NITC
NATIONAL INSTITUTE for TRANSPORTATION and COMMUNITIES

Final Report 1180
September 2024

Connected Streetcar Project

Kristin Tufte
John MacArthur
Shruti Nuchhi
Larry Head
Niraj Vasant Altekar
Debashis Das

Portland State UNIVERSITY

Connected Streetcar Pilot Project

Final Report

NITC-RR-1182

by

Kristin Tufte (co-PI)
John MacArthur
Shruti Nuchhi
Portland State University

Larry Head (co-PI)
Niraj Vasant Altekar
Debashis Das
The University of Arizona

for

National Institute for Transportation and Communities (NITC)
P.O. Box 751
Portland, OR 97207

August 2024

Technical Report Documentation Page		
1. Report No. NITC-RR-1182	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Connected Streetcar Pilot Project		5. Report Date August 2024
		6. Performing Organization Code
7. Author(s) Kristin Tufte, Larry Head, John MacArthur, Shruti Nuchhi, Niraj Vasant Altekar. Debashis Das		8. Performing Organization Report No.
9. Performing Organization Name and Address National Institute for Transportation and Communities (NITC) P.O. Box 751 Portland, OR 97207		10. Work Unit No. (TRAIS)
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address U.S. Department of Transportation Office of the Assistant Secretary for Research and Technology 1200 New Jersey Avenue, SE, Washington, DC 20590		13. Type of Report and Period Covered
		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

This project looked to create a foundation for deploying connected vehicle infrastructure and facilitating connected vehicle research for the NITC UTC. Connected Vehicle (CV) technology and the associated data have the potential to help better understand vehicle, streetcar, and transit performance as well as improve safety for vehicles, pedestrians, and bicyclists. Specifically, this project piloted the Multi-Modal Intelligent Traffic Signal System (MMITSS), a Dynamic Mobility Application that provides traffic signal priority for multiple modes of travelers developed at the University of Arizona. Data from the MMITSS platform was integrated into the PORTAL data archive. The connected vehicle deployment began with one Portland Streetcar vehicle and included three intersections along the "Art Museum Corridor." This pilot project was a top-to-bottom implementation of a foundational platform for connected vehicle research in hopes to provide lessons learned of an applied application of CV technology in an urban transit environment. Unfortunately, the project didn't achieve the goal of achieving an operational system that could be analyzed, evaluated, and an expansion planned, but City of Portland staff did learn about CV technologies, implementation of advanced traffic management systems, and the challenges of emerging technologies.

17. Key Words Connected Vehicle, Public transportation, Transit, Multi-Modal, Intelligent Traffic Signal Systems		18. Distribution Statement No restrictions. Copies available from NITC: www.nitc-utc.net	
19. Security Classification (of this report) Unclassified	20. Security Classification (of this page) Unclassified	21. No. of Pages	22. Price

ACKNOWLEDGEMENTS
This project was funded by the National Institute for Transportation and Communities (NITC; grant number 1182), a U.S. DOT University Transportation Center.”

The project team would like to acknowledge the support from the City of Portland, Oregon, and Portland Streetcar. We appreciate the support from the Portland Bureau of Transportation team: Alison Tanaka, Willie Rotich, Anne Hill, Adam Moore, Michael Kerr, Kirk McEwen, and Eric Hesse; and from Portland Streetcar: Kathryn Levine and Erick Moe.

RECOMMENDED CITATION

Tufte, Kristin, Larry Head, John MacArthur, Shruti Nuchhi, Niraj Vasant Altekar, and Debashis Das. *Connected Streetcar Pilot Project*. NITC-RR-1182. Portland, OR: Transportation Research and Education Center (TREC), 2024.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

1.0 EXECUTIVE SUMMARY

This project looked to create a foundation for deploying connected vehicle infrastructure and facilitating connected vehicle research for the NITC UTC. Connected Vehicle (CV) technology and the associated data have the potential to help better understand vehicle, streetcar, and transit performance as well as improve safety for vehicles, pedestrians, and bicyclists. Specifically, this project piloted the Multi-Modal Intelligent Traffic Signal System (MMITSS), a Dynamic Mobility Application that provides traffic signal priority for multiple modes of travelers developed at the University of Arizona. Data from the MMITSS platform was integrated into the PORTAL data archive. The connected vehicle deployment began with one Portland Streetcar vehicle and included three intersections along the “Art Museum Corridor.” In parallel to the CV technology deployment, the team investigated user needs, potential usages of the data, and designed visual analytics for the CV data aimed at evaluating and improving streetcar performance. The long-term vision of this project includes analyzing the data to determine which modes of travel should be given a higher level of priority. For example, during commute times, buses, light rail, and bicycles may be given higher priority than passenger vehicles. Additional research projects based on this CV platform could include the addition of smartphone capabilities for pedestrians and bicycles and the capture of high-fidelity data from connected travelers (pedestrians, bicycles, passenger vehicles, transit, trucks, and emergency vehicles) to study the ability to impact traffic flow using priority policy-based control and other CV applications. This pilot project was a top-to-bottom implementation of a foundational platform for connected vehicle research in hopes to provide lessons learned of an applied application of CV technology in an urban transit environment.

Unfortunately, the project didn’t achieve the goal of achieving an operational system that could be analyzed, evaluated, and an expansion planned, but City of Portland staff did learn about CV technologies, implementation of advanced traffic management systems, and the challenges of emerging technologies.

The project faced several significant technical challenges. One significant challenge was that the Federal Communications Commission (FCC) didn’t issue a regulation in 2016 that would require all new passenger vehicles manufactured and sold in the U.S. after 2021 to include DSRC OBUs. Since the future was uncertain, the City of Portland was reluctant to invest in technology that could be obsolete in the near future. The University of Arizona had spare RSUs and OBUs that they agreed to loan to the project. This allowed the City of Portland to gain connected vehicle systems experience and for the project team to pursue the goals of data collection and analysis.

Another significant challenge was realized once the OBU was installed on the streetcar. The team observed the behavior of the system when the connected streetcar approached the connected intersections and the behavior wasn’t as expected. The issues in vehicle positioning are likely related to the urban characteristics of downtown

Portland. The tall buildings, trees, and lack of clear sky view contribute to the poor GPS performance. A possible solution would be to enable and use the SAE J2735 RTCM message (RTCM - Radio Technical Commission for Maritime Services that provides GPS differential corrections). However, this would require each RSU to have an internet connection that would allow two-way data exchange outside of the closed City of Portland IT network. This implementation was not possible based on the existing city IT policies. Unfortunately, with the poor streetcar positioning, it was not possible to proceed with the evaluation of operations and it was decided to end the pilot project. The City of Portland is partnering with TriMet on their Next Generation Transit Signal Priority (Next-Gen TSP) project, which is instrumenting buses outside of the downtown core and testing CV infrastructure.

2.0 INTRODUCTION

The City of Portland is the owner and operator of Oregon’s third-largest transit system (by ridership), Portland Streetcar. Portland Streetcar provides critical transit service to Portland’s central city and adjoining neighborhoods where the state’s largest populations, employers, institutions, and hospitals are located. Portland Streetcar previously provided roughly 15,000 trips per day - more than all other modern streetcar systems in the country combined. People use the service to access jobs, education, recreation, and social services. Like most transit systems across the country, average weekday ridership dropped nearly 80% in spring 2020. Ridership has rebounded to 60% of prior averages. As of July 2023, the average weekly ridership was approximately 7,800 trips.

Portland Streetcar operates three lines around 16 miles of track in Portland’s central city (see Figure 1). The North/South (NS) Line operates on eight miles of track from SW Lowell & Bond in the South Waterfront to NW 23rd & Marshall in the Alphabet District connecting PSU, the Central Business District, and the Pearl District along the way. The A & B Loops operate two circular routes connecting the Pearl District, Lloyd District, Central Eastside Industrial District, Central Business District, and Portland State University in clockwise (A Loop) and counter-clockwise (B Loop) loops around the central city. Portland Streetcar operates and maintains a fleet of 17 trams. Portland Streetcar intersects with regional TriMet Max light rail, 80% of the region’s frequent-service bus routes, the Portland Aerial Tram, and C-TRAN’s interstate bus service (Vancouver, WA transit agency).

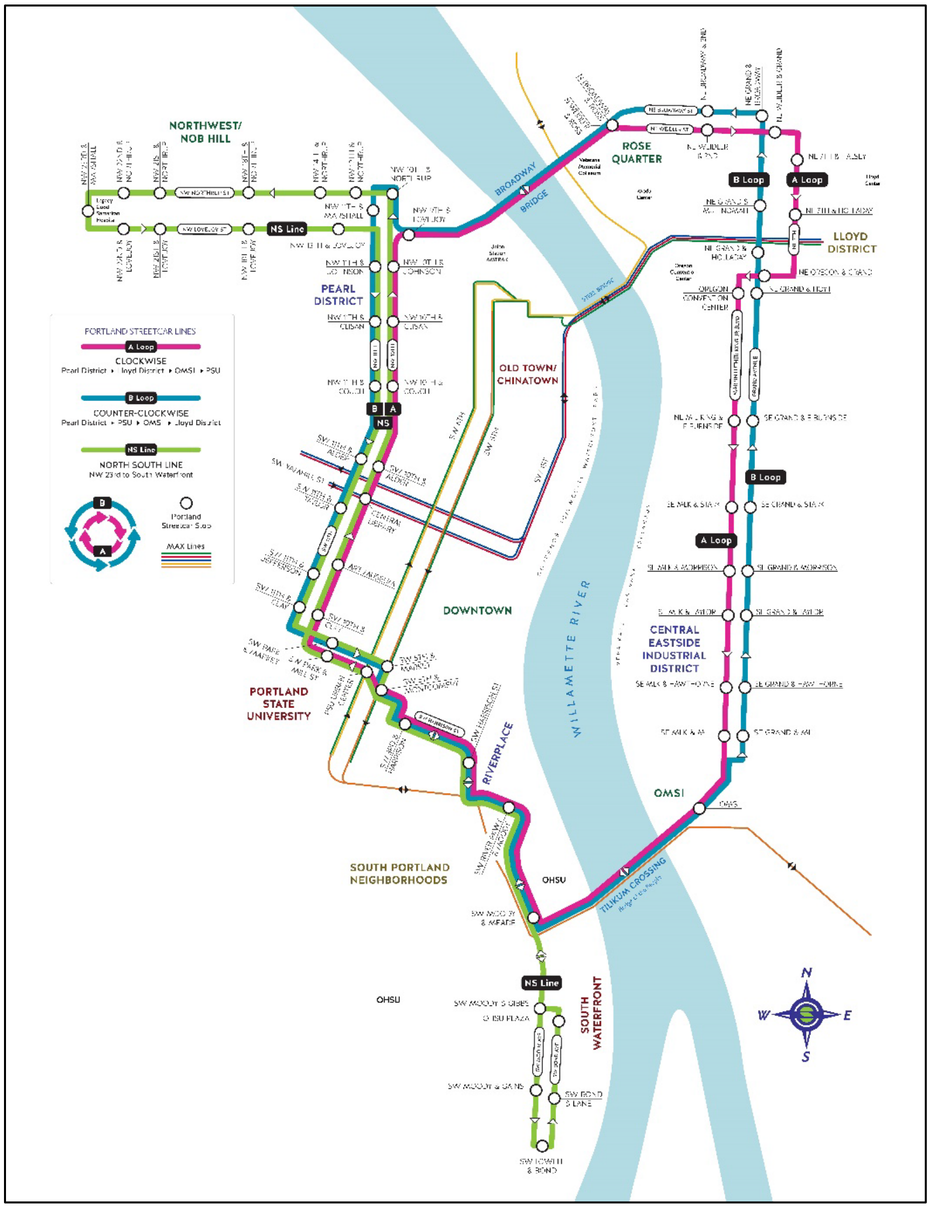


Figure 1: Portland Streetcar service map

The long-term vision and promise of CV technology is to leverage communication between vehicles and infrastructure to improve the safety and efficiency of our transportation network. CV data has the potential to help us better understand vehicle, streetcar, and transit performance as well as improve the safety for vehicles, pedestrians, and bicyclists.

This project looked to deploy the Multi-Modal Intelligent Traffic Signal System (MMITSS), a dynamic mobility application, on the Portland Streetcar and at four intersections along the “Art Museum Corridor” in downtown Portland. Through this deployment, the project created a foundation (applied experience and understanding) for deploying connected vehicle infrastructure in the city of Portland.

To achieve its goals, the project deployed the MMITSS on the Portland Streetcar in the summer of 2021. Developed by Dr. Larry Head of the University of Arizona, MMITSS is a dynamic mobility application that provides traffic signal priority for multiple modes of travelers.

Figure 2: The Portland Streetcar near the study corridor

Following the deployment, the research team established the data transfer storage mechanisms of CV data into PORTAL, the multimodal transportation data archive for the Portland metropolitan area. In addition, the project team tried to evaluate the potential effectiveness of the MMITSS system for improving streetcar operations and safety given data characteristics. For example, the fidelity of the data (speed, location, acceleration, system status, etc.) from the MMITSS is much higher than data available from currently used detectors and in-vehicle data.

The City of Portland used this project to evaluate the benefits and drawbacks of implementing CV infrastructure. For example, the Portland Streetcar, one of the project partners, hoped to be able to use the data to assess if CV deployment could meet and exceed established performance and safety goals. If so, a more extensive deployment of this technology could help improve streetcar system performance.

Using MMITSS gave the City of Portland the opportunity to understand and use dedicated short-range communications (DSRC). While MMITSS is currently implemented with DSRC, the MMITSS concept is compatible with other wireless communication technologies such as 5G and LTE-Direct. However, DSRC allows the agency (the city in this case) to control its use, access, deployment, and operations.

The goal of this project was to create a foundation for deploying CV infrastructure and facilitate CV research among our city and university partners. The CV deployment will begin with transit (bus and/or light rail) in Portland using the MMITSS. The project developed a management policy to allow the operating agency to determine which modes should be given a higher level of priority. Implications on practice will include development of guidance for the deployment of advanced traffic signal control in a connected and autonomous vehicle environment.

In parallel to the technology deployment, this project investigated user needs and usage of the data. Visual analytics were developed using the combined data sets and implemented select analytics in PORTAL, the official transportation data archive for the Portland metropolitan region. The specific goal was to create results that could be transferred to other cities.

2.1 CONOPS - MMITSS

The Multi-Modal Intelligent Traffic Signal System (MMITSS)[1] project is part of the Connected Vehicle Pooled Fund Study (CV PFS) entitled "Program to Support the Development and Deployment of Connected Vehicle System Applications." The CV PFS was developed by a group of state and local transportation agencies and the Federal Highway Administration (FHWA). The Virginia Department of Transportation (VDOT) serves as the lead agency and is assisted by the University of Virginia's Center for Transportation Studies, which serves as the technical and administrative lead for the PFS.

The USDOT has identified six mobility application bundles under the Dynamic Mobility Applications (DMA) program for the connected vehicle environment where high-fidelity data from vehicles, infrastructure, pedestrians, etc. can be shared through wireless communications. Each bundle contains a set of related applications that are focused on similar outcomes. Since a major focus of the CV PFS members – who are the actual owners and operators of transportation infrastructure – lies in traffic signal-related applications, the CV PFS team led the project entitled "Multi-Modal Intelligent Traffic Signal System" in cooperation with US DOT's Dynamic Mobility Applications Program (USDOT - Dynamic Mobility Applications). As one of the six DMA application bundles, MMITSS includes five applications: Intelligent Traffic Signal Control (I-SIG), Transit Signal Priority (TSP), Mobile Accessible Pedestrian Signal System (PED-SIG), Freight Signal Priority (FSP), and Emergency Vehicle Preemption (PREEMPT). The MMITSS prototype was developed based on traffic controllers using the National Transportation Communications for Intelligent Transportation System Protocol (NTCIP) communications protocol and new algorithms for providing priority control (emergency vehicle, transit, and truck priority) and intelligent signal control (e.g., adaptive control using connected vehicle data).

Using high-fidelity data collected from vehicles through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) wireless communications as well as pedestrian and non-motorized travelers, the Intelligent Traffic Signal System (ISIG) proposed application seeks to control signals and maximize flows in real time. The ISIG application also plays the role of an overarching system optimization application, accommodating transit or freight signal priority, preemption, and pedestrian movements to maximize overall

[1] https://github.com/mmitss

network performance. The TSP application allows transit agencies to manage bus service by adding the capability to grant bus priority based on a number of factors. The proposed application provides the ability for transit vehicles to communicate passenger count data, service type, scheduled and actual arrival time, and heading information to roadside equipment via an on-board device.

The long-term vision and promise of CV technology is to use communications between vehicles and between vehicles and infrastructure to improve the safety and efficiency of our transportation network. CV technology and the associated data has the potential to help us better understand vehicle, streetcar, and transit performance as well as improve safety for vehicles, pedestrians and bicyclists.

2.2 PROJECT OUTCOMES

The objective of this project was to create a *foundation for deploying connected vehicle infrastructure and facilitating connected vehicle research* within the NITC UTC. This project will provide an initial deployment of CV infrastructure and CV data collection in Portland using the Multi-Modal Intelligent Traffic Signal System (MMITSS) and the PORTAL data archive (Portland, Oregon Regional Transportation Archive Listing)[2]. This foundation will enable connected vehicle research by NITC researchers and regional agency staff, including City of Portland staff, and in the longer term national researchers.

This project is intended to lay a foundation for CV deployment and research in Portland and nationally. Planned project outcomes include:

- CV infrastructure deployment: Pilot deployment of the MMITSS CV infrastructure on the Portland Streetcar and at signalized intersections and evaluate the feasibility of expansion.
- CV data collection and analytics: Collection and storage of CV data in the PORTAL data archive. Design and development of select visual analytics in PORTAL using CV data (e.g., assessing streetcar performance with and without CV/MMITSS technology).
- Deployment guidance: Lessons learned for the deployment of advanced traffic signal control in a CV environment for use by other researchers and public agencies.
- Management policy: The development of a management policy to allow the operating agency to determine which modes of travel should be given higher priority**.** For example, during commute times, buses, light rail, and bicycles may be given higher priority than passenger vehicles.
- Foundation for future work: The project will be designed to add future mobile data sources allowing travelers to communicate with traffic signals and other infrastructure using their smartphones and capture high-fidelity data from

[2] https://new.portal.its.pdx.edu/

connected travelers (pedestrians, bicycles, passenger vehicles, transit, trucks, and emergency vehicles).

MMITSS has the potential to include passenger cars, transit vehicles, freight, pedestrians, bicycles, and emergency vehicles. In its initial phase, the project focused on streetcar vehicles. Other potential phases of the project could look to integrate communication with and data collection from TriMet buses, bicycles and pedestrians.

Distinguishing it from other projects, this project is the first deployment of integrating CV and DSRC with an existing data archive (PORTAL). An important aspect to the project is the existing cross-agency collaborations, which provided for significant future opportunities. A key value of the project is also its ability to lay the foundation for future CV research by providing a platform for researchers to develop and execute CV projects.

