FHWA-NJ-2022-002

REAL-TIME TRAFFIC SIGNAL SYSTEM PERFORMANCE MEASUREMENT
PHASE II: DATA AND FUNCTIONALITY ENHANCEMENT, LARGE SCALE DEPLOYMENT, CONNECTED AND AUTONOMOUS VEHICLES INTEGRATION
FINAL REPORT
VOLUME II

May 2022

Submitted by

Peter J. Jin, Ph.D.
Associate Professor
Department of Civil and Environmental Engineering
Rutgers, The State University of New Jersey

Tianya Zhang,
Graduate Research Assistant,
Department of Civil and Environmental Engineering
Rutgers, The State University of New Jersey

Thomas M. Brennan JR., Ph.D., P.E.
Professor
Department of Civil Engineering
The College of New Jersey

Mohammad Jalayer, Ph.D.
Associate Professor
Department of Civil and Environmental Engineering
Rowan University

NJDOT Research Project Manager –
Priscilla Ukpah

In cooperation with
New Jersey Department of Transportation
Bureau of Research
And
U.S. Department of Transportation
Federal Highway Administration

DISCLAIMER STATEMENT

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the New Jersey Department of Transportation (NJDOT) or the Federal Highway Administration (FHWA). This report does not constitute a standard, specification, or regulation.

ACKNOWLEDGMENTS

This research was supported by the New Jersey Department of Transportation (NJDOT). We are grateful to the Research Selection and Implementation Panel members for giving us the opportunity to undertake this important research. We thank Project Manager Ms. Priscilla Ukpah of the NJDOT Bureau of Research, and Ms. Amanda Gendek, manager of Bureau, for their consistent support and advice. We are grateful to Ekaraj Phomsavath and Eddie Curtis from FHWA for providing insights and materials. We are grateful to Mr. Kelly McVeigh, Mohamed Elhefnawi, LaDanya Friday, Sanjaykumar Patel, Mark Renner, Virginia Todd and Katelin Barone from NJDOT Cherry Hill Office and Wasif Mirza, Director of the Division of Mobility and System for providing insights, feedback, and case study materials. We are grateful to all the members of the stakeholder panel, Tom Brennan from The College of New Jersey (TCNJ), Mohammad Jalayer and Deep Patel from Rowan University, Amy Lopez, Rick Schuman and Mike Massaro from INRIX, Robert Meyer from Transcore, Jason Simmons and Katie Elliott from SJTPO, Mark Taylor (Derrick) from UDOT, Richasrd Cippoletti from North Jersey Transportation Planning Authority (NJTPA), Allen Davis from Georigia DOT, Steve Remias from Wayne State University, and Christopher M. Day from Iowa State University, who volunteered to participate in some meetings, contributed to tasks, shared their experiences, and provided insights and on pertinent issues.

TABLE OF CONTENT

LIST OF FIGURES

EXCLUSIVE SUMMARY

Volume II provides supplementary material for Volume I. The content includes: the operational manual for ATSPM (Automated Traffic Signal Performance Measures) metadata processing and archiving program, training modules and field test data for CV (Connected Vehicle) OBU (On-Board Unit) and RSU (Roadside Unit), and details of the meeting discussion and participant feedbacks in the stakeholder meeting.

EVENT TRANSLATOR METADATA FILE PROCESSING GUIDE

Step 1: Looking from the intersection timing Info .pdf to find the min green, clearance, change, and max green time for each phase.

1103104d

Directive No. 185-20

Route US 1 and Bakers Basin Road/
Franklin Corner Road
Lawrence Twp., Mercer Co.

PHASE	SIGNAL INDICATIONS 1-6	7-8	9	10-11	12	13,14 16,17	15,18	19,20	NORMAL OPERATION Adaptive Mode***	TIME (SECONDS) BACK UP TOD PLANS PLAN I (127-179)	PLAN II (99-143)	PLAN III (122-176)	PLAN IV (105-145)	PLAN V (120-140)	PLAN X (140-180)
A) Route US 1 ROW	G	R	R	R	R	W	DW	DW	7 (min)	68	40	63	46	61	81
Pedestrian Clearance	G	R	R	R	R	FDW	DW	DW	22	22	22	22	22	22	22
Change	Y	R	R	R	R	DW	DW	DW	6	6	6	6	6	6	6
Clearance	R	R	R	R	R	DW	DW	DW	2	2	2	2	2	2	2
B) Bakers Basin Road WB ROW	R	R	R	G/<G-	G	DW	DW	DW	7 (min)	7-32	7-28	7-34	7-26	7-16	7-27
Change	R	R	R	Y	Y	DW	DW	DW	4	4	4	4	4	4	4
Clearance	R	R	R	R	R	DW	DW	DW	4	4	4	4	4	4	4
C) Franklin Corner Road EB ROW	R	G/<G-	G	R	R	DW	DW	DW	7 (min)	7-34	7-30	7-34	7-28	7-18	7-27
Change	R	Y	Y	R	R	DW	DW	DW	4	4	4	4	4	4	4
Clearance	R	R	R	R	R	DW	DW	DW	3	3	3	3	3	3	3
Emergency Flash	Y	R	R	R	R	DARK	DARK	DARK		-	-	-	-	-	-

intersectionID	phaseID	phaseLetter	Movements_Phase	minGreen	maxGreen	planID	planStartTime	planEndTime	PlanDOW	Change	Clearance	PlanType
10012	1	A	NT,ST(2,6)	7	81	None	None	None	1-2-3-4-5-6-7	6	2	Adaptive
10012	2	B	WLTR(3)	7	34	None	None	None	1-2-3-4-5-6-7	4	3	Adaptive
10012	3	C	ELTR(4)	7	34	None	None	None	1-2-3-4-5-6-7	4	3	Adaptive

Step 2: Find the Phase Letter for each Signal Group number from SCATS SG to Phase Table (Rt 1 SCATS SG to Phase[28556].xlsx). In this example, Phase A corresponds to SG 2&6 and Phase B corresponds to SG 3, and Phase C corresponds to SG 4.

Row Labels
10012 A100
10012: SG 2 Dets 1-2
10012: SG 2 Dets 17-18
10012: SG 6 Dets 21-23
10012: SG 6 Dets 6-8
10012 B100
10012: SG 3 Dets 9-11
10012 C100
10012: SG 4 Dets 3-5

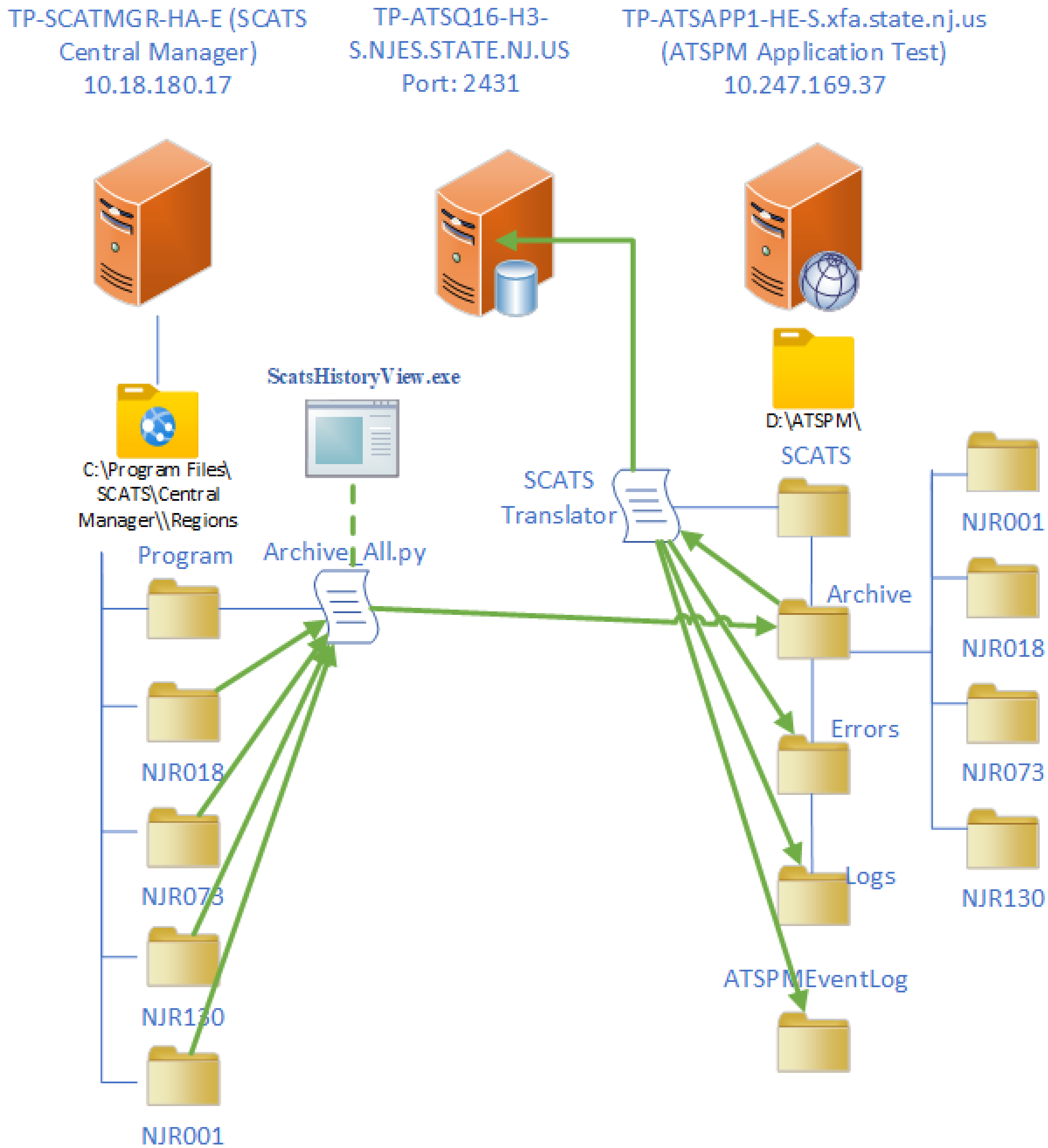
Step 3: From Signal Phasing Table (Rt. 1 SCATS Phasing.xlsx), Find the Direction for each Signal Group Number

intersectionID	phaseID	phaseLetter	Movements_Phase	minGreen	maxGreen	planID	planStartTime	planEndTime	PlanDOW	Change	Clearance	PlanType
10012	1	A	NTR,STR(2,6)	7	81	None	None	None	1-2-3-4-5-6-7	6	2	Adaptive
10012	2	B	WLTR(3)	7	34	None	None	None	1-2-3-4-5-6-7	4	3	Adaptive
10012	3	C	ELTR(4)	7	34	None	None	None	1-2-3-4-5-6-7	4	3	Adaptive

			Movement Phase Number											
Mainline	Cross Street	Signal ID	SBL	SBT	SBR	EBL	EBT	EBR	NBL	NBT	NBR	WBL	WBT	WBR
	Bakers Basin	10012		2	2	4	4	4		6	6	3	3	3
	Carnegie Ctr	10015		6	6	4	4	4		2	2	3	3	3
	Washington	10017		6	6	8	8	8		2	2	4	4	4
	Fisher	10018		2	2	8	8	8		6	6	4	4	4
	Harrison	10019		6	6	4	4	4		2	2	3	3	3
	Independence	10020		6	6	7	4	4		2	2	3		3
	Ridge	10021		6	6	3	8	8		2	2	7	4	4
	Raymond	10022		2	2	8		8		6	6	4	4	4
	Promenade	10023		6	6	3	8	8		2	2	7	4	4
	Whispering Woods	10024		6	6	3	3	3		2	2	4	4	4
	New	10025		2	2	3	8	8		6	6	7	4	4

For this example, STR represents Southbound Through and Right Turn Traffic, signal group 2, and NTR represents northbound through and right turn traffic, signal group 6.

ATSPM HISTORICAL DATA ARCHIVING GUIDE



TRAINING MODULE 1

NJDOT ATSPM (Automated Traffic Signal Performance Measures) Training

- SCATS and InSync Deployment

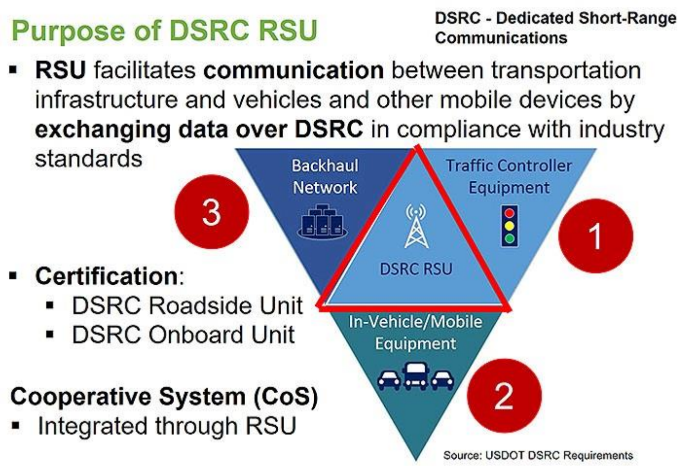
September 2020

Outline

- **CV101**
- **CV Applications**
- **NJ CV ConOps**
- **CV Instruments**
- **Security Certificates**
- **Training Module Overview**

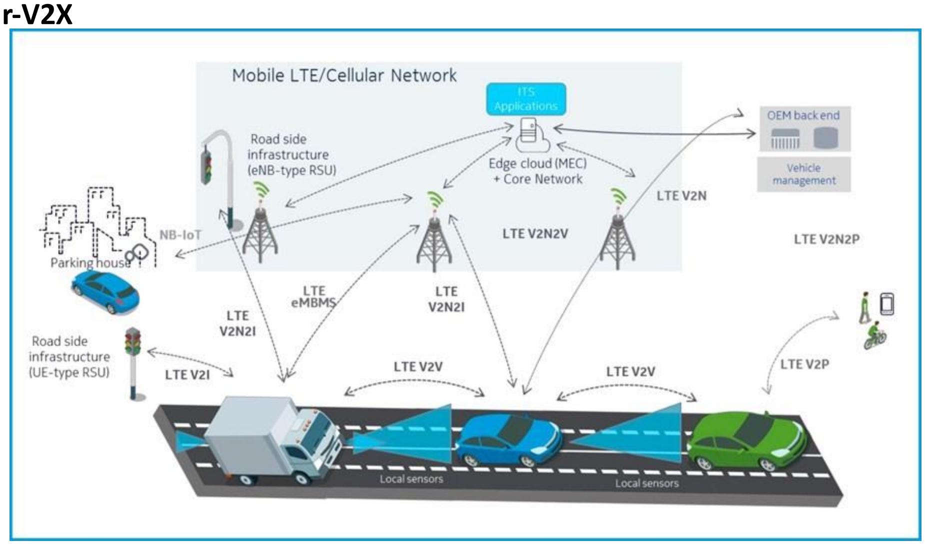
CV101 -Technologies

Dedicated Short-Range Communications (DSRC):



CV101- Technologies

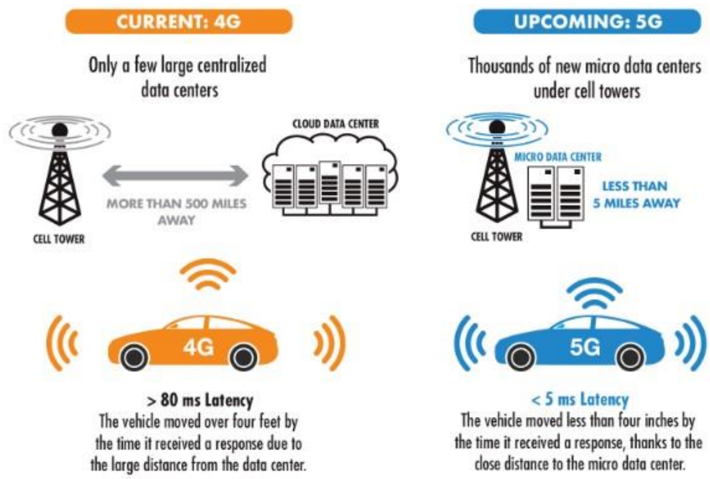
Cellular-V2X



Source Guy Daniels. NGMN Alliance selects C -V2X technology for the connected car.
https://www.telecomtv.com/content/automotive/ngmn -alliance-selects-c-v2x-technology-for-the-connected-car-31854/

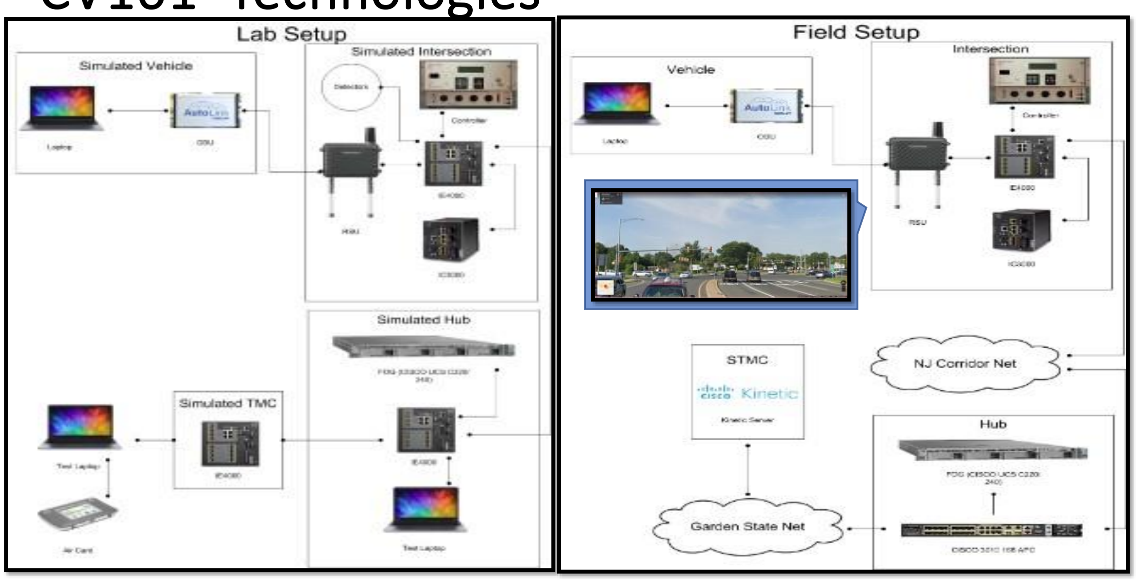
CV101- Technologies

5G



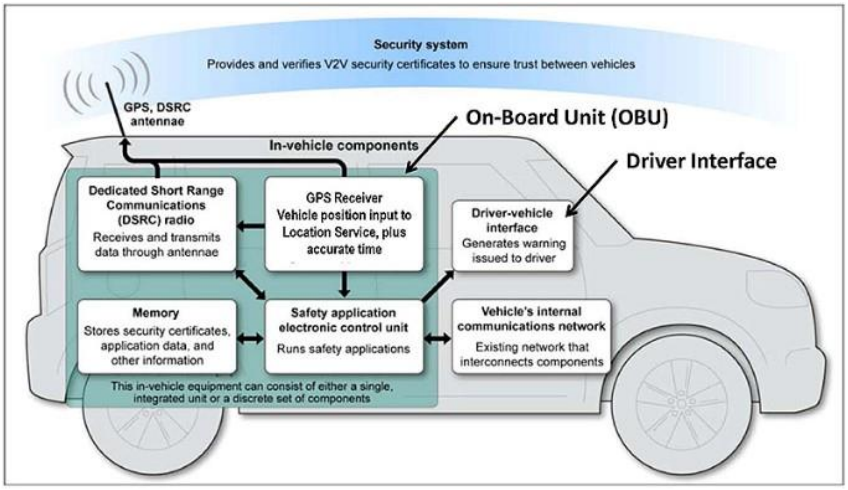
Source: https://www.connectorsupplier.com/how -the-rise-of-edge-computing-will-reshape-the-data-center-landscape/

CV101 -Technologies



CV101 -Terms

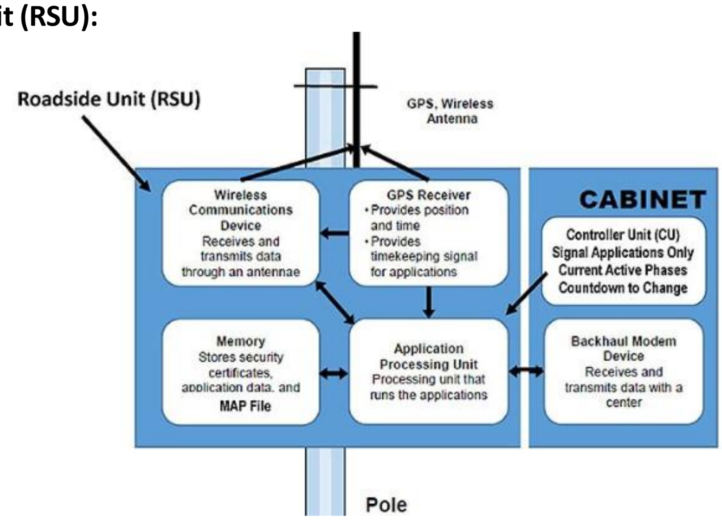
On-board Unit (OBU):



Source: USDOT, CV T160: Connected Vehicle Certification Testing Introduction.
https://www.pcb.its.dot.gov/StandardsTraining/mod57/ppt/m57ppt.htm

CV101- Terms

Roadside Unit (RSU):



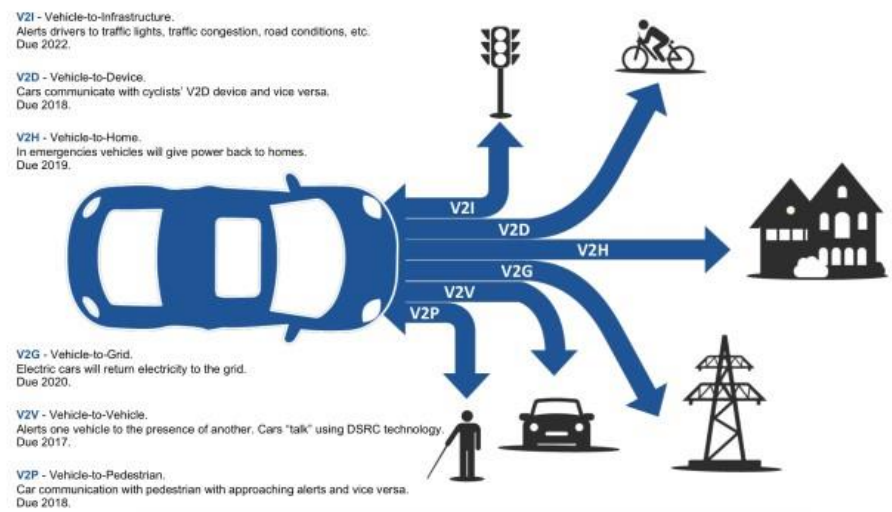
Source: USDOT, CV T160: Connected Vehicle Certification Testing Introduction.
https://www.pcb.its.dot.gov/StandardsTraining/mod57/ppt/m57ppt.htm

CV101- Terms

Signal Phase and Timing (SPaT):

Source: https://yhfeng.weebly.com/research.html

CV Applications – V2V, V2I, V2X



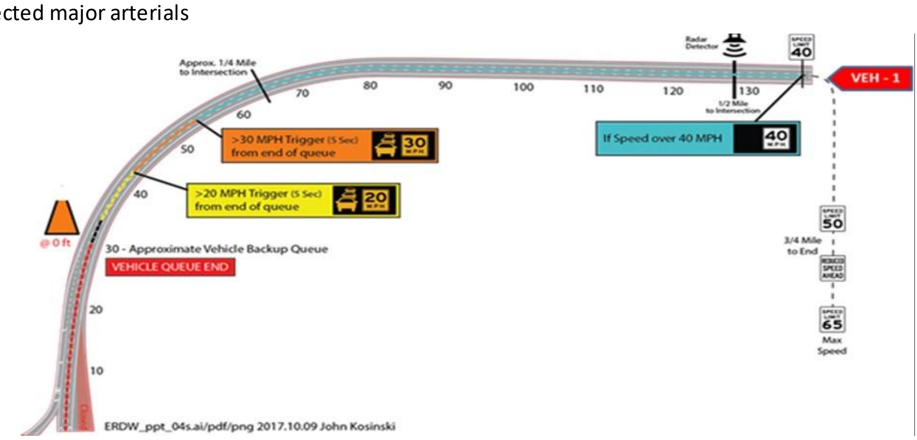
Source: https://medium.com/datadriveninvestor/is- 5g-friend-or-foe-for-autonomous-vehicle-72ee70800031

CV Applications – Mobility

TRANSIT SIGNAL PRIORITY AND FREIGHT SIGNAL PRIORITY (TSP/FSP):
Provides signal priority to buses and freight vehicles if current roadway operations meet predefined criteria that would result in priority being given

