; t++){
               cout << checkingvector[t] << ",";
            }
            cout << endl;
            checkingvector.clear();
            cout << "test 2" << endl;
            to_vehicles.resize(5);
            to_vehicles[0] = 0;
            to_vehicles[1] = vehicle[0].get_vehicle_id();
            to_vehicles[2] = vehicle[0].get_current_location();
            to_vehicles[3] = vehicle[0].get_desired_location();
            to_vehicles[4] = 1;
            cout << "test 3" << endl;
            publish_to_vehicles();  //publishes string through xbee to all vehicles

         }

         else {
//check all vehicles on road and priority for matching closed roads. if match, vehicle must be
added to priority list. if no match, vehicle can go.
            sleep(1);
            size = vehicle.size();
            vehicle.resize(size + 1);
            vehicle[size].set_vehicle_id(msg->data[1]);
            vehicle[size].set_current_location(msg->data[2]);
            vehicle[size].set_desired_location(msg->data[3]);
            vehicle[size].set_status(msg->data[4]);

            vehicle[size].open_all();
            cout<< "test 4" << endl;
            vehicle[size].close_roads();
            cout<< "test 5" << endl;
            go = true;
            checkingvector.resize(12);
            establishedvector.resize(12);
            checkingvector = vehicle[size].get_road_status();
            for(r = 0; r < 12; r++){
               cout << checkingvector[r] << ",";
            }
            cout << endl;
            for(j = 0; j < size; j++){
               establishedvector = vehicle[j].get_road_status();
               for(s = 0; s < 12; s++){
                  cout << establishedvector[s] << ",";
               }
               cout << endl;
               for(k = 0; k < 12; k++){
                  if( (checkingvector[k] == 1) && (establishedvector[k] == 1) ){
                     go = false;

                  }
               }
               establishedvector.clear();
            }
            checkingvector.clear();
            cout<< "test 6" << endl;
/*
if go = true, it can be sent out without affecting any robots on the road or any in the
priority queue
if go = false, it has to be put at the end of the priority list
*/

            if(go == false){
               newpriority = get_highest_priority();
               vehicle[size].set_priority(newpriority);
               newpriority = 0;
               cout<< "test 7" << endl;
               cout << "The vehicle number : " << size << " : has a priority value of: " <<
vehicle[size].get_priority() << endl;

            }
            if(go == true){
            vehicle[size].set_priority(0);
            to_vehicles.clear();
               to_vehicles.resize(5);
               to_vehicles[0] = 0;
               to_vehicles[1] = vehicle[size].get_vehicle_id();
               to_vehicles[2] = vehicle[size].get_current_location();
               to_vehicles[3] = vehicle[size].get_desired_location();
               to_vehicles[4] = 1;
               << "-----------to_vehicles[] values-----<< endl;
               for(x = 0; x < 5; x++){
                  cout << to_vehicles[x] << ",";
               }
               cout << endl;
               cout << "-----------------------------------------------" << endl;

               publish_to_vehicles();  //publishes string through xbee to all vehicles
               //cout<< "test 8" << endl;
            }

            //publish_to_priority();
            //assign_priority_callback();
```

```
         }
         //cout<< "test 9" << endl;
      }
/***************/


/********************
DONE_VEHICLE_CALLBACK
********************/

      void done_vehicle_callback(const V2I_communication::read_vector::ConstPtr& donemsg){
         sleep(1);
         object_id = donemsg->data[1];//changed donemsg.data[1]

         for(l = 0; l < size; l++){
         test_id = vehicle[l].get_vehicle_id();
//Here we are matching up the ROS's vehicle id's with the id from the done msg.
            if(test_id == object_id){
            object_num = l;
            }
         }
         cout << "object_num is: " << object_num << endl;
         vehicle[object_num].open_all();


         size = vehicle.size();
         car_in_waiting = false;
         for(w =0; w < size; w++){
            if(vehicle[w].get_priority() > 0){
               car_in_waiting = true;
            }
         }
         if(car_in_waiting == true){
            loop = true;
            while(loop == true){
               for(u =0; u < size; u++){
                  checkingvector.clear();
                  checkingvector = vehicle[u].get_road_status();
                  for(v = 0; v < 12; v++){
                     cout << checkingvector[v] << ",";
                  }
                  cout << endl;
               }



               go = true;
               cout<< "test 10" << endl;
                                 cout<< "in done vehicle callback function" << endl;