Future work may include: additional deployments on light rail or buses within the Portland metropolitan region; more detailed assessment of and research into the benefits of CV technologies; identification of multimodal CV application opportunities, including the capture of high-fidelity data from pedestrians and bicyclists; research on the integration of bicycles and pedestrians into the CV environment, such as allowing individuals to receive information from signals about signal status and to make signal calls; and additional development of visual analytics and data processing paradigms for CV data.

Additional work may also include the use of CV data to manage traffic operations. For example, CV data can potentially be used to improve travel time reliability by prioritizing different modes at signals. This may include giving buses priority during rush hour, which will also reduce stops and accelerations of buses at signals and increase their fuel efficiency. Given the frequent capture of CV data (10 vehicle positions per second) it is possible to use new and innovative ways to examine transportation problems. Data may be used, for example, to create fine-grained and multi-perspective “visualizations” of vehicle movements through an intersection during different signal phases. Visualizations can then be used to evaluate signal failures (e.g., significant delays at a signal) and the data can be used to model potential solutions to the problem.

2.3 PILOT OVERVIEW

The initial pilot deployed the MMITSS (Multi-Modal Intelligent Traffic Signal Systems) CV system on the Portland Streetcar and City of Portland signalized intersections. The initial pilot deployed on one streetcar (not the entire fleet) and at three select intersections in the “Art Museum Corridor” of downtown Portland located between SW 10th/Alder & SW 10th/Clay. The Art Museum Corridor was selected for the initial deployment because all three streetcar line loops run on this corridor and the corridor contains three consecutive intersections that could be instrumented. Streetcar service along this corridor is shown in Figure 3. The initial plan was to deploy the system for nine months in 2019 and 2020. Due to operational, hardware, and other constraints, the units were only deployed in the summer of 2021.

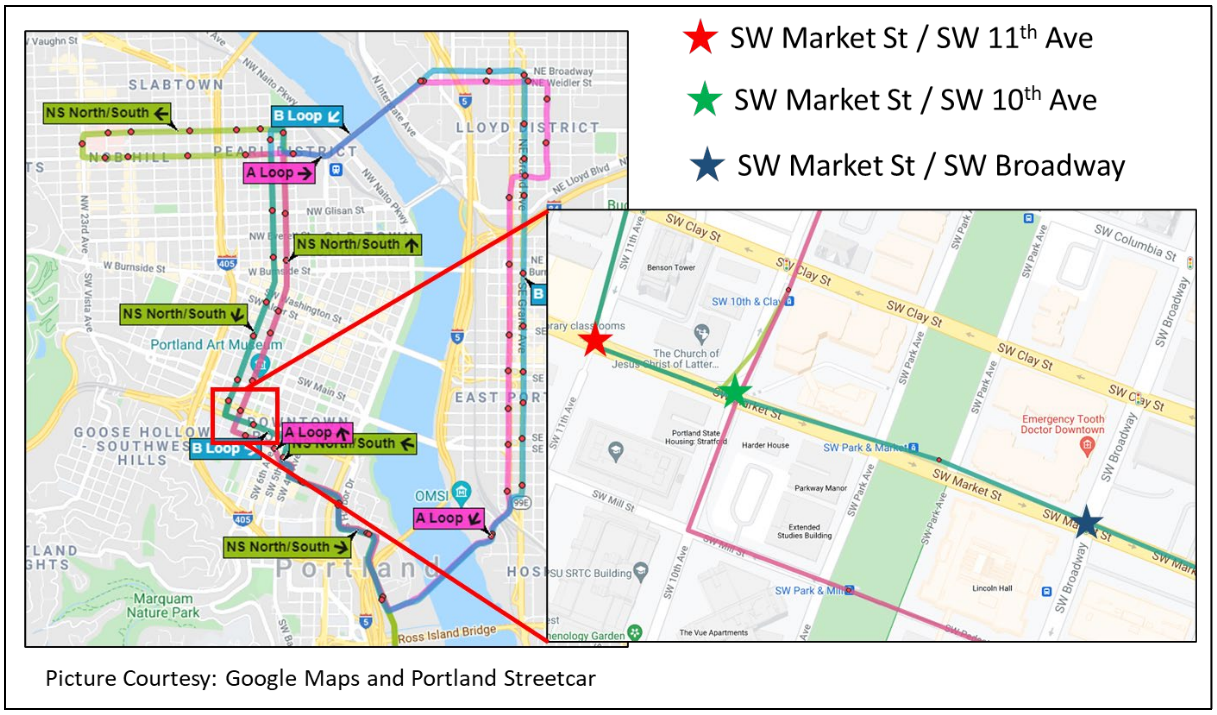


Figure 3: Streetcar Service – Art Museum Corridor with study corridor (red box) [Streetcar Map, Portland, Oregon]

Deployment of MMITSS requires deploying a dedicated short-range communications (DSRC) Road Side Unit (RSU) at each intersection (the RSU is effectively an advanced signal controller) and an On-Board Unit (OBU) on the streetcar (Figure 4). RSUs interfaced with the signal controllers and the city communications network along the study corridor. In addition, Bluetooth sensors were installed along the study corridor to collect vehicle travel time data. The pilot deployed these units, collected data from that deployment, integrated that data into the PORTAL archive, and performed preliminary analytics to evaluate the potential effectiveness of the MMITSS system for improving streetcar operations and safety.

MMITSS is currently based on DSRC technology, which will allow the City of Portland to install and begin to understand DSRC technology, its benefits, and drawbacks. However, while MMITSS is currently implemented with DSRC, the MMITSS concept is compatible with other wireless communications technologies such as 5G, LTE-Direct, etc. MMITSS should work with other technologies, but DSRC allows the agency (city in this case) to control the use, access, deployment, and operations.

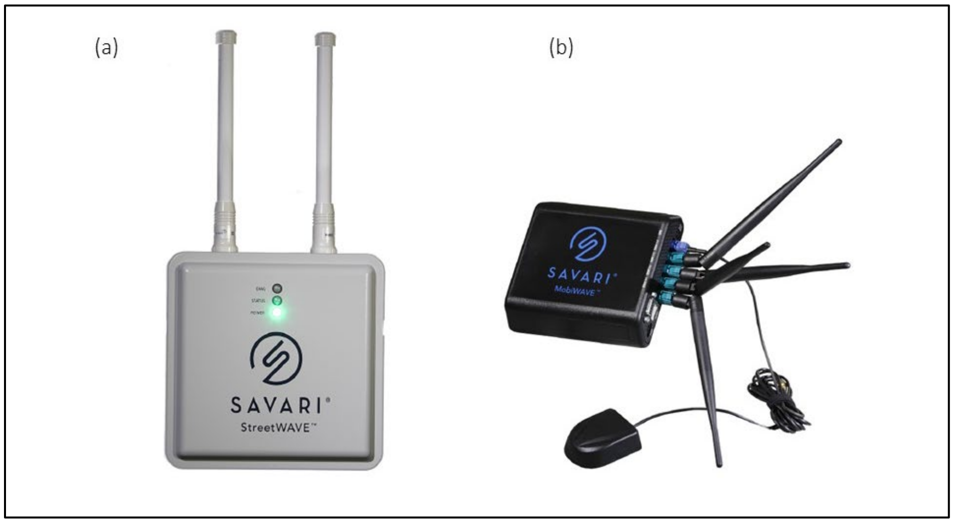


Figure 4: The Multi-Modal Intelligent Traffic Signal System (MMITSS) includes the (a) RSU: Savari SW-1000 Road-Side-Unit and (b) OBU: the MobiWAVE On-Board-Unit. See the specification pages for further details on CV hardware.

2.4 POTENTIAL BENEFITS TO PORTLAND STREETCAR

Portland Streetcar has established goals around system performance and safety that CV technology could improve. First, Portland Streetcar wants to maintain at least an 85% on-time performance measure. Currently, two of the loops just meet this measure while one loop has 75% on-time performance. On-time reliability is important to overall system performance but also to ridership. Another measure focuses on safety related to collisions. The goal is to have less than .65 collisions per 1,000 hours of service. In 2016, Portland Streetcar had .71 collisions per 1,000 hours of service. CV technologies and infrastructure could improve performance, reduce collisions, and provide prioritization to the mode so ridership could be increased. The outcome of this project looks to support the City of Portland in identifying the direction of future CV research, application opportunities, and deployment methods in the city.

Deployment of MMITSS requires deploying an RSU at the intersection and an OBU on the streetcar. This project will deploy those units, collect data from that deployment, integrate that data into the PORTAL archive, and perform preliminary analytics with the goal of evaluating the potential effectiveness of the MMITSS system for improving streetcar operations and safety. The fidelity of the data (speed, location, acceleration, system status, etc.) from the MMITSS is much higher than data available from currently used detectors leading to the expectation that this data may improve streetcar performance. The Portland Streetcar has established goals around system performance and safety that CV technologies could improve, including an 85% on-time performance goal and a goal of less than .65 collisions per 1,000 hours of service. The promise of CV technologies is to reduce collisions and improve on-time performance for the streetcar.

MMITSS has the potential to include passenger cars, transit vehicles, freight, pedestrians, bicycles, and emergency vehicles. In this initial phase, this project will focus on transit vehicles. The goal of this phase is to deploy the infrastructure and establish the data transfer and storage mechanisms. If time and budget allow, this project will integrate communication with and data collection from bicycles and pedestrians. This project can be distinguished from other projects by it being the first deployment of CV infrastructure in Portland, and by the integration of this project with an existing data archive (PORTAL) and with existing cross-agency collaborations in Portland which provide for significant future opportunities. Figure 5 shows how the CV infrastructure could be used in a multimodal signal priority control, but could be enhanced for other signal priority applications, such as freight signal priority (FSP) and emergency vehicle preemption.

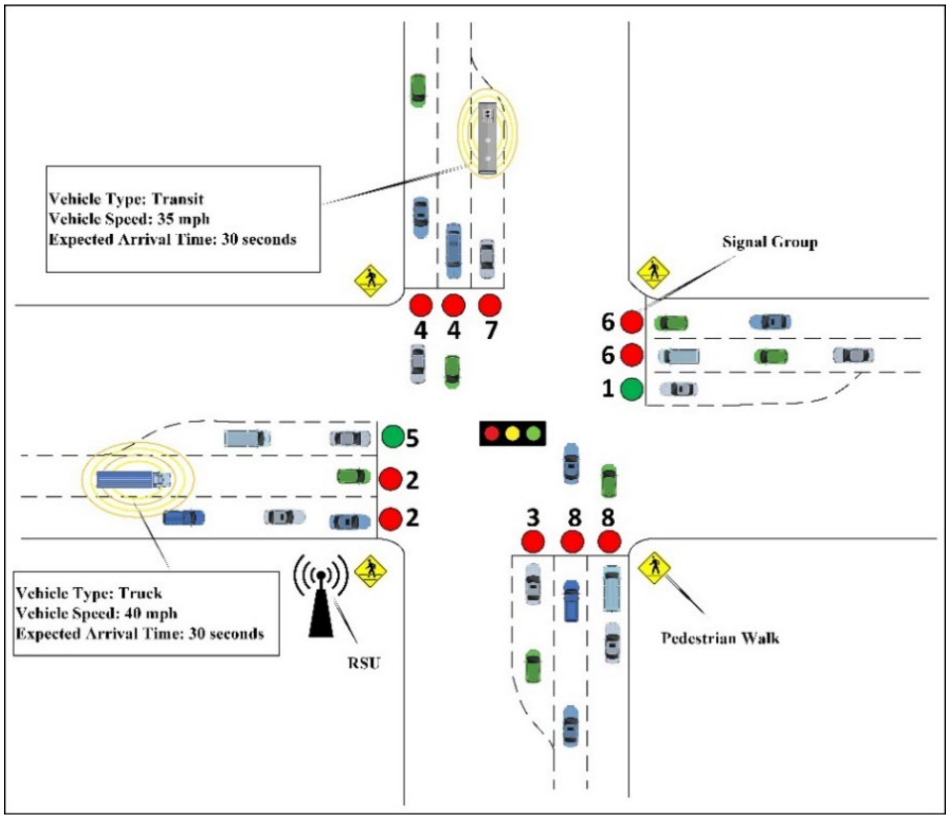


Figure 5: Multimodal Signal Priority Control

3.0 MMITSS ARCHITECTURE

The physical MMITSS architecture is shown in Figure 6 as a UML Deployment Diagram. In the Unified Modeling Language (UML), nodes are shown as 3D blocks and represent physical devices that have a processor (at least one), memory, and physical interfaces (e.g., Ethernet, RS-232, or wireless – 3G/4G, 5.9 GHz DSRC, or other such as CAN-bus). The basic architecture is applicable at each MMITSS-controlled intersection. In a system that consists of several intersections, each intersection would have an RSE Radio and traffic control equipment as shown in the deployment diagram. There would generally be only one instance of the Traffic Management System, but there could be multiple instances of the Fleet Management System (e.g., Transit Fleet, commercial truck fleet, emergency vehicles).

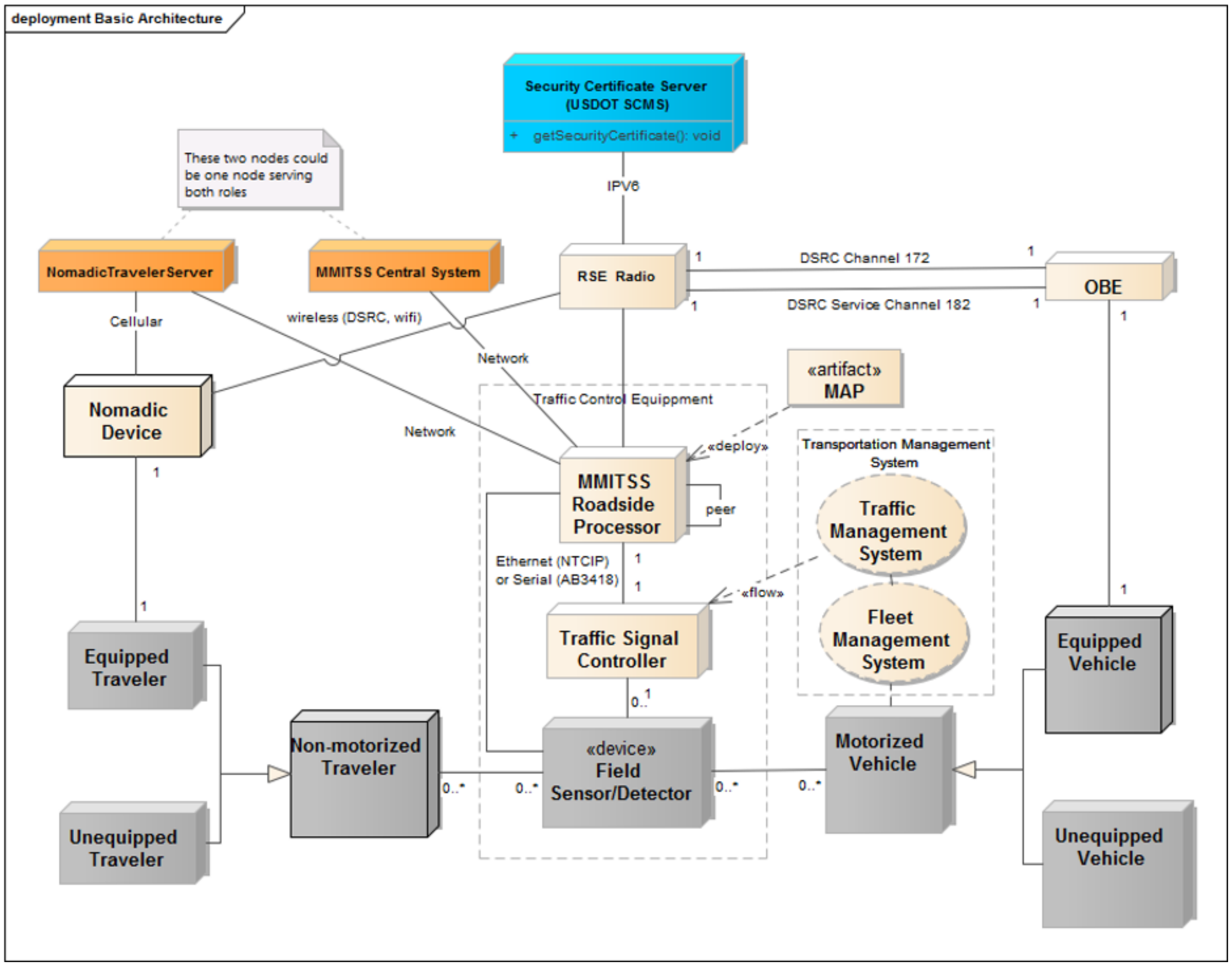

deployment Basic Architecture
Security Certificate Server (USDOT SCMS)
+ getSecurityCertificate(): void
IPV6
These two nodes could be one node serving both roles
NomadicTravelerServer
MMITSS Central System
RSE Radio
DSRC Channel 172
DSRC Service Channel 182
OBE
Cellular
wireless (DSRC, wifi)
Network
Network
Nomadic Device
«artifact» MAP
Traffic Control Equippment
«deploy»
Transportation Management System
MMITSS Roadside Processor
peer
Traffic Management System
Fleet Management System
Ethernet (NTCIP) or Serial (AB3418)
«flow»
Traffic Signal Controller
Equipped Traveler
Unequipped Traveler
Non-motorized Traveler
«device» Field Sensor/Detector
Motorized Vehicle
Equipped Vehicle
Unequipped Vehicle
1
0..1
0..*

Figure 6 MMITSS physical architecture

The nodes have been shaded such that the light-colored nodes are part of the connected vehicle system. The Traffic Management and Fleet Management systems are shown as UML collaborations and aren't part of the MMITSS system. The gray colored nodes represent the vehicles, travelers and infrastructure sensors (e.g., inductive loops or video detectors). The Security Certificate Server (e.g., USDOT Security Credential Management System – SCMS) is interfaced to the RSE using a backhaul network and is used to provide security certificates to trusted OBEs and a user revocation list to the RSE. The orange-colored nodes are the MMITSS Central System and Nomadic Traveler Server as described below. These two nodes may be realized as a single node for the test bed implementations.

In this view of the system, there are two types of travelers – motorized vehicles and non-motorized travelers. Motorized vehicles consist of passenger vehicles, trucks, transit vehicles, emergency vehicles, and motorcycles. This type of traveler includes any vehicle that must be licensed to operate on the public roadway. Non-motorized travelers include pedestrians, bicyclists, and other modes such as equestrian that are not required to be licensed to operate on the public roadway. These travelers are either unequipped or equipped, meaning that they have some type of OBE (on-board equipment) or nomadic device that is connected-vehicle (or MMITSS) aware and can operate as part of the traffic control system.

Motorized vehicles can be part of a fleet management system such as a transit management system, commercial freight management system, emergency vehicle dispatch system, or taxi dispatch, which is shown as a UML collaboration (oval in Figure 1), meaning that a collection of entities work together to perform the traffic management functions, but there may be many different systems involved in this collaboration.