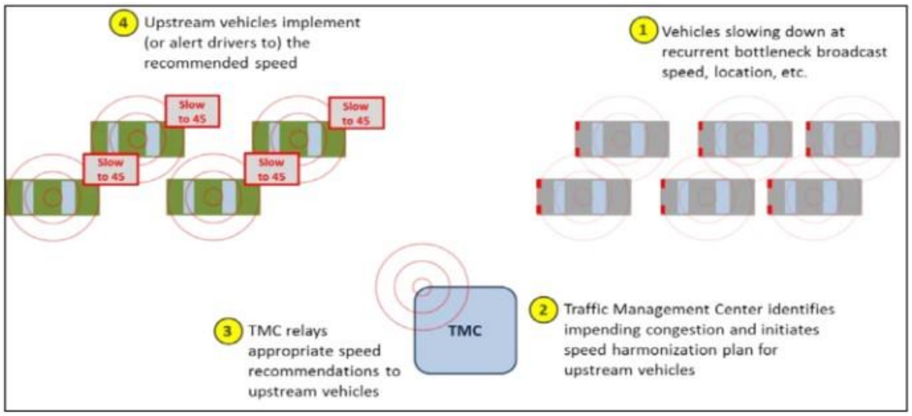
CV Applications – Mobility

QUEUE WARNING (Q-WARN):
Advises the driver with advance notification of an impending queue along freeway, expressways or selected major arterials



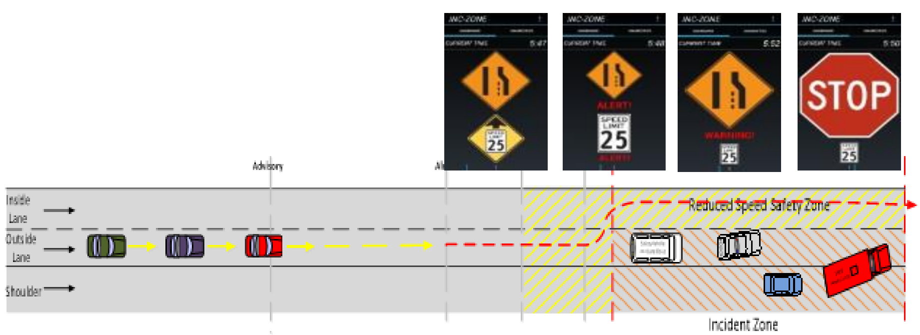
CV Applications – Mobility

DYNAMIC SPEED HARMONIZATION (SPD -HARM):
Allows for vehicle platoons within a roadway network to improve throughput and harmonize speed along arterials and freeways, maximizing the green band



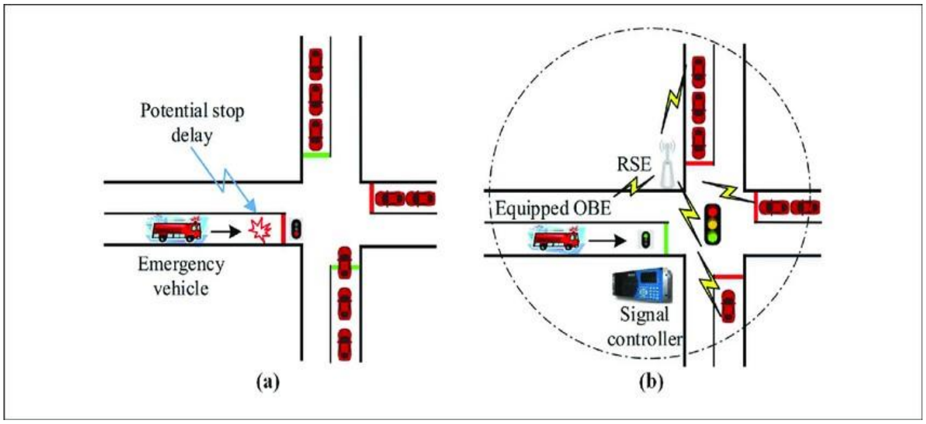
CV Applications – Mobility

INCIDENT SCENE WORK ZONE ALERTS FOR DRIVERS AND WORKERS (INC -ZONE):
- A communication approach that will improve protection of responders at incident sites where there have been events impacting traffic
- Near-term: driver gets alert they are approaching incident
- Long-term: responder gets alert a vehicle is approaching a closed lane or at a high rate of speed



CV Applications– Mobility

EMERGENCY VEHICLE PREEMPTION (PREEMPT):
Provides signal preemption to emergency vehicles and accommodates multiple emergency requests



CV Applications– Mobility

INCIDENT SCENE PRE -ARRIVAL STAGING GUIDANCE FOR EMERGENCY RESPONDERS (RESP -STG):

- Provides situational awareness
- Coordination among emergency responders upon dispatch
- Coordinates if circumstances require additional dispatch and staging
- Data includes:
  - Staging Plans
  - Satellite Imagery
  - GIS map graphics
  - Camera images
  - Current weather data
  - Traffic conditions
  - Dynamic Routing Guidance

CV Applications – Mobility

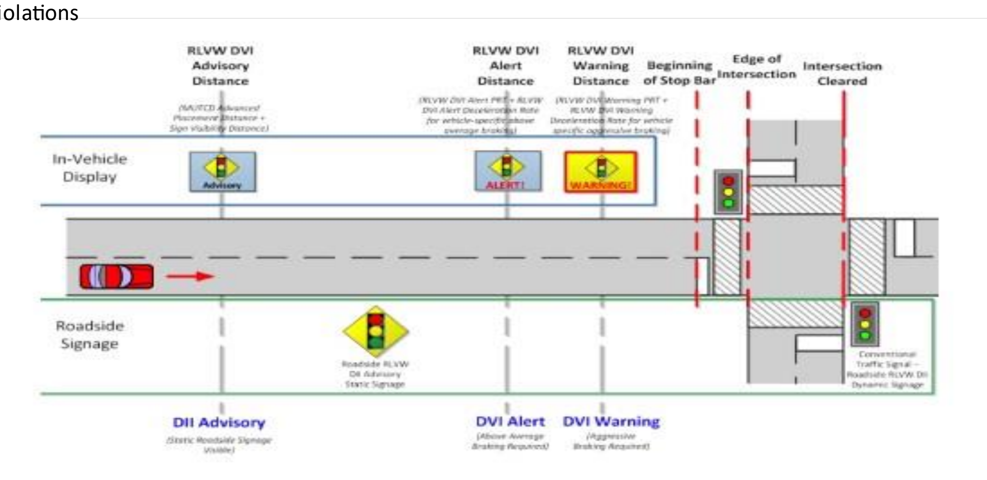
EMERGENCY COMMUNICATIONS AND EVACUATION (EVAC):

- Transmits location-specific directions for evacuation, location restrictions for entry, global emergency information, and route-specific information
- Messages can be reviewed by the TOC before being sent to the RSUs and broadcasted to the OBUs

CV Applications – Safety

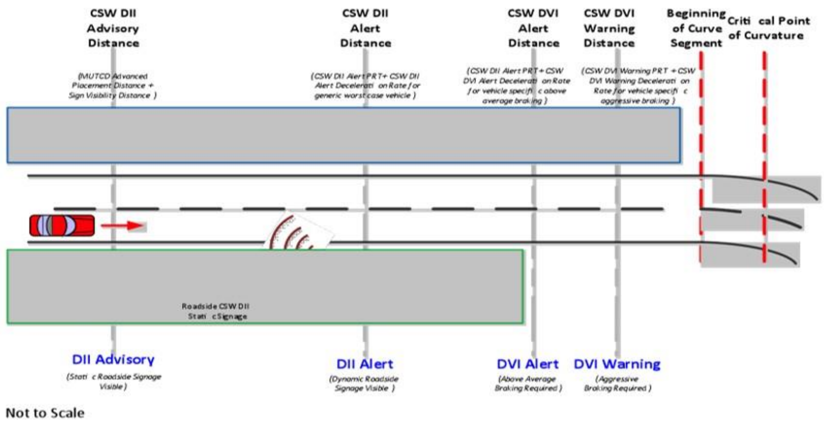
RED LIGHT VIOLATION WARNING (RLVW) :

Broadcasts SPaT and other data to the in-vehicle device, allowing warnings for impending red-light violations



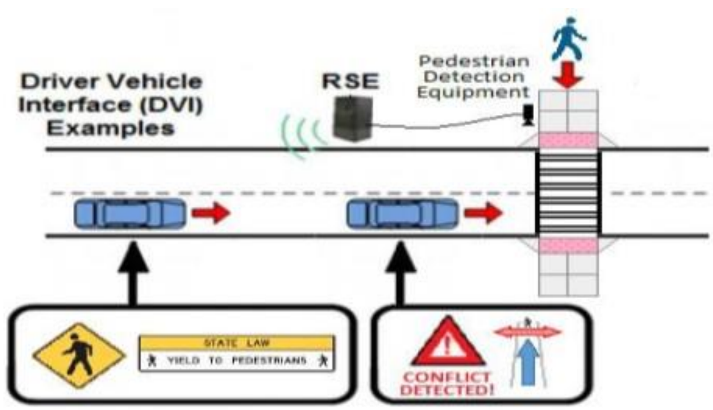
CV Applications– Safety

CURVE SPEED WARNING (CSW):
Provides an alert to the driver that they are approaching a curve at an unsafe speed



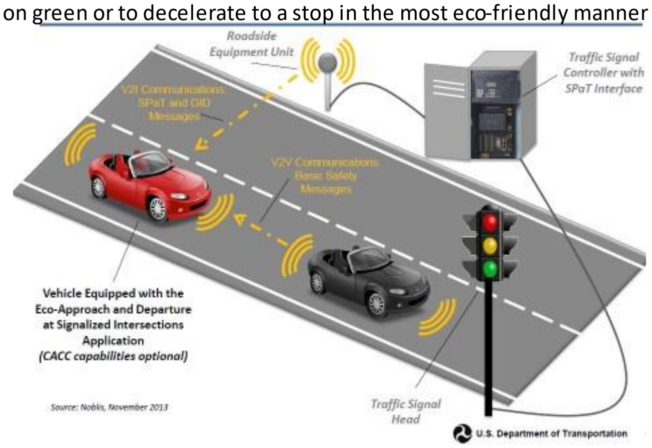
CV Applications– Safety

PEDESTRIAN IN SIGNALIZED CROSSWALK WARNING (PCW):
Capable of providing warnings to a vehicular driver that pedestrians within a crosswalk at a signalized intersection are in the intended path of the vehicle



CV Applications – Environmental

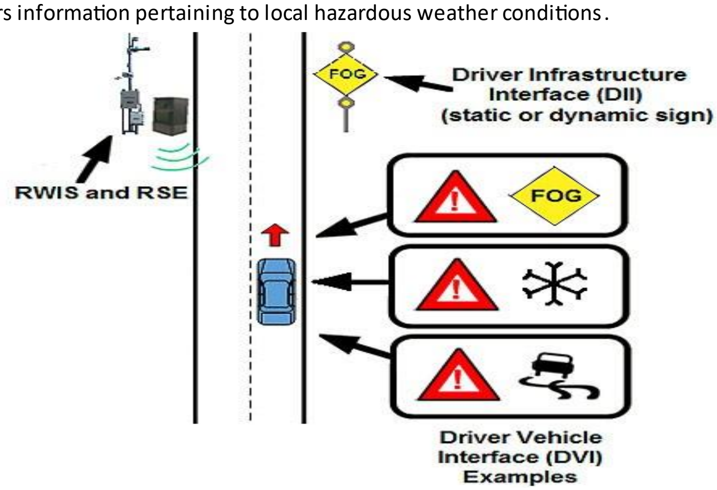
ECO APPROACH AND DEPARTURE AT SIGNALIZED INTERSECTIONS:
use connected vehicle technology to provide speed advice to drivers to adapt the vehicle's speed to pass the next traffic signal on green or to decelerate to a stop in the most eco-friendly manner



Resource: Image from JPO presentation http://www.its.dot.gov/research_archives/aeris/pdf/UCR_eco-approach_final2.pdf

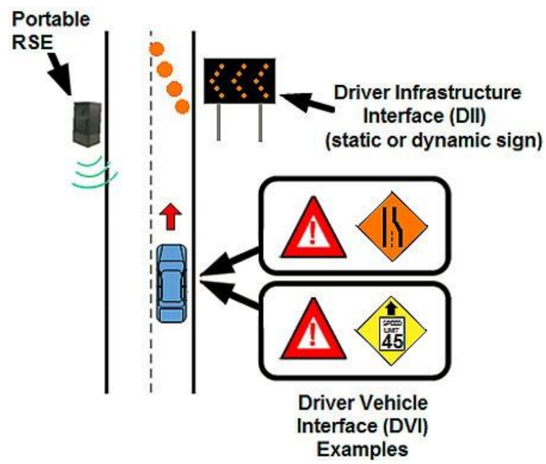
CV Applications – Road Weather

SPOT WEATHER IMPACT WARNING (SWIW):
Provides drivers information pertaining to local hazardous weather conditions.

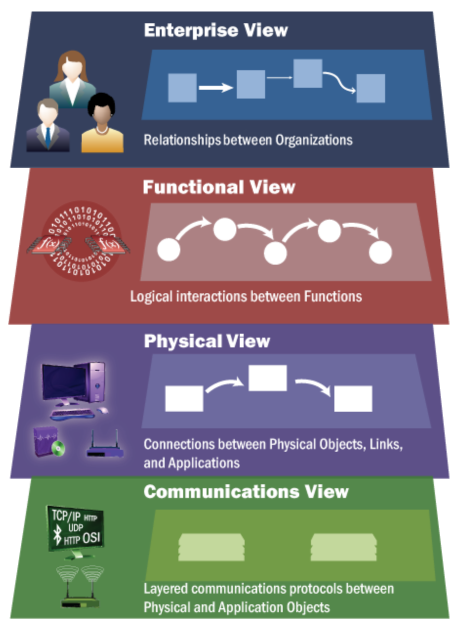


CV Applications – Work Zone

REDUCED SPEED/WORK ZONE WARNING (RSWZ) :
Provides an alert to drivers to reduce speed when approaching a work zone



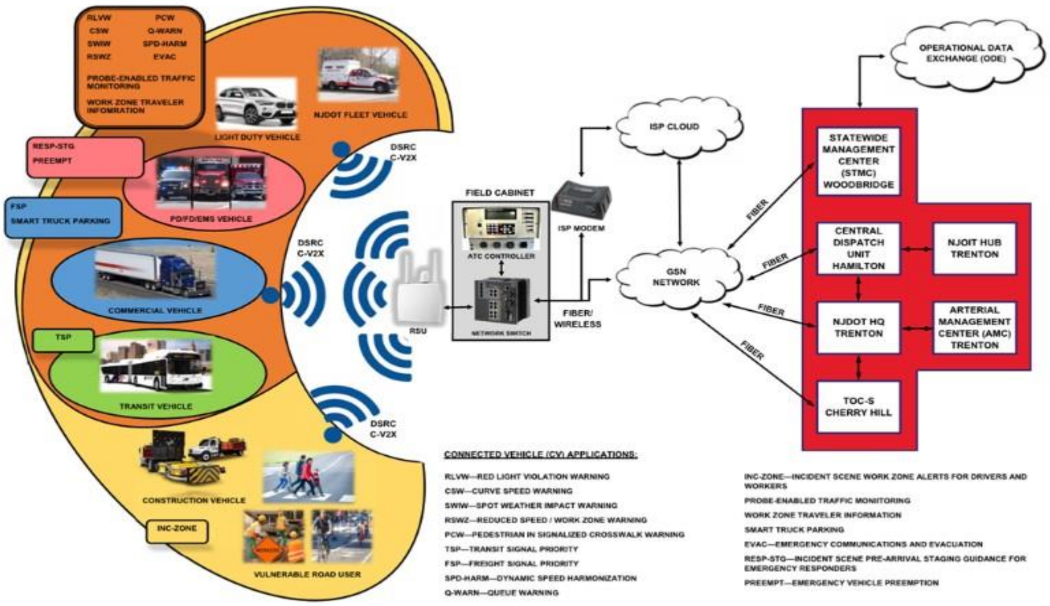
NJ CV ConOps – CVRIA



Connected Vehicle Reference Implementation Architecture (CVRIA)

- Enterprise - relationships between organizations
- Functional - abstract functional elements and their logical interactions
- Physical - physical objects and their application objects as well as the high -level interfaces
- Communications - layered sets of communications protocols

NJ CV ConOps CV Framework and App Diagrams



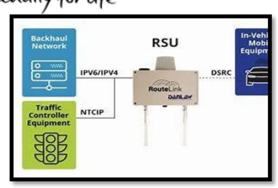
CV Instruments RSU

- **OmniAir Certified Device/Vendor List**

Vendor	Model
Intersect	Intersect-ECO/ ISECT-ECO-A RSU RSU
Kapsch TrafficCom AG	ROADSIDE ITS STATION / RIS-9160
Kapsch TrafficCom AG	ROADSIDE ITS STATION / RIS-9260-QAE(X)
Danlaw Inc.	ROUTELINK DSRC ROADSIDE UNIT / RSU001
Siemens	SITRAFFIC ESCOS ROADSIDE UNIT RSU

SIEMENS
Ingenuity for life

kapsch



DANLAW

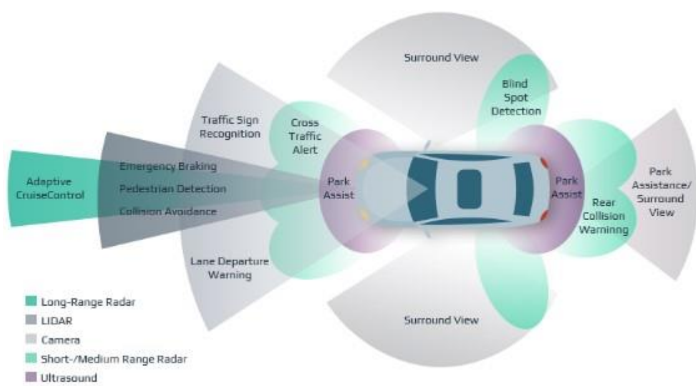
intersect

CV Instruments OBU

Vendor	Model
SiriusXM	Auriga (PN SX-7800-0357)
Danlaw Inc.	AUTOLINK ASD100 OBU
Lear Corporation	LOCOMATE ROADSTAR OBU
Savari, Inc.	MOBIWAVE MW1000
Commsignia	ITS-OB4 OBU VERSION: V1.17.45 B186782

CV Instruments Sensors

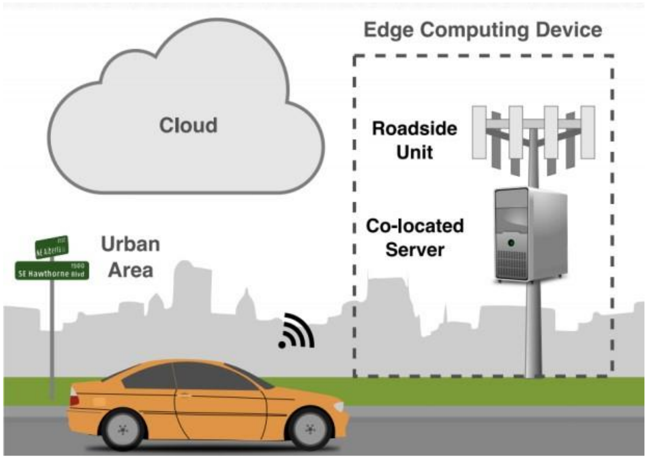
- **Ultrasound:** to detect obstacles in the immediate vicinity
- **GPS:** to calculate longitude, latitude, speed, and course
- **Radar:** uses radio waves to detect objects and is less expensive than LiDAR
- **LiDAR:** transmits optical laser light in pulses to determine distance to obstacles
- **Camera:** to detect, classify and determine the distance from objects



Source: Intellias. How Sensor Fusion for Autonomous Cars Helps Avoid Deaths on the Road.
https://www.intellias.com/sensor -fusion-autonomous-cars-helps-avoid-deaths-road/

CV Instruments- Edge Computing

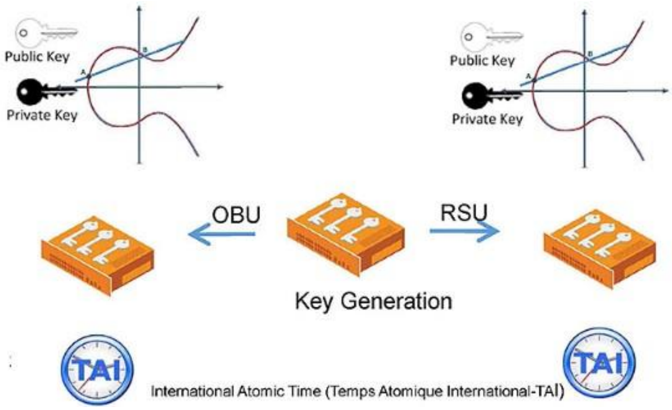
- **Edge Computing** pushes computing applications, data, and services away from centralized nodes to network edges, enabling analytics and knowledge generation to occur close to data sources;



Source: Yu Huang. Edge Computing for V2X and Autonomous Driving.
https://www.slideshare.net/yuhuang/edge -computing-for-v2x-and-autonomous-driving?from_action=save

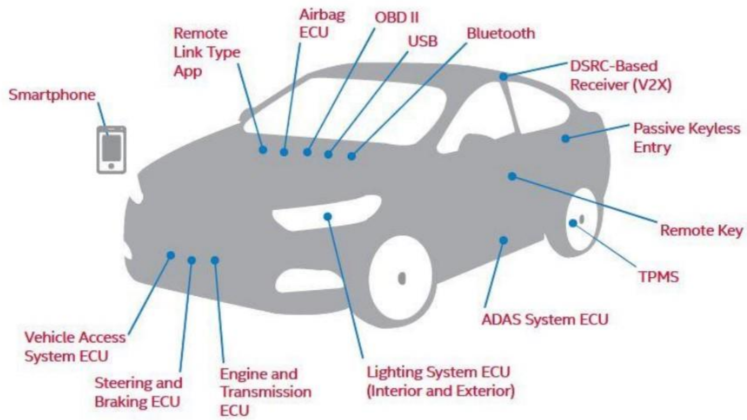
Cyber Security- Security Certificate Applications

Security Credential Management System (SCMS):
provides DSRC devices with digital certificates that the devices use to sign (authenticate) and encrypt DSRC messages and revokes certificates



Cyber Security - Potential Attack Gateways

- Electrical Control Units (ECUs)
- Airbag, Advanced Driver Assistant System, Engine, Steering & Brakes, etc.
- On-Board Diagnostics (OBD) II Diagnostic Port
- Dedicated Short-Range Communications-Based Receiver
- USB Ports
- Passive Keyless Entry/ Remote Key
- Remote Link Type App
- Tire Pressure Monitoring System (TPMS)



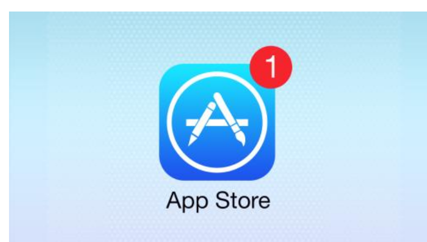
Source: Telematics Wire: Cybersecurity – A Gating Issue for Safety in A Connected and Automated Vehicle Future.
https://www.mcca.com/wp-content/uploads/2018/04/Autonomous-Vehicles.pdf

Cyber Security - Common Security Vulnerabilities

- **Software Glitches** – Connected vehicles today contain more than 100 million lines of code. More code means more opportunity for bugs and mistakes

- **No Single Source of Knowledge of or Control Over Source Code** – Software for different components of connected vehicles is being written by different developers, installed by different supplies, and no one source has knowledge of or control over the source code.

- **Increase Use of Apps Leave Vulnerabilities** – Consumers are using an increasing number of smartphone apps to interface with their connected cars. Some of these apps are Likely to see spread in use of malware.



- **Need for Constant Updates May be Overlooked** – There is a risk these updates could be overlooked or that malicious actors could infect routine updates.