               //need to find object with priority number 1;
               for(m = 0; m < size; m++){
                  if(vehicle[m].get_priority() == 1){
                     priority_obj_num = m;
                  }
               }

//Here we are getting the road status for the next priority and comparing them to all the
vehicles
               checkingvector = vehicle[priority_obj_num].get_road_status();
               cout << "priority 1 vehicle has a checking vector of: " << endl;
```

```
//<< checkingvector[0] << endl;
               for(p = 0; p <12; p++){
                  cout << checkingvector[p] << ",";
               }
               cout << endl;

//for (p = checkingvector.begin(); p != checkingvector.end(); ++p)//
//                                    std::cout << *p << ' ';//

               cout<< "test 11" << endl;
               cout<< "priority object number is " << priority_obj_num << endl;
               for(n = 0; n < size; n++){
                  cout << "Currently checking vehicle[" << n <<"] for closed road matching" <<
endl;
                  if(n != priority_obj_num){
                     cout << "not referring to itself... check if on road..." << endl;
                     if(vehicle[n].get_priority() == 0){
                        cout << "Checking vehicle[" << n << "]'s road closures because it is
on road..." << endl;
                        establishedvector = vehicle[n].get_road_status();
                        //cout << "established vector is: " << establishedvector << endl;
                        cout << "established vector of: " << endl; //<< checkingvector[0] <<
endl;

                        for(q = 0; q < 12; q++){
                           cout << establishedvector[q] << ",";
                        }
                        cout << endl;

                        for(k = 0; k < 12; k++){
                           if( (checkingvector[k] == 1) && (establishedvector[k] == 1) ){
                              go = false;
                              cout << "priority vehicle has a match and can not go yet...go is
set to false..." << endl;
                           }
                        }
                     }
                  }
                  else{
                     cout << "priority_obj_num is matched... vehicle is referring to itself...
do nothing" << endl;}
               establishedvector.clear();
               }
               checkingvector.clear();
               cout<< "test 12" << endl;
               cout << "status of go is: " << go << endl;
               //Here the state of 'go' was either changed to false or stayed true,
               //If true, we send this vehicle out, close its roads, and change all priority
values -1, and recheck the priority list.
               //if false, close while loop, and do nothing.


               if(go == true){
                  cout<< "test 12.1" << endl;
                  //cin.ignore();
                  vehicle[priority_obj_num].set_priority(0);
                  //to_vehicles.clear();
                  to_vehicles[0] = 0;
                  to_vehicles[1] = vehicle[priority_obj_num].get_vehicle_id();
                  to_vehicles[2] = vehicle[priority_obj_num].get_current_location();
                  to_vehicles[3] = vehicle[priority_obj_num].get_desired_location();
                  to_vehicles[4] = 1;
                  publish_to_vehicles();  //publishes string through xbee to all vehicles
                  last = false;
                  for(o = 0; o < size; o++){
                     if(vehicle[o].get_priority() > 0){
                        last = true;
                        placeholder = vehicle[o].get_priority();
                        placeholder = placeholder - 1;
                        vehicle[o].set_priority(placeholder);
                     }
                  }
                  if(last == false){
                     loop = false;
                  }
                  cout<< "test 13" << endl;
               }
               else{
                  loop = false;
                  cout<< "test 14" << endl;
               }
            }
         }
         cout<< "exited done vehicle callback" << endl;
      }
   /********************/


      int get_highest_priority(){
         priority = 0;
         for(int i = 0; i < size; i++){
            testpriority = vehicle[i].get_priority();
            if(testpriority >= priority){
               priority = testpriority + 1;
            }
         }

         return priority;
      }


      void publish_to_vehicles(){
      // Convert all but the last element to avoid a trailing ","
         cout << "TESTESTESTESTESTESTEST" << endl;
         cout << "size of to_vehicles[] is: " << to_vehicles.size() << endl;
         for(x = 0; x < 5; x++){
            cout << to_vehicles[x] << ",";
         }
         cout << endl;


         copy(to_vehicles.begin(), to_vehicles.end()-1, ostream_iterator<int>(output, ","));
         //Now add the last element with no delimiter
         output << to_vehicles.back();
         cout << "THIS IS PUBLISHING TO /WRITE: " << output.str() << endl;
         gomsg.data = output.str();