The infrastructure-based traffic signal control equipment consists of the traffic signal controller, field sensors/detectors, an optional MMITSS Roadside Processor (MRP) and the RSE. The RSE Radio is the hardware device that is responsible for managing all of the 5.9 GHz DSRC communications between the vehicles and the infrastructure. Two channels are utilized for communications between the RSE and the vehicle OBE. Channel 172 is called the safety-of-life channel and is used by vehicles for broadcasting basic safety messages (BSM) and by the infrastructure (RSE) for broadcasting the map message (MAP) and the signal phase and timing message (SPaT). Channel 182 is used for exchanging the signal request messages (SRM) and signal status messages (SSM). The RSE hardware used for the MMITSS development and testing was supplied by Savari, Inc. and is called the StreetWave unit.

The OBE is a hardware device deployed on the vehicle. MMITSS was developed and tested using Savari MobilWave units (called aftermarket safety devices - ASD) for the OBE. Recent developments include adapting the RSU 4.1 application interface (API) with Cohda OBEs. These units are general purpose and provide a powerful and flexible platform for development and testing. The OBE devices exchange messages with the RSE based on the SAE J2735 (2016) standard using the Dedicated Short-Range Communications standards – IEEE 1609.

The MMITSS system was originally tested using Econolite ASC/3 (or Cobalt) traffic signal controllers. The Econolite ASC/3 and the Cobalt controllers are based on NEMA standards and support NTCIP over ethernet communications. MMITSS has been tested with McCain NTCIP controllers at the FHWA Saxton Transportation Lab, and has been tested using the Intelight MaxTime controller software. The MMITSS MRP is a Linux based general-purpose computer and is currently based on the Econolite Connected Vehicle Co-Processor module.

The MMITSS MRP is a general-purpose computer that can host the core intersection level infrastructure applications for MMITSS. The RSE contains (deploys) the MAP artifact, which is the digital description of the intersection geometry and associated traffic control definitions.

Both motorized and non-motorized travelers can be detected by the Field Sensor/Detector node at the intersections using a variety of detection technologies, including inductive loop detectors, video detection, microwave, radar, pedestrian push button, etc. The detection system at an intersection provides information to the traffic signal controller that stimulates the control algorithms. For example, a vehicle that triggers a detector will call a signal control phase for service or extension. A pedestrian may activate a pedestrian push button to request the traffic signal pedestrian interval associated with a crosswalk movement. The direct communications path from the field sensor to the MRP allows the communication of information from the sensor. This information might include vehicle count from presence detectors or possibly vehicle trajectories from radar-based sensors in the future.

3.1 OVERVIEW OF THE MMITSS PROTOTYPE

The MMITSS prototype was developed with the philosophy that connected vehicle data, both vehicle basic safety messages and priority request messages, would provide significant additional information that could be used to develop a new generation of traffic signal control algorithms. Goodall (2013) demonstrated that connected vehicle data could improve traffic signal control performance. Together, with recent advances in adaptive traffic signal control, such as OPAC (Gartner, 1983), RHODES (Mirchandani and Head, 2001), and ACS Lite (Gettman et al., 2007) and the results of the NCHRP 3-66 project (Urbanik et al, 2003), there has been significant advancement in the development of traffic control algorithms for both general vehicles (adaptive signal control) and for special classes of vehicle (priority control). In addition, it was recognized that the market penetration rate of connected vehicles would be low for several years, but priority-eligible vehicle fleets could be good candidates for early adoption projects, such as transit priority corridors, freight priority corridors, and emergency vehicle systems. If MMITSS could provide effective priority control with low market penetration rates for personal vehicles, then the entire traffic management system could benefit as the market penetration increases.

One aspect of this strategy was the recognition that actuated traffic signal control would continue to be the standard at most traffic signals, so MMITSS should be able to integrate innovative priority control with actuated signal control. Coordination is also a critical traffic control strategy and MMITSS should be able to integrate priority control with coordination and actuation. As such, coordination was considered to be a form of priority which could be addressed in a unified priority control framework.

An important concept adopted in the MMITSS prototype was the concept of multiple levels of importance for different modes in traffic control. An operating agency should have the ability to determine which mode of priority should be the most important and which should be the least important when serving multiple requests at one time. For example, in one corridor, coordination might be the most important consideration during the AM and PM peak commute times or during an incident on a parallel freeway. In the off-peak times, the priority might favor transit vehicles in an urban shopping/school area or might favor freight in a commercial warehouse/factory area. The ability to establish a "priority policy" would allow the operating agency to have a powerful tool for traffic management.

Finally, the recent development and agency adoption of the NTCIP communication standards has provided a favorable path for deployment of MMITSS. NTCIP-compliant traffic signal controllers can easily provide the current signal timing plan data to MMITSS and can interface with the MMITSS algorithms. The standards provide a well-defined and common interface that is now widely used across the United States. Most major traffic signal controller manufacturers support NTCIP, and NTCIP provides the interfaces necessary for working with the controllers.

In the following sections, an overview of the MMITSS system is presented. The goal in this section is to familiarize the reader with the design and algorithm concepts used in the MMITSS prototype. Details of the software implementation can be found in the MMITSS Detailed Design document. The following section is divided into two main subsections. The first presents the MMITSS architecture and describes the software components that provide the platform for the MMITSS traffic control algorithms. The second section provides an introduction to the adaptive signal control, the priority control, and the performance observer components.

3.2 ARCHITECTURE OF MMITSS

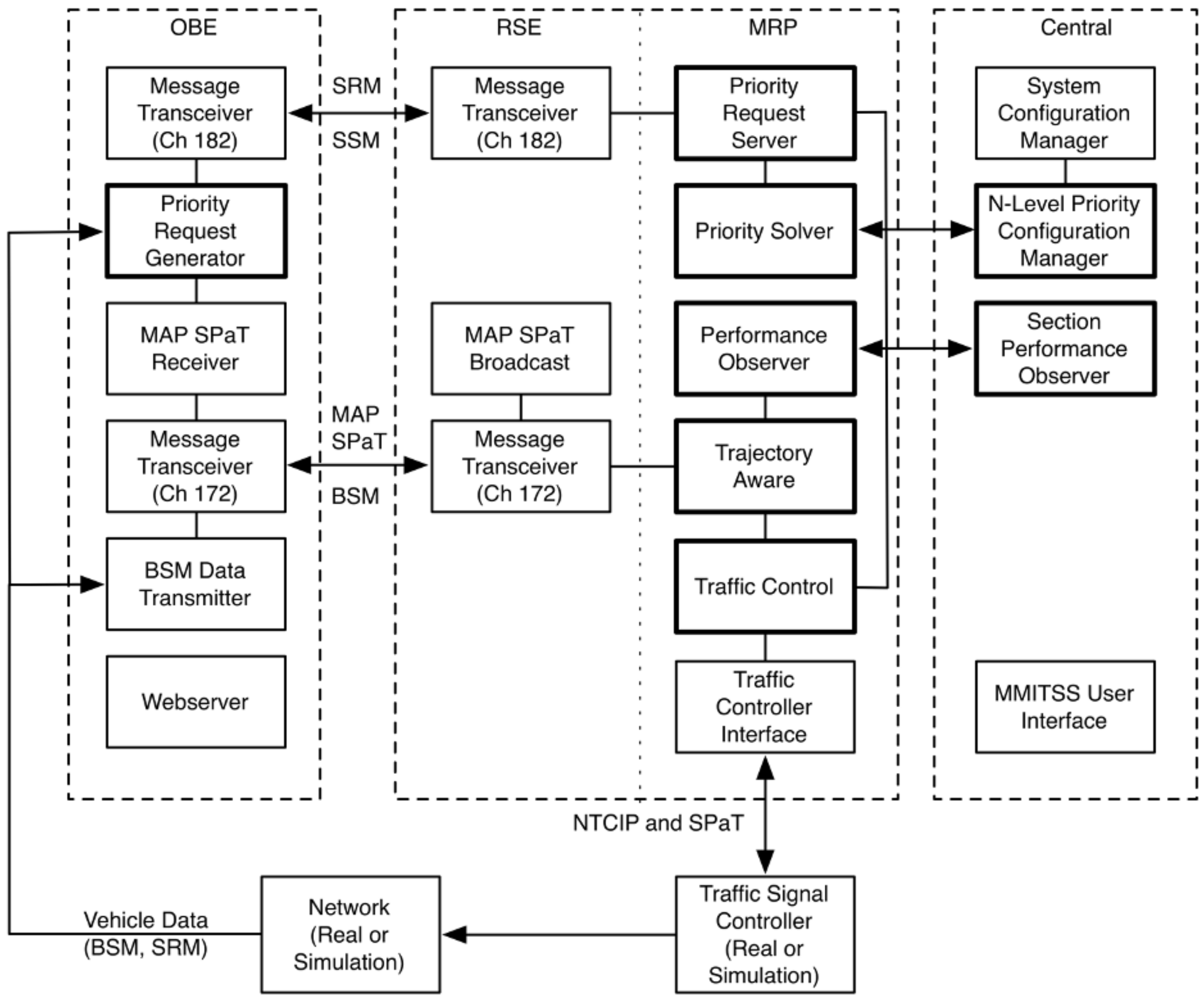

OBE
RSE
MRP
Central
Message Transceiver (Ch 182)
SRM
SSM
Message Transceiver (Ch 182)
Priority Request Server
System Configuration Manager
Priority Request Generator
Priority Solver
N-Level Priority Configuration Manager
MAP SPaT Receiver
MAP SPaT Broadcast
Performance Observer
Section Performance Observer
MAP
SPaT
BSM
Message Transceiver (Ch 172)
Message Transceiver (Ch 172)
Trajectory Aware
BSM Data Transmitter
Traffic Control
Webserver
Traffic Controller Interface
MMITSS User Interface
NTCIP and SPaT
Vehicle Data (BSM, SRM)
Network (Real or Simulation)
Traffic Signal Controller (Real or Simulation)

Figure 7: Architecture of MMITSS system

The MMITSS architecture is shown in Figure 7. The architecture was designed based on a component-based, or distributed, software design where the components use peer-to-peer communications (sockets) to share information. Figure 7 shows three key MMITSS architecture elements: the on-board equipment (OBE) and vehicle processor (for offloading processes from the OBE to support hardware agnostics), Roadside Processor and equipment (RSE and MRP), and the central system. Software components are deployed on each of the key architecture elements. There are two kinds of components: MMTISS traffic control algorithms (shown with thick black borders) and MMTISS interface and general connected vehicle components. The interface components allow the MMITSS system to acquire data and actuate controls, as well as to interface to the two DSRC communication channels that are used. The MMITSS traffic control components implement the new control algorithms developed to use the connected vehicle data.

3.2.1 Architecture Components

The architecture includes six software components that run on the OBE and vehicle processor, 10 software components that run on the MMITSS MRP, and four that run on the central system (Note, these four components are not supported under the current MMITSS 3.0 version at this time). The peer-to-peer socket interfaces allow the components to be installed on any processor in the subnetwork and configured to communicate with the desired peer component(s). For example, all the algorithm components (thick black lines in Figure 7) could be run on a separate processor, or some could be on the RSE and some on the MRP, or all on the RSE. Currently, all run on the MRP using the RSU 4.1 application interface to the RSE.

Connected Vehicles – Message Description

Basic Safety Message (BSM)
- Broadcasted by connected vehicles at 10Hz
- Contains vehicle's dynamic state (position and motion) information

Intersection Map Message (MAP)
- Broadcasted by connected intersections at 1Hz
- Contains digital description of intersection geometry and physical features

Signal Phase and Timing (SPaT)
- Broadcasted by connected intersections at 10Hz
- Contains intersection's signal state information

Signal Request Message (SRM)
- Generated when priority-eligible vehicle enters the geo-fenced area of the connected intersection
- Contains information about estimated arrival time, requested signal state, vehicle type, and other vehicle-state information
- Generally broadcasted by connected vehicles but can also be generated at connected intersections

Signal Status Message (SSM)
- Broadcasted by connected intersections as an acknowledgement of SRMs

All messages follow the SAE J2735 message structure standard

Applications, including adaptive signal control and dilemma zone protection, utilize only BSMs while coordination and signal priority utilize SPaT, MAP, SRM and SSM. Two DSRC channels, 172 and 182, are used to send/receive different types of messages. Channel 172 is the safety channel, which is used to transmit safety-related messages (e.g., BSM, SPaT and MAP). Channel 182 is one of the available service channels and was selected to transmit priority application-specific messages (e.g., SRM and SSM for priority control). Any of the other service channels could be used if the transmitting and receiving radios are configured for this use.

This architecture has been implemented in both simulation (VISSIM 2021 with the Econolite ASC/3 software-in-the-loop (SIL) signal controller) and in the field using NTCIP-compliant signal controllers. [Under the current Portland MMITSS/NITC project, the Intelight Maxtime 2.0 software in the loop controller was tested].

3.2.2 Message Transceiver (OBE/VSP and MRP)

The Message Transceiver application is configurable to utilize either of the two 5.9GHz radios. One is configured to use channel 172 and the other to use channel 182. Channel 172 is used by the OBE to broadcast basic safety messages (BSM) and to receive MAP and SPaT messages that are broadcast by the RSE (MAP SPaT Broadcast component). Channel 182 is used to broadcast and receive Signal Request Messages (SRM) and Signal Status Messages (SSM). The Message Transceiver interfaces to RSE using the FHWA RSU 4.1 API.

3.2.3 MAP SPaT Receiver (OBE)

The MAP SPaT Receiver is deployed on the OBE/vehicle processor and is responsible for receiving and unpacking the J2735 (2016) MAP and SPaT messages and making the data available to the Priority Request Generator. The MAP SPaT receiver can supply MAP and SPaT data to other components on the OBE if desired. The MAP data part of the configuration is developed using the USDOT MAP Tool (https://webapp.connectedvcs.com/isd/).

3.2.4 BSM Data Transmitter (OBE)

The BSM Data Transmitter is responsible for building the J2735 (2016) Basic Safety Message (BSM). For the MMITSS prototype, the BSM contains a temporary identification number (ID, the vehicle’s GPS position, speed (from GPS) and the vehicle type). Vehicle configuration data is stored in a json configuration file. The message is built 10 times per second and sent to the Message Transceiver for broadcast on Channel 172. Currently, the BSM Data Transmitter does not read vehicle data from the vehicle’s CAN bus and it does not do any position correction. The position accuracy was sufficient for field testing in the Arizona Connected Vehicle Test Bed where lane level positions were generally received. (Note: This was not true for Portland due to the trees and other foliage in the study area).

3.2.5 Traffic Controller Interface (RSE, MRP)

The Traffic Control Interface component is responsible for all communications with the traffic signal controller. There are three main communication interactions between the traffic signal controller and the Traffic Controller Interface component: 1) register and receive SPaT data, 2) query the controller for configuration data, and 3) command the controller to implement the desired traffic control schedule. All communications with the traffic controller (Econolite ASC/3 or Cobalt or MaxTime) are through the local ethernet network using standard NTCIP objects. The Econolite ASC/3 controller is configured with the IP address and port where it can stream SPaT messages. The SPaT messages are defined according to the Battelle (2012) SPaT specification that was developed for FHWA. SPaT data is received, unpacked and made available to the Traffic Control and Priority Request Server components 10 times per second.

The controller configuration data is queried when it is started and provides the Phase Timing Interval Data defined in Table 1. [Note: the frequency of this query can be changed to allow MMITSS to be aware of changes to the controller configuration by a traffic technician using the controller front panel or by a central traffic management system.] In addition, Detector Parameter configuration is read on start up, and Detector Data (Phase Calls) are read once per second. This data is made available to Traffic Control, Priority Request Server, and the Performance Observer components as input for their functions. All controller configuration and detector data are queried using NTCIP 1202 v2 objects (2005).

The commands from MMITSS to the traffic signal controller are implemented using NTCIP Phase Control Objects (Table 1). MMITSS utilizes the controller configuration together with the data from vehicle detectors, connected vehicle basic safety messages and signal request messages to determine an optimal signal timing schedule that consists of the phase sequence and durations. The Traffic Control Interface component monitors the controller status and sends the appropriate NTCIP Phase Control command to the controller to implement the signal timing schedule. The optimal signal timing schedule is updated by the MMITSS Traffic Control and Priority Request Server components as new information is available. As the schedule is updated, the Traffic Control Interface component issues the commands to the traffic signal controller.

Table 1: Traffic Controller Configuration Data

Data Object	**Parameters**	**NTCIP 1202 Protocol Objects**	**Usage**
PhaseTimingIntervals	PhaseNumber PhaseWalk PhasePedestrianClear PhaseMinimumGreen PhasePassage PhaseMaximum PhaseYellowChange PhaseRedClear	Phase Parameters	Input to MRP_TrafficControl & MRP_PRS_PriorityServer for phase allocation
DetectorParameters	DetectorID DetectorType DetectorCallPhase NTCIPOccupancy NTCIPVolume	Detector Parameters	Input to MRP_PerformanceObser for performance estimation MRP_TrafficControl for phase allocation
PhaseControl	PhaseOmit PhaseHold PhaseForceOff VehicleCall PedCall	Phase Control	Output from MRP_TrafficControlInterface to the traffic signal controller

3.2.6 Priority Request Generator (OBE)

The Priority Request Generator (PRG) is deployed on the OBE/vehicle processor. The PRG embodies the vehicle-based logic that determines when to send a priority request and the contents of the request message. In the MMITSS prototype it is assumed that any priority-eligible vehicle (e.g., emergency vehicles, transit vehicles, and trucks) can send a request when they are approaching an equipped intersection. The Priority Request Generator (PRG) checks to see if the MAP SPaT Receiver has new map data. Once a MAP, or several MAPs, are received, the PRG determines which MAP describes the intersection that the vehicle is approaching and which lane/approach the vehicle is located on. Once this is determined, the PRG will compute the travel time to the stop bar (from the MAP data and the current position) and will build and send the Signal Request Message. The PRG will wait for a Signal Status Message (SSM) to confirm receipt of the request. If no SSM is received, or an SSM is received and the vehicle's request isn't in the table of active requests, the PRG will update the SRM and send it again. If the vehicle speed changes significantly, the estimated time of arrival is updated and an updated request is sent. Once the vehicle crosses the stop bar, a cancel request is sent. Once a SSM is received showing the canceled request is no longer active, the PRG will check any available MAPs or wait for a new MAP to be received.

3.2.7 Trajectory Aware (RSE, MRP)

The Equipped Vehicle Trajectory Aware component is responsible for collecting data from each of the connected vehicles that is broadcasting Basic Safety Messages (BSM) when in the range of an RSE. Each BSM received by the Message Transceiver (Channel 172) is forwarded to the Trajectory Aware component. The Trajectory Aware component checks to see if the vehicle is within the geo-fence area created by the MAP GPS waypoints, and creates a record for each vehicle's temporary ID that is received and within the geo-fence. Checking the geo-fence location of each vehicle allows the system to filter out vehicles that are in nearby parking lots or other non-traffic controller facilities.