Source: Telematics Wire: Cybersecurity – A Gating Issue for Safety in A Connected and Automated Vehicle Future.
https://www.mcca.com/wp-content/uploads/2018/04/Autonomous-Vehicles.pdf

Cyber Security – Sample Attacks

- **Denial of Service (DoS)**
  - Interruption targeted at crashing system
  - Distributed DoS- overwhelm bandwidth
- **Malware**
  - Malicious inserted software
  - Gain access & gather information
- **Ransomware**
  - Type of Malware
  - Blocks access to data

Source: CNN. Jeep remotely carjacked, shut down on highway.
https://www.cnn.com/videos/tv/2015/07/23/exp_-chrysler-hack-simon-dnt-erin.cnn

Cyber Security – Impacts to Transportation Operations

- **Legacy Systems**
  - Can interact with other network devices
  - Has vulnerability due to lack of patching
- **Brute Force**
  - 1st step in gaining access
  - Combined to deliver malware / ransomware
- **Physical Vulnerabilities**
  - Unlocked/Exposed ITS infrastructure
  - Readily available Vendor & detailed product information

Cyber Security – Countermeasures

- **Provide Multi-layered protection** – Beginning at level of individual ECUs, moving up a level to include software to protect vehicle's internal network.
- **Defend against externally-facing potential gateways** – Ensure weakest links in car's security are viewed as potential threats and defenses are built into system.
- **Ensure vendors and suppliers have strong security** – It is critical to review and monitor vendor and supplier policies and practices.
- **Promote timely updates** – Companies should push timely and effective fixes as soon as problems are identified.

Source: Telematics Wire: Cybersecurity – A Gating Issue for Safety in A Connected and Automated Vehicle Future.
https://www.mcca.com/wp-content/uploads/2018/04/Autonomous-Vehicles.pdf

Training Module Overview

- Module 1 (Virtual): Overall Review:
- Module 2 (Hands-on): Controller Setup and Networking needed for RSU
- Module 3 (Hands-on): Individual RSU SPaT Messaging/OBU Configuration
- Module 4 (Hands-on): Intersection Configuration and Troubleshooting Scenarios
- Module 5 (Hands-on): SPaT and Adaptive Signal Control Configuration
- Module 6 (Hands-on): Maintenance and Troubleshooting Scenarios
- Module 7 (Hands-on): Other Cisco Configurations and RSU Brands

TRAINING MODULE 2

Introduction

Provided herein is a list of steps to set up a Road Side Unit (RSU) -Controller Connection in a laboratory setting. The setup is used, in part, to establish a Connected and Automated Vehicle (CAV) communication system that allows a connected vehicle to send and receive information from NJDOT infrastructure. This process will require fundamental changes (as noted) to conduct a similar setup in the field. This document is independent of any field contract used to evaluate an established CAV. This document intends to provide general training/knowledge/skills to NJDOT staff on the primary communication setup between a traffic signal controller and an RSU. The RSU provides a communication conduit between a traffic signal controller's Signal Phase and Timing Data (SPaT)/Pedestrian Data and a vehicle's On-Board-Unit (OBU). Conversely, the OBU can provide data to the NJDOT infrastructure, but that portion is not covered in this document. The network requirements to set up an RSU and traffic signal controller are covered. Not all RSU systems or traffic signal controllers are alike, and some may have a wireless connection, while others require a direct Ethernet connection (Cat-5 or Cat-6 cable). A static IP from a laptop is required for a direct Ethernet connection. Some other notes, the RSU may already be connected to a controller or require the controller to be set up before the RSU is connected. It is suggested that the latter be adhered to and the controller set up to transmit SPaT data before the RSU is set up to receive the data. Doing so will assure that the RSU and Controller communicate once the RSU is active. That stated, provided is a list of the general materials used for the laboratory setup of an RSU and a traffic signal controller:

- **Items necessary:**
  - Assorted Ethernet cables
  - Siemens Sitraffic ESCoS RSU with POE injector (Setup reflects DSRC protocol)
  - Test Commsignia ITS-OB4 OBU with associated Android tablet with Commsignia App (For testing OBU/RSU communication in Lab)
  - A controller:
    - Trafficware Commander ATC Controller (CONTROLLER USED)
  - All available hardware manuals for the above items
  - Windows Laptop with Ethernet ports, **Admin access to Laptop**, Wifi internet access, Google Chrome browser (or Firefox) installed. Some RSU's only allow specific browsers.

!!! Note, you must ensure all antennas and wires are attached to the RSU unit before powering on. Failing to do so could damage the unit.

Section 1: Controller Network and Communication Configuration

The recommendation is to have a controller setup and outputting SPaT messages that, once set up correctly, the RSU will be able to receive. Some controllers may require a direct connection via an Ethernet cable. If that is the case, go to *Section 2: Initial Laptop Network Setup to communicate with RSU* and follow the same Laptop Network IP procedure requiring establishing a static IP on a local laptop. For the Trafficware

Commander, the IP address can be set on the front panel. This section assumes that any controller is set up for the standard 8 phase and properly configured to run a specific intersection. The following are the standard controller setup configurations for a Trafficware Commander:

Trafficware Commander ATC Controller:

The Configuration for Network and Spat is through the Front Panel and should not require an Ethernet connection. The information is the recommended first step to pairing a Trafficware Commander ATC controller with a specific RSU. Any IP changes in the Controller will require a reboot once the IP is changed.

To initialize the Controller for DSRC communication, follow this sequence. Please note that DSRC communication could change to C-V2X. Hit Home → 8 'Login and Utilities' → 4 'Initialize' → 2 'Run Options'

Figure 1. Trafficware Commander Initializing Controller 'Home Key'

Figure 2. Trafficware Commander Initializing Controller 'Home Key' again if needed.

Figure 3. Trafficware Commander Initializing Controller-Numeric Menus

Figure 4. Trafficware Commander Initializing Controller-Utilities

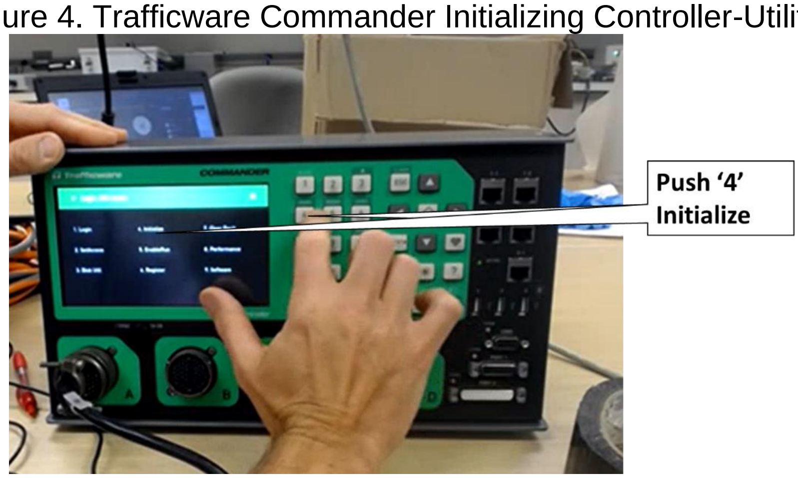


Figure 5. Trafficware Commander Initializing Controller-Initialize

Figure 6. Trafficware Commander Initializing Controller-Run Options

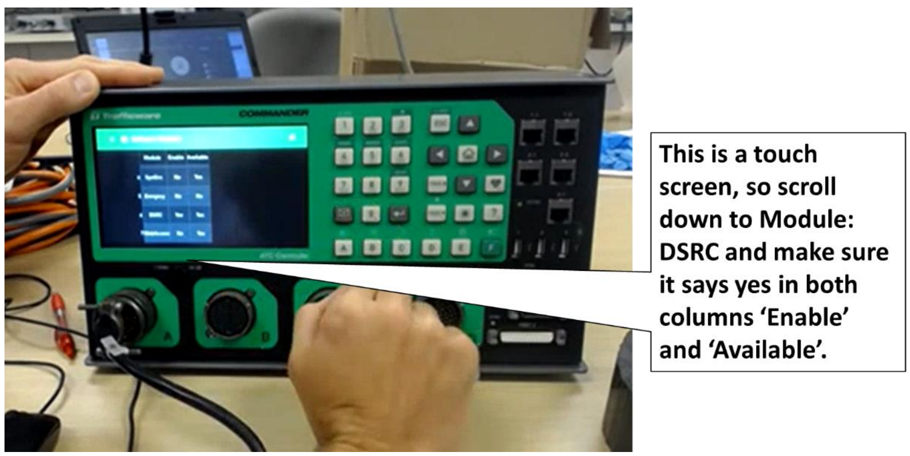


Figure 7. Trafficware Commander Initializing Controller- Scroll to DSRC

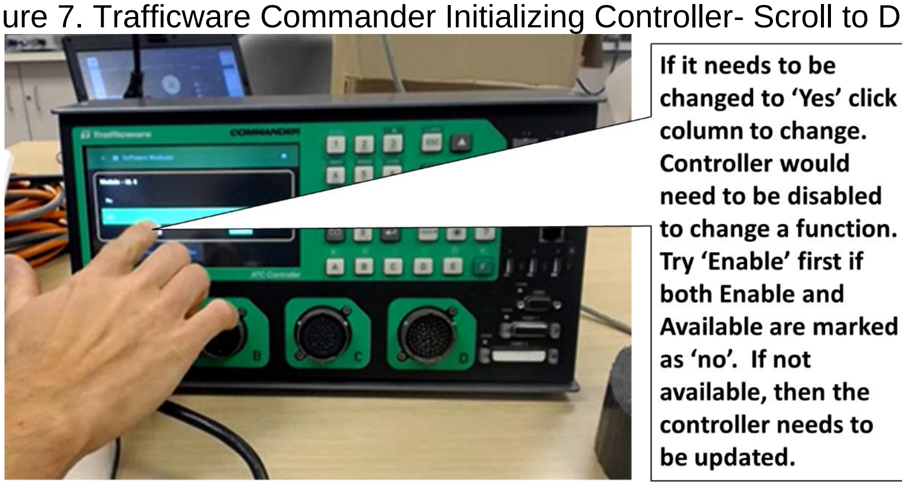


Figure 8. Trafficware Commander Initializing Controller – Assure setup

Once the initialization of the DSRC is completed, the Controller’s IP configuration needs to be set. The Controller will also have the RSU IP address placed in its configuration (Host 2), and the RSU will have the controller IP information. All the IPs must match:
To change the IP, follow this sequence. Hit Home → 6 ‘Comm’ → 5 ‘IP Setup’

Figure 9. Trafficware Commander Initializing Controller-IP Setup

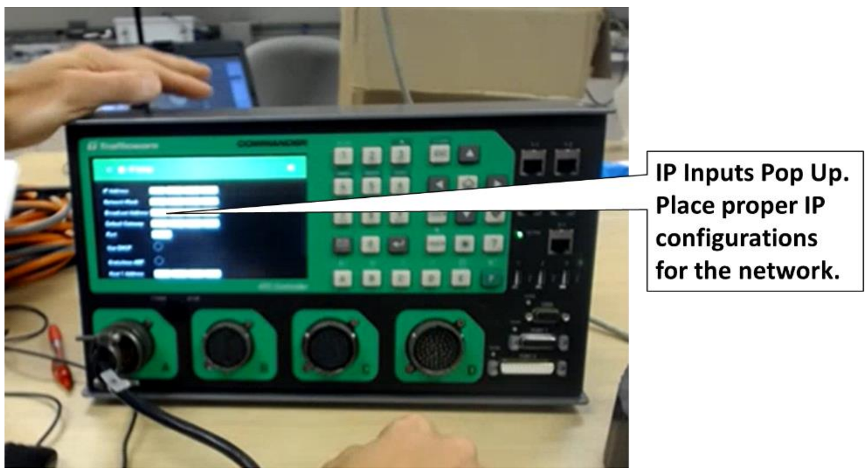


Figure 10. Trafficware Commander Initializing Controller –IPs need to be predetermined

IP, gateway, and subnet should already be set if Controller is communicating over ATSM.now. If out of the box, this information would need to be obtained from OIT and placed in the Controller. NTCIP port for the lab is set to 501 and needs to match the RSU port. Some RSUs have a pre-set NTCIP 'Port,' so the Controller's port must match the **RSU NTCIP Port.**

RSU Host Address 2 needs to be set to the RSU. In theory, Host1 is the ATMS.now server.

PING address to match the RSU address (or planned RSU IP). Usually, this is the ATMS.now server. It should have defaulted to ATSM.now, but for testing of the future RSU setup, set up the PING to send to the RSU. **Note that only one traffic controller can be paired with one RSU**. The RSU will require the Controller's IP address to be set in the RSU configuration. CISCO will resolve the pairing in the field once the IPs are set up.

Once all setup, hit 'Home,' then save changes.

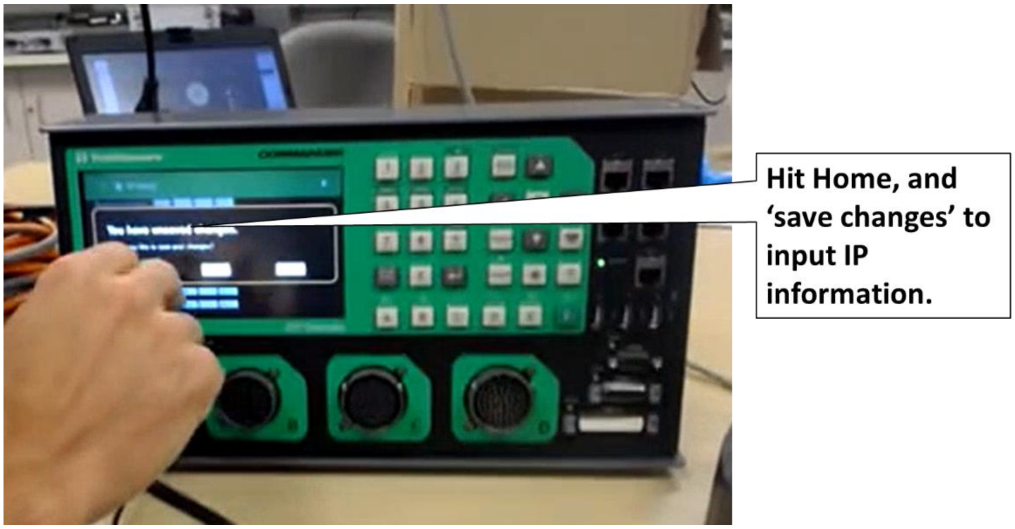


Figure 11. Trafficware Commander Initializing Controller- Saving IP data

Once the RSU is set up, a Ping Test can be performed to see if the Controller can talk with the RSU over the network. It can only be done once the RSU is set up to respond to the ping.

To ping the RSU, follow this sequence. Hit Home → 6 'Comm' → 8 'Ping'

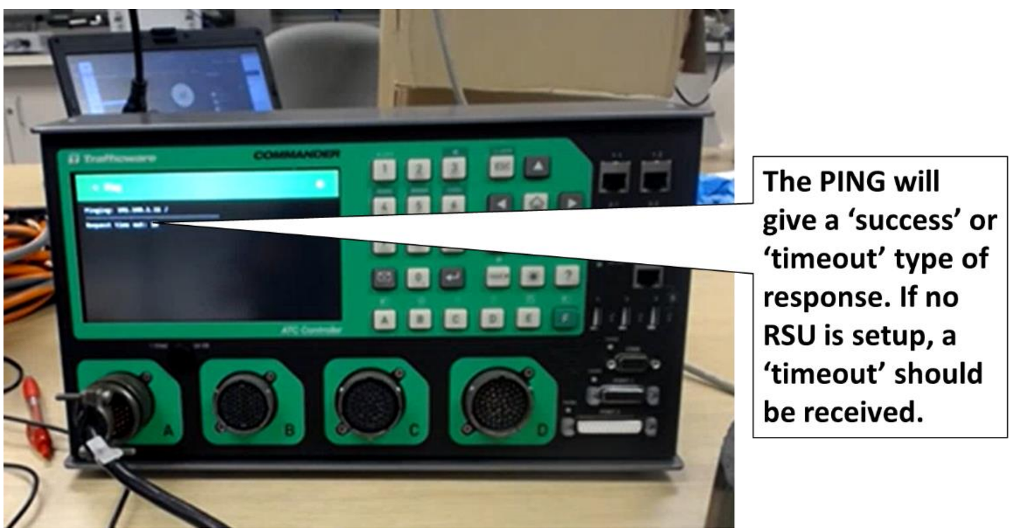


Figure 12. Trafficware Commander Initializing Controller –Ping Test

After initialization and all the IP addresses are established, the DSRC needs to be set up (DSRC MAY BE CHANGED ACCOMMODATE C-V2X PROTOCOL). **To set up the DSRC, follow this sequence: Home → 6 'Comm' → 9 DSRC Proprietary Setup**

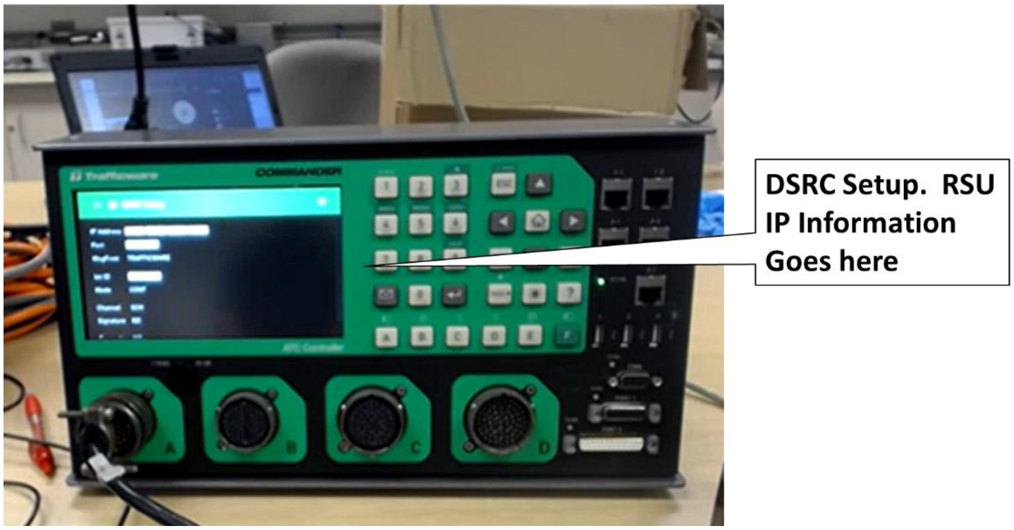


Figure 13. Trafficware Commander Initializing Controller –DSRC setup

The following need to be set at the DSRC:
PORT: Set to **6053**, but can be changed in Trafficware and needs to match RSU. Message Format…no RSU will work on J2735 (Although this is the standard). All of the RSU's seem to prefer 'Trafficware Proprietary,' so leave this on default.
Intersection ID needs to match the MAP message being used for the intersection. It is supercritical, the RSUs will emit a separate map and SPaT if the IDs are different, but the MAP MESSAGE (https://webapp.connectedvcs.com/isd/) and Intersection ID need to match.
MODE: Alternative 'ALT' or Continuous 'CONT.' Keep on Continuous.
Channel: Don't touch.
Signature: Don't touch.
Encryption: Don't touch for now.

The Trafficware Commander controller is now set up to communicate with an RSU.

Section 2: Initial Laptop Network Setup to communicate with RSU:

The setup of an RSU is unit-specific; this section shows the primary method for setting up a laptop to communicate with an RSU. IPs are dependent on the unit type and configuration status. The same methodology for setting up a laptop to connect to an RSU can be done for a traffic signal controller requiring an Ethernet cable connection. The Trafficware Commander allows input on the front panel. Some RSUs have a cellular connection, and some do not (e.g., Siemens does not). For those RSUs that DO NOT have a cellular connection, a PC laptop (with admin writes) requires a static IP address setting to be adjusted to be able to communicate with the network that the RSU resides (same first three octets of the connecting IP, different 4th octet set on the laptop that has Ethernet connectivity). There are two situations for non-cellular connections:

- The first is that the RSU needs to be reconfigured from its default IP settings, requiring a direct connection with a laptop via Ethernet cable.

- The second is that the **RSU is already set up** or had been set up and communicating with the existing network. In either case, the laptop used requires a new static IP, with the 4th octet being a high number in the range 0-255. (Suggested 251; therefore, if connecting to a default Siemens RSU the laptop should be 172.24.4.25**1**). To change the IP on a laptop, please follow Figures 14-18, which start by typing in 'Control Panel' in windows explorer and transition through:

Control Panel/ Network and Internet / Network and Sharing Center/

Once at the Network and Internet (Figure 14), Figure 15 pops up. Right-click active Ethernet connection and click on properties (Figure 16). Double Click Internet Protocol Version 4 in Figure 17. THE ADDRESS USED MUST BE UNIQUE AND OBTAINED FROM OIT…UNLESS IT IS A NEW RSU, then use the default manufacturer IP to change the IP to be on the same network as the Controller. Ultimately this IP needs to be placed in a controller if one has not already been set up. The subnet mask should also match current network configurations. The necessary IP configurations placed in Figure 17 should be obtained from NJDOT. For this document, a Siemen's RSU is used, whose default IP should be, e.g., Siemens default IP is 172.24.4.254, meaning that the laptop should be set to 172.24.4.25**1**.