//         cout << "THIS IS PUBLISHING TO /WRITE: " << output.str() << endl;
         output.str( string() );
         output.clear();
         cout << "THIS IS IMMEDIATLY AFTER CLEAR: " << output.str() << endl;
         //publish to "write" topic
         robot_pub.publish(gomsg);
      }

      void roads_on(){
         p1_2 = 0;
         p1_3 = 0;
         p1_4 = 0;
         p2_1 = 0;
         p2_3 = 0;
         p2_4 = 0;
         p3_1 = 0;
         p3_2 = 0;
         p3_4 = 0;
         p4_1 = 0;
         p4_2 = 0;
         p4_3 = 0;
      }



   private:
      ros::NodeHandle nh;
      ros::Publisher robot_pub;
      ros::Subscriber new_vector_sub;
      ros::Subscriber done_vector_sub;
      vector<Robot> vehicle;
      int size;
      int p1_2, p1_3, p1_4, p2_1, p2_3, p2_4, p3_1, p3_2, p3_4, p4_1, p4_2, p4_3;
      vector<int> to_vehicles;
      ostringstream output;
      std_msgs::String gomsg;
      V2I_communication::read_vector prioritylist;
      int newpriority;
      int priority;
      int i;
      int j;
      int k;
      int testpriority;
      bool go;
      vector<int> checkingvector, establishedvector;

      /********/
      int object_num;
      //vector<int> donemsg;
      int object_id;
      int test_id;
      int l, m, n, o, p, q, r, s, t, u, v, w, x;
      int priority_obj_num;
      bool loop;
      bool last;
      bool car_in_waiting;
      int placeholder;
};
#endif


#pragma once
#ifndef ROBOT_H
#define ROBOT_H

using namespace std;
```

```
class Robot {

   public:
      Robot();

      ~Robot();

      //Mutator Functions:
      void set_vehicle_id(int value){
       vehicle_id = value;
      }
      void set_current_location(int value){
         current_location = value;
      }
      void set_desired_location(int value){
         desired_location = value;
      }

      void set_priority(int value){
         priority = value;
      }

      void set_status(int value){
         status = value;
      }

      void open_all(){
         p1_2 = 0;
         p1_3 = 0;
         p1_4 = 0;
         p2_1 = 0;
         p2_3 = 0;
         p2_4 = 0;
         p3_1 = 0;
         p3_2 = 0;
         p3_4 = 0;
         p4_1 = 0;
         p4_2 = 0;
         p4_3 = 0;
      }

      void close_1_2(){ //left turn
         p1_2 = 1;
         p2_3 = 1;
         p2_4 = 1;
         p3_1 = 1;
         p3_2 = 1;
         p3_4 = 1;
         p4_1 = 1;
         p4_2 = 1;
         p4_3 = 1;//not necessary but just to be safe depending on the size of the
intersection
      }

      void close_1_3(){ //straight through
         p1_3 = 1;
         p2_3 = 1;
         p2_4 = 1;
         p3_4 = 1;
         p4_1 = 1;
         p4_2 = 1;
```

```
            p4_3 = 1;
        }

        void close_1_4(){ //right turn
            p1_4 = 1;
            p2_3 = 1;  //not necessary but just to be safe depending on the size of the
intersection
            p2_4 = 1;
            p3_4 = 1;
    }

        void close_2_1(){ //right turn
            p2_1 = 1;
            p3_1 = 1;
            p3_4 = 1;  //not necessary but just to be safe depending on the size of the
intersection
            p4_1 = 1;
        }

        void close_2_3(){ //left turn
            p1_2 = 1;
            p1_3 = 1;
            p1_4 = 1;  //not necessary but just to be safe depending on the size of the
intersection
            p2_3 = 1;
            p3_1 = 1;
            p3_4 = 1;
            p4_1 = 1;
            p4_2 = 1;
            p4_3 = 1;
        }

        void close_2_4(){ //straight through
            p1_2 = 1;
            p1_3 = 1;
            p1_4 = 1;
            p2_4 = 1;
            p3_1 = 1;
            p3_4 = 1;
            p4_1 = 1;
        }

        void close_3_1(){ //straight through
            p1_2 = 1;
            p2_1 = 1;
            p2_3 = 1;
            p2_4 = 1;
            p3_1 = 1;
            p4_1 = 1;
            p4_2 = 1;
        }

        void close_3_2(){ //right turn
            p1_2 = 1;
            p3_2 = 1;
            p4_1 = 1;  //not necessary but just to be safe depending on the size of the
intersection
            p4_2 = 1;
        }
        void close_3_4(){ //left turn
            p1_2 = 1;
```