Each record is time stamped and contains the position, speed, vehicle type information, traffic control phases, and estimated time of arrival (ETA) at the stop bar (assuming current speed). If a vehicle changes its temporary ID, a new record will be created for the new vehicle ID. Vehicle data is retained for five minutes after the vehicle leaves the RSE range or after it changes its temporary ID. This data is deleted as part of the privacy assurance requirements in the design. The collection of all vehicles that are sending BSM is available to any other MMITSS component that requests data about the location, and trajectory history, of each connected vehicle. Trajectory Aware provides data for Traffic Control and the Performance Observer.

3.2.8 Traffic Control (RSE, MRP)

The Traffic Control component is responsible for the allocation of green time to the traffic control phases based on the information available in Trajectory Aware and from vehicle detector data from the Traffic Control Interface component. Traffic Control was developed to provide traffic adaptive control using concepts from the COP algorithm (Sen and Head, 1997) that was developed as part of the RHODES (Mirchandani and Head, 2001) and to provide dilemma zone protection. The MMITSS Traffic Control component is based on three algorithms: 1) Estimation of the location of unequipped vehicles (called EVLS); 2) a phase allocation algorithm that determines the duration of the green phases based on the data from Trajectory Aware and the estimated locations of unequipped vehicles; and 3) a dilemma zone protection algorithm. Details of these algorithms can be found in Feng et al. (2016).

3.2.9 Priority Request Server (RSE, MRP)

The Priority Request Server component and the Priority Solver component provide the multimodal priority control for priority requests from multiple vehicles. The AZ MMITSS approach to priority utilizes a new approach to priority and preemption rather than implementing legacy algorithms that exist in current traffic signal controllers. The new approach is based on research from the NCHRP 3-66 project "Traffic signal state transition logic using enhanced sensor information" (Urbanik et al., 2003) that investigated improved approaches to priority control. The algorithms implemented in the MMITSS system use information in the connected vehicle Signal Requests Messages that are received from multiple vehicles of multiple modes. At any time, there may be several active requests that are simultaneously considered through a mathematical programming model that captures the ring-barrier and phase-interval based structure of the traffic signal controller. The formulation considers how to best serve all active requests based on a policy (N-level policy) set by the operating agency by selecting the mode priority weights in the mathematical programming model. The model is formed and managed by the Priority Request Server and solved by the Priority Solver. The Priority Solver uses an open-source optimization solver – the GNU Linear Programming Kit (GLPK) to solve the optimization each time a new request is received, updated, or canceled. The output of the Priority Request Server is a flexible signal timing schedule that the Traffic Control component can take as input when allocating phase green time based on the estimated vehicle arrivals. Details of the algorithms can be found in Zamanipour (2016).

3.3 HARDWARE DEVELOPMENT, ARCHITECTURE, AND INSTALLATION

The City of Portland is well known for its implementation of transit, including the streetcar, bicycle facilities, and pedestrian considerations in its transportation management system. It was decided to implement MMITSS to provide traffic signal priority for transit using streetcar vehicles. The City of Portland identified three intersections (see Figure 8) that are part of the Smart City PDX Initiative: SW Market St. and SW 11th Ave., SW Market St. and SW 10th Ave., and SW Market St. and SW Broadway. Each of the three intersections was equipped with a DSRC Roadside Unit (RSU) and a Roadside Coprocessor (called the MMITSS Roadside Processor). The University of Arizona provided three Savari RSUs and three Raspberry Pi 4+s for the deployment. The Raspberry Pi units are not field-hardened computers, but they meet the processing requirements of MMITSS and they are much less expensive. One DSRC OBU (Savari) was also provided by the University of Arizona with an enclosure that was installed on one streetcar. Future expansion of the system was possible if this initial deployment was successful and provided value to the City of Portland.

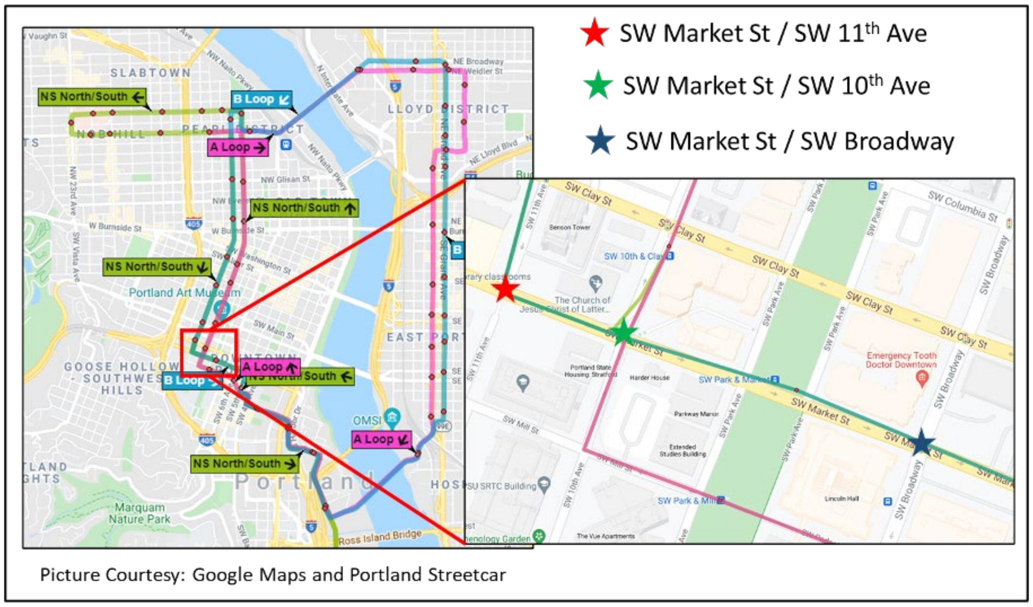

★ SW Market St / SW 11th Ave
★ SW Market St / SW 10th Ave
★ SW Market St / SW Broadway
Picture Courtesy: Google Maps and Portland Streetcar

Figure 8: Intersections selected for deployment of MMITSS in the Portland Smart City PDX Initiative

3.4 SYSTEM ARCHITECTURE

The MMITSS software architecture was refactored to support use of the Savari OBU. The Savari OBU doesn’t support the RSU 4.1 API that is used to allow the vehicle-side processor (VSP) to talk to the OBU. The VSP includes the priority request generator (PRG) that uses the vehicle's position (GPS) and MAP message information to generate and send a Signal Request Message (SRM) from the vehicle to the intersection RSU. Figure 9 shows the modified architecture for a single intersection. In this configuration, the OBU in the streetcar sends a Basic Safety Message (BSMs) to the RSU. The RSU forwards the BSM to the Trajectory Aware component of MMITSS that stores a collection of BSMs for each vehicle for computing performance measures, estimating vehicle arrivals (non-connected vehicles), and support for other applications that need vehicle data. Trajectory Aware forwards the BSMs to a Message Distributor that is actually from the MMITSS Simulation Tools package. In simulation, the Message Distributor forwards BSMs to the geographically correct intersection since in the simulated world there is no physical separation between wireless devices. The Message Distributor forwards the BSM to the Priority Request Generator Server (PRGS), another MMITSS Simulation Tools package that instantiates multiple Priority Request Generators (PRG) that would normally be deployed in the VSP on the vehicle. The PRGS sends the PRS to the Priority Request Server (PRS) which manages all requests that are received and generates the Signal Status Message (SSM) that is forwarded to the PRGS.

Figure 10 illustrates the Portland MMITSS deployment and the capture and storage of connected vehicle data. Each intersection receives BSMs from the vehicle (streetcar) as it progresses through the system. Each RSU broadcasts MAP and SPaT data.

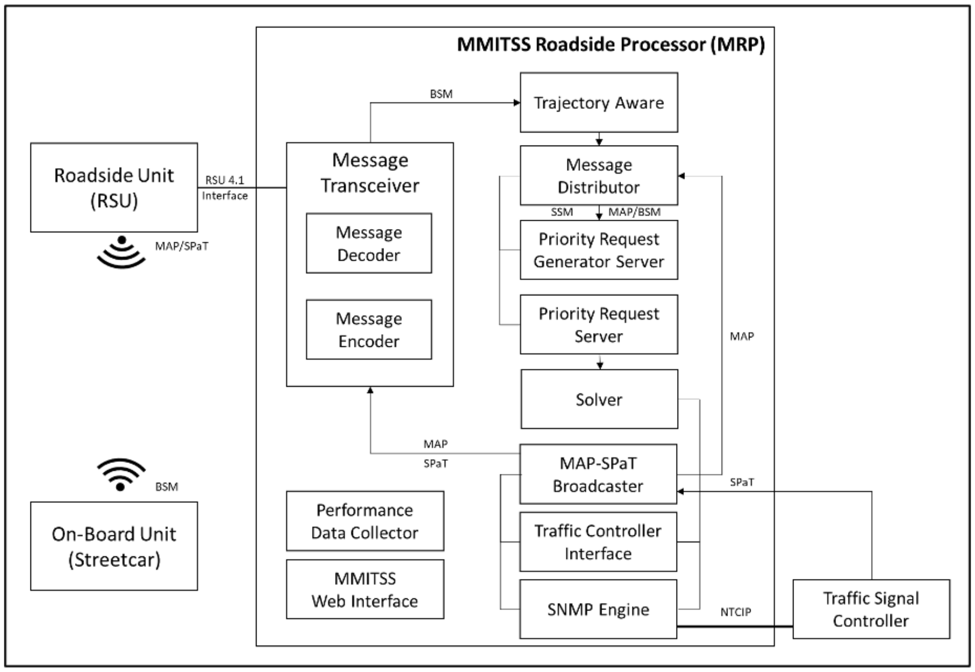


Figure 9: Refactored MMITSS architecture for a single intersection

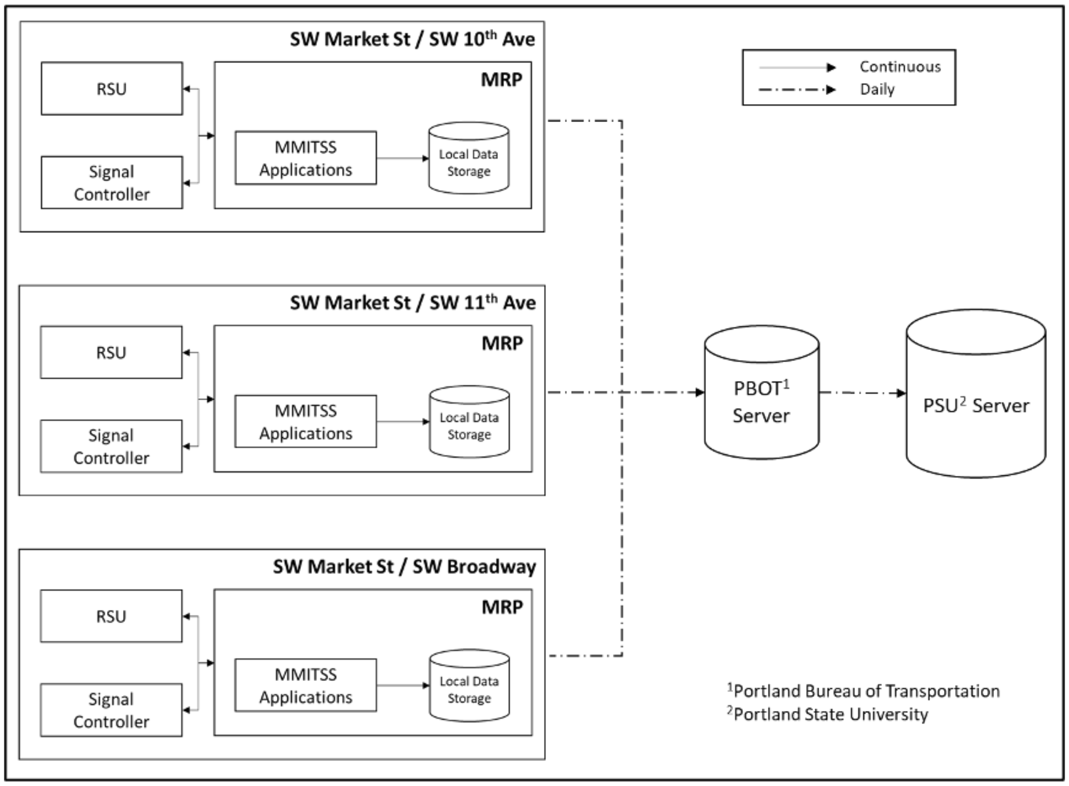


Figure 10: MMITSS system architecture for data capture and storage

All of this data, plus the SRM and SSM messages, is captured on the MRP and forwarded daily to a server hosted at the Portland Bureau of Transportation (PBOT) and then to a server at PSU where it is stored and analyzed.

3.4.1 Simulation Modeling and Operation Validation

To test the MMITSS system, including validation that the integration with the Intelight MaxTime controllers works correctly, a VISSIM simulation model of the study area was created. Figure 11 and Figure 12 show screenshots of the simulation model, including the Intelight MaxTime controller (software in the loop). The model includes the streetcar platform just before the intersection of SW Market St and SW Broadway where passengers board and alight. Regular vehicles are shown as blue vehicles and the streetcar as a long green vehicle.

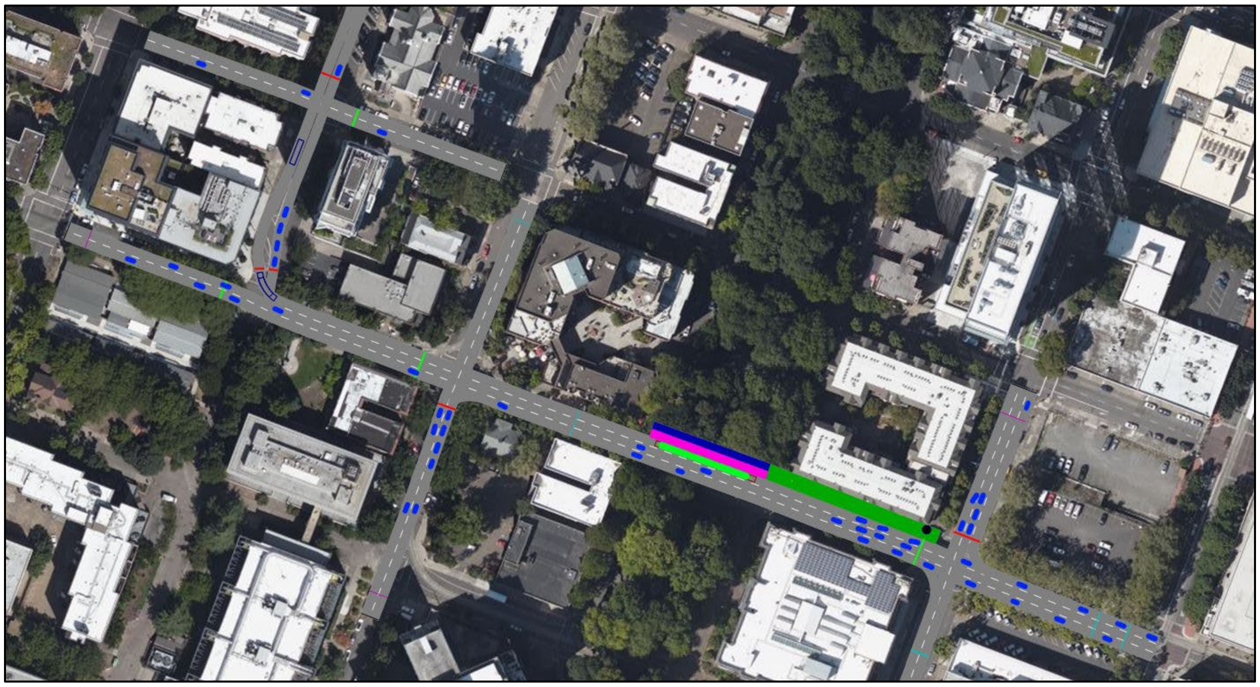
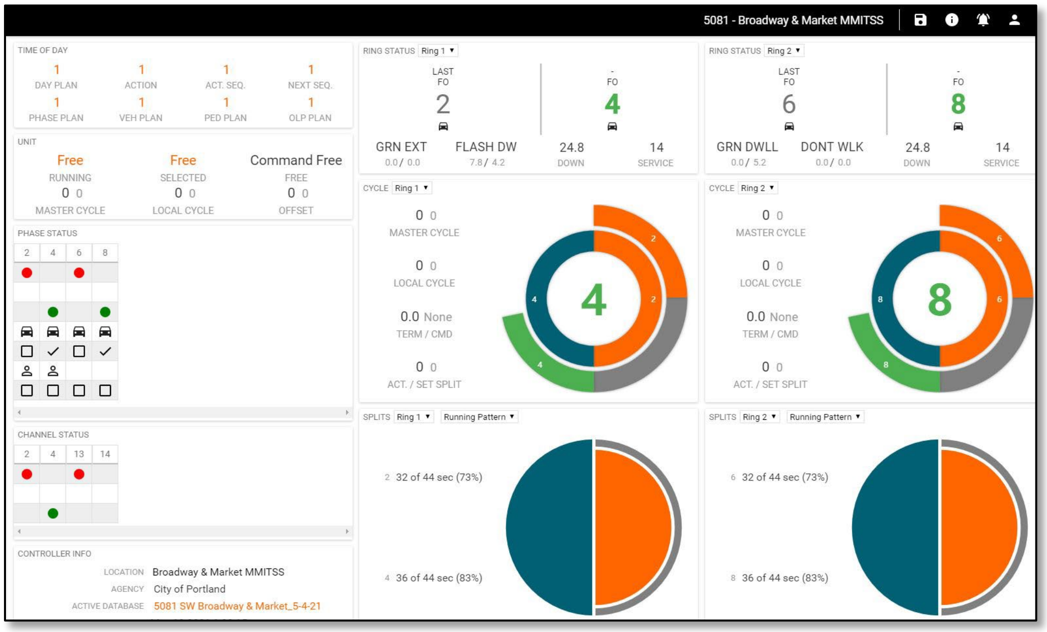


Figure 11: VISSIM simulation of three intersections on streetcar route


5081 - Broadway & Market MMITSS

TIME OF DAY

DAY PLAN	ACTION	ACT. SEQ.	NEXT SEQ.
1	1	1	1

PHASE PLAN	VEH PLAN	PED PLAN	OLP PLAN
1	1	1	1

UNIT

Free	Free	Command Free
RUNNING	SELECTED	FREE
0 0	0 0	0 0
MASTER CYCLE	LOCAL CYCLE	OFFSET

PHASE STATUS

2	4	6	8
🟥		🟥	
	🟩		🟩
🚘	🚘	🚘	🚘
☐	✓	☐	✓
👤	👤		
☐	☐	☐	☐

CHANNEL STATUS

2	4	13	14
🟥		🟥	
	🟩		

CONTROLLER INFO

LOCATION	Broadway & Market MMITSS
AGENCY	City of Portland
ACTIVE DATABASE	5081 SW Broadway & Market_5-4-21

RING STATUS Ring 1

LAST FO 2 — FO 4

GRN EXT 0.0 / 0.0 | FLASH DW 7.8 / 4.2 | 24.8 DOWN | 14 SERVICE

RING STATUS Ring 2

LAST FO 6 — FO 8

GRN DWLL 0.0 / 5.2 | DONT WLK 0.0 / 0.0 | 24.8 DOWN | 14 SERVICE

CYCLE Ring 1

0 0 MASTER CYCLE
0 0 LOCAL CYCLE
0.0 None TERM / CMD
0 0 ACT. / SET SPLIT

CYCLE Ring 2

0 0 MASTER CYCLE
0 0 LOCAL CYCLE
0.0 None TERM / CMD
0 0 ACT. / SET SPLIT

SPLITS Ring 1 Running Pattern

2 32 of 44 sec (73%)
4 36 of 44 sec (83%)

SPLITS Ring 2 Running Pattern

6 32 of 44 sec (73%)
8 36 of 44 sec (83%)

Figure 12: Software-in-loop Intelight MaxTime traffic signal controller output

The traffic signals along this corridor operate as coordinated signals with fixed cycle lengths, splits, and offset. MMITSS is designed to utilize some flexibility in the traffic signal timing that results from actuated signal control. Fixed time signals don't allow MMITSS to provide priority signal control for the transit vehicle (streetcar). The simulation model was used iteratively with the city traffic signal engineer to create some flexibility in the signal timing. For example, the pedestrian walk intervals were programmed to rest in Walk to allow maximum crossing time for pedestrians during the phase split timing. The rest in Walk was changed to a fixed Walk interval to provide some flexibility.