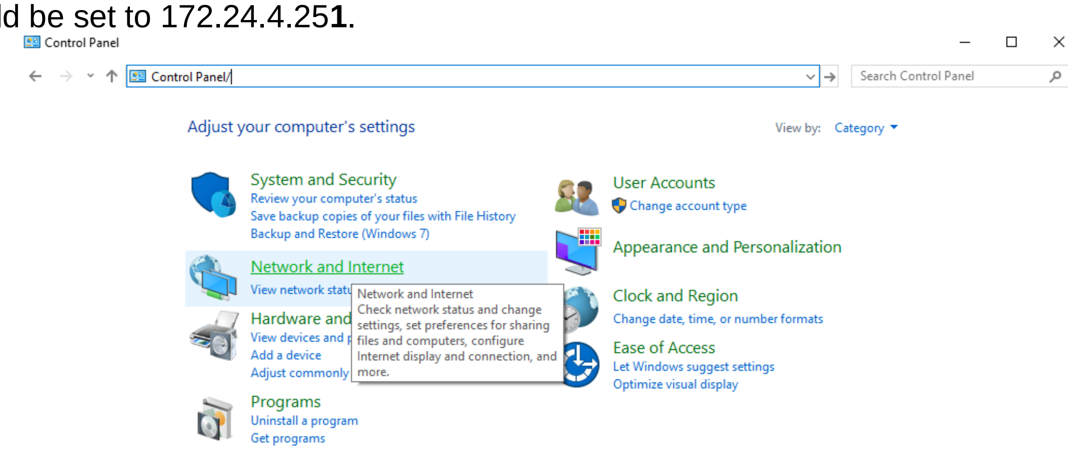


Figure 14. Control Panel/Network and Internet

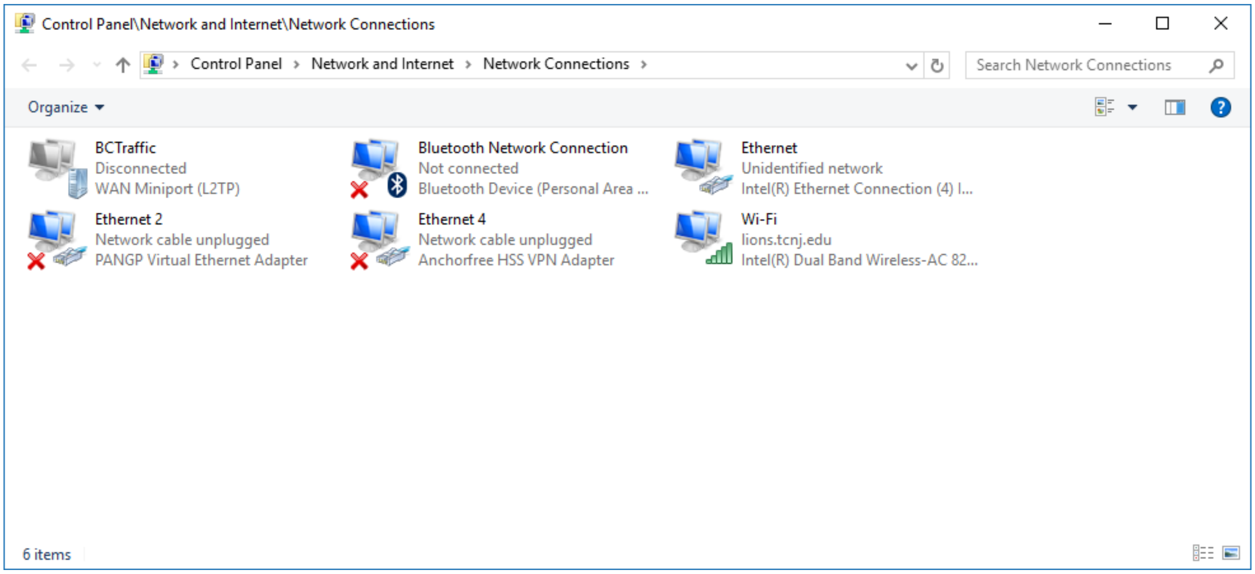


Figure 15. Ethernet Connection

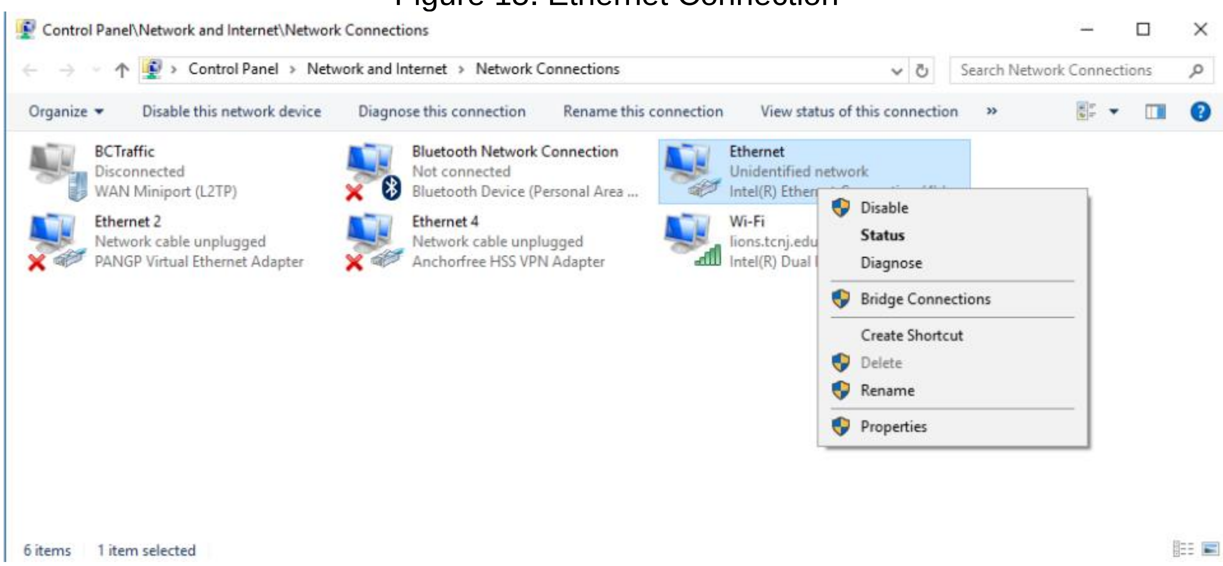


Figure 16. Adjust Properties

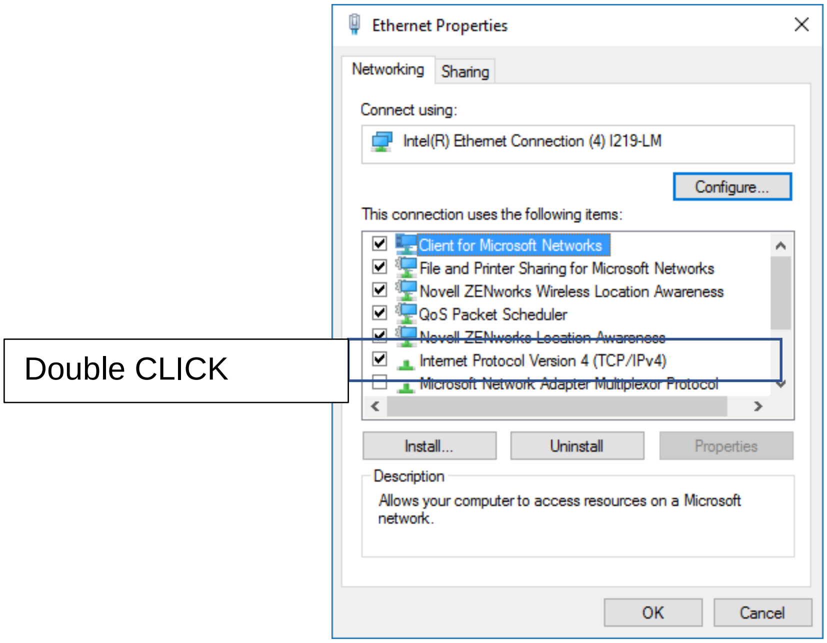


Figure 17. Select Properties for Internet Protocol Version 4 (TCP/IPv4)

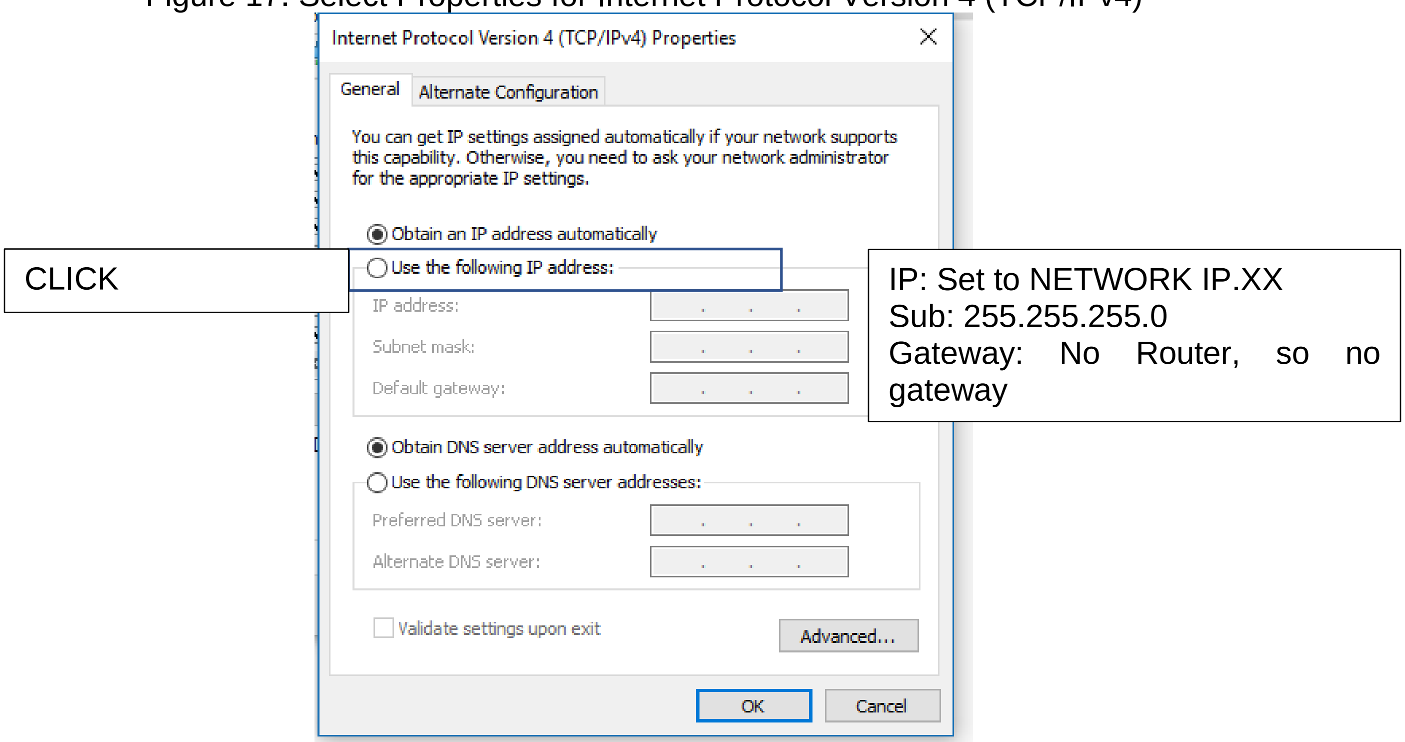


Figure 18. The IP address needs to be set for the laptop. 'Use the following IP Address.'

Once Internet Protocol Version 4 (TCP/IPv4) has the new IP address that matches either the default RSU or network IP, you **must hit okay** to set the laptop IP. Once this is completed, the laptop's initial network scheme has been configured.

With the laptop set to an appropriate network, the next step is to configure RSU's to the network. **You must ensure all antennas and wires are attached to the unit before powering on an RSU unit.** Each of these different pieces of hardware will have a unique process defined under the specific hardware direction. When connecting directly to the

RSU, the default IP address will be needed (e.g., **Siemens's default IP is 172.24.4.254**). The next step is for the RSU can be reassigned to the appropriate network. However, the laptop will need to be reset (Figures 14-18) to match the network configurations (not the default RSU) to communicate with an RSU that is now configured for the network. In summary, you need to have your laptop talk with the RSU on the same IP network, reset the RSU IP to a new network, and reset the laptop to the new IP network. Go to the RSU setup for additional information:

Section 3: RSU Network Configuration

Siemens

Siemens does not have a cellular connection and therefore requires a direct connection to the RSU via CAT6/5 (See network setup). If out of the box, the RSU will have a pre-defined default address that needs to be changed to the new network. You must ensure all antennas and wires are attached to the unit before powering it on. The layout of the RSU unit from box to assembly is shown in Figure 19 - Figure 32.

Figure 19. Siemens in the box

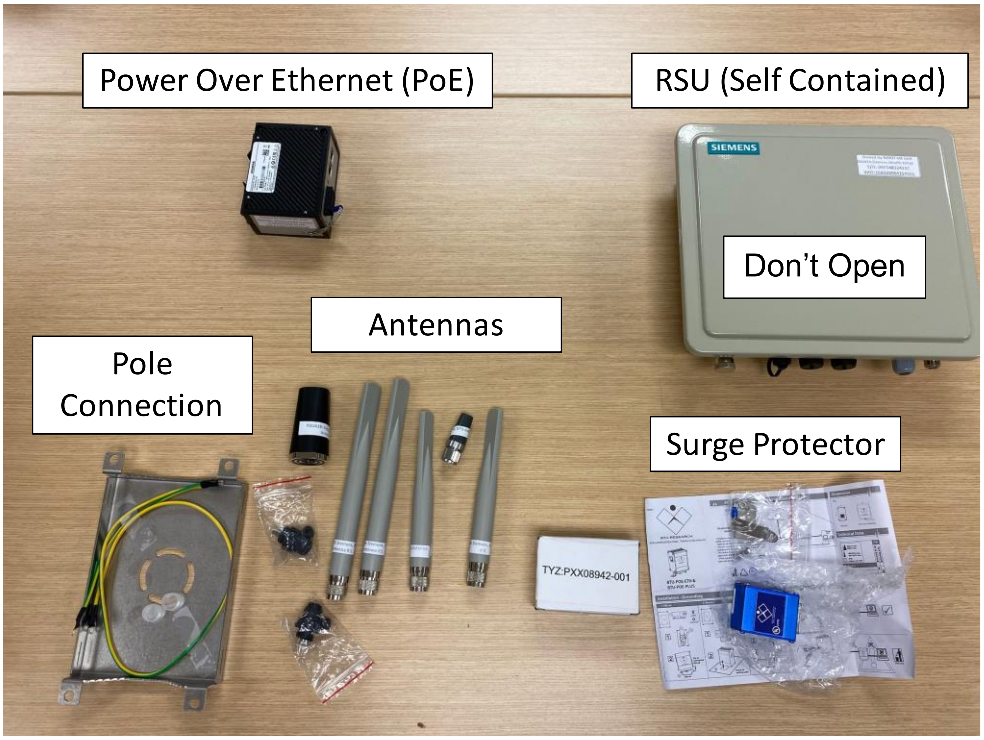


Figure 20. Siemens out of the box

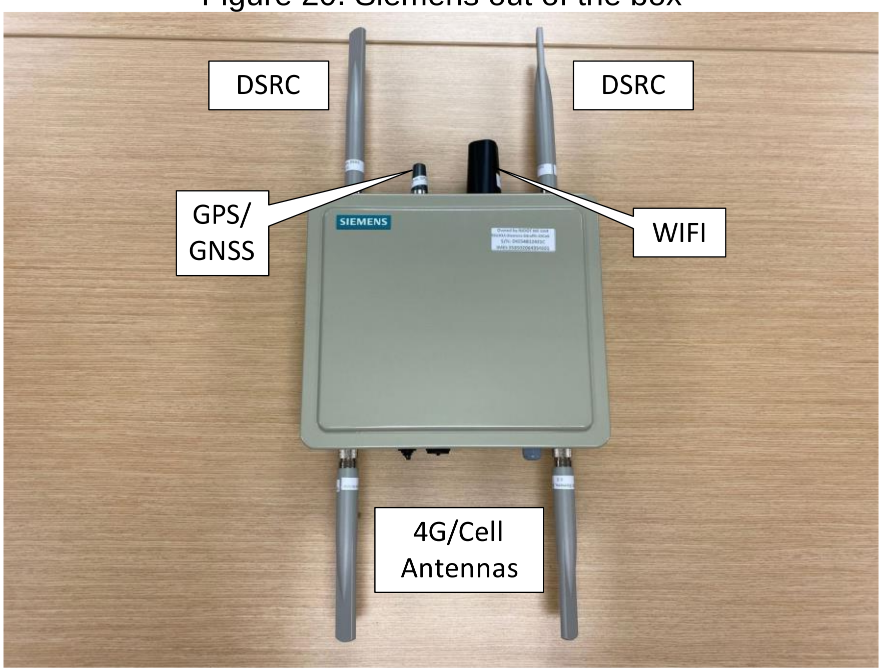


Figure 21. Assembled RSU

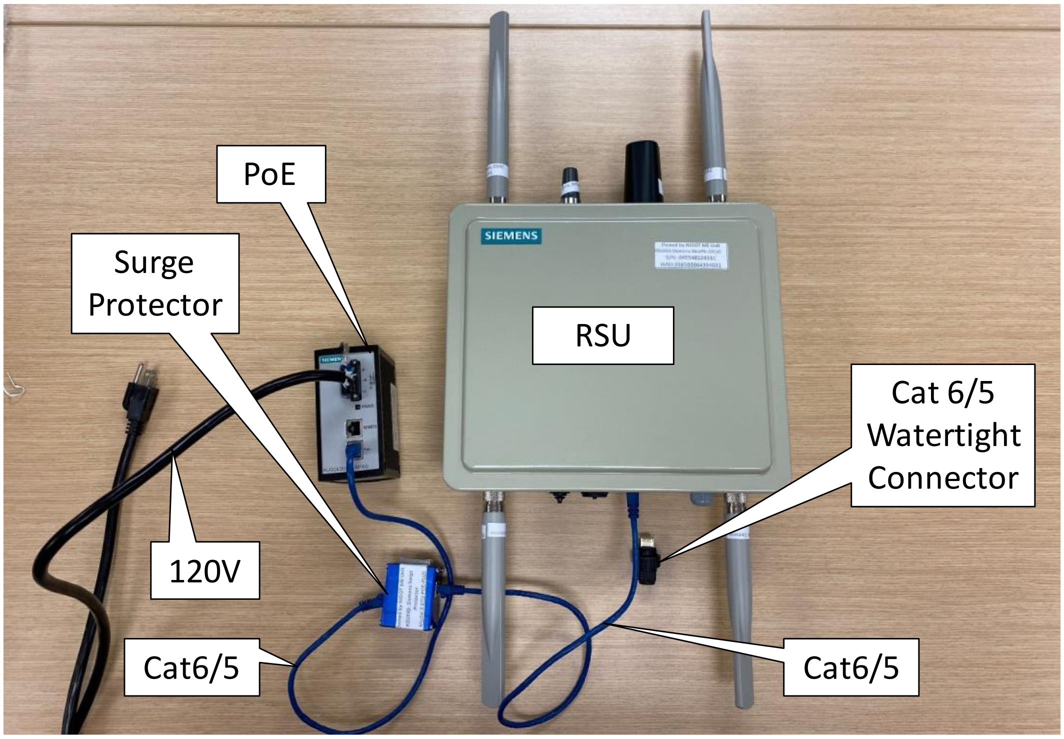


Figure 22. Setup of RSU connections

The Wire Power Supply is 120V to PoE. To power on the Siemens RSU, the connection for power is as follows:

Power (120V) → PoE Injector → Cat6/5 → Surge Protector → Cat6/5 → RSU

The Cat 6/5 connector to Siemens needs to be a watertight connector in the Field. A close-up picture of the connections and RSU on the button is shown in Figure 23.

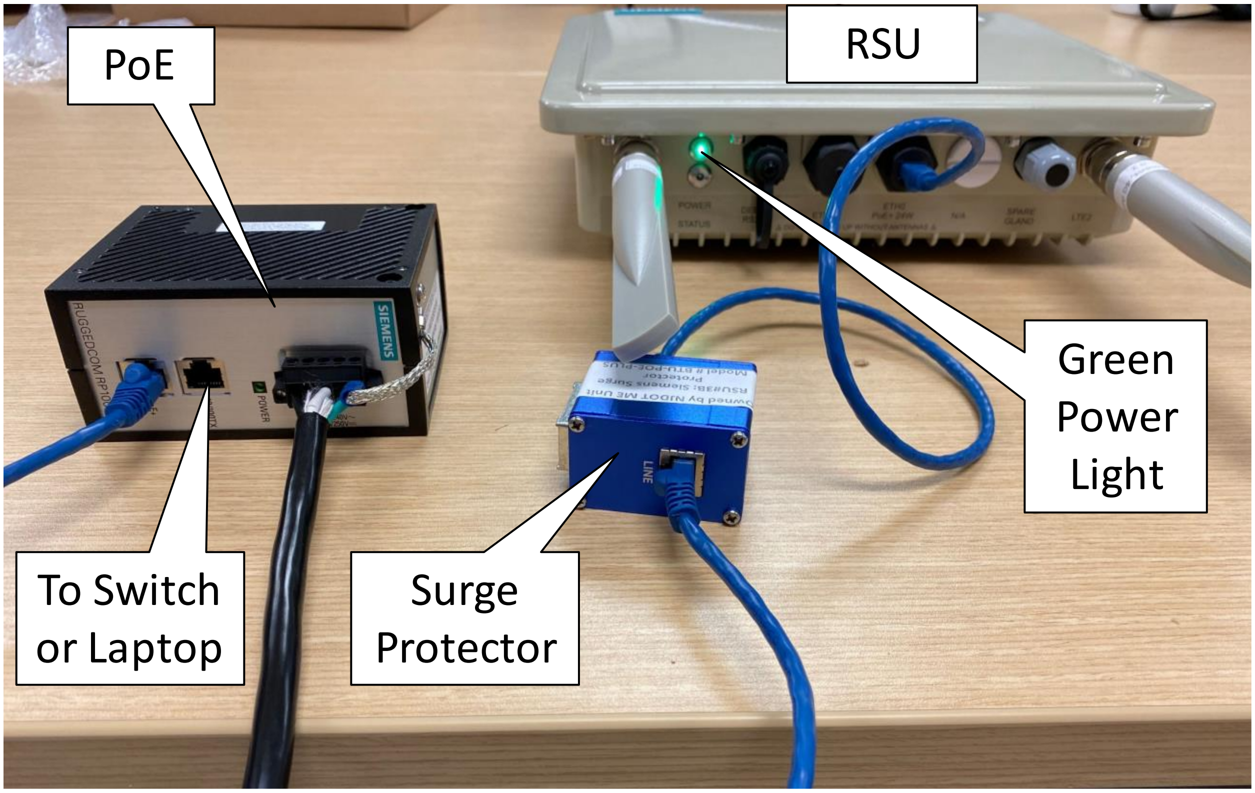


Figure 23. RSU powered up (Max 300’ Cat 6 in the field)

The Siemens unit is configured via a web GUI, so all modifications need to be done locally with CAT5/6 cable as outlined in the initial network setup. Connect the laptop via Ethernet to the POE OR network; again, make sure the laptop, with Ethernet port, is set to the appropriate address. Siemens does not have DHCP.

At this point, the RSU should be able to be ‘pinged,’ which can be done by opening up a DOS Command Prompt (type ‘CMD’ at the start window) and typing the word PING followed by a space and the RSU’s IP address, Figure 24; if default it should be 172.24.5.254.

Figure 24. Command Prompt to ‘ping’ RSU

Use a browser (not internet explorer, preferred Chrome/Firefox) to access the RSU GUI by typing in the default Siemens IP address – 172.24.5.254 or 172.24.5.254: 4443. See Figure 25, which shows the Siemens GUI when first putting the IP address in the browser. Default login credentials – Username “admin” Password – “**T_JKFM;5qH**”

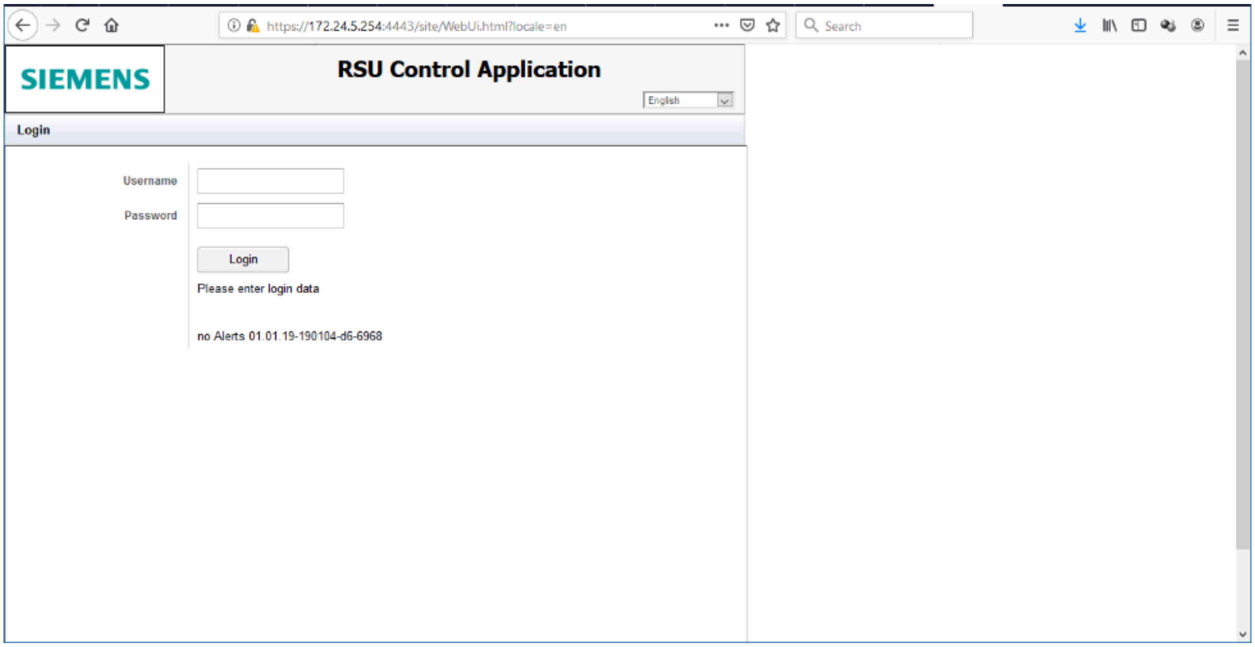


Figure 25. Siemens GUI

After successful login, the entire web GUI should be displayed (Figure 26). Now, make sure the firmware is updated. To do this, navigate to the “Network or Settings” tab and then “Wired Interface” (Figure 27). Ensure that Auto Mode DHCP – CLIENT is off and input the IP address information for the configured local network (this is the new address on the same network as the Controller previously set). After inputting and applying these settings, reboot the unit (power cycle).