```
         p1_3 = 1;
         p1_4 = 1;
         p2_1 = 1; //not necessary but just to be safe depending on the size of the
intersection
         p2_3 = 1;
         p2_4 = 1;
         p3_4 = 1;
         p4_1 = 1;
         p4_2 = 1;
      }

      void close_4_1(){ //left turn
         p1_2 = 1;
         p1_3 = 1;
         p2_1 = 1;
         p2_3 = 1;
         p2_4 = 1;
         p3_1 = 1;
         p3_2 = 1; //not necessary but just to be safe depending on the size of the
intersection
         p3_4 = 1;
         p4_1 = 1;
      }

      void close_4_2(){ //straight through
         p1_2 = 1;
         p1_3 = 1;
         p2_3 = 1;
         p3_1 = 1;
         p3_2 = 1;
         p3_4 = 1;
         p4_2 = 1;
      }

      void close_4_3(){ //right turn
         p1_2 = 1;  //not necessary but just to be safe depending on the size of the
intersection
         p1_3 = 1;
         p2_3 = 1;
         p4_3 = 1;
      }

      void close_roads(){
         int cp = current_location;
         int dp = desired_location;
         if(cp == 1 && dp == 2){
            close_1_2();
            cout << "close_1_2 called" << endl;
         }

         if(cp == 1 && dp == 3){
            close_1_3();
            cout << "close_1_3 called" << endl;
         }

         if(cp == 1 && dp == 4){
            close_1_4();
            cout << "close_1_4 called" << endl;
         }

         if(cp == 2 && dp == 1){
```

```
      close_2_1();
      cout << "close_2_1 called" << endl;
   }

   if(cp == 2 && dp == 3){
      close_2_3();
      cout << "close_2_3 called" << endl;
   }

   if(cp == 2 && dp == 4){
      close_2_4();
      cout << "close_2_4 called" << endl;
   }

   if(cp == 3 && dp == 1){
      close_3_1();
      cout << "close_3_1 called" << endl;
   }

   if(cp == 3 && dp == 2){
      close_3_2();
      cout << "close_3_2 called" << endl;
   }

   if(cp == 3 && dp == 4){
      close_3_4();
      cout << "close_3_4 called" << endl;
   }

   if(cp == 4 && dp == 1){
      close_4_1();
      cout << "close_4_1 called" << endl;
   }

   if(cp == 4 && dp == 2){
      close_4_2();
      cout << "close_4_2 called" << endl;
   }

   if(cp == 4 && dp == 3){
      close_4_3();
      cout << "close_4_3 called" << endl;
   }

}


//Accessor Functions:
int get_vehicle_id(){
   return vehicle_id;
}
int get_current_location(){
   return current_location;
}
int get_desired_location(){
   return desired_location;
}
int get_priority(){
   return priority;
}
vector<int> get_road_status(){
```

```
            road_status.resize(12);
            road_status[0] = p1_2;
            road_status[1] = p1_3;
            road_status[2] = p1_4;
            road_status[3] = p2_1;
            road_status[4] = p2_3;
            road_status[5] = p2_4;
            road_status[6] = p3_1;
            road_status[7] = p3_2;
            road_status[8] = p3_4;
            road_status[9] = p4_1;
            road_status[10] = p4_2;
            road_status[11] = p4_3;
            return road_status;
        }


   private:
        int vehicle_id;
        int current_location;
        int desired_location;
        int status;
        int priority;
        int p1_2, p1_3, p1_4, p2_1, p2_3, p2_4, p3_1, p3_2, p3_4, p4_1, p4_2, p4_3;
        int i;
        vector<int> road_status;