MMITSS provides coordination control as a form of priority and was configured to match the coordination plan(s) that the City of Portland normally utilized. The coordination behavior, with the fixed Walk interval, instead of rest in Walk, was validated using the simulation model before being implemented in the field.

Since only one streetcar was equipped with an OBU, the existing method of providing traffic signal priority was retained for all the other vehicles. For the non-connected streetcars, a detector was used to call a preempt in the controller when the streetcar was approaching each intersection. The streetcar was released from the intersection just upstream from SW Market St. and 11th Ave. at the right time in the cycle so that the preemption call would be received at SW Market St. and 11th Ave. at a logical time in the cycle. If the streetcar is a connected streetcar, it is desired to omit the preemption and allow MMITSS to provide the priority signal timing. To accomplish this behavior, a special function on the controller was set to TRUE when a SRM was received by the RSU and forwarded to the MRP. The controller mapped the special function to the OMIT PREEMPT input on the controller. If the streetcar is not a connected streetcar, no

SRM is sent and the special function is FALSE and the detector call triggers the PREEMPT. This behavior was tested and validated in the simulation model before being deployed at the intersections.

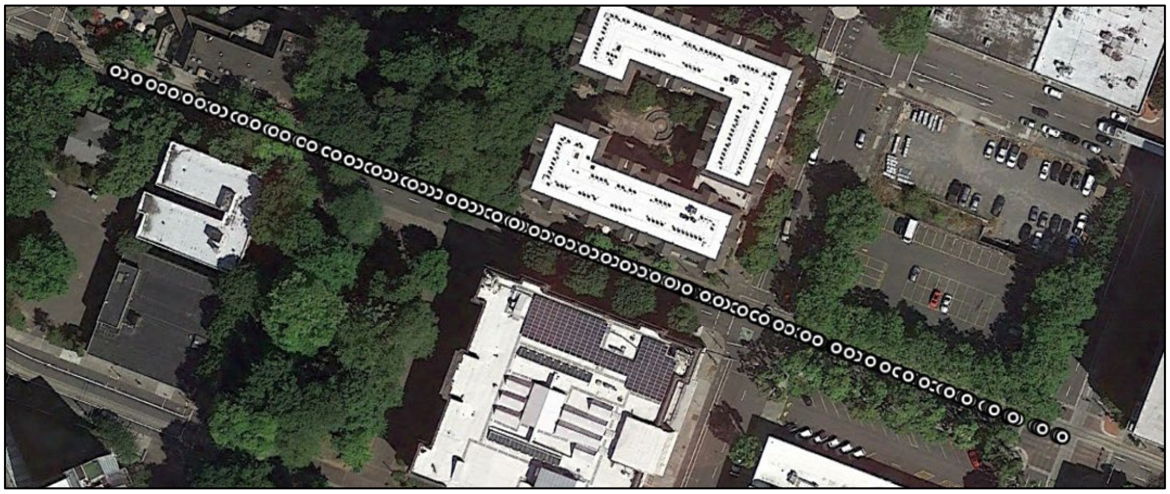


Figure 13: Basic Safety Messages (BSMs) on SW Broadway to Market St. intersection

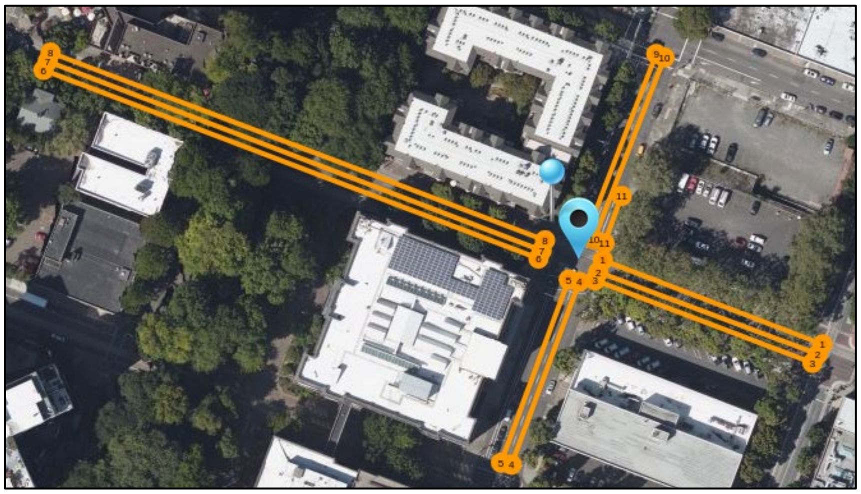


Figure 14: Intersection MAP Message (MAP) on SW Broadway to Market St. intersection

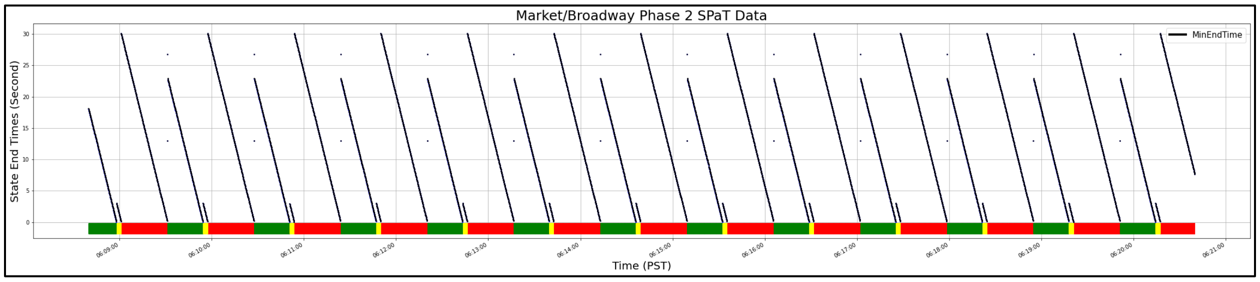


Figure 15: Signal Phase and Timing (SPaT)

Streetcars broadcasted BSMs throughout the route. There is no on-board computation from the OBU. These BSMs are received at the intersections by the RSUs, which in turn broadcast SPaT and MAP and generate SRM and SSM. Figures 13-15 show the collected BSM, MAP and SPaT data. The computation is developed for priority logic. In the case of the MMITSS system, all resource-intensive computations are isolated at each intersection and prioritize the unique OBU-equipped streetcar.

3.4.2 Visualizations

Using data from the simulations, single-vehicle and multi-vehicle time-space phase visualizations were developed. The time-space phase diagrams (Figures 16-17) are a tool to analyze traffic stream progression through a series of signalized intersections. These diagrams show movement of individual vehicles in time and space along with signal state timing sequences. The X-Axis represents time and the Y-Axis represents the distance travelled by a vehicle on its path and the locations of intersections. The slopes of lines represent the speeds of vehicle. These graphs can be used to assess performance of signal priority applications and coordination plans. The visualization tools were developed using Pandas and Plotly (Python). These plots are further described in Appendix 7.1.

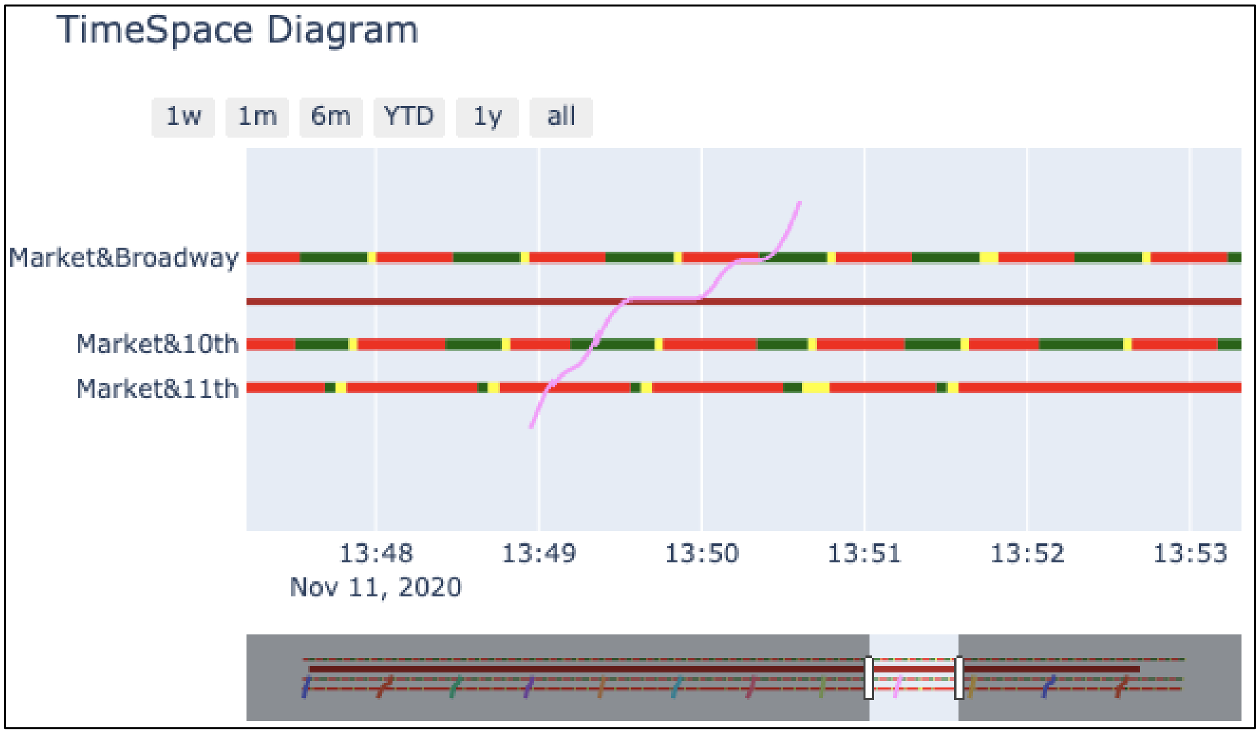


Figure 16: Time-space diagram for single vehicle

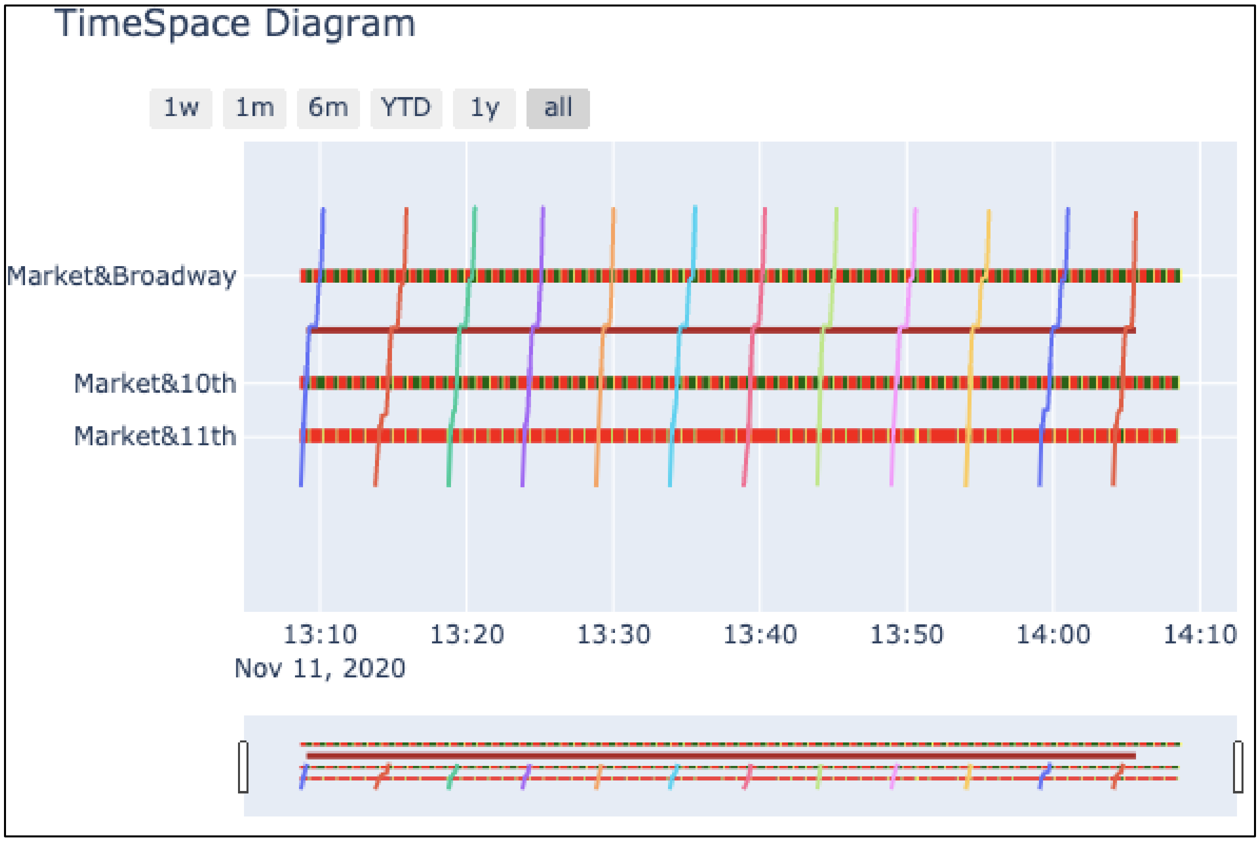


Figure 17: Time-space diagram for multiple vehicles

4.0 FIELD DEPLOYMENT AND TESTING

Once MMITSS was validated using the simulation model, the MRPs were installed at each of the study intersections and the OBU was installed in a City of Portland vehicle that was driven along the streetcar route to test the functionality. Since the City of Portland and the City of Tucson use the same streetcar vehicles, an OBU mount from the Tucson Sunlink system was adopted. Figure 18 shows pictures of the installation on the Tucson streetcar from overhead and from outside the windshield.

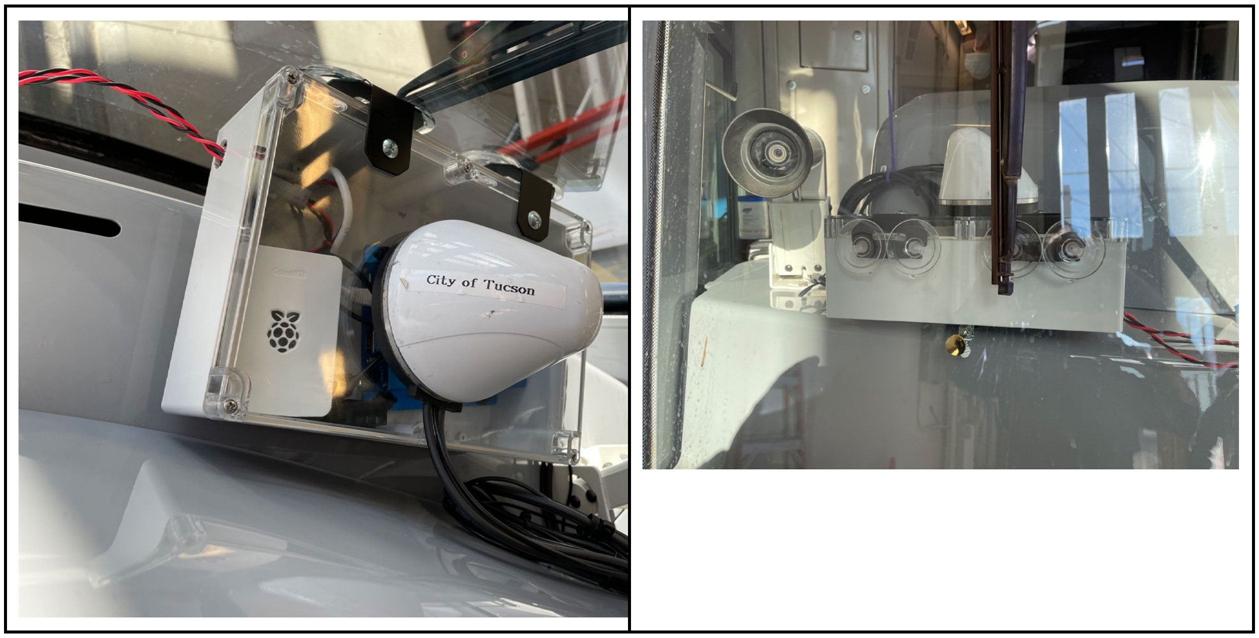


Figure 18: Streetcar OBU installation (based on Tucson Sunlink Streetcar installation)

The system was observed operating without and with the connected streetcar. When the connected streetcar was not present, MMITSS operated the signals as a coordinated system using the MMITSS priority-based coordination approach. The signals behaved as expected and in coordination with surrounding traffic signals. When a non-connected streetcar approached the intersections, the PREEMPT call was placed as expected and the signal controller used the pre-programmed PREEMPTION behavior. When a connected streetcar approached, the special function was set to TRUE and the PREEMPTION did not control the intersection. However, the MMITSS priority control did not respond as expected. MMITSS would start to control, then stop, then start again, then stop, etc. (See discussion in Section 5 about challenges of the pilot project). The issue was determined to be related to the use of GPS for positioning in the urban environment that consists of tall buildings and trees. Since no appropriate solution could be found under current conditions and funding, it was decided to stop the project.

Several important lessons can be taken from this project. First, the use of GPS in the urban environment should be carefully considered. Some systems use a combination of

GPS and dead reckoning that can improve positioning or the use of the RTCM should be considered and appropriate networking accommodations developed. Second, the MMITSS architecture that utilizes Docker as a container technology was very flexible and allowed the team to accommodate the challenges associated with the current DSRC political landscape. Third, the use of the VISSIM simulation tool with the Intelight MaxTime software in the loop controller proved invaluable to integrating MMITSS with the existing traffic signal control. The collaboration between the city staff and the research team allowed the development of good solutions to some challenging traffic control problems. Some features, such as rest in Walk, are highly desirable and the MMITSS research team needs to consider how these features can be accommodated in a priority traffic control system. Unfortunately, the project didn’t achieve the goal of achieving an operational system that could be analyzed, evaluated, and an expansion planned, but City of Portland staff did learn about connected vehicle technologies, implementation of advanced traffic management systems, and the challenges of emerging technologies.