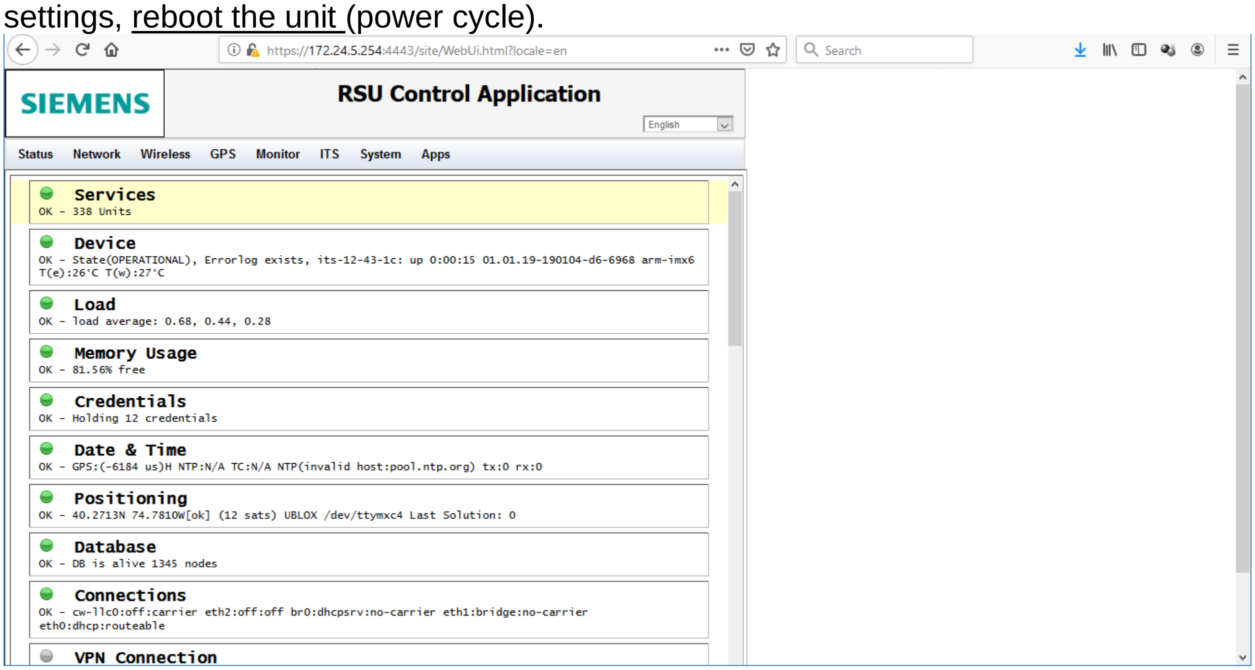


Figure 26. RSU services

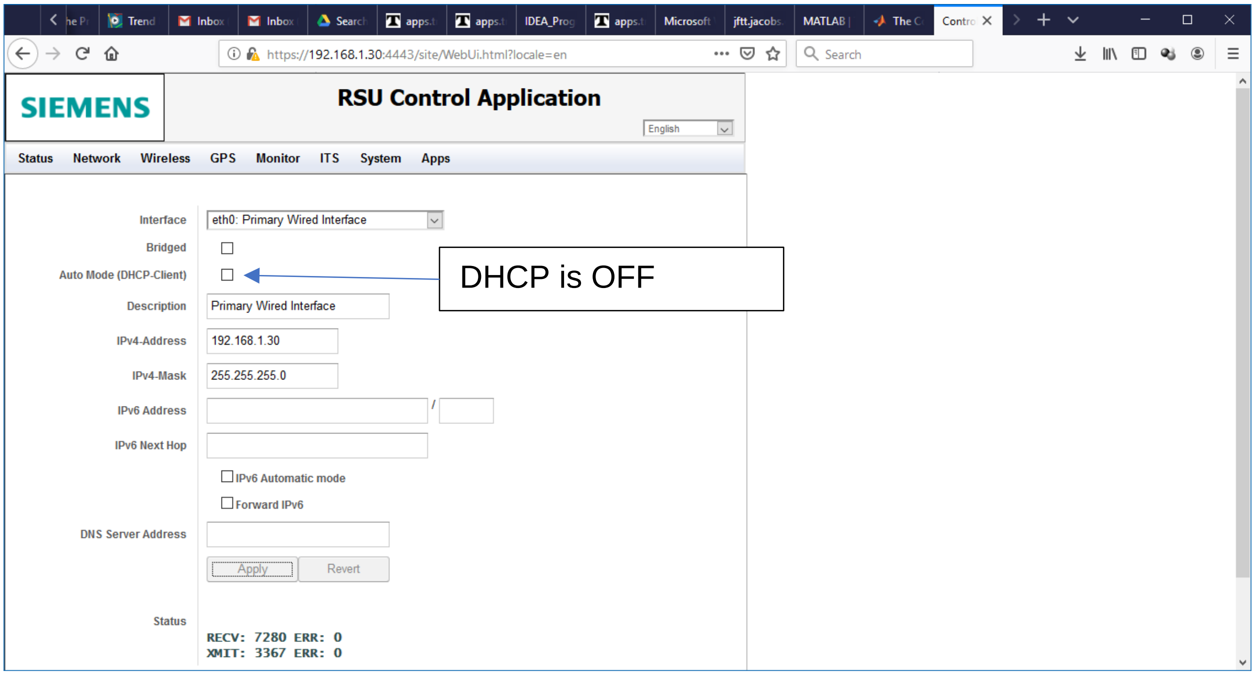


Figure 27. RSU IP Settings

Once rebooted, <u>you will lose contact with the RSU</u> because the RSU is on a new network that the laptop is no longer on. Therefore, it is necessary to go through Section 2 (Figures 14 -18) to reset the laptop to have its IP network reconfigured to regain access to RSU. Once completed, navigate back to the web GUI (Figure 26) with the **new IP** address to confirm new settings.

The next step is to activate the MAP/SPaT, which allows the Controller to send messages through the RSU. To activate MAP/SPaT output, a MAP message will need to be uploaded to the unit using the '**SPaT/MAP extension'** under the "APPS" tab. An XML must have already been created for the' Map Upload.' Note, contractor to make the XML map message created from this site: https://webapp2.connectedvcs.com/). The files are shown in Figure 28. The GUI should say "MAP message okay" if this has been appropriately configured. Also, confirm under the "Status" tab that the MAP/SPaT indicator is green and active (Figure 29). It will only be green if connected to a controller. It will be yellow otherwise. In this case, the controller still needs to be set up, but if the Controller was set up correctly, the RSU is set up, and the MAP is correct, it should be green.

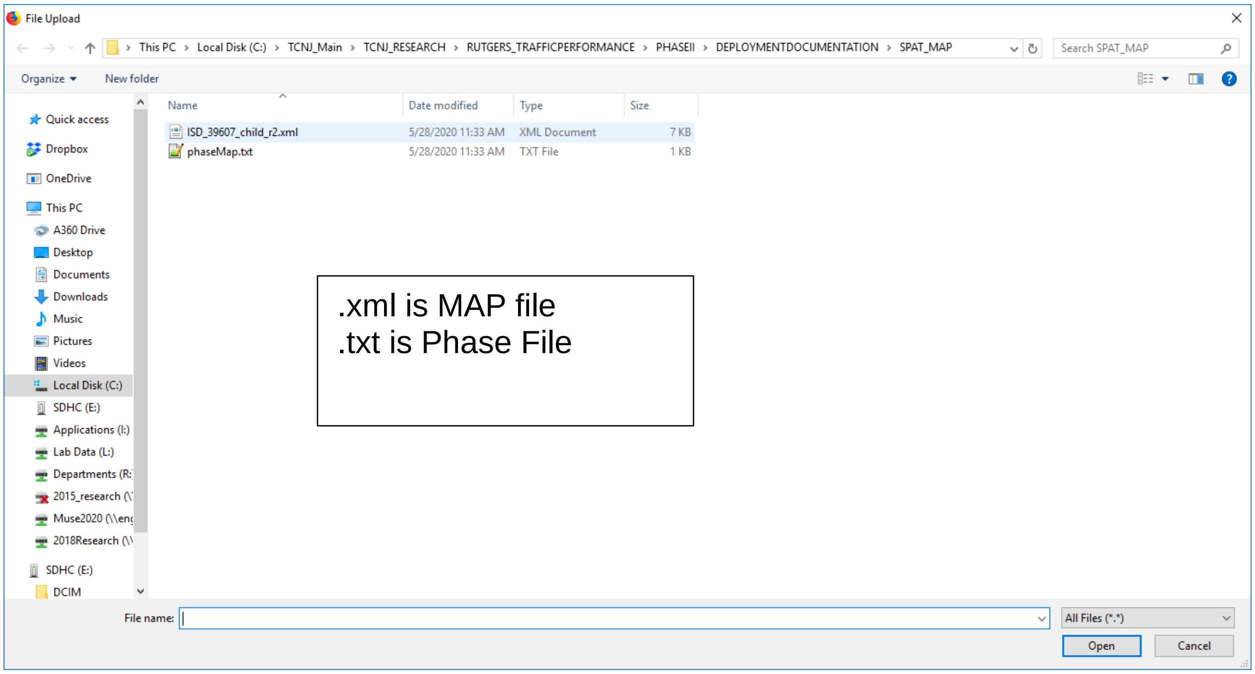


Figure 28. RSU XML (txt is the translation between map message and timing, XML is map message)

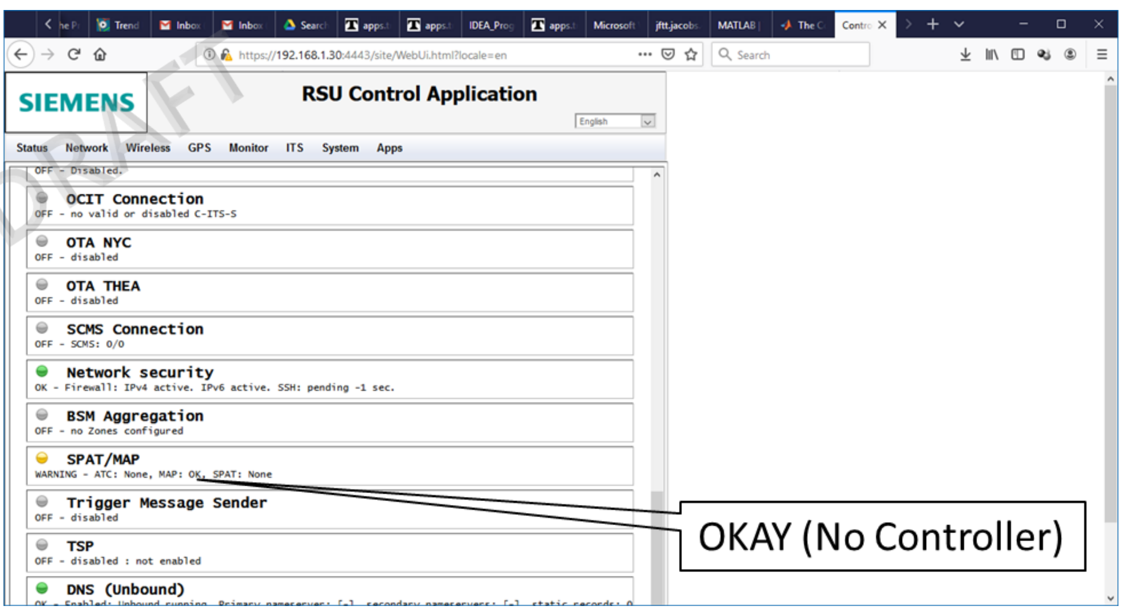


Figure 29. RSU setup, but no controller set with MAP message

Make sure it is set to controller type (in Apps/ SPAT). It will communicate with a controller on TCP Port 6053, which needs to match the Controller (Figure 30). Note that Status will not show Info w/o Controller.

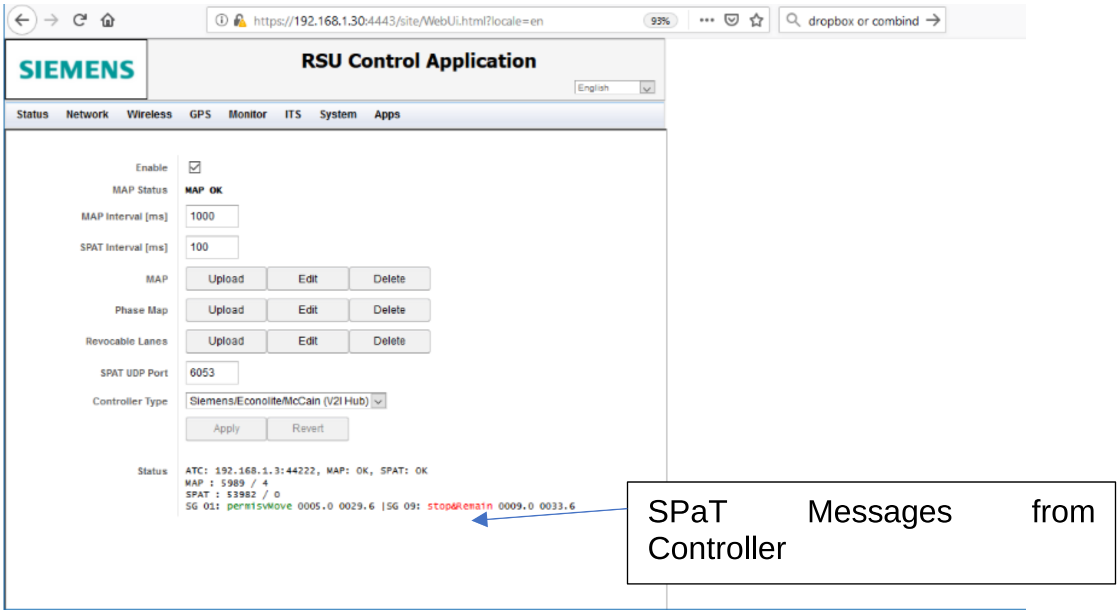


Figure 30. SPAT/MAP messages (MOVE IN ORDER)

Under APPs, under Controller, the communications port needs to be set to the standard communications port 501; this is not the same as the SPAT UDP Port. Ensure the address in this screen matches the **controller IP** (Figure 31).

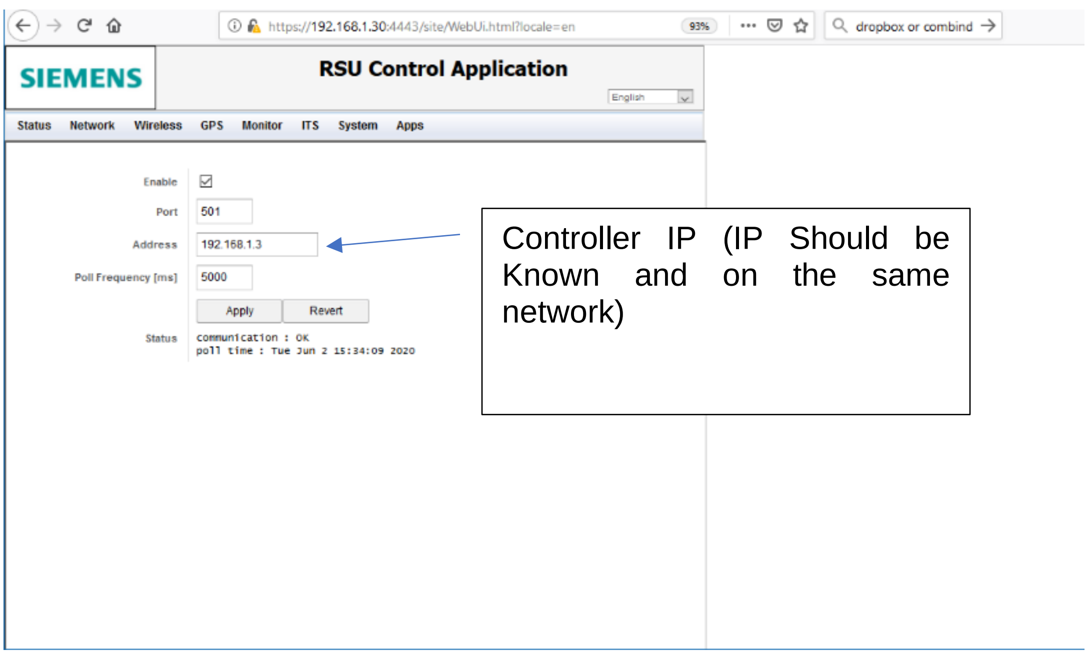


Figure 31. RSU Controller Configuration

If this has been appropriately configured, the RSU should now be outputting SPaT/MAP messages to any OBU in the area (Figure 30, Status), and Figure 32 will show the green 'OK' under SPAT/MAP. Reboot the RSU. Once completed, the controller and RSU should be communicating, and the RSU should be sending out SPAT information. To verify this outside of the RSU interface (Figure 30), an OBU would need to be used.

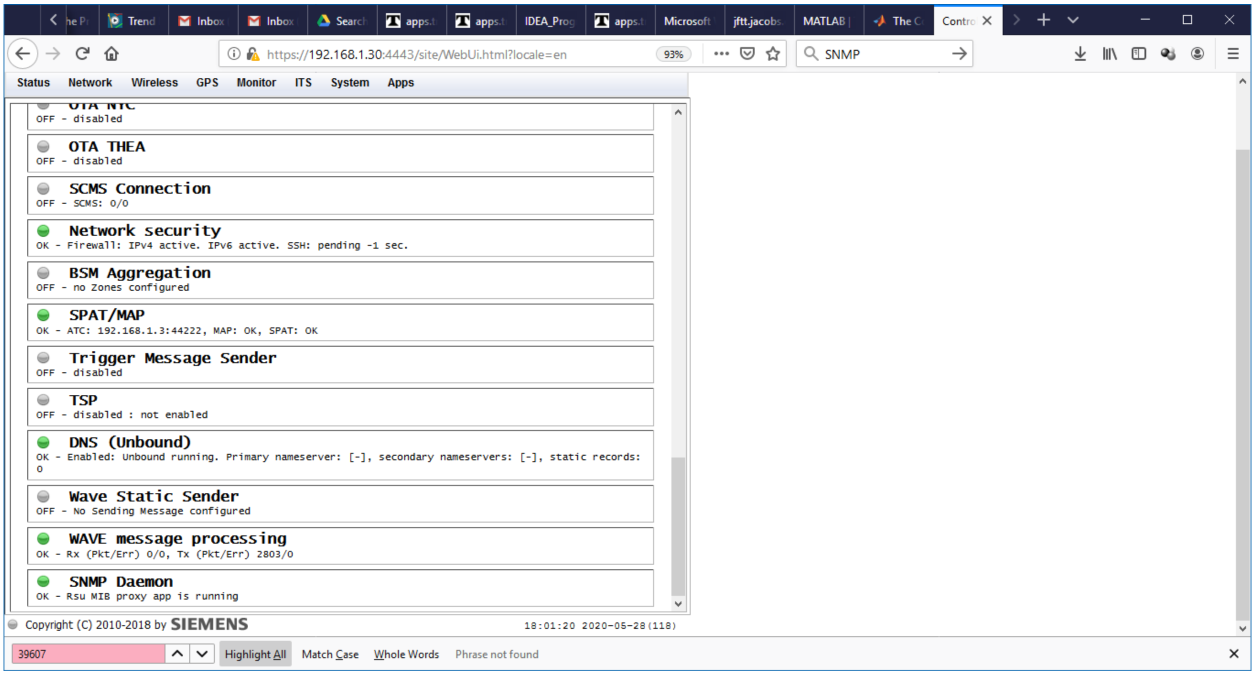


Figure 32. SPAT/MAP green, messages being sent

Section 4: OBU Configuration

SCMS

- The outer layers of the PDU contain information such as the version and creation time of the request.
- The inner request contains the to-be-signed certificate itself.
- The to-Be-Signed certificate request contains the essential fields pertinent to the enrollment certificate created.
    - The id field contains the serial number that is required for the device
    - cracaId and crlSeries are information about who is authorized to revoke this device and put it on a CRL. ECA (enrollment cert authority) can overwrite this information, so it is safe to put all 0s for the cracaId and 4 (enrollment cert revocation series) for those fields
    - validityPeriod is how long your enrollment cert will be valid. ECA can coerce this too.
    - ‘region’ is where this cert will be valid. The most common value for that is 840 (USA country code).
    - certRequestPermissions are the PSID/SSP combinations which are a set of permissions for applications that the enrollment cert can request application/pseudonym certs. One typical example of that is the BSM PSID which has a value of 0x20. The IEEE publishes those lists of PSIDs.
    - The verificationKey field contains the public signing key that is required to have to complete the enrollment certificate

- Overall the request is self-signed by the private signing key generated during the bootstrapping process.

Useful Links:
https://stash.campllc.org/projects/SCMS/repos/scms-asn/browse/scms-protocol.asn?at=release/1.2.1#827-847
https://stash.campllc.org/projects/SCMS/repos/scms-asn/browse/eca-ee.asn?at=release/1.2.1#68-98

Jason file to be prepared for OER file generation


```
{
 "protocolVersion": 3,
 "content": {
  "signedCertificateRequest": {
   "hashId": "sha256",
   "tbsRequest": {
    "version": 1,
    "content": {
     "eca-ee": {
      "eeEcaCertRequest": {
       "version": 1,
       "currentTime": 11,
       "tbsData": {
        "id": {
         "none": null
        },
        "cracaId": "AAAAAA",
        "crlSeries": 153,
        "validityPeriod": {
         "start": 26214,
         "duration": {
          "years": 40
         }
        },
        "region": {
         "identifiedRegion": [
          {
           "countryOnly": 124
          },
          {
           "countryOnly": 484
```

```
              },
              {
                "countryOnly": 840
              }
            ]
          },
          "certRequestPermissions": [
            {
              "subjectPermissions": {
                "explicit": [
                  {
                    "psid": 32
                  },
                  {
                    "psid": 38
                  }
                ]
              },
              "minChainDepth": 0
            }
          ],
          "verifyKeyIndicator": {
            "verificationKey": {
              "ecdsaNistP256": {
                "compressed-y-0":
"6162636465666162636465666162636465666162636465666162636465666162"
              }
            }
          }
        }
      }
    }
  }
},
"signer": {
  "self": null
},
"signature": {
  "ecdsaNistP256Signature": {
    "r": {
      "compressed-y-1":
"1D4D192AE43FDF44600083562D0084821678673BB56837E9C0BBF50810209B15"
    },
    "s":
"D5209F416FAECF9419BD52C3CBF7DC1914B1BB6060C334672B4A3563D3E0F0C4
"
```

```
        }
      }
    }
  }
}
```

Date and Time format Conversion

Use the following to convert the Data and time format.

```
import datetime
import netCDF4

times = [datetime.datetime(2016, 10, 1) + datetime.timedelta(hours=hour)
        for hour in range(84)]

# Create netCDF file
calendar = 'standard'
units = 'days since 1970-01-01 00:00'
ds = netCDF4.Dataset('test.nc', 'w')
timedim = ds.createDimension(dimname='time', size=len(times))

# Write timestamps to netCDF file using 32bit float
timevar32 = ds.createVariable(varname='time32', dimensions=('time',),
                              datatype='float32')
timevar32[:] = netCDF4.date2num(times, units=units, calendar=calendar)

# Write timestamps to netCDF file using 64bit float
timevar64 = ds.createVariable(varname='time64', dimensions=('time',),
                              datatype='float64')
timevar64[:] = netCDF4.date2num(times, units=units, calendar=calendar)

# Read timestamps from netCDF file
times32 = netCDF4.num2date(timevar32[:], units=units, calendar=calendar)
times64 = netCDF4.num2date(timevar64[:], units=units, calendar=calendar)
for time, time32, time64 in zip(times, times32, times64):
    print("original  ", time)
    print ("  32 bit  ", time32)
    print ("  64 bit  ", time64)
```

Public key and private key generation

```
import pycoin
import hashlib, secrets

def sha3_256Hash(msg):
```

```
  hashBytes = hashlib.sha3_256(msg.encode("utf8")).digest()
  return int.from_bytes(hashBytes, byteorder="big")

def signECDSAsecp256k1(msg, privKey):
  msgHash = sha3_256Hash(msg)
  signature = sign(generator_secp256k1, privKey, msgHash)
  return signature

def verifyECDSAsecp256k1(msg, signature, pubKey):
  msgHash = sha3_256Hash(msg)
  valid = verify(generator_secp256k1, pubKey, msgHash, signature)
  return valid

# ECDSA sign message (using the curve secp256k1 + SHA3-256)
msg = "Message for ECDSA signing"
privKey = secrets.randbelow(generator_secp256k1.order())
signature = signECDSAsecp256k1(msg, privKey)
print("Message:", msg)
print("Private key:", hex(privKey))
print("Signature: r=" + hex(signature[0]) + ", s=" + hex(signature[1]))

# ECDSA verify signature (using the curve secp256k1 + SHA3-256)
pubKey = (generator_secp256k1 * privKey).pair()
valid = verifyECDSAsecp256k1(msg, signature, pubKey)
print("\nMessage:", msg)
print("Public key: (" + hex(pubKey[0]) + ", " + hex(pubKey[1]) + ")")
print("Signature valid?", valid)

# ECDSA verify tampered signature (using the curve secp256k1 + SHA3-256)
msg = "Tampered message"
valid = verifyECDSAsecp256k1(msg, signature, pubKey)
print("\nMessage:", msg)
print("Signature (tampered msg) valid?", valid)


from tinyec.ec import SubGroup, Curve

# Domain parameters for the `secp256k1` curve
# (as defined in http://www.secg.org/sec2-v2.pdf)
name = 'secp256k1'
p = 0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f
n = 0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
a = 0x0000000000000000000000000000000000000000000000000000000000000000
b = 0x0000000000000000000000000000000000000000000000000000000000000007
g = (0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,
    0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8)
```

```
h = 1
curve = Curve(a, b, SubGroup(p, g, n, h), name)
print('curve:', curve)

privKey = int('0x51897b64e85c3f714bba707e86791454a1377a7463a9dae8', 16)
print('privKey:', oct(privKey)[2:])

pubKey = curve.g * privKey
pubKeyCompressed = '0' + str(2 + pubKey.y % 2) + str(oct(pubKey.x)[2:])
print('pubKey:', pubKeyCompressed)

import ecdsa
# SECP256k1 is the Bitcoin elliptic curve
sk = ecdsa.SigningKey.generate(curve=ecdsa.SECP256k1)
print(sk)
vk = sk.get_verifying_key()
print(vk)
sig = sk.sign(b"message")
print(sig)
vk.verify(sig, b"message") # True

import ecdsa
from hashlib import sha256
message = b"message"
public_key =
'021307267760606135552437340361175672722654521173621253052432507362056
3073610613723737131'
# sig =
'740894121e1c7f33b174153a7349f6899d0a1d2730e9cc59f674921d8aef73532f63edb9
c5dba4877074a937448a37c5c485e0d53419297967e95e9b1bef630d'

vk = ecdsa.VerifyingKey.from_string(bytes.fromoct(public_key),
curve=ecdsa.SECP256k1, hashfunc=sha256) # the default is sha1
print(vk)
# vk.verify(bytes.fromhex(sig), message) # True

# Generate Keys
privateKey =
2430457331164134176705135647017641474424251204673647214352355350724652
13442411061431
publicKey =
2130726776060613555243734036117567272265452117362125305243250736205630
73610613723737131
message = "My test message"

##Generate Signature##
```

```
signature = ecdsa.sign(message, privateKey)

# Verify if signature is valid
print(ecdsa.verify(message, signature, publicKey))
```

Basic configuration for US Region

The following steps are used to implement a basic default configuration on the device compliant with US regional standards. The following requirements must be met before the configuration steps:

• You must have a V2X device running the Commsignia software stack.