};
Robot::Robot(){
}

Robot::~Robot(){
}

#endif
```

APPENDIX C – Photographs of test conditions

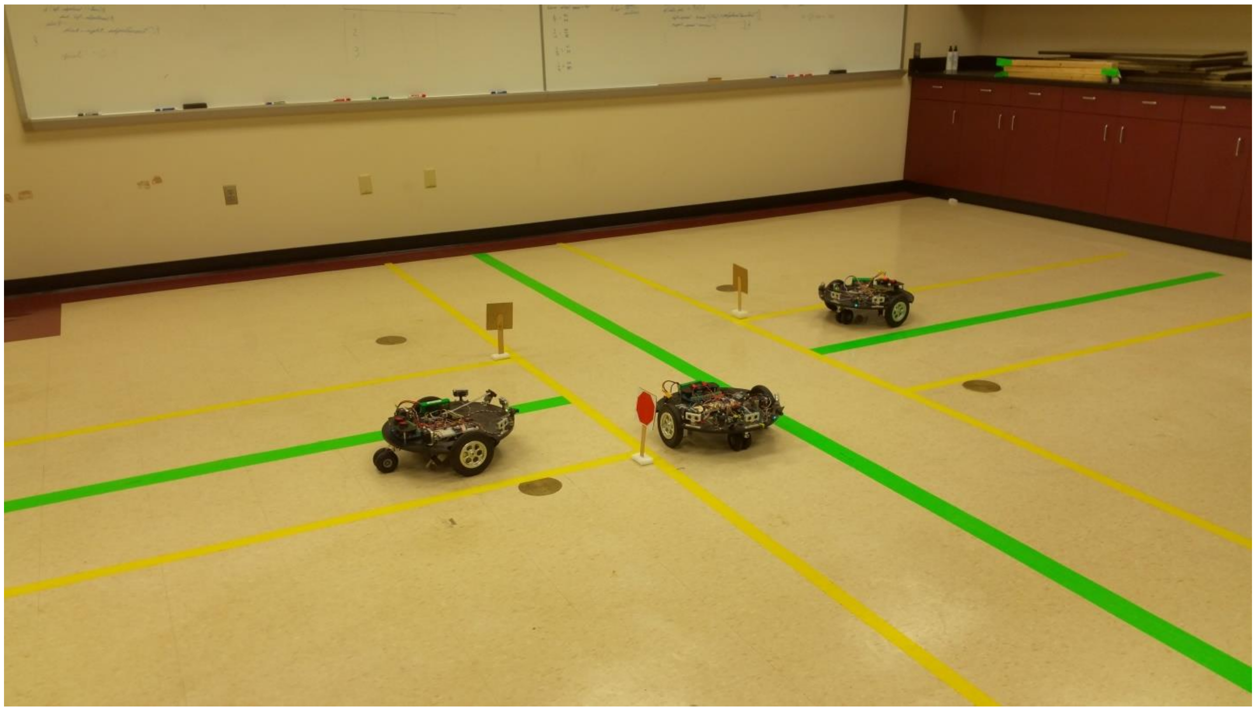
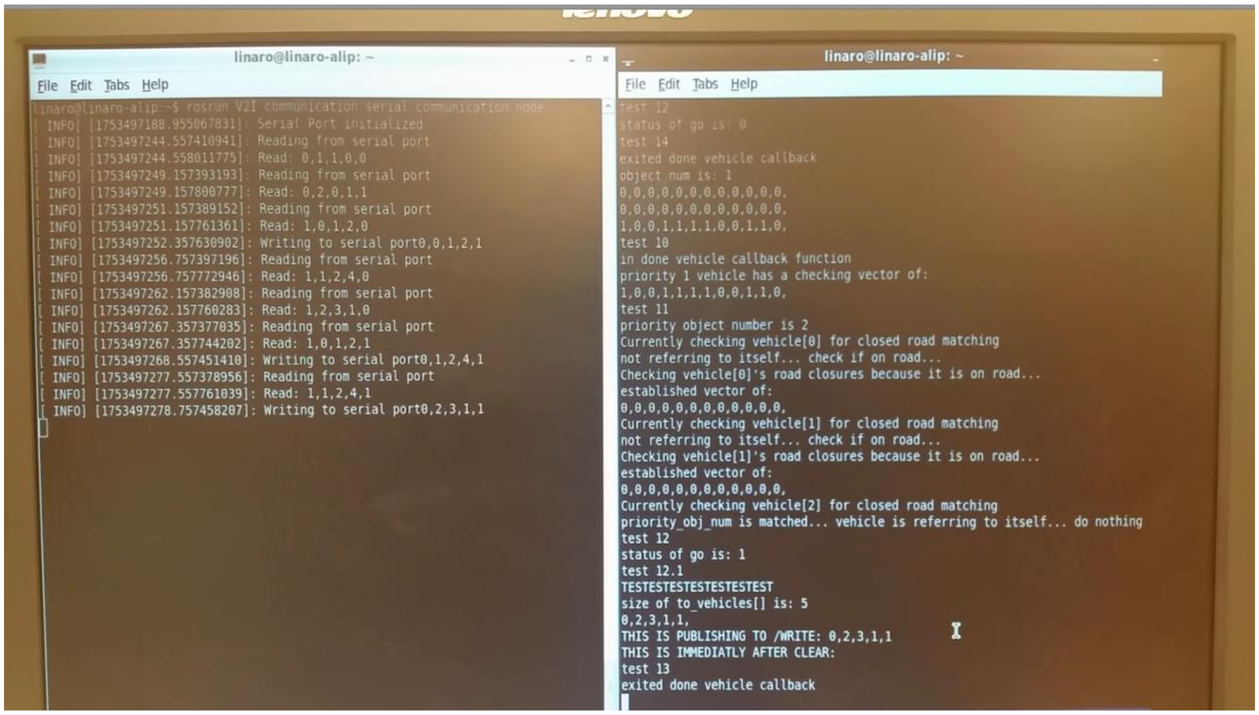


Figure A-1 Controlled scaled four way intersection and vehicles used in testing