5.0 CHALLENGES

The project faced several significant technical challenges. One significant challenge was that the Federal Communications Commission (FCC) didn't issue a regulation in 2016 that would require all new passenger vehicles manufactured and sold in the U.S. after 2021 to include DSRC OBUs. This placed the future of DSRC as the selected wireless communication technology in jeopardy. Alternative technology, namely C-V2X (cellular V2X), was being discussed as an improvement to DSRC. C-V2X was included in the 5G standards development (3GPP Version 14) and would accommodate a technology evolution path that DSRC did not allow. Since the future was uncertain, the City of Portland was reluctant to invest in technology that could be obsolete in the near future. The University of Arizona had spare RSUs and OBUs that they agreed to loan to the project. This allowed the City of Portland to gain connected vehicle systems experience and for the project team to pursue the goals of data collection and analysis.

Another challenge was that the Connected Vehicle Pooled Fund decided to pursue an MMITSS Phase III project to improve the deployment readiness of MMITSS. The original MMITSS proof-of-concept system was developed by graduate students with a focus on algorithms more than on software quality. The MMITSS Phase III project provided an opportunity to add support for users, such as a web-based user interface to configure, enable/disable MMITSS, and view data that was collected. The MMITSS Phase III project started at about the same time as this project and had a duration of 24 months. The desire to use the improved software delayed the start of the deployment but still allowed the development of the VISSIM simulation model, testing the integration with the City of Portland's Intelight MaxTime traffic signal controllers, and development of an alternative architecture where the Priority Request Generator (PRG) was hosted on the Roadside Processor (MRP) rather than in the vehicle. This was required since the Savari OBU that the University of Arizona loaned the project doesn't support the RSU 4.1 API specification so there was no way to talk to the OBU. The MMITSS Phase III software was ready for deployment in late 2019.

The final challenge was realized once the OBU was installed on the streetcar. The team observed the behavior of the system when the connected streetcar approached the connected intersections and the behavior wasn't as expected. The SRM requests would be received, then canceled, then received, then canceled, etc. The BSM data was collected and plotted on Google Earth to observe the path to the vehicle. Figures 19 and 20 show two separate trajectories through the corridor. Clearly, there is an issue with the vehicle location. If the vehicle isn't located within the roadway (e.g., on the area described by the MAP message), the SRM will not be generated. If the vehicle is located on the roadway, but then appears to leave, a cancel SRM will be sent. This is the behavior that was observed during the field integration testing.

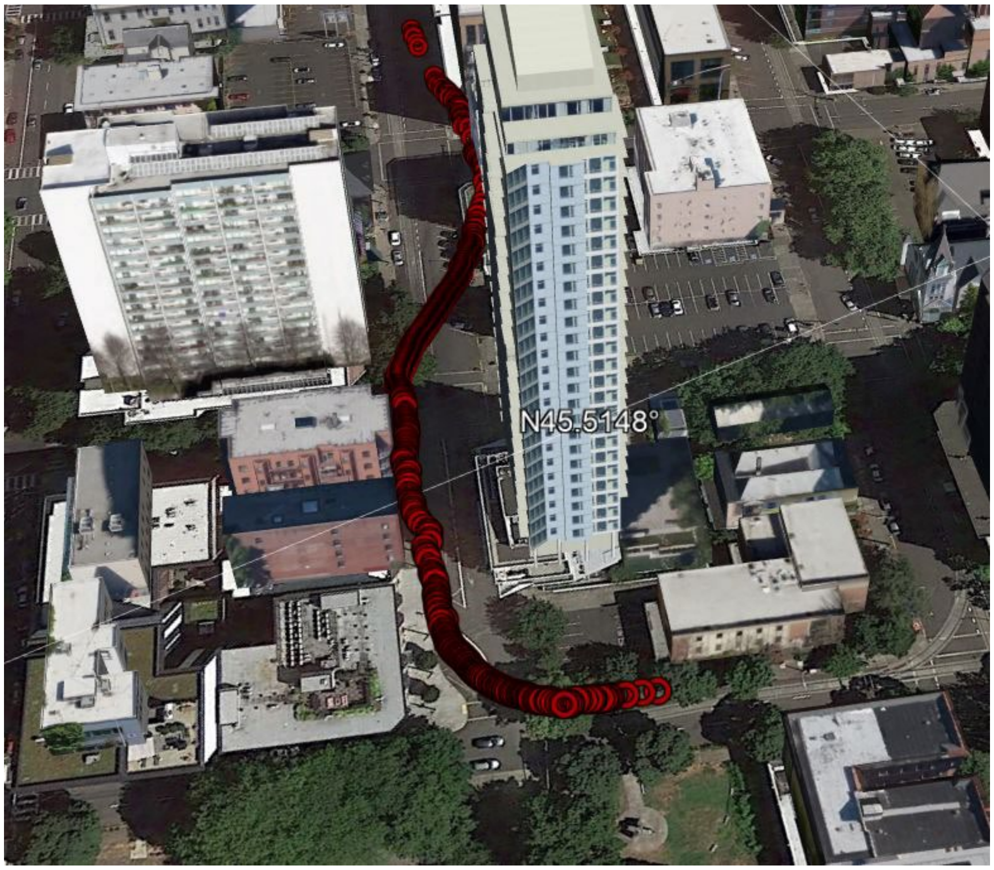


Figure 19: Streetcar trajectory approaching SW Market St and 11th Ave (1st run)

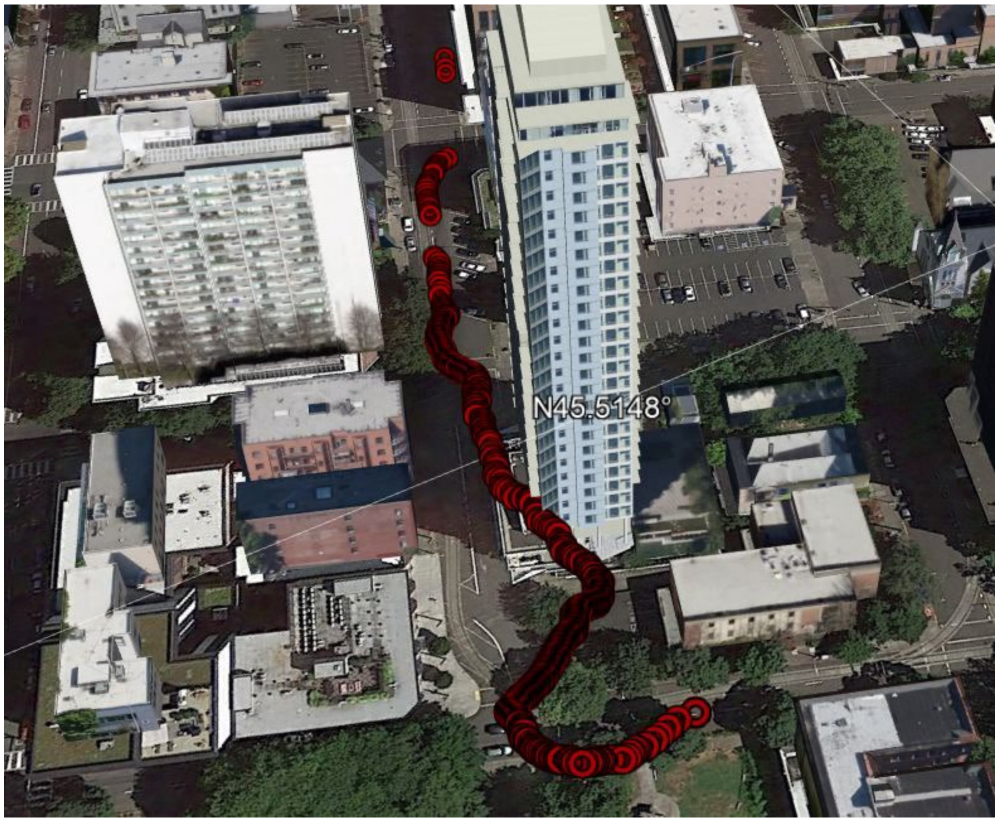


Figure 20: Streetcar trajectory approaching SW Market St and 11th Ave (2nd run)

The issues in vehicle positioning are likely related to the urban characteristics of downtown Portland. The tall buildings, trees, and lack of clear sky view contribute to the poor GPS performance. A possible solution would be to enable and use the SAE J2735 RTCM message (RTCM - Radio Technical Commission for Maritime Services that provides GPS differential corrections). However, this would require each RSU to have an internet connection that would allow two-way data exchange outside of the closed City of Portland IT network. This implementation was not possible based on the existing city IT policies. Unfortunately, with the poor streetcar positioning, it was not possible to proceed with the evaluation of operations and it was decided to end the pilot project. The City of Portland is partnering with TriMet on their Next Generation Transit Signal Priority (Next-Gen TSP) project, which is instrumenting buses outside of the downtown core and testing CV infrastructure.

6.0 REFERENCES

AASHTO, ITE, and NEMA. (2005) NTCIP 1202:2005 v02.19 National Transportation Communications for ITS Protocol Object Definitions for Actuated Traffic Signal Controller (ASC) Units – version 02.

Feng, Y., M. Zamanipour, L. Head, S. Khoshmagham. Connected vehicle–based adaptive signal control and applications. *Transportation Research Record* 2558 (1), 11-19. https://doi.org/10.3141/2558-02

Gartner, N. (1983) OPAC: a demand-responsive strategy for traffic signal control. *Transportation Research Record* 906, 75–81.

Gettman, D., SG Shelby, L. Head, DM Bullock, N. Soyke. (2007) Data-Driven Algorithms for Real-Time Adaptive Tuning of Offsets in Coordinated Traffic Signal Systems. *Transportation Research Record*. 2007;2035(1):1-9. doi:10.3141/2035-01

Goodall, N., B. Smith, B. Park. (2013) Traffic signal control with connected vehicles. *Transportation Research Record*. 2381, 65–72. http://dx.doi.org/10.3141/2381-08.

Mirchandani, P., L. Head. (2001) A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technology* 9, 415–432. http://dx.doi.org/10.1016/S0968-090X(00)00047-4.

Sen, S., L. Head. (1997) Controlled Optimization of Phases at an Intersection. *Transportation Science* Vol. 31, No. 1. https://doi.org/10.1287/trsc.31.1.5

Urbanik, T., D. Bullock, L. Head, D. Gettman, R. Campbell, M. Abblett, S. Quayle. (2006). NCHRP 3-66: Traffic Signal State Transition Logic Using Enhanced Sensor Information. *National Cooperative Highway Research Program-Transportation Research Board*.

USDOT, Dynamic Mobility Applications, https://www.its.dot.gov/research_archives/dma/index.htm, last accessed August 1, 2024.

USDOT, Dynamic Mobility Applications (DMA) Program, Multi-Modal Intelligent Traffic Safety System (MMITSS) https://www.its.dot.gov/research_archives/dma/bundle/mmitss_plan.htm, last accessed August 1, 2024.

Zamanipour, M., L. Head, Y. Feng, S. Khoshmagham. (2016) Efficient Priority Control Model for Multimodal Traffic Signals. *Transportation Research Record*. 2557, 44-54. https://doi.org/10.3141/2557-09

7.0 APPENDIX

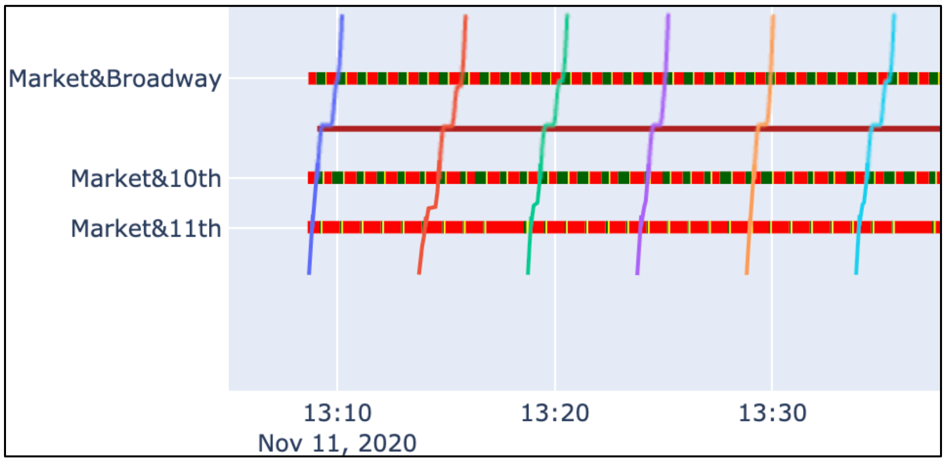
7.1 TIME-SPACE DIAGRAM DESCRIPTION

This describes the details of the project, what files are used, what intersections are used for testing, and the detailed description of the code required to produce the visualizations.

7.1.1 Description of the Project

The agenda is to generate the Timespace plots for Portland Streetcar vehicles to test the MMITSS device. TimeSpace plots provide the details of the vehicle travel path across various intersections and the change in traffic signals through the time. The overall idea for generating visualization is to use the horizontal Gantt bars along with line plots to generate the required timespace plots. The reason to choose Gantt bars is because Gantt bars seemed to be the best way to represent the traffic signal changes with respect to time, and also customize the colors for the bars and provide a customized distancing between the intersections.

- x-axis - defines the travel time
- y-axis - distance travelled by vehicle on its path across the traffic signals & locations of intersections



7.1.2 Details of Intersections and Files Used

The intersections used for testing purposes are Market-Broadway, Market-10th and Market-11th. These intersections have an associated id listed below:

- Market-Broadway: 34108
- Market-10th: 35925
- Market-11th: 36447

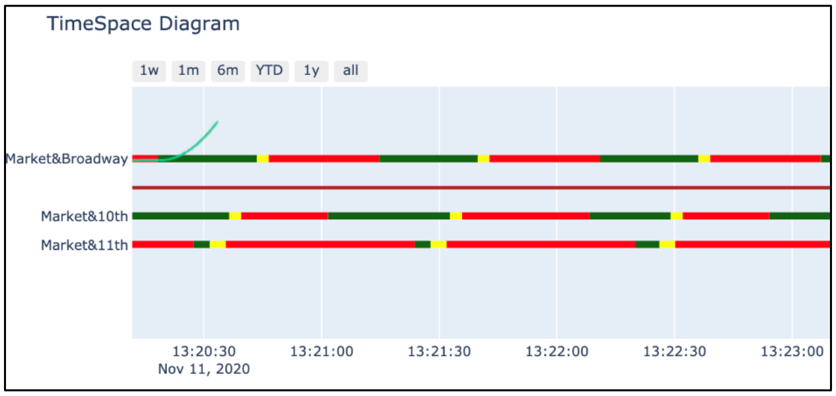
The files used for the generation of the visualizations are BSM and SPaT files. The BSM and SPaT files are created for each intersection on a daily basis and pushed from city to PSU servers.

The BSM file consists of parameters describing the location and motion of the connected vehicle, at an instant of the time such as vehicleID, the distance travelled by the vehicle, the type of vehicle, the location coordinates of the vehicle, etc.

The SPaT data describes the current state of each phase of the intersection and how long this state of each phase will persist.

7.1.3 Description of Code that Generates the Plot for the Horizontal Traffic Signal Status using Plotly Gantt Charts

This plot shows a horizontal bar that provides traffic signal status at a given timestamp. The horizontal bar is colored red-green-yellow to indicate the traffic signal status. The plots are developed using the SPaT data (signal phase and timing data), which is in the csv format. A sample of the bar graph shown in the image below.



<u>SPaT data required columns for plots</u>

1. Timestamp_verbose - data details recorded at every 1/10th of a second
2. intersectionId - the unique id of the intersection
3. v1_currState - v8_currState - vehicle phase status. Represents the states for every phase.
   Current state of the vehicle phases -
   0. Unavailable
   1. Dark
   2. Stop-then-Proceed (flashing red)
   3. Stop-and-Remain (redlight)
   4. Premovement (notusedintheUnitedStates)
   5. Permissive-Movement-Allowed (permissive green)
   6. Protected-Movement-Allowed (protected green)
   7. Permissive-Clearance (permissive yellow)
   8. Protected-Clearance (protected yellow)

Details for creating the plot

1. Initialize the filename variables to the correct SPaT data files.

```
SpatFileNameMarketBroadway = "PDX_MULTIPLE/spatLog_Market_Broadway_11112020_130839.csv"
SpatFileNameMarket10 ="PDX_MULTIPLE/spatLog_Market_10th_11112020_130838.csv"
SpatFileNameMarket11 ="PDX_MULTIPLE/spatLog_Market_11th_11112020_130838.csv"
```

2. Using Pandas, the code creates a dataframe and reads the columns from the csv file that are required to create the plot. The columns required are: intersectionId, log_timestamp_verbose, end_timestamp , v4_currState. The function used is pd.read_csv (“file name”)
Command ---> df = pd.read_csv (SpatFileNameMarket10)

3. The code uses the particular phase (either v4_currState or v5_currState) which is the column from the csv file. The phases contain the details of the traffic signal status at the given timestamp (red, yellow, green). Select the signal phase to be plotted. To select different signal phases to be plotted, the code needs to be edited to change the dataframe to select the particular column.

4. A column with a name signal_phase_change is created and the only rows that have a change in traffic signal are selected from the previous dataframe and the data frame is updated with the required rows. Example: when a signal changes from red to green we would only select the two rows, first row that has the details of when the signal is red and the second row that has the details when the signal is green.

5. Create a new data frame with columns new_df ['Task','Start','Finish','Resource'] And insert the above computed data accordingly into this new dataframe.

Task - intersectionId
Start - timestamp_verbose
Finish- end_timestamp
Resource- v1_currstate (phase)

6. Repeat the above to create one data frame for each intersection to be plotted. These three data frames are then merged into one dataframe to produce the plots. The idea is that we would need one dataframe for each of the intersections to ensure that the traffic signal details are put into the respective data frame to plot the Gantt bars.