• The device must be connected to power and operational. For more information, see the Starting up the device chapter.

• You must be connected to the device using a web browser (for the graphical user interface (GUI)) or over an SSH connection from a Windows or Linux-based computer.

These configuration steps aim to provide a default backup configuration set on the device fully compliant with US regional V2X standards. You can use this configuration as a starting point for any custom changes you want to make or for debugging purposes.

1. Set the Navigation mode for the device.

a. Using the GUI, navigate to the V2X → Stack menu and click the Navigation tab. You can select the navigation source (GPSD, real, or manual

GPSD mode means communicating with a GPSD server for a navigation source, Real mode

uses the navigation chip on the device as the source, and manual mode means you can specify navigation details manually.

b. In the case of the manual navigation source, set the values you want the device to use for a navigation fix. Manual navigation method is recommended for any indoor or lab testing.

2. Set the vehicle information details in the Identifiers tab in the V2X → Stack menu.

a. Set the station ID and set the vehicle type in case of an onboard unit.

b. Set the vehicle details if you have all the details.

3. Go to the Radio interface tab in the V2X → Stack menu.

a. For C-V2X communication, turn on C-V2X. It cannot be configured or enabled by default.

b. For US regional standards, you can no longer use DSRC radio.

c. Set all radio frequency values according to your regional standards.

4. Turn on the WSMP module for US regional standard compliance.

5. Turn on the BSM module for US regional compliant essential V2X messages.

The V2X is ready and configured for the US regional standard.

Establish Connectivity with OBU to visualize the data recoding

The following steps were followed to log and visualize the messages:

1. Before your test, please run the following commands on your device:

uci set uplconfig.c2p.capture_enable='1'
uci set uplconfig.c2p.capture_server_ip='127.0.0.1'
uci commit
unplugged-rt-restart.sh

2. Check if the parameters were set correctly with the following commands:

uci show uplconfig.c2p.capture_enable
uci show uplconfig.c2p.capture_server_ip

3. Check if the C2P is running on the OB4:

tcpdump -i lo -n "udp dst port 7943"

If packets appear on your stdout, you're good.

4. Before going on the field, run the following command in the background on the OB4:

tcpdump -i lo -n "udp dst port 7943" -w /rwdata/test01.pcap

A packet must appear on your stdout

5. After your on-field test, kill the process:

killall tcpdump

6. Copy the /rwdata/test01.pcap file to your computer.

Note:

For recoding the BSM and SPAT, use following commands:

unplugged-rt-status-gen 127.0.0.1 | grep MAP; and
unplugged-rt-status-gen 127.0.0.1 | grep SPAT

Installation of Capture application and connect the OBU

This section describes the necessary steps to enable the Commsignia Capture Protocol (C2P) using the command line for a device running the Commsignia V2X software stack.

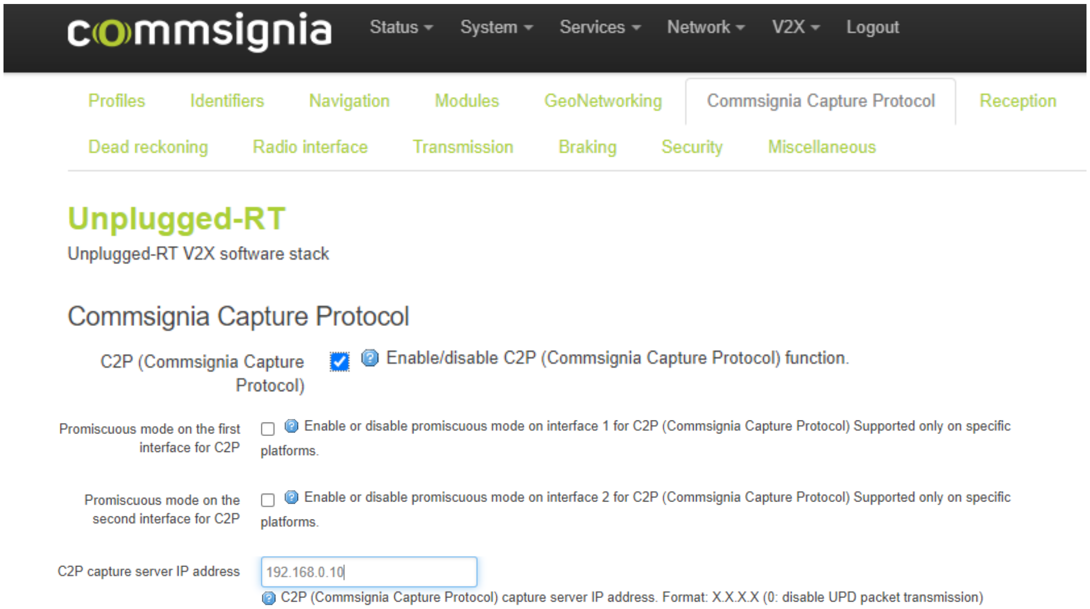
• You must be connected either directly or over a network to the V2X device running the Commsignia V2X software stack.

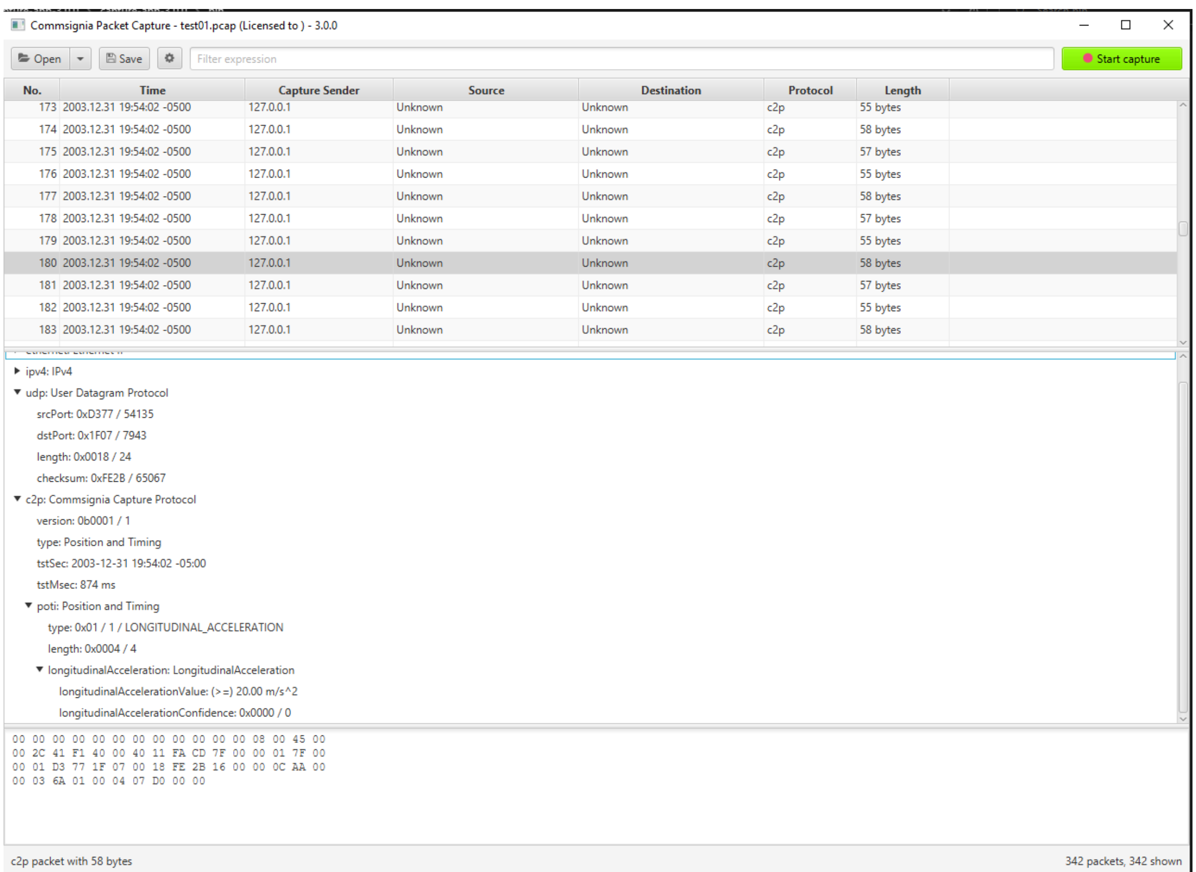
• The device must be powered on and operational, and all antennas and accessories must be properly
connected.

• Your computer must be running either a Windows or a Linux-based operating system, and you must have access to the command line.

Use the following commands for installing the Capture app:

1. Log in to the V2X device using an SSH connection.
ssh root@192.168.0.54
The default IP address is
192.168.0.54
and the
root
user's default password is
UK5BJLFZVBPZLIM55Y
2. Enable the C2P module in the configuration.
a. Enable show command.
uci show | grep c2p
It will show a list of all the configuration parameters related to the C2P
b. Enable the C2P module using the uci set command. Set the value of the
uplconfig.c2p.cap-
ture_enable to 1.
uci set uplconfig.c2p.capture_enable='1'
c. Set the IP address and the port number for the C2P module using the uci set command.
uci set uplconfig.c2p.capture_port='7943'
uci set uplconfig.c2p.capture_server_ip='192.168.0.196'
Use the IP address and port number of your choice. It will be the address you can use to subscribe to the C2P module's data stream.
3. Commit the changes to the configuration.
uci commit
4. Restart the device to enable the C2P module's data stream to be broadcast. Use the following
command to restart:
unplugged-rt-restart.sh
You are successfully connected and logged in to the V2X device, and the C2P data stream is enabled.


Status
System
Services
Network
V2X
Logout
Profiles
Identifiers
Navigation
Modules
GeoNetworking
Commsignia Capture Protocol
Reception
Dead reckoning
Radio interface
Transmission
Braking
Security
Miscellaneous
Unplugged-RT
Unplugged-RT V2X software stack
Commsignia Capture Protocol
C2P (Commsignia Capture Protocol)
Enable/disable C2P (Commsignia Capture Protocol) function.
Promiscuous mode on the first interface for C2P
Enable or disable promiscuous mode on interface 1 for C2P (Commsignia Capture Protocol) Supported only on specific platforms.
Promiscuous mode on the second interface for C2P
Enable or disable promiscuous mode on interface 2 for C2P (Commsignia Capture Protocol) Supported only on specific platforms.
C2P capture server IP address
192.168.0.10
C2P (Commsignia Capture Protocol) capture server IP address. Format: X.X.X.X (0: disable UPD packet transmission)


Commsignia Packet Capture - test01.pcap (Licensed to) - 3.0.0
Open
Save
Filter expression
Start capture
No. Time Capture Sender Source Destination Protocol Length
173 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 55 bytes
174 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 58 bytes
175 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 57 bytes
176 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 55 bytes
177 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 58 bytes
178 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 57 bytes
179 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 55 bytes
180 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 58 bytes
181 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 57 bytes
182 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 55 bytes
183 2003.12.31 19:54:02 -0500 127.0.0.1 Unknown Unknown c2p 58 bytes
ipv4: IPv4
udp: User Datagram Protocol
srcPort: 0xD377 / 54135
dstPort: 0x1F07 / 7943
length: 0x0018 / 24
checksum: 0xFE2B / 65067
c2p: Commsignia Capture Protocol
version: 0b0001 / 1
type: Position and Timing
tstSec: 2003-12-31 19:54:02 -05:00
tstMsec: 874 ms
poti: Position and Timing
type: 0x01 / 1 / LONGITUDINAL_ACCELERATION
length: 0x0004 / 4
longitudinalAcceleration: LongitudinalAcceleration
longitudinalAccelerationValue: (>=) 20.00 m/s^2
longitudinalAccelerationConfidence: 0x0000 / 0
00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00
00 2C 41 F1 40 00 40 11 FA CD 7F 00 00 01 7F 00
00 01 D3 77 1F 07 00 18 FE 2B 16 00 00 0C AA 00
00 03 6A 01 00 04 07 D0 00 00
c2p packet with 58 bytes
342 packets, 342 shown

Automated SPAT/MAP/BSM logger recording

```
from pywinauto.application import Application
import time

app = Application ().start (cmd_line=u'putty -ssh root@192.168.0.54 -pw UK5BJLFZVBPZLIM55Y command > log')
putty = app.PuTTY
putty.wait ('ready')
time.sleep (1)
# putty.type_keys("password")
# putty.type_keys ("UK5BJLFZVBPZLIM55Y")
time.sleep (1)
# putty.TypeKeys ("ls")
putty.type_keys ("unplugged-rt-status-gen {SPACE}127.0.0.1 {SPACE}|{SPACE} grep{SPACE} SPAT")
putty.type_keys ("{ENTER}")
```

```
import paramiko
import datetime

host = '192.168'
user = 'root'
secret = ''
port = 22

ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy()) #Set policy to use when connecting to servers without a known host key
ssh.connect(hostname=host, username=user, password=secret, port=port)
ct = datetime.datetime.now()
stdin, stdout, stderr = ssh.exec_command('unplugged-rt-status-gen 127.0.0.1 | grep SPAT')
ct1 = datetime.datetime.now()
stdin1, stdout1, stderr1 = ssh.exec_command('unplugged-rt-status-gen 127.0.0.1 | grep BSM')
output1= stderr.readlines()
output = stdout.readlines()
print (''.join(output1))
print (''.join(output))

output11= stderr1.readlines()
output12 = stdout1.readlines()
print (''.join(output11))
print (''.join(output1))
file = open('SPAT.txt', 'a')
```

```
file.write('----------------------------------------------------------------------------------------------------
---------------------------\n')
file.write(f'current time:{ct}\n')
file.write(''.join(output1))
file.write(''.join(output))
file1 = open('BSM.txt', 'a')
file1.write('---------------------------------------------------------------------------------------------------
-----------------------------\n')
file1.write(f'current time: {ct1}\n')
file1.write(''.join(output11))
file1.write(''.join(output12))
file1.close()
```

Initiate five windows

1. Open the ATSPM Website (10.247.169.37/ATSPM), log in as an admin
   a. Email: defaultadmin@spm.gov
   b. Password: **Input your password here**
2. RDP to AutoScope Server (10.18.180.12)
   a. Double click Autoscope v10.5.0 on the desktop to run it
   b. Switch from Channel to Custom (bottom left) to display Intersections.
3. Open Google Sheet: NJDOT ATSPM Deployment Status
4. Open Google Sheet: ATSPM Autoscope Detector Records
5. Open the metadata folder for intersections, and it should contain the .xlsx files like this:

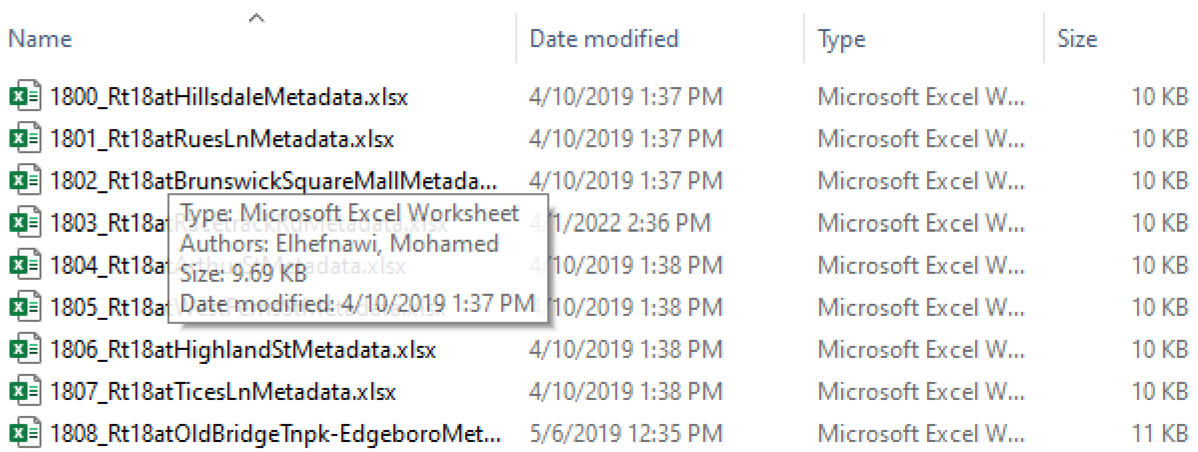


Figure 33. Metadata samples

Main Workflow

1. Go to Autoscope Network Browser on window #2
   a. Pick one intersection and left-click it (Here, we use Rt.1 at Henderson Rd. for example)

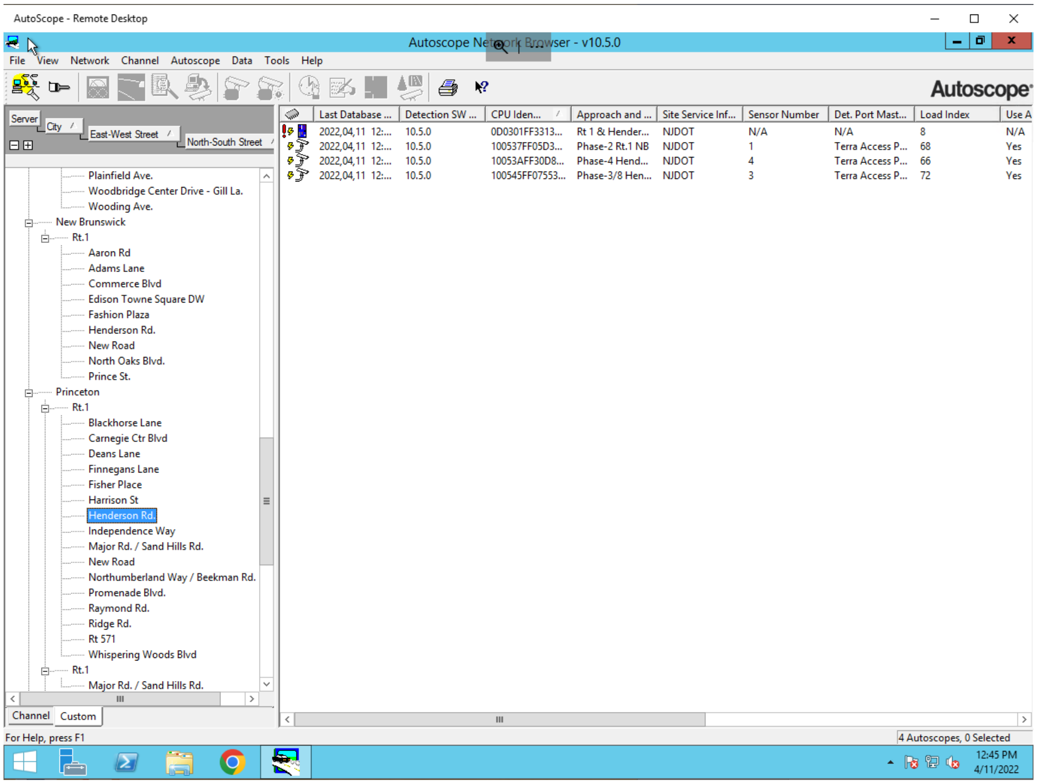


Figure 34. Autoscope network browser interface

b. Then we right-click the “Approach and Phases” one by one and choose “Edit Detectors.”

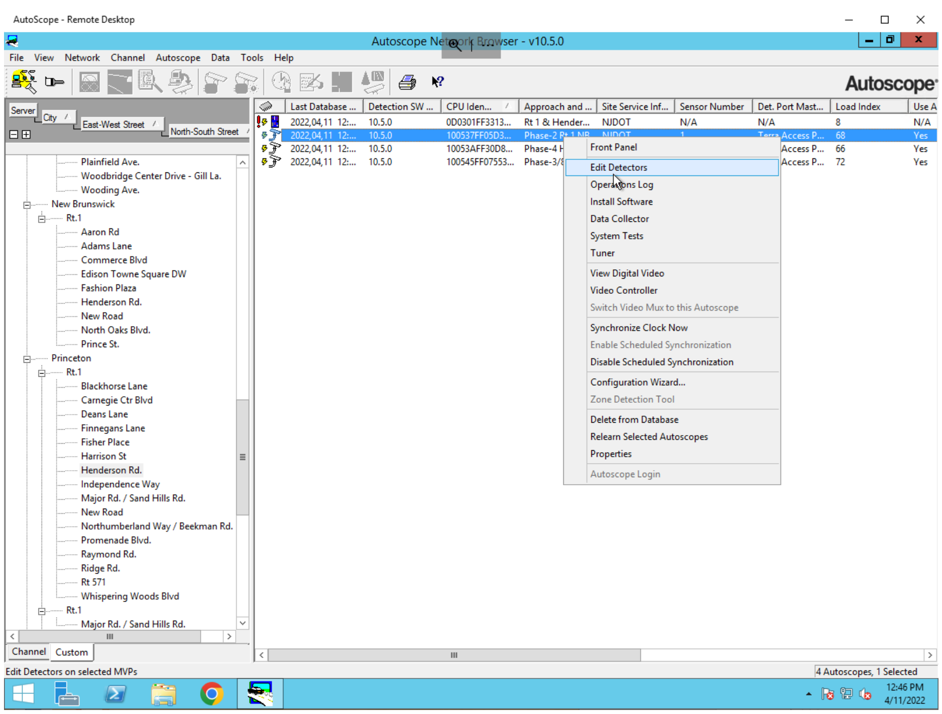


Figure 35. Autoscope network browser interface- properties

c. Confirm “Yes” to “Would you like to upload a new copy.”
d. Left-click to any space to cancel the selections
e. Move your cursor onto a Stop Line, and the number “120” will be the input of Window #4 – Google Sheet: “ATSPM Autoscope Detector Records,” Sheet 2 “SignalASDetectors,” Column E “AutoScopeDetID.” Meanwhile, the lane for this detector is northbound (see from Phase-2 Rt.1 NB on the top left of the camera view), so “NBT” is the input of Window #4 – Google Sheet: “ATSPM Autoscope Detector Records,” Sheet 2 “SignalASDetectors,” Column F “Direction.”