```
linaro@linaro-alip:~$ rosrun V2I communication serial communication node
[ INFO] [1753497188.955067831]: Serial Port initialized
[ INFO] [1753497244.557410941]: Reading from serial port
[ INFO] [1753497244.558011775]: Read: 0,1,1,0,0
[ INFO] [1753497249.157393193]: Reading from serial port
[ INFO] [1753497249.157800777]: Read: 0,2,0,1,1
[ INFO] [1753497251.157389152]: Reading from serial port
[ INFO] [1753497251.157761361]: Read: 1,0,1,2,0
[ INFO] [1753497252.357630902]: Writing to serial port0,0,1,2,1
[ INFO] [1753497256.757397196]: Reading from serial port
[ INFO] [1753497256.757772946]: Read: 1,1,2,4,0
[ INFO] [1753497262.157382908]: Reading from serial port
[ INFO] [1753497262.157760283]: Read: 1,2,3,1,0
[ INFO] [1753497267.357377035]: Reading from serial port
[ INFO] [1753497267.357744202]: Read: 1,0,1,2,1
[ INFO] [1753497268.557451410]: Writing to serial port0,1,2,4,1
[ INFO] [1753497277.557378956]: Reading from serial port
[ INFO] [1753497277.557761039]: Read: 1,1,2,4,1
[ INFO] [1753497278.757458207]: Writing to serial port0,2,3,1,1
```

```
test 12
status of go is: 0
test 14
exited done vehicle callback
object num is: 1
0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,
1,0,0,1,1,1,1,0,0,1,1,0,
test 10
in done vehicle callback function
priority 1 vehicle has a checking vector of:
1,0,0,1,1,1,1,0,0,1,1,0,
test 11
priority object number is 2
Currently checking vehicle[0] for closed road matching
not referring to itself... check if on road...
Checking vehicle[0]'s road closures because it is on road...
established vector of:
0,0,0,0,0,0,0,0,0,0,0,0,
Currently checking vehicle[1] for closed road matching
not referring to itself... check if on road...
Checking vehicle[1]'s road closures because it is on road...
established vector of:
0,0,0,0,0,0,0,0,0,0,0,0,
Currently checking vehicle[2] for closed road matching
priority_obj_num is matched... vehicle is referring to itself... do nothing
test 12
status of go is: 1
test 12.1
TESTESTESTESTESTEST
size of to_vehicles[] is: 5
0,2,3,1,1,
THIS IS PUBLISHING TO /WRITE: 0,2,3,1,1
THIS IS IMMEDIATLY AFTER CLEAR:
test 13
exited done vehicle callback
```

Figure A-2 Base station response to request strings from navigating vehicles