7. Create a color dictionary based on the states

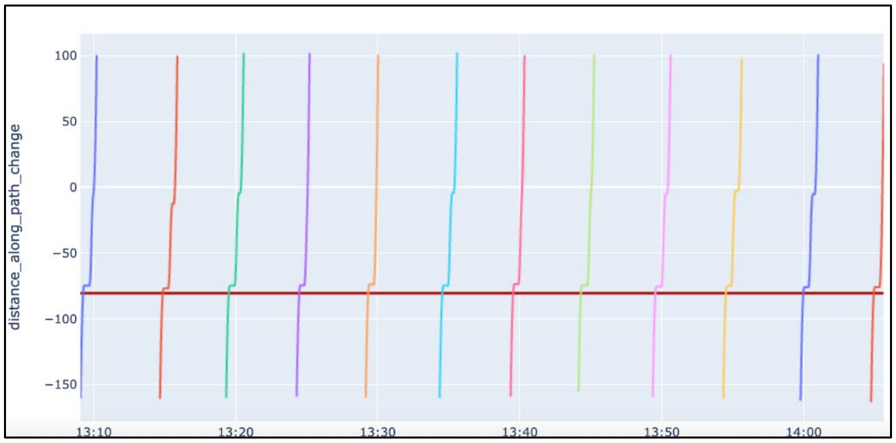
```
colors = {'red': 'rgb(255,0,0)',
          'red': 'rgb(255,0,0)',
          'green': 'rgb(0,128,0)',
          'green': 'rgb(0,100,0)',
          'yellow': 'rgb(255,165,0)',
          'yellow': 'rgb(255,255,0)',}
```

8. Construct the dictionary that is needed to separate the Gantt bars by distance.
Example - groupedtask = [0,-160,-240] {the numbers are negative because of the way the Gantts work and that spacings is the absolute value of the distances between the intersections).

9. Use this new dataframe (df_spat_pdx_final) to plot the Gantt chart using Plotly. Provide the color dictionary for the color codes to be reflected and the grouped task for the distance between intersections to be reflected.

7.1.4 Description of Code that Generates the Plot for Vehicle Trajectory Using Plotly Line Charts

This plot shows the vertical lines on the time-space plots that represent the vehicle path across the intersections and traffic signal statuses represented by horizontal Gantt bars. The x-axis represents the distance along the path and y axis is the travel time for the vehicles. The plots are developed using the BSM data (Basic Safety Messages) which is in the csv format. An example of the line graph is shown in the image below.



<u>BSM data required columns for plots</u>
1. timestamp_verbose = data details recorded at every 1/10th of a second
2. temporaryId = unique vehicle id
3. trajectory_signal_group = describes the details of the signal group we are using to plot the Gantt bars (4 ,5)
4. type = the type of the vehicle (400 = streetcar)
5. dist_to_stopbar = the distance to the stop bar from the current position
6. distance_along_path = the distance travelled by the vehicle along its path
7. onmap_status = the status on the vehicle (True/False)

<u>Details for creating the plot</u>
1. Initialize various variables
    a. filename variables
```
BsmFileNameMarketBroadway = "PDX_MULTIPLE/BsmLogMarket_Broadway_11112020_130839.csv"
BsmFileNameMarket10 ="PDX_MULTIPLE/BsmLogMarket_10th_11112020_130838.csv"
BsmFileNameMarket11 ="PDX_MULTIPLE/BsmLogMarket_11th_11112020_130838.csv"
```
    b. signal group variables
        ▪ trajectory_signal_group_4 = 4
        ▪ trajectory_signal_group_5 = 5
    c. vehicle type variable
        • vehicle_type = 400

d. the distance of the intersection from the 0 y-axis plot. These are negative due to the data being normalized the distance along the path (this would make anything below the signal bar as negative distances and anything above the traffic signal bar as positive distances). The distance to the stopbar is roughly estimated based on the data and the approximate measurements on Google maps.

- distancefrombroadwayto10th = -240
- distancefrombroadwayto11th = -320
- distancetostopbar = -80.5

2. Once all the variables are initialized create the dataframes with the required fields for the respective BSM files. In our case, we create three dataframes to represent three intersections.

3. Clean up the data to only have the rows that are required for generating the plots (i.e., keep only the rows that cater to the signal_group, vehicle type, and onmap status).

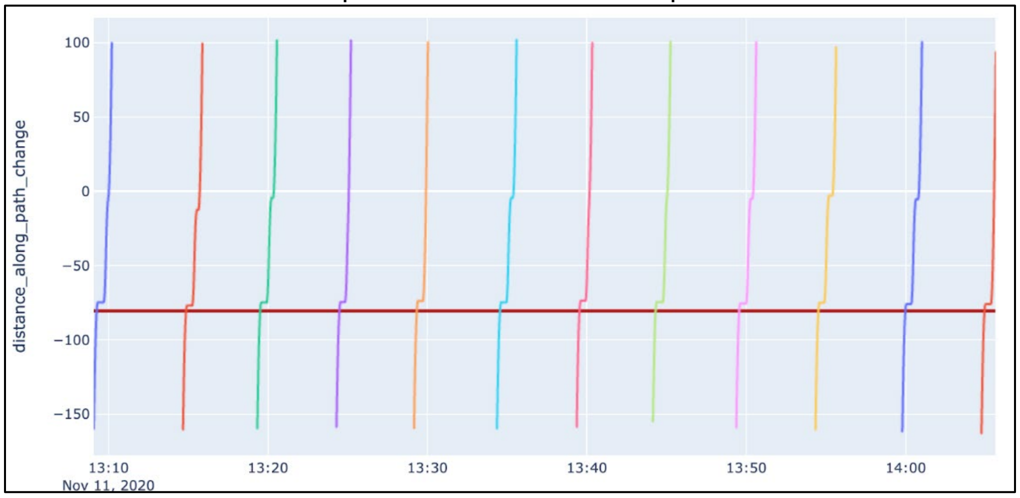
4. Normalize the data to reset the indexes of the data frames, as we have eliminated some data.

5. Group the data by the vehicle id and trajectory signal groups and place these groups into new data frames as we would need to do some calculations after grouping the vehicles to plot the distance traveled along the y-axis.

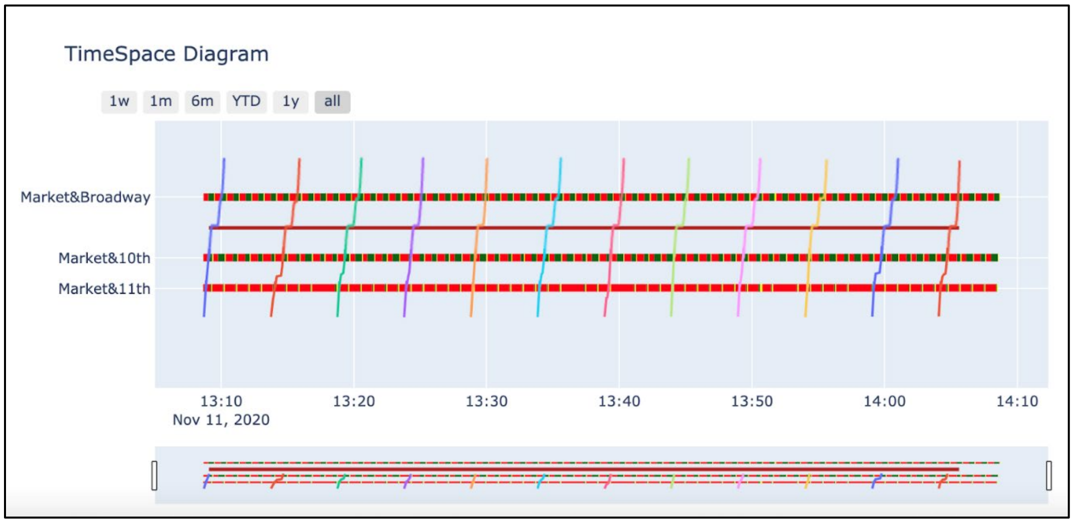
6. The distance along the path changes for the other two intersections that do not start at 0 axis are calculated by adding the distances mentioned earlier as part of the variable distancefrombroadway10th etc.

7. Create a new dataframe to plot the line graph for the STOP sign for the streetcar at approximately -80.5.

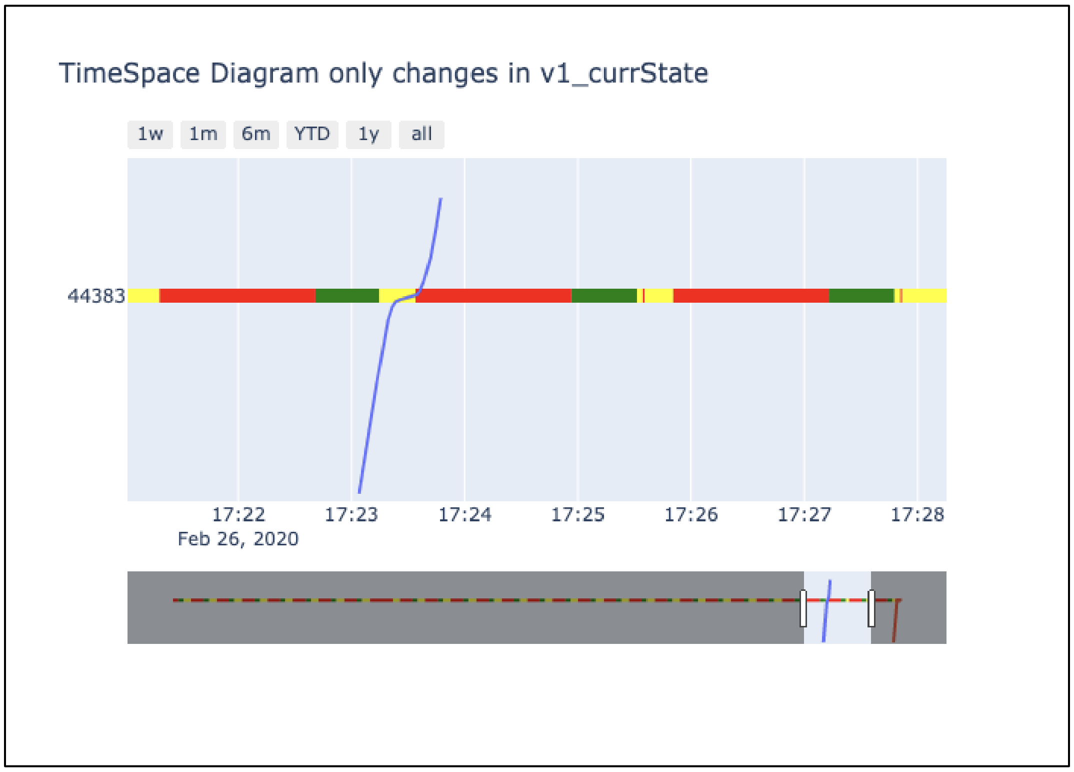
8. Plot the vehicle's travel path across the timestamp.



9. Once the dataframes for the vehicle plots for each intersection and the Gantt bar graphs are ready. Add the line plots on top the Gantt bars.

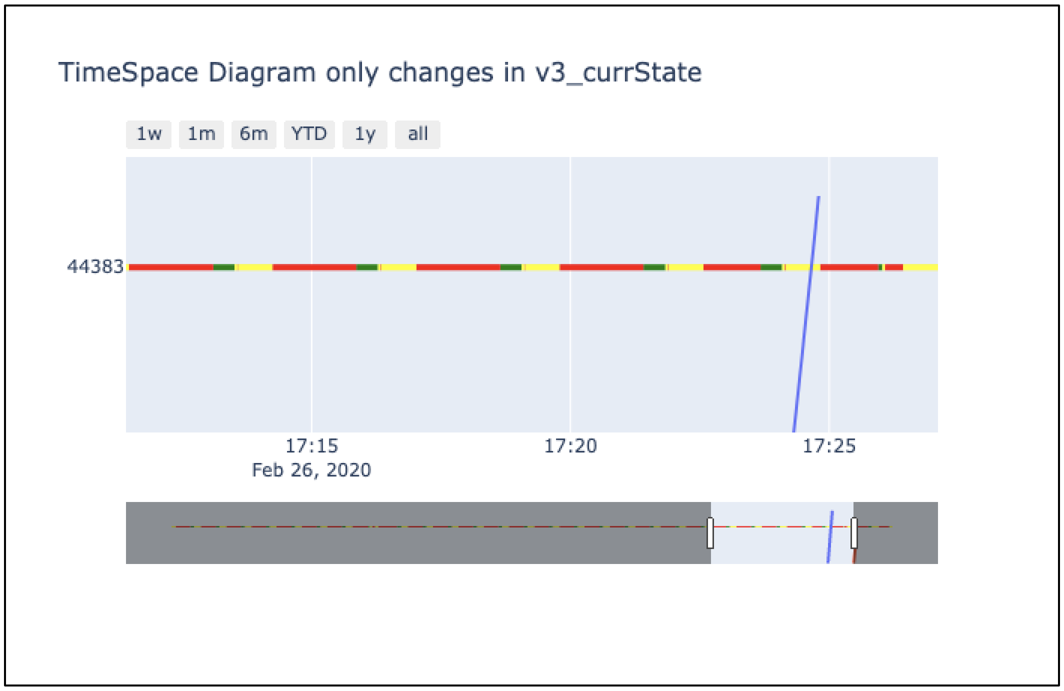


This was the initial work when the plots were generated for a single vehicle and a single intersection.



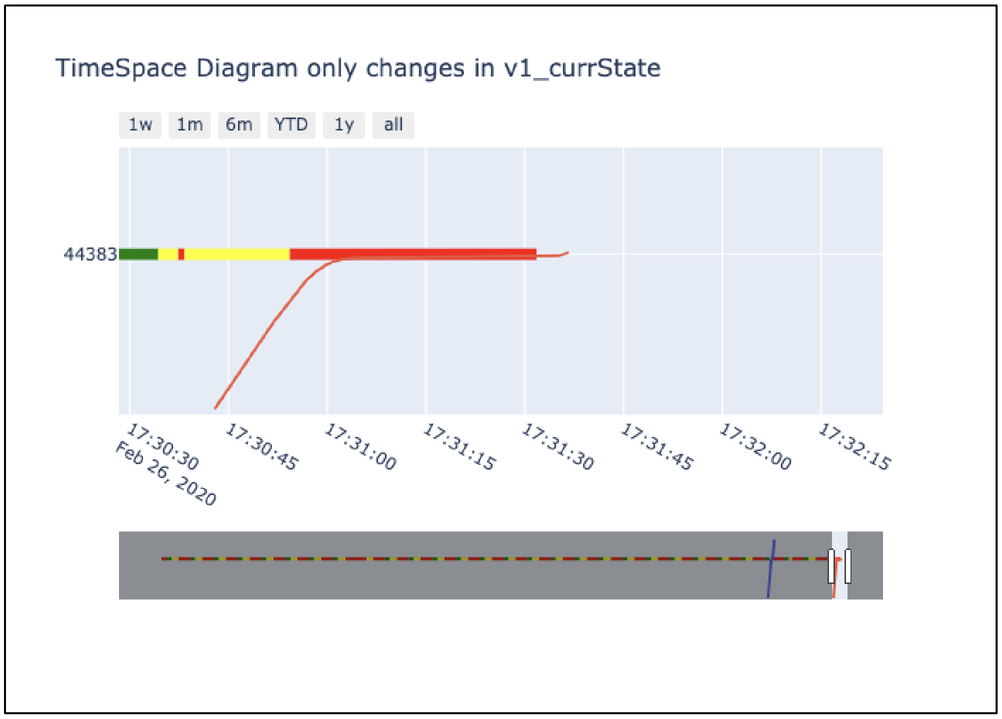
Data used for this plot:

- temporaryId == -713872729
- initial_approach == 3 (v3_currState)
- Distance to the stopbar == 361.12
- Distance along path group =1



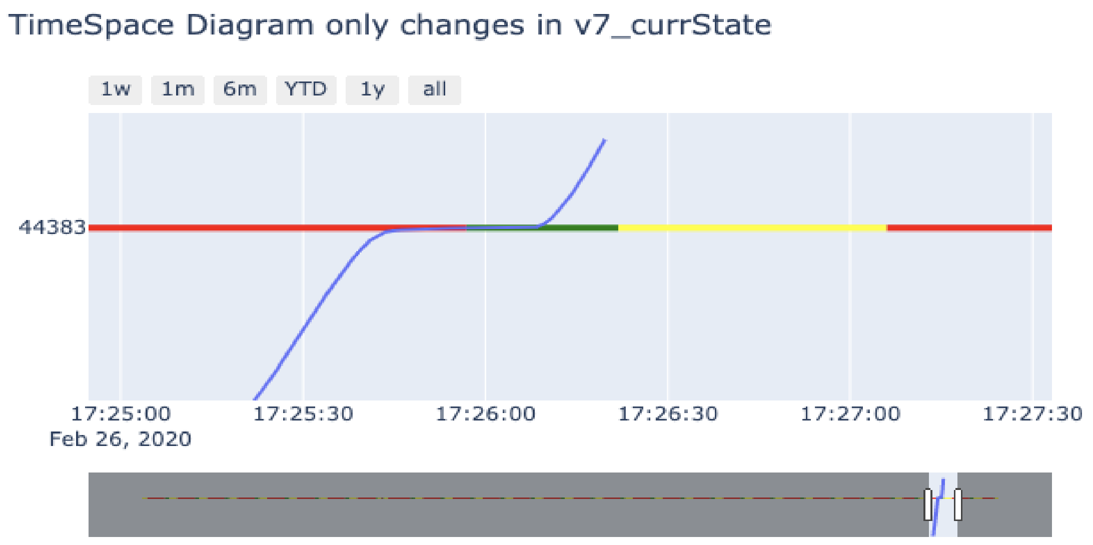
Data used for this plot:

- temporaryId == -713872729
- initial_approach == 1 (v1_currState)
- Distance to the stopbar ==290.29
- Phase = 1
- Distance along path group =2



Data used for this plot:

- temporaryId == -713872729
- initial_approach == 7 (v1_currState)
- Distance to the stopbar == 328.66
- Phase = 7
- Distance along path group =1



7.2 GENERATING VISUALIZATIONS FOR PORTLAND STREETCARS

GitHub for Portland Streetcar MMITSS Project:
https://github.com/mmitss/portlandStreetcar/tree/issue12-push-to-psu

Software Requirements

- pip install python 3.0 and above
- pip install plotly
- pip install pandas
- pip install numpy
- pip install matplotlib

Details about Plotly
1.Why Plotly was chosen
a. It is an open source tool/library.
b. The visualization needed for the time-space plot required a specific color to be assigned when the traffic signals change (red, yellow, green) and this seemed easily achievable with Plotly rather than with Matplotlib or Seaborn. We tried to generate the plots with Matplotlib, Seaborn and Tableau, but didn't find the exact inbuilt functions that helped to generate the plots that were required.

2. Why still it didn't work exactly as required
However, the Plotly tool does not let us plot the horizontal Gantt bars with a customized spacing (i.e., the distance between intersections) of the bars that are required for the signal change plot of the time-space plot.

3. How the issue was fixed
As this is an open-source library we can change the code as per our requirements and, hence, there is a tweak made to cater to our needs (provide customized distance between the intersections) in the code that generates the _Gantt.py.