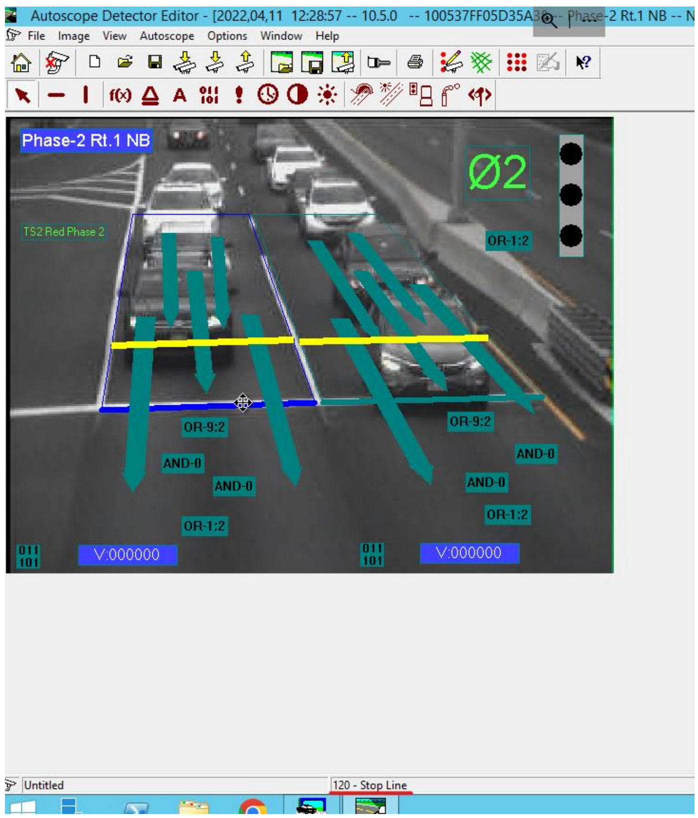


Figure 36. Autoscope detector editor of one phase

f. For column A “SignalID” in Window #4 – Google Sheet: “ATSPM Autoscope Detector Records,” it will be found from Window #3 - Google Sheet: “NJDOT ATSPM Deployment Status.” Search for “Henderson” in Window #3 and find the “10029_Rt1 at Henderson Rd Metadata.xlsx” cell, so “10029” is the “SignalID” in Window #4.
g. Open the “10029_Rt1 at Henderson Rd Metadata.xlsx” (from step f.) in Window #5 metadata folder, find the “NBT” (from step e.) in column D “Movements_Phase.” In this case, it’s in “NBTR” corresponding to phase “2”. Use “2” as the input of Window #4 Column C “PhaseNum.”
h. Then go to Window #1 – ATSPM website, Admin - Signal Configuration

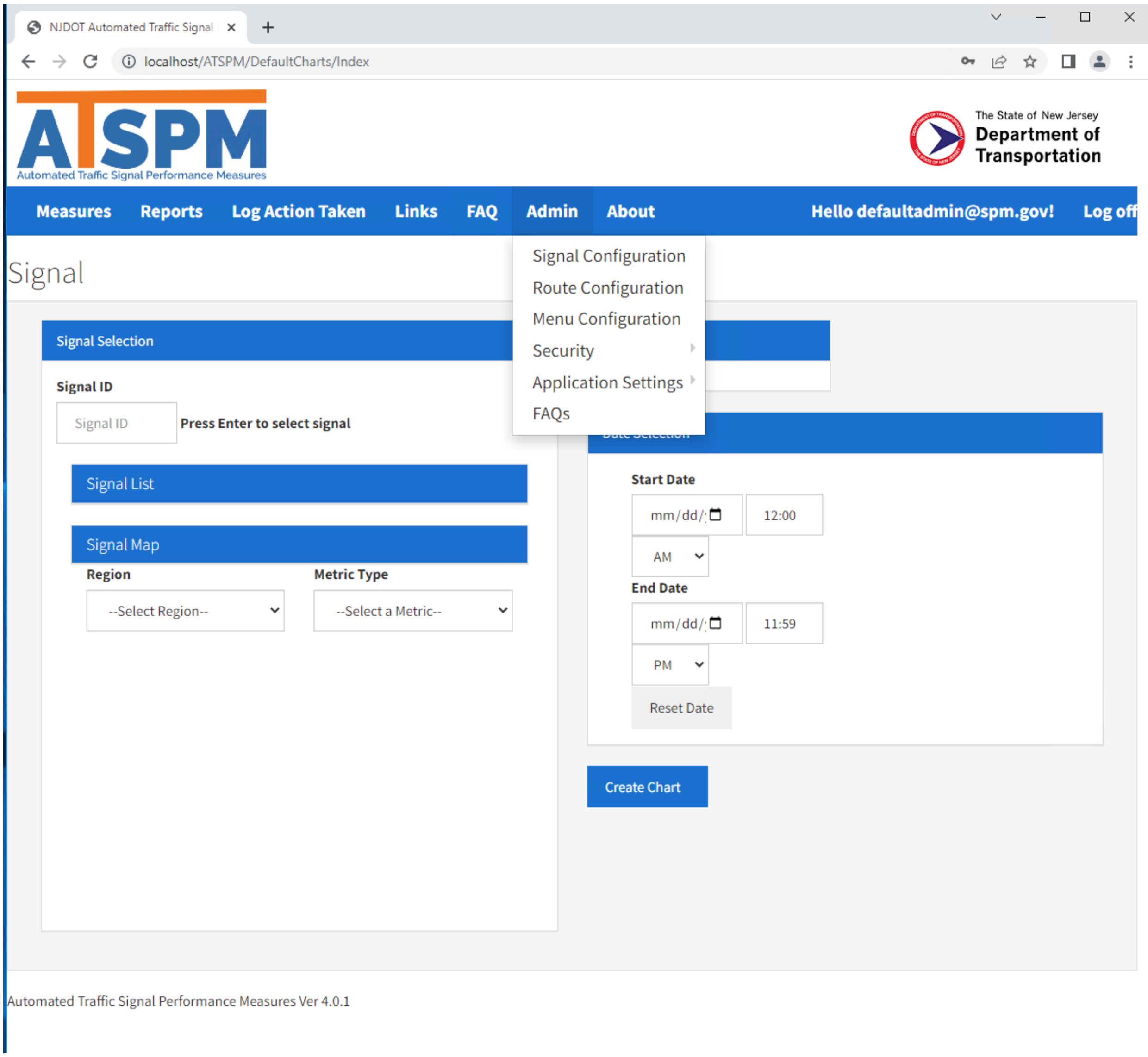


Figure 37. ATSPM website, Admin - Signal Configuration

i. Click “Create New Signal” if signal 10029 hasn't been created, set “Primary Name,” “Secondary Name,” “latitude,” “longitude,” and “region” accordingly. Set the controller type as “ASC3”, check the “Display On Map.”

j. Click “Phase/Direction+” to add a new phase
   i. Set the Direction as “NB” ((from step e.), add descriptions accordingly, then click “Detectors+” to add detectors.
      1. The “Det Channel” has been generated automatically, which is the input of Column D: “DetChannel” in Window #4 – Google Sheet: “ATSPM Autoscope Detector Records.”
      2. For “Detection Types,” check “Advanced Count,” “Lane-by-lane Count,” and “Stop Bar Presence.”

    3. Set “Detection Hardware” as “Wavetronix Advance,” set “Lane Number (Lane-by-lane Count)” accordingly, set the “Movement Type (Lane-by-lane Count” as “Thru” due to “NBT” (from step e.), set “Movement Type (Lane-by-lane Count” accordingly.
    4. **Remember to click “Save” (Top right)**
  k. The “DetectorID” in Window #4 Google Sheet: ATSPM Autoscope Detector Records can be updated automatically after “DetChannel” is filled.
2. Repeat Step 1 to complete other configurations.

SCATS EVENT-RELATED ARCHIVING AND CONFIGURATION

SCATS Event Export and Archiving Guide

It is an introduction to a program running in a Python environment to finish the following processes automatically:

1. Call the SCATSHistoryViewer software to read all the historical data in .hst formats.
2. Export all the historical data into .csv excel files for batch loading.

QuickStartGuide:

Step 0. If there are existing files you archived before and want to keep them, please go to the archive folder and rename the folder **Archive**, e.g., "**Archive + Date,**" and create a new empty Archive folder (the code will create region subfolders).

Otherwise, the code will delete all existing files in the region subfolders and overwrite them.

Step 1. Open Command Prompt as administrator: Open **Start Menu** → searches **Command Prompt** → Right Click **Command Prompt icon** → click **'Run as administrator.'**

Figure 38. Snapshot **of Opening Command Prompt**

Step 2. Type in **cd D:\Program Files\SCATS\Central Manager\Regions**

This query gets to the program folder

Step 3. Type in **Python archive_HistRetrieval.py StartingDate EndingDate**

It runs the python code that takes two arguments: Starting and Ending Date:

All dates should follow the format of **YYYYMMDD**.

For example, if you want archive data from 03/18/2022 to 03/31/2022, the input you need to type is **python archive_HistRetrieval.py 20220318 20220331**
Notice that the ending date will be interpreted as '[date] 23:59:59.'
Special case 1:
If you don't indicate a specific period, it means you only type:
python archive_HistRetrieval.py
That's okay, and the program will automatically archive the data for seven days from now.
Special case 2:
If you only want to archive the data for one day, you need to use the same starting and ending date. For example, if you want to archive the data for 03/18/2022, you should type in:
python archive_HistRetrieval.py 20220318 20220318
Step 4. Suppose you don't see any errors, congratulations! The code will be working and will archive the data you want.

If you need to set up path or server information, please follow the steps below or find one that describes your demand.
Step 1. Find the folder of the programs on the server 10.18.180.17. The path is: 'D:\Program Files\SCATS\Central Manager\Regions.'
The figure shows the folder and files. Please check the integrity of the folder and files.

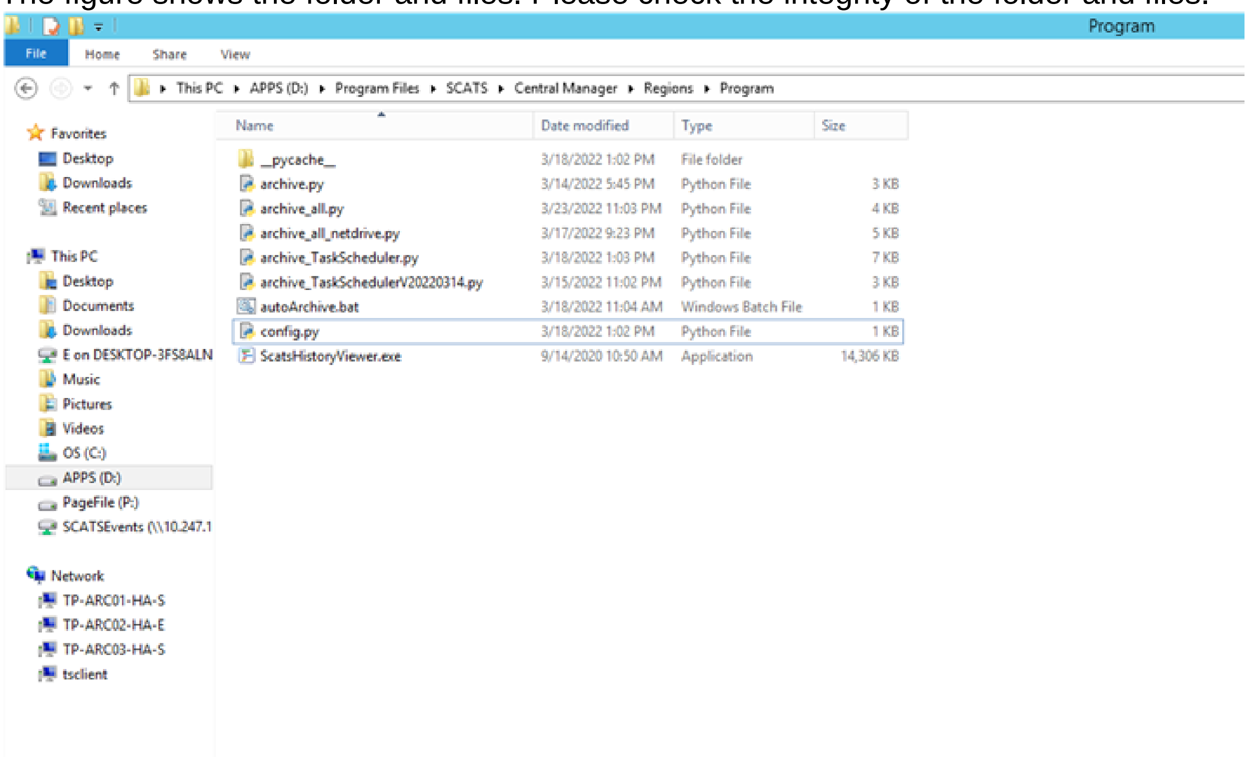


Figure 39. Snapshot of Python Program Files for batch loading

Step 2. Open the config.py file by right-clicking the file and selecting 'Edit with IDLE' → 'Edit with IDLE 3.10 (64-bit)'. This file saves all the information on paths. Line 3 is the destination folder where you want to archive the .csv excel files within them. You can modify the paths where you want to archive. The format of the path should be: “Drive:\\Folder\\Subfolder\\...\\” or “//Server address/Folder/Subfolder/…/”. Please keep the double quotes and only replace the words inside with your input when updating your information. Line 4 is the path direct to folder Central Manager, which stores the necessary software and programs for ATSPM. You can modify the path of the Central Manager, but PLEASE NOTE: the path in line 4 must be the same as your Central Manager folder location. So, if you find your Central Manager has a different path from the path in line 4 in this file, please modify either the path of the whole Central Manager folder or the path in line 4 in this file to make sure they are the same. Line 5 is the Region information you want to call and export, and you can insert any available regions or remove them based on your demands. Please keep the double quotes and only replace the words inside with your input when updating your information.

config.py - D:\Program Files\SCATS\Central Manager\Regions\Program\config.py (3.10.2)

File Edit Format Run Options Window Help

```
1 class FolderPaths:
2     SCATSCsvLiveFolder = "//10.247.169.37/SCATSEvents/Live/"
3     SCATSCsvArchiveFolder = "//10.247.169.37/SCATSEvents/Archive/"
4     SCATSCntrlMngrFolder = "D:\\Program Files\\SCATS\\Central Manager\\"
5     SCATSRegions = "NJR001,NJR018,NJR130,NJR073"
6     SCATSCsvLiveBackupFolder = "D:\\Program Files\\SCATS\\Central Manager\\Regions\\SCATSEventsCSV\\"
7     SCATSCsvLogsFolder = "//10.247.169.37/SCATSEvents/Logs/"
```

Figure 40. Snapshot of Information of Configuration File

Step 3: For the first time, running the archive_HistRetrieval.py, here are the necessary python packages that need to be installed. Each line means a package of the Python environment. Among them, the dependencies of line 12 come from the config.py file, and please ensure the integrity of these files. The others are the necessary python environment to run these files. Please download and ensure the integrity of these python packages on your computer if you don’t have them before.

```
 6 from datetime import datetime
 7 from datetime import timedelta
 8 import os
 9 import time
10 import subprocess
11 import sys
12 from config import FolderPaths
```

Figure 41. Snapshot of Necessary Python Packages and Environment

Step 4. Running the archive_HistRetrieval.py file can call the SCATSHistoryReviewer first to find the corresponding historical data and then export them as the .csv excel files to your destination file you identified in step 2 and zip them. The core query in this file is the line 76, which shows in the figure:

```
60        for date_time in all_date_times:
61
62            try:
63
64                timestampStr = date_time.strftime("%d-%b-%Y %H:%M:%S.%f")
65
66                year = date_time.strftime("%Y")
67                month = date_time.strftime("%m") if len(date_time.strftime("%m")) == 2 else "0" + date_time.strftime("%m")
68                day = date_time.strftime("%d") if len(date_time.strftime("%d")) == 2 else "0" + date_time.strftime("%d")
69
70                # cmds = f'ScatsHistoryViewer.exe -files Regions\\{region}\\SCATSData\\History\\{region + "_" + year + month + day} -sites all -out {dst_folder + region + "_" + yea
71
72                hst_file_name = region + "_" + year + month + day
73
74                # test = ScatsHistoryViewer.exe -files Regions\\NJR001\\SCATSData\\History\\.hst -sites all -e -out p:\\SCATSEventsFiles\\NJR001\\NJR001_20220314-all-events.csv
75
76                cmds = f'ScatsHistoryViewer.exe -files Regions\\{region}\\SCATSData\\History\\{hst_file_name}.hst -sites all -e -out {dst_folder}\\{hst_file_name}-all-events.csv'
77
78                print(cmds)
79                # print(os.getcwd())
```

Figure 42. Snapshot of Core Query in the Program

Here, “f'ScatsHistoryViewer.exe” means to open the ScatsHitoryViewer software. “-files Regions\\{region}\\SCATSData\\History\\{hst_file_name}.hst” means to find the locations of the historical data with the regional information. You can modify the path of the historical data, but PLEASE NOTE: the path must be the same as your historical data location. So, if you find your historical data files have a different path from the path indicated above, please modify either the path of the whole historical data file folder or the path above in line 76 in this file to make sure they are the same. You don’t need to specify the region and file name in the query. The sample historical data is shown in the figure:

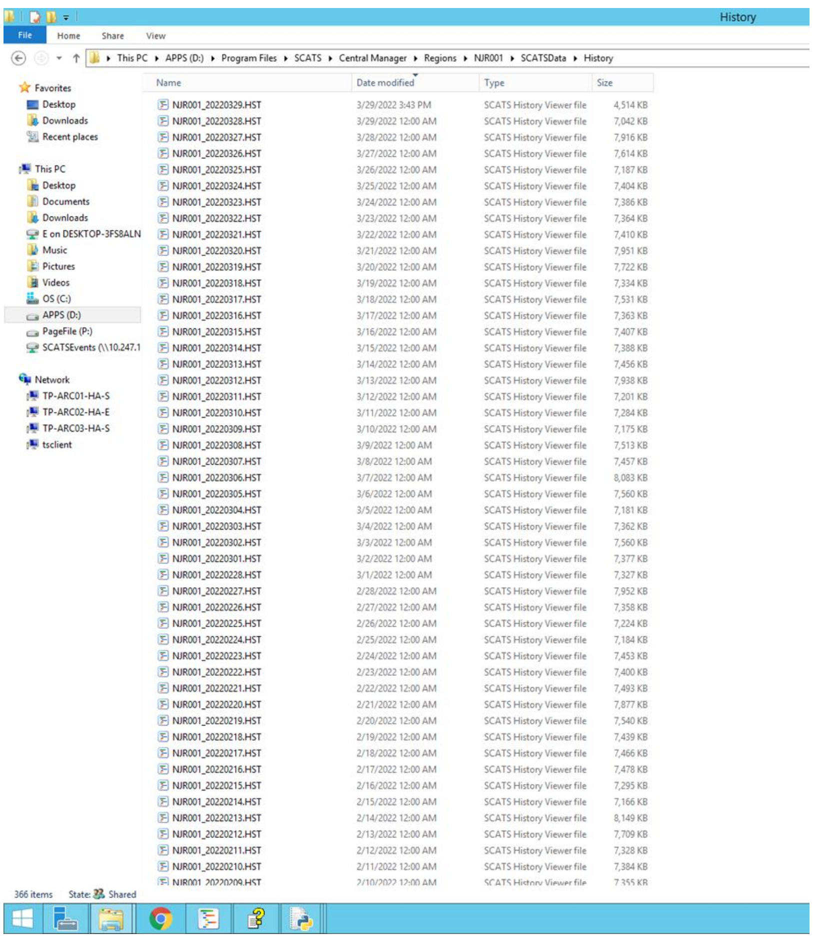


Figure 43. Snapshot of the .hst Historical Data File

Continuing back to the query in line 76, "-sites all" means to get all sites in the historical file. You may not need to modify this one. "-e" means the EventTypeIDs; you may not need to modify this one. "-out {dst_folder}\\{hst_file_name}-all-events. CSV" means to output the .csv excel files to your destination path that you identified in Step 2. You don't need to modify any characters in curly braces because they are already defined before. You can change the "-all-events" parts here to change the name of the exported .csv excel files directly.

SCATS and ATSPM Server Configuration and Archiving

QuickStartGuide:
Step 0. If there are existing files you archived before and want to keep them, please go to the archive folder and rename the folder Archive, e.g., "Archive + Date," and create a new empty Archive folder (the code will create region subfolders).
Otherwise, the code will delete all existing files in the region subfolders and overwrite them.
Step 1. Open Command Prompt: Open Start Menu, → search **Command Prompt**→ click **Command Prompt** icon.
Step 2. Type in cd D:\ATSPM\Program/Program
This query gets to the program folder
Step 3. Type in Python SCATS_ATSPM_hst_archive.py
It runs the python code that archives the SCATS event files.
Step 4. There may be some warnings that you can ignore if you don't see any errors. Congratulations! The code will be working and will archive the data you want. This program may take a few minutes to finish, so please let it run until it finishes.

If you need to set up path or server information, please follow the steps below or find one that describes your demand.
Step 1: Copy Python Program Files for bulk loading into a project folder (e.g., ATSPM) on the target server. Note: your path must be the same as entered in the command prompt

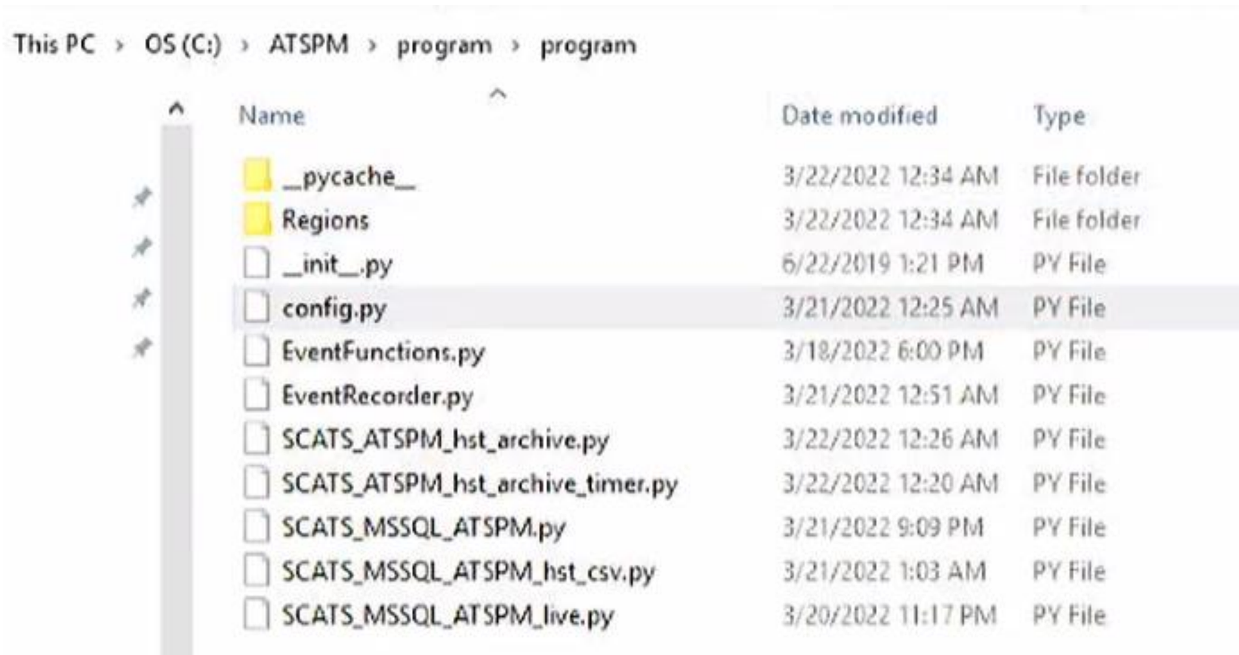


Figure 44. Snapshot of Python Program Files for bulk loading

Step 2: Load all the historical event files to a designated folder. On the ATSPM server, there is an Archive folder under the SCATS Event folder reserved for bulk loading.
Bulk loading time: Due to the complexity of the event logic, a one-day SCATS hist file of 30 intersections takes around one hour to run on the ATSPM server.