- Once we pip install the Plotly tool we need to change the program that generates the Gantt bars in the tool
- Replace the _Gantt.py from the below location if you are using the anaconda to run the jupyter notebook with the code in the mentioned github link:**anaconda3/pkgs/plotly-4.8.2-py_0/site-packages/plotly/figure_factory/_Gantt.py**

https://github.com/mmitss/portlandStreetcar/tree/issue12-push-to-psu/tools/v2x-data-transfer

If Anaconda is not used to run the notebook and instead it's done in a virtual environment then find the location where the Plotly was installed and follow the same steps mentioned above *(maybe something like this is where the Plotly could be located pkgs/plotly-4.8.2-py_0/site-packages/plotly/figure_factory/_Gantt.py).*

Generating Visualizations Steps

1. The files are pushed from the city server onto the PSU server (it's on the ITS network platoon.its.pdx)

   a. The directory structure on PSU server -

      mmitss

remoteBsm
intersectioname-remoteBsm-timestamp.csv
intersectioname-remoteBsm-timestamp.csv
ssm
intersectioname-ssm-timestamp.csv
intersectioname-ssm-timestamp.csv
srm
intersectioname-srm-timestamp.csv
intersectioname-srm-timestamp.csv
spat
intersectioname-spat-timestamp.csv
intersectioname-spat-timestamp.csv

msgCount
intersectioname-msgCount-timestamp.csv
intersectioname-msgCount-timestamp.csv

b.The user used to ssh to platoon is portaldu.

c. The part of the code that does this is collaborated with Arizona team and is present in the below github link:
https://github.com/mmitss/portlandStreetcar/tree/issue12-push-to-psu/tools/v2x-data-transfer

d. If anything needs to be changed in the code we need to look at 3 files
1. v2x-data-transfer-main.py
2. PushToPSU.py
3. v2x-data-transfer-config.json

e. The program uses the ssh to connect to the PSU server and the files are transferred over sftp. The json element for the psu part -

```
"psu":
        {"name":"mmitss",
        "ip_address": "216.25.220.74"
        "username": "portaldu",
        "key_file":"config/keys/privatekey.ppk",
        "psu_location":"mmitss"},
```

The ssh key file needs to be a .pem file if the code is run on mac, else it needs to be a .ppk file if the code is run on a Windows machine. Use the ip_address as "platoon.its.pdx.edu" if the 216.25.220.74 does not work on your local machine for testing anything if required.

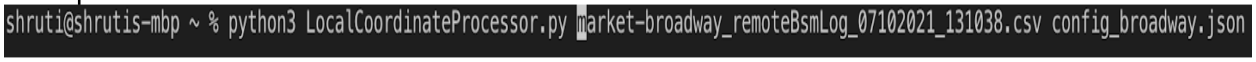
2. Once the files are in the PSU server, copy the BSM and SPaT files from the remoteBsm and SPaT folders required for the particular day (the file names have the dates mentioned) for the intersections required onto your local machine.

3. Run the code mentioned below that would add the various fields for the BSM files
https://github.com/mmitss/portlandStreetcar/tree/master/tools/map-bsm-processor/src
The programs required to run are the

a. LocalCoordinateProcessor.py
(https://github.com/mmitss/portlandStreetcar/blob/master/tools/map-bsm-processor/src/LocalCoordinateProcessor.py)
b. Position3D.py
(https://github.com/mmitss/portlandStreetcar/blob/master/tools/map-bsm-processor/src/Position3D.py)
c. geoCoord.py
(https://github.com/mmitss/portlandStreetcar/blob/master/tools/map-bsm-processor/src/geoCoord.py)

The LocalCoordinateProcessor is the main Python file requires two command line inputs:

a. the location of the input file (BSM data file)
b. the location of the config file
example of the command



```
shruti@shrutis-mbp ~ % python3 LocalCoordinateProcessor.py market-broadway_remoteBsmLog_07102021_131038.csv config_broadway.json
```

c. The config file needs to be created for each intersection separately and the config files are available here:
https://github.com/mmitss/portlandStreetcar/tree/issue12-push-to-psu/tools/map-bsm-processor/config

d. The details required for the config file are contained in this folder:
https://github.com/mmitss/portlandStreetcar/tree/master/config/corridor/simulation

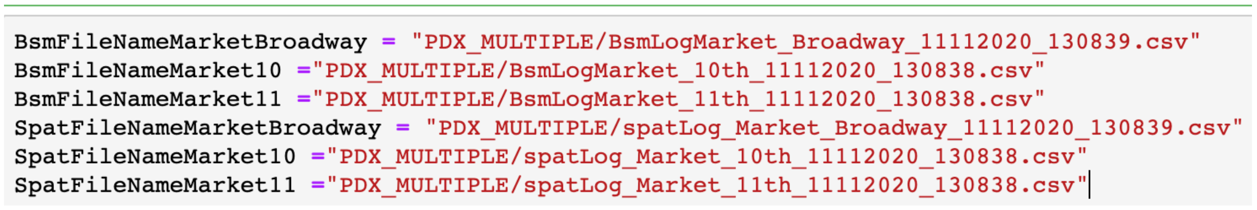
The example for the market broadway is in this json file… the details required for config file are used from this file to create the config file:
(https://github.com/mmitss/portlandStreetcar/blob/master/config/corridor/simulation/market-broadway/nojournal/bin/mmitss-phase3-master-config.json)
Sample of the config file for broadway ---{

```
   "IntersectionName": "market-broadway",

   "MapPayload": "copy this from the link mentioned above the from the json",

   "IntersectionID": 34108,

   "RegionalID": 0,

   "ReferenceLocation":

   {

       "Latitude_DecimalDegree":45.513363,

       "Longitude_DecimalDegree":-122.682865,

       "Elevation_Meter":124.7

   }

}
```

4. Run the visualization - the code is in the name of **StreetCar_Connected_Viz** which is available in the github link: https://github.com/mmitss/portlandStreetcar/tree/issue12-push-to-psu/tools/visualizations

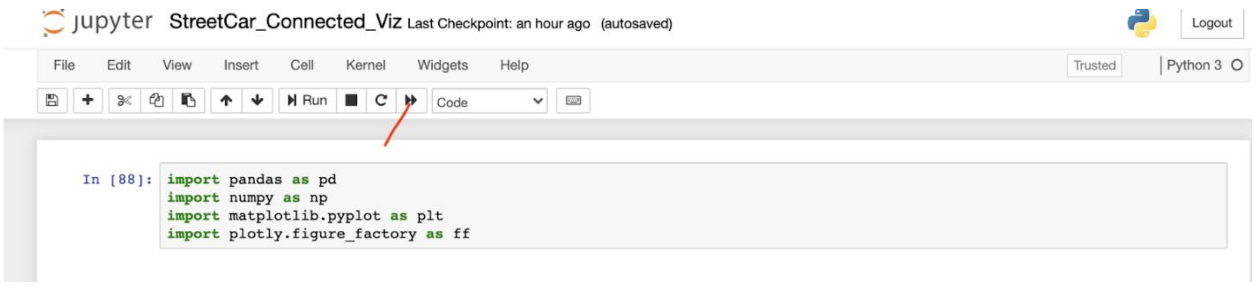
The details for how to run the visualizations -

- The visualization code is written in Jupyter Notebook and requires various libraries mentioned above to run the notebook.
- Once you are on the Jupyter notebook, open the code StreetCar_Connected_Viz.ipnyb that produces the visualizations.
- The strings that consist of the BSM and SPaT file names need to be replaced with the required filename for which the visualizations need to be run for. The filenames are string variables. Below is an example of the filenames for all three intersections. Replace the string with the location of the respective filename.



```
BsmFileNameMarketBroadway = "PDX_MULTIPLE/BsmLogMarket_Broadway_11112020_130839.csv"
BsmFileNameMarket10 ="PDX_MULTIPLE/BsmLogMarket_10th_11112020_130838.csv"
BsmFileNameMarket11 ="PDX_MULTIPLE/BsmLogMarket_11th_11112020_130838.csv"
SpatFileNameMarketBroadway = "PDX_MULTIPLE/spatLog_Market_Broadway_11112020_130839.csv"
SpatFileNameMarket10 ="PDX_MULTIPLE/spatLog_Market_10th_11112020_130838.csv"
SpatFileNameMarket11 ="PDX_MULTIPLE/spatLog_Market_11th_11112020_130838.csv"
```

Once the correct filenames are input the code can be run by clicking on the double arrows (marked in red in the image below) to run the whole code at once or click on the run button to run line by line.



```
In [88]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import plotly.figure_factory as ff
```

A detailed description of the code that generates the time-space plot is located in Appendix 7.1.

7.3 DATA SCHEMA CODEBOOK

BSM_Data

log_timestamp_verbose log_timestamp_posix timestamp_verbose timestamp_posix **BSM_ID - PK (sequence generator)** temporaryId secMark latitude longitude elevation local_x local_y local_z speed heading type -FK(Type) length	width position_on_map in_queue current_approach current_lane current_signal_group trajectory_signal_group dist_to_stopbar time_to_stopbar distance_along_path onmap_status

```
CREATE TABLE BSM_DATA
(
BSM_ID  serial PRIMARY KEY,
log_timestamp_verbose  TIMESTAMP,
log_timestamp_posix      TIMESTAMP,
timestamp_verbose     TIMESTAMP,
timestamp_posix         TIMESTAMP,
temporaryID   INT NOT NULL,
secMark INT,
latitude NUMERIC,
longitude NUMERIC,
elevation NUMERIC,
local_x NUMERIC,
local_y NUMERIC,
local_z NUMERIC,
speed NUMERIC,
heading NUMERIC,
type INT,
length NUMERIC,
width NUMERIC,
position_on_map VARCHAR,
in_queue BOOLEAN,
current_approach INT,
current_lane INT,
current_signal_group INT,
trajectory_signal_group INT,
dist_to_stopbar NUMERIC,
time_to_stopbar VARCHAR,
distance_along_path NUMERIC,
onmap_status  BOOLEAN,
FOREIGN KEY(temporaryId)
REFERENCES INTERSECTION_DETAILS(IntersectionId),
FOREIGN KEY(current_signal_group)
REFERENCES SIGNAL_GROUP(SignalGroupId),
FOREIGN KEY(trajectory_signal_group)
REFERENCES SIGNAL_GROUP(SignalGroupId),
FOREIGN KEY(type)
REFERENCES VEHICLE_TYPE(typeid)
);
```

Attribute_name	**Attribute_type**	**Example**
log_timestamp_verbose	timestamp(UTC)	2020-11-11 13:08:51
log_timestamp_posix	POSIX	1605100131
timestamp_verbose	timestamp(UTC)	2020-11-11 13:08:51
timestamp_posix	POSIX	1605100131
BSM_ID - PK (sequence generator)	Integer	1
temporaryId	Integer	1647
secMark	float	44000
latitude	float	45.51420673
longitude	float	-122.6860654
elevation	float	172.825
local_x	float	22.25612637
local_y	float	-52.29313448
local_z	float	0.000252888063
speed	float	5.741
heading	float	104.782
type	Integer	400
length	float	376.5
width	float	26.9
position_on_map	varchar	inbound
in_queue	boolean	FALSE
current_approach	Integer	3
current_lane	Integer	7
current_signal_group	Integer	8
trajectory_signal_group	Integer	4
dist_to_stopbar	float	46.78
time_to_stopbar	Varchar - has float numbers and true false	8.148406201
distance_along_path	float	2.30572991
onmap_status	boolean	TRUE

SPAT_Data

log_timestamp_verbose log_timestamp_posix timestamp_verbose timestamp_posix regionalId **intersectionId - FK(Intersection)** msgCount moy msom v1_currState v1_minEndTime v1_maxEndTime v1_elapsedTime v2_currState v2_minEndTime v2_maxEndTime v2_elapsedTime v3_currState v3_minEndTime v3_maxEndTime v3_elapsedTime v4_currState v4_minEndTime v4_maxEndTime v4_elapsedTime	v5_currState v5_minEndTime v5_maxEndTime v5_elapsedTime v6_currState v6_minEndTime v6_maxEndTime v6_elapsedTime v7_currState v7_minEndTime v7_maxEndTime v7_elapsedTime v8_currState v8_minEndTime v8_maxEndTime v8_elapsedTime p1_currState p1_minEndTime p1_maxEndTime p1_elapsedTime p2_currState p2_minEndTime p2_maxEndTime p2_elapsedTime	p3_currState p3_minEndTime p3_maxEndTime p3_elapsedTime p4_currState p4_minEndTime p4_maxEndTime p4_elapsedTime p5_currState p5_minEndTime p5_maxEndTime p5_elapsedTime p6_currState p6_minEndTime p6_maxEndTime p6_elapsedTime p7_currState p7_minEndTime p7_maxEndTime p7_elapsedTime p8_currState p8_minEndTime p8_maxEndTime p8_elapsedTime

CREATE TABLE SPAT_DATA
(

```
log_timestamp_verbose TIMESTAMP,
log_timestamp_posix  VARCHAR,
timestamp_verbose    TIMESTAMP,
timestamp_posix      VARCHAR,
regionalId           INT,
intersectionId       INT,
msgCount             INT,
moy                  INT,
msom                 INT,
v1_currState         VARCHAR,
v1_minEndTime        INT,
v1_maxEndTime        INT,
v1_elapsedTime       INT,
v2_currState         VARCHAR,
v2_minEndTime        INT,
v2_maxEndTime        INT,
v2_elapsedTime       INT,
v3_currState         VARCHAR,
v3_minEndTime       INT,
v3_maxEndTime       INT,
v3_elapsedTime      INT,
v4_currState        VARCHAR,
v4_minEndTime       INT,
v4_maxEndTime       INT,
v4_elapsedTime      INT,
v5_currState        VARCHAR,
v5_minEndTime       INT,
v5_maxEndTime       INT,
v5_elapsedTime      INT,
v6_currState        VARCHAR,
v6_minEndTime       INT,
v6_maxEndTime       INT,
v6_elapsedTime      INT,
v7_currState        VARCHAR,
v7_minEndTime       INT,
v7_maxEndTime       INT,
v7_elapsedTime      INT,
v8_currState        VARCHAR,
v8_minEndTime       INT,
v8_maxEndTime       INT,
v8_elapsedTime      INT,
p1_currState        VARCHAR,
p1_minEndTime       INT,
p1_maxEndTime       INT,
p1_elapsedTime      INT,
p2_currState        VARCHAR,
p2_minEndTime       INT,
p2_maxEndTime       INT,
p2_elapsedTime      INT,
p3_currState        VARCHAR,
p3_minEndTime       INT,
p3_maxEndTime       INT,
p3_elapsedTime      INT,
p4_currState        VARCHAR,
p4_minEndTime       INT,
p4_maxEndTime       INT,
p4_elapsedTime      INT,
p5_currState        VARCHAR,
p5_minEndTime       INT,
p5_maxEndTime       INT,
p5_elapsedTime      INT,
p6_currState        VARCHAR,
p6_minEndTime       INT,
p6_maxEndTime       INT,
p6_elapsedTime      INT,
p7_currState        VARCHAR,
p7_minEndTime       INT,
p7_maxEndTime       INT,
p7_elapsedTime      INT,
p8_currState            VARCHAR,
p8_minEndTime           INT,
p8_maxEndTime           INT,
p8_elapsedTime          INT
FOREIGN KEY(intersectionId)
REFERENCES INTERSECTION_DETAILS(IntersectionId));
```

Attribute_name	**Attribute_type**	**Example**
log_timestamp_verbose	timestamp	2020-11-11 13:08:39
log_timestamp_posix	posix	1605100119
timestamp_verbose	timestamp	2020-11-11 13:08:39
timestamp_posix	posix	1605100119
regionalId	integer	0
intersectionId	integer	35925
msgCount	integer	1
moy	integer	454388
msom	integer	38843
v1_currState	varchar	0
v1_minEndTime	integer	0
v1_maxEndTime	integer	0
v1_elapsedTime	integer	16051001188
v2_currState	varchar	green
v2_minEndTime	integer	53
v2_maxEndTime	integer	222
v2_elapsedTime	integer	0
v3_currState	varchar	red
v3_minEndTime	integer	0
v3_maxEndTime	integer	0
v3_elapsedTime	integer	0
v4_currState	varchar	red
v4_minEndTime	integer	262
v4_maxEndTime	integer	262
v4_elapsedTime	integer	0
v5_currState	varchar	0
v5_minEndTime	integer	0
v5_maxEndTime	integer	0
v5_elapsedTime	integer	16051001188
v6_currState	varchar	0
v6_minEndTime	integer	0
v6_maxEndTime	integer	0
v6_elapsedTime	integer	16051001188
v7_currState	varchar	0
v7_minEndTime	integer	0
v7_maxEndTime	integer	0
v7_elapsedTime	integer	16051001188
v8_currState	varchar	red
v8_minEndTime	integer	662
v8_maxEndTime	integer	662

Attribute_name	**Attribute_type**	**Example**
v8_elapsedTime	integer	0
p1_currState	varchar	0
p1_minEndTime	integer	0
p1_maxEndTime	integer	0
p1_elapsedTime	integer	16051001188
p2_currState	varchar	walk
p2_minEndTime	integer	53
p2_maxEndTime	integer	53
p2_elapsedTime	integer	0
p3_currState	integer	0
p3_minEndTime	integer	0
p3_maxEndTime	integer	0
p3_elapsedTime	integer	16051001188
p4_currState	varchar	do_not_walk
p4_minEndTime	integer	0
p4_maxEndTime	integer	0
p4_elapsedTime	integer	0
p5_currState	varchar	0
p5_minEndTime	integer	0
p5_maxEndTime	integer	0
p5_elapsedTime	integer	16051001188
p6_currState	varchar	0
p6_minEndTime	integer	0
p6_maxEndTime	integer	0
p6_elapsedTime	integer	16051001188
p7_currState	varchar	0
p7_minEndTime	integer	0
p7_maxEndTime	integer	0
p7_elapsedTime	integer	16051001188
p8_currState	varchar	0
p8_minEndTime	integer	0
p8_maxEndTime	integer	0
p8_elapsedTime	integer	16051001188

Intersection_Details

IntersectionID -PK
Longitude
Latitude
Description

```
CREATE TABLE INTERSECTION_DETAILS
(
        IntersectionId  INT PRIMARY KEY,
        Longitude  NUMERIC,
        Latitude NUMERIC,
        Description VARCHAR
        );
```

Attribute_name	Attribute_type	Example
IntersectionID -PK	integer	24
Longitude	float	45.23455
Latitude	float	46.2222
Description	varchar	market&10th

Phase_Details

PhaseId -PK IntersectionId -FK(Intersection) Description

```
CREATE TABLE PHASE_DETAILS
(
        PhaseId INT PRIMARY KEY,
        IntersectionId INT,
        DESCRIPTION VARCHAR,
        FOREIGN KEY(IntersectionId)
        REFERENCES INTERSECTION_DETAILS(IntersectionId)
);
```

Attribute_name	Attribute_type	Example
PhaseId -PK	integer	1
IntersectionId -FK(Intersection)	integer	24
Description	varchar	

Signal_Group

SignalGroupID- PK PhaseID- FK Description

```
CREATE TABLE SIGNAL_GROUP
(
        SignalGroupId INT PRIMARY KEY,
        PhaseId INT,
        Description VARCHAR,
        FOREIGN KEY(PhaseId)
        REFERENCES PHASE_DETAILS(PhaseId)
);
```

Attribute_name	**Attribute_type**	**Example**
SignalGroupID-PK	integer	1
PhaseID- FK	integer	1
Description	varchar	

Type(VehicleType)

TypeId - PK
Description