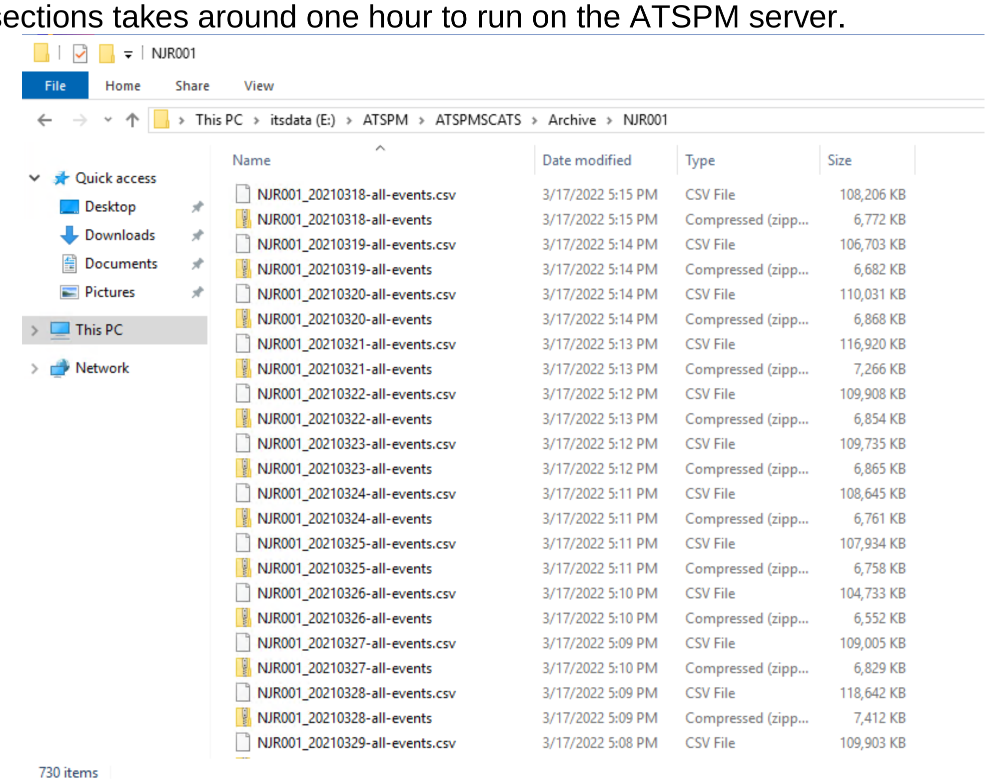


Figure 45. Snapshot of Events Archive Folder

Step 3: For the first time running the code, here are the necessary python packages to be installed. Each line means a package of the Python environment. Among them, the dependencies of line 10 to line 13 come from the config.py file; therefore, it is crucial to ensure the integrity of these files. The others are the necessary python environment to run these files. Please download and ensure the integrity of these python packages on your computer if you don't have them before.

```
 2  import numpy as np
 3  import datetime
 4  import os
 5  import pandas as pd
 6  import sys
 7  import pdb
 8  import pyodbc
 9  import time
10  from config import DatabaseConfig
11  from config import FileFolders
12  from config import DatabaseTbls
13  import config
14  import re
15  import glob
```

Figure 46. Snapshot of Necessary Python Packages and Environment

Additional package to install after standard python installation

> pip install pandas

```
C:\Users\Administrator>pip install pandas
Collecting pandas
  Downloading pandas-1.4.1-cp39-cp39-win_amd64.whl (10.5 MB)
     |████████████████████████████████| 10.5 MB ...
Collecting python-dateutil>=2.8.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     |████████████████████████████████| 247 kB ...
Collecting pytz>=2020.1
  Downloading pytz-2022.1-py2.py3-none-any.whl (503 kB)
     |████████████████████████████████| 503 kB ...
Requirement already satisfied: numpy>=1.18.5 in c:\users\administrator\appdata\local\programs\python\python39\lib\s
ackages (from pandas) (1.22.3)
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pytz, python-dateutil, pandas
Successfully installed pandas-1.4.1 python-dateutil-2.8.2 pytz-2022.1 six-1.16.0
WARNING: You are using pip version 21.2.4; however, version 22.0.4 is available.
You should consider upgrading via the 'C:\Users\Administrator\AppData\Local\Programs\Python\Python39\python.exe -m
nstall --upgrade pip' command.

C:\Users\Administrator>
```

>pip install pyodbc

```
C:\Users\Administrator>pip install pyodbc
Collecting pyodbc
  Downloading pyodbc-4.0.32-cp39-cp39-win_amd64.whl (72 kB)
     ---------------------------------------- 73.0/73.0 KB ? eta 0:00:00
Installing collected packages: pyodbc
Successfully installed pyodbc-4.0.32
```

Figure 47. Snapshots of Python Packages installation

Step 4: Confirm and modify database access information using a different ATSPM server. Please open the config.py file and go from line 3 to line 7, and you can input the database access information there. Please keep the double quotes and only replace the words inside with your access information when you update your information.

C:\Users\49762\AppData\Local\Temp\Temp1_program.zip\program\config.py • - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

SCATS_MSSQL_ATSPM.py | config.py

```
1  class DatabaseConfig:
2
3      server_driver = "Please indicate your server_driver here!"
4      server = "Please indicate your server here!"
5      database = "Please indicate your database here!"
6      username = "Please input your database username here!"
7      password = "Please input your database password here!"
8
```

Figure 48. Snapshot of Database Info Modification

Step 5: Confirm and modify paths for metafile, region information, and data bulk loading if using different paths in your computer. Please open the config.py file and go to line 15 to line 18, and you can input the corresponding path information for a different type of archiving and the region information there. In line 15, you can modify the paths for the Folder of SCATS Live .csv files. The path format should be: "Drive:\\Folder\\Subfolder\\...\\". In line 16, you can modify the Folder of SCATS metafiles paths. The path format should be: "Drive:\\Folder\\Subfolder\\...\\". In line 17, you can modify the paths for the Folder of archiving SCATS files. The path format should be: "Drive:\\Folder\\Subfolder\\...\\". In line 18, you can modify which regions you want. The path format should be: "Region1, Region2, …". When you update your information, please keep the double quotes and only replace the words inside with your inputs.

```
13
14  class FileFolders:
15      SCATSCsvLiveFolder = "Please input your path here: Drive:\\Folder\\Subfolder\\...\\"
16      SCATSMetaFileFolder = "Please input your path here: Drive:\\Folder\\Subfolder\\...\\"
17      SCATSArchiveFolder = "Please input your path here: Drive:\\Folder\\Subfolder\\...\\"
18      Regions = "Please input the region here!" #e.g. NJR001...
19
```

Figure 49**.** Snapshot of Paths and Region Info Modification

Step 6: You finished the SCATS and ATSPM Server configuration! It should be a preprocessing file that you should finish before data archiving.

SCATS Signal Event Conversion

In the SCATS log files, all events have their unique identity code. The project team uses the SCATS message of events, phase status, and timestamps to build up a SCATS translator for archiving all the events in order.

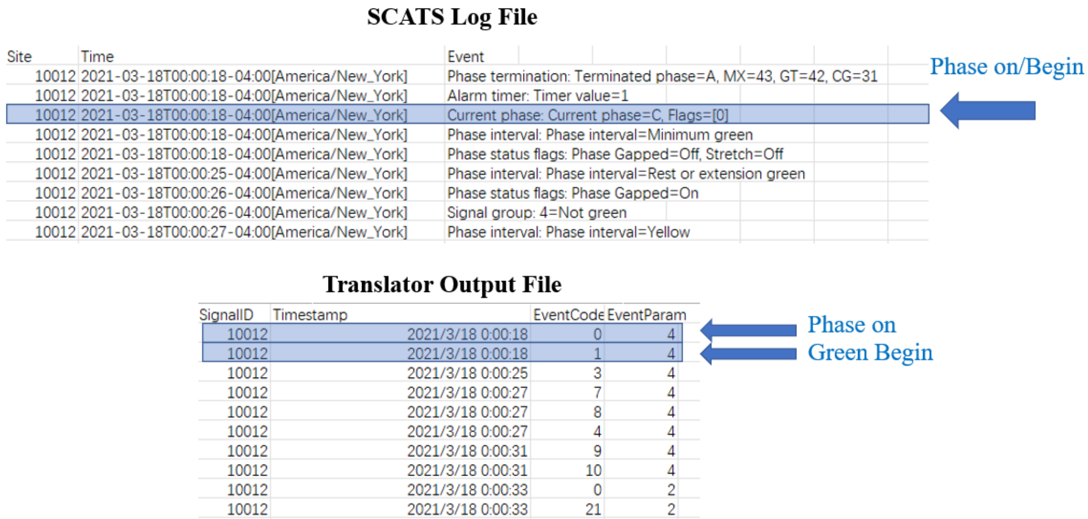
Table 1 shows the SCATS signal event and corresponding event code and Translator logic in the SCATS message. When looking for these events in the log file, the premise is that these events occur within the target phase time

Table 1. Convertible SCATS Signal Event

Event Code	Event	SCATS Translator Logic
0	Phase On	"Current Phase" in SCATS message
1	Phase Begin Green	" Current Phase" in SCATS message
2	Phase Check	"Phase demand" in SCATS message
3	Phase Min Complete	calculate the phase min complete using the minimum green value from the metadata file
4	Phase Gap-Out	Green Duration < Maximum Green
5	Phase Max-Out	Green Duration > Maximum Green
7	Phase Green Termination	"Phase interval: Yellow" in SCATS message
8	Phase Begin Yellow Clearance	"Phase interval: Yellow" in SCATS message
9	Phase End Yellow Clearance	"Phase interval: All Red" in SCATS message
10	Phase Begin Red Clearance	"Phase interval: All Red" in SCATS message
11	Phase End Red Clearance	keyword: "Phase termination."
12	Phase Inactive	If a movement does not exist in a specific cycle, then create Phase Inactive
21	Pedestrian Begin Walk	keywords: "Walk" + "Active=On"
22	Pedestrian Begin Clearance	Keywords: "Walk" + "Active=Off"
45	Pedestrian Call Registered	keywords: "Walk" + "Demand=On"

Event #0: Phase On and #1: GreenBegin

For each phase in an intersection, the phase starts at the beginning of greenlight. Therefore, Event#0: Phase On and #1: GreenBegin has the same translator logic in the SCATS message, and they should appear simultaneously in the SCATS log files. The same message statement identifies their program logic: for event #0, 'If "Current Phase = X" is in the log event i, create event #0 for movement m that in the phase X in translator output'. $\mathrm{t_{m,PhaseOn}} = \mathrm{t_i}$; For event #1, that is 'If "Current Phase = X" is in the log event i, create event #1 for movement m that in the phase X in translator output'. $\mathrm{t_{m,GreenBegin}} = \mathrm{t_i}$.



SCATS Log File

Site	Time	Event
10012	2021-03-18T00:00:18-04:00[America/New_York]	Phase termination: Terminated phase=A, MX=43, GT=42, CG=31
10012	2021-03-18T00:00:18-04:00[America/New_York]	Alarm timer: Timer value=1
10012	2021-03-18T00:00:18-04:00[America/New_York]	Current phase: Current phase=C, Flags=[0]
10012	2021-03-18T00:00:18-04:00[America/New_York]	Phase interval: Phase interval=Minimum green
10012	2021-03-18T00:00:18-04:00[America/New_York]	Phase status flags: Phase Gapped=Off, Stretch=Off
10012	2021-03-18T00:00:25-04:00[America/New_York]	Phase interval: Phase interval=Rest or extension green
10012	2021-03-18T00:00:26-04:00[America/New_York]	Phase status flags: Phase Gapped=On
10012	2021-03-18T00:00:26-04:00[America/New_York]	Signal group: 4=Not green
10012	2021-03-18T00:00:27-04:00[America/New_York]	Phase interval: Phase interval=Yellow

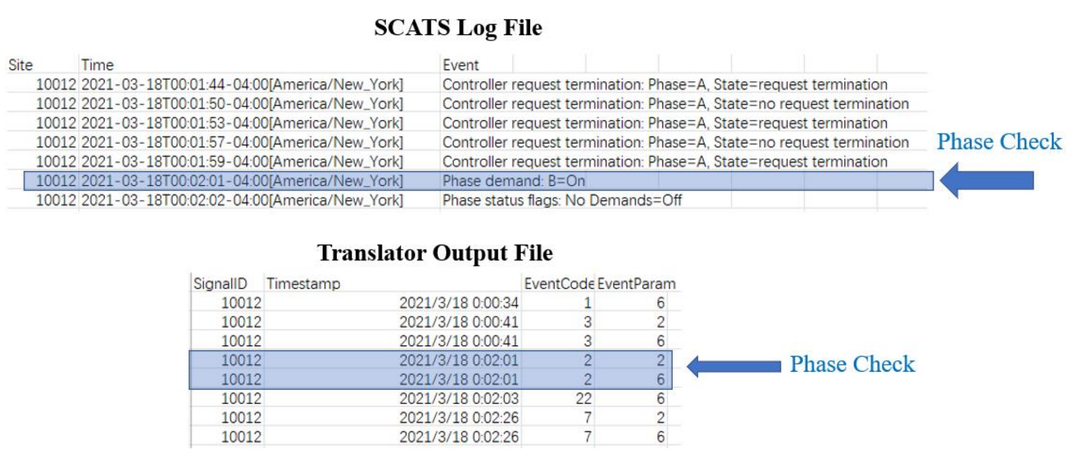
Translator Output File

SignalID	Timestamp	EventCode	EventParam
10012	2021/3/18 0:00:18	0	4
10012	2021/3/18 0:00:18	1	4
10012	2021/3/18 0:00:25	3	4
10012	2021/3/18 0:00:27	7	4
10012	2021/3/18 0:00:27	8	4
10012	2021/3/18 0:00:27	4	4
10012	2021/3/18 0:00:31	9	4
10012	2021/3/18 0:00:31	10	4
10012	2021/3/18 0:00:33	0	2
10012	2021/3/18 0:00:33	21	2

Figure 50. SCATS Log File and Corresponding Event #0 and Event #1 in Translator Output

Event #2: PhaseCheck

PhaseCheck event is used to determine whether a phase is needed. We need to check the demand of a phase is 'On' while another phase is 'On.' Therefore, the program logic of event #2 is identified by: 'If phase $PhaseOn_{X,m} = True$, and log event i = "Phase demand: $X' = On$," then create PhaseCheck for movement m in translator output'. $t_{m,PhaseCheck} = t_i$



SCATS Log File

Site	Time	Event
10012	2021-03-18T00:01:44-04:00[America/New_York]	Controller request termination: Phase=A, State=request termination
10012	2021-03-18T00:01:50-04:00[America/New_York]	Controller request termination: Phase=A, State=no request termination
10012	2021-03-18T00:01:53-04:00[America/New_York]	Controller request termination: Phase=A, State=request termination
10012	2021-03-18T00:01:57-04:00[America/New_York]	Controller request termination: Phase=A, State=no request termination
10012	2021-03-18T00:01:59-04:00[America/New_York]	Controller request termination: Phase=A, State=request termination
10012	2021-03-18T00:02:01-04:00[America/New_York]	Phase demand: B=On
10012	2021-03-18T00:02:02-04:00[America/New_York]	Phase status flags: No Demands=Off

Translator Output File

SignalID	Timestamp	EventCode	EventParam
10012	2021/3/18 0:00:34	1	6
10012	2021/3/18 0:00:41	3	2
10012	2021/3/18 0:00:41	3	6
10012	2021/3/18 0:02:01	2	2
10012	2021/3/18 0:02:01	2	6
10012	2021/3/18 0:02:03	22	6
10012	2021/3/18 0:02:26	7	2
10012	2021/3/18 0:02:26	7	6

Figure 51. SCATS Log File and Corresponding Event #2 in Translator Output

Event #3: Phase Min Complete

Event #3 is used to record the timestamp of minimum green time in the log file. We use this event to calculate the phase minimum green. First, compare the phase with minimum green by using timestamps in the log file. The duration is based on the time interval between the phase beginning and the timestamp of the phase turning to yellow. Create the record if the duration is longer than the minimum green time. Therefore, the program logic of event #3 is identified by: If the duration of time $d_m = t_{m,\,PhaseGreenTerminate}$- $t_{m,\,PhaseOn} > m_{MinimumGreen}$, then create Phase Min Complete event for all movement m in translator output'. $t_{m,\,MinimumGreen} = t_i^{(SCATS)} + t_{m,MinimumGreen}$.

SCATS Log File

Site	Time	Event
10012	2021-03-18T00:00:34-04:00[America/New_York]	Phase interval: Phase interval=Minimum green
10012	2021-03-18T00:00:34-04:00[America/New_York]	Phase status flags: No Demands=On, Phase Gapped=Off, Stretch=On
10012	2021-03-18T00:00:40-04:00[America/New_York]	Controller request termination: Phase=A, State=request termination
10012	2021-03-18T00:00:41-04:00[America/New_York]	Phase interval: Phase interval=Rest or extension green
10012	2021-03-18T00:00:41-04:00[America/New_York]	Phase status flags: Request Termination=On
10012	2021-03-18T00:00:41-04:00[America/New_York]	Controller request termination: Phase=A, State=no request termination
10012	2021-03-18T00:00:42-04:00[America/New_York]	Phase status flags: Request Termination=Off
10012	2021-03-18T00:00:43-04:00[America/New_York]	Controller request termination: Phase=A, State=request termination
10012	2021-03-18T00:00:44-04:00[America/New_York]	Phase status flags: Request Termination=On

Phase Min Complete

Translator Output File

SignalID	Timestamp	EventCode	EventParam
10012	2021/3/18 0:00:34	1	6
10012	2021/3/18 0:00:41	3	2
10012	2021/3/18 0:00:41	3	6
10012	2021/3/18 0:02:01	2	2
10012	2021/3/18 0:02:01	2	6
10012	2021/3/18 0:02:03	22	6
10012	2021/3/18 0:02:26	7	2
10012	2021/3/18 0:02:26	7	6

Phase Min Complete

Figure 52. SCATS Log File and Corresponding Event #3 in Translator Output

Event #4: Phase Gap-out and Event #5: Phase Max-out

Events #4 and #5 are used to evaluate the green time duration. We can create Phase gap-out, phase max-out, and even phase force-off entries by comparing actual green duration time to maximum phase green time. There are different event description messages in the original log file to indicate how the phase ends instead of calculating the actual duration of phase green. Therefore, we can search the SCATS message directly for the program logic of event #4 and event #5: 'If "Phase Gapped=On" or "No Demands=On" in the log file, create event #4 Phase Gap-out for movement m that in the phase X in translator output.' $t_{m,\,GapOut} = t_i^{SCATS}$; 'If "Mx Ack=Off" in or "Max Due=Off" in the log file, create event #4 Phase Gap-out for movement m that in the phase X in translator output.' $t_{m,\,GapOut} = t_i^{SCATS}$; 'If "Cycle generator" in the log file, create Phase Force-off for movement m that in the phase X in translator output.' $t_{m,Forceoff} = t_i^{SCATS}$; If no corresponding messages appear, create event #5 Phase Max-out for movement m that in the phase X in translator output.' $t_{m,\,Maxout} = t_i^{SCATS}$;

SCATS Log File

Site	Time	Event
10012	2021-03-18T00:00:25-04:00[America/New_York]	Phase interval: Phase interval=Rest or extension green
10012	2021-03-18T00:00:26-04:00[America/New_York]	Phase status flags: Phase Gapped=On
10012	2021-03-18T00:00:26-04:00[America/New_York]	Signal group: 4=Not green
10012	2021-03-18T00:00:27-04:00[America/New_York]	Phase interval: Phase interval=Yellow
10012	2021-03-18T00:00:31-04:00[America/New_York]	Phase interval: Phase interval=All red
10012	2021-03-18T00:00:33-04:00[America/New_York]	Phase demand: A=Off
10012	2021-03-18T00:00:33-04:00[America/New_York]	Signal group: 2=Green, 6=Green, 18=Green
10012	2021-03-18T00:00:33-04:00[America/New_York]	Pedestrian movement (Region 6.9.4+): Ped 2=[Demand=Off, Interval=Walk]
10012	2021-03-18T00:00:34-04:00[America/New_York]	Phase termination: Terminated phase=C, MX=65, GT=16, CG=47

Phase Gap-out

Translator Output File

SignalID	Timestamp	EventCode	EventParam
10012	2021/3/18 0:00:18	0	4
10012	2021/3/18 0:00:18	1	4
10012	2021/3/18 0:00:25	3	4
10012	2021/3/18 0:00:27	7	4
10012	2021/3/18 0:00:27	8	4
10012	2021/3/18 0:00:27	4	4
10012	2021/3/18 0:00:31	9	4
10012	2021/3/18 0:00:31	10	4
10012	2021/3/18 0:00:33	0	2

Phase Gap-out

Figure 53. SCATS Log File and Corresponding Event #4 in Translator Output

SCATS Log File

Site	Time	Event
10012	2021-03-18T00:02:25-04:00[America/New_York]	Signal group: 2=Not green, 6=Not green
10012	2021-03-18T00:02:26-04:00[America/New_York]	Phase interval: Phase interval=Yellow
10012	2021-03-18T00:02:31-04:00[America/New_York]	Phase demand: A=On
10012	2021-03-18T00:02:32-04:00[America/New_York]	Phase interval: Phase interval=All red
10012	2021-03-18T00:02:33-04:00[America/New_York]	Phase demand: B=Off
10012	2021-03-18T00:02:33-04:00[America/New_York]	Signal group: 3=Green
10012	2021-03-18T00:02:33-04:00[America/New_York]	Pedestrian movement (Region 6.9.4+): Ped 2=[Demand=On]
10012	2021-03-18T00:02:34-04:00[America/New_York]	Phase termination: Terminated phase=A, MX=-32, GT=120, CG=67
10012	2021-03-18T00:02:34-04:00[America/New_York]	Alarm timer: Timer value=1

Phase Max-out

Translator Output File

SignalID	Timestamp	EventCode	EventParam
10012	2021/3/18 0:02:26	8	2
10012	2021/3/18 0:02:26	8	6
10012	2021/3/18 0:02:26	5	2
10012	2021/3/18 0:02:26	5	6
10012	2021/3/18 0:02:31	2	2
10012	2021/3/18 0:02:31	2	6
10012	2021/3/18 0:02:32	9	2
10012	2021/3/18 0:02:32	9	6
10012	2021/3/18 0:02:32	10	2

Phase Max-out

Figure 54. SCATS Log File and Corresponding Event #5 in Translator Output

Event #7: Phase Green Termination

Event #7 is used to record the timestamp of the green light starts turning to yellow. That is, the termination of the green light occurs at the moment when the signal light turns yellow. Therefore, green termination is defined in the log file and occurs at the timestamp recorded by the ‘Phase interval: Yellow’ message. Program logic of event #7 is identified by: ‘If log event i is “Phase interval=Yellow” and movement $PhaseOn_{X,m} = True$, Create Phase Green Termination in translator output’. $t_{m,\, PhaseGreenTermination} = t_i^{SCATS}$

SCATS Log File

Site	Time	Event	
10012	2021-03-18T00:00:25-04:00[America/New_York]	Phase interval: Phase interval=Rest or extension green	
10012	2021-03-18T00:00:26-04:00[America/New_York]	Phase status flags: Phase Gapped=On	
10012	2021-03-18T00:00:26-04:00[America/New_York]	Signal group: 4=Not green	
10012	2021-03-18T00:00:27-04:00[America/New_York]	Phase interval: Phase interval=Yellow	Phase Termination
10012	2021-03-18T00:00:31-04:00[America/New_York]	Phase interval: Phase interval=All red	
10012	2021-03-18T00:00:33-04:00[America/New_York]	Phase demand: A=Off	
10012	2021-03-18T00:00:33-04:00[America/New_York]	Signal group: 2=Green, 6=Green, 18=Green	
10012	2021-03-18T00:00:33-04:00[America/New_York]	Pedestrian movement (Region 6.9.4+): Ped 2=[Demand=Off, Interval=Walk]	
10012	2021-03-18T00:00:34-04:00[America/New_York]	Phase termination: Terminated phase=C, MX=65, GT=16, CG=47	

Translator Output File

SignalID	Timestamp	EventCode	EventParam	
10012	2021/3/18 0:00:18	0	4	
10012	2021/3/18 0:00:18	1	4	
10012	2021/3/18 0:00:25	3	4	
10012	2021/3/18 0:00:27	7	4	Phase Termination
10012	2021/3/18 0:00:27	8	4	
10012	2021/3/18 0:00:27	4	4	
10012	2021/3/18 0:00:31	9	4	
10012	2021/3/18 0:00:31	10	4	
10012	2021/3/18 0:00:33	0	2	

Figure 55. SCATS Log File and Corresponding Event #7 in Translator Output

Event #8: Phase Begin Yellow Clearance and #9: Phase End Yellow Clearance

Events #8 and #9 identify the yellow time interval. The time point of Begin Yellow Clearance is the timestamp of ‘Phase interval: Yellow’ in the SCATS log file. The time point of End Yellow Clearance is when the yellow light turns red, which is the timestamp of Phase interval: All Red in the log file. The difference between them is the duration time of the yellow light. Therefore, for the program logic of events #8 and #9, we only need to search the corresponding SCATS message: ‘If log event i is “Phase interval=Yellow,” Create Phase Begin Yellow Clearance in translator output.’ $t_{m,YellowBegin} = t_i^{(SCATS)}$; ‘If log event i is “Phase interval=All Red,” Create Phase End Yellow Clearance in translator output.’ $t_{m,YellowEnd} = t_i^{(SCATS)}$

SCATS Log File

Site	Time	Event	
10012	2021-03-18T00:00:25-04:00[America/New_York]	Phase interval: Phase interval=Rest or extension green	
10012	2021-03-18T00:00:26-04:00[America/New_York]	Phase status flags: Phase Gapped=On	
10012	2021-03-18T00:00:26-04:00[America/New_York]	Signal group: 4=Not green	
10012	2021-03-18T00:00:27-04:00[America/New_York]	Phase interval: Phase interval=Yellow	Phase Begin Yellow Clearance
10012	2021-03-18T00:00:31-04:00[America/New_York]	Phase interval: Phase interval=All red	Phase End Yellow Clearance
10012	2021-03-18T00:00:33-04:00[America/New_York]	Phase demand: A=Off	
10012	2021-03-18T00:00:33-04:00[America/New_York]	Signal group: 2=Green, 6=Green, 18=Green	
10012	2021-03-18T00:00:33-04:00[America/New_York]	Pedestrian movement (Region 6.9.4+): Ped 2=[Demand=Off, Interval=Walk]	
10012	2021-03-18T00:00:34-04:00[America/New_York]	Phase termination: Terminated phase=C, MX=65, GT=16, CG=47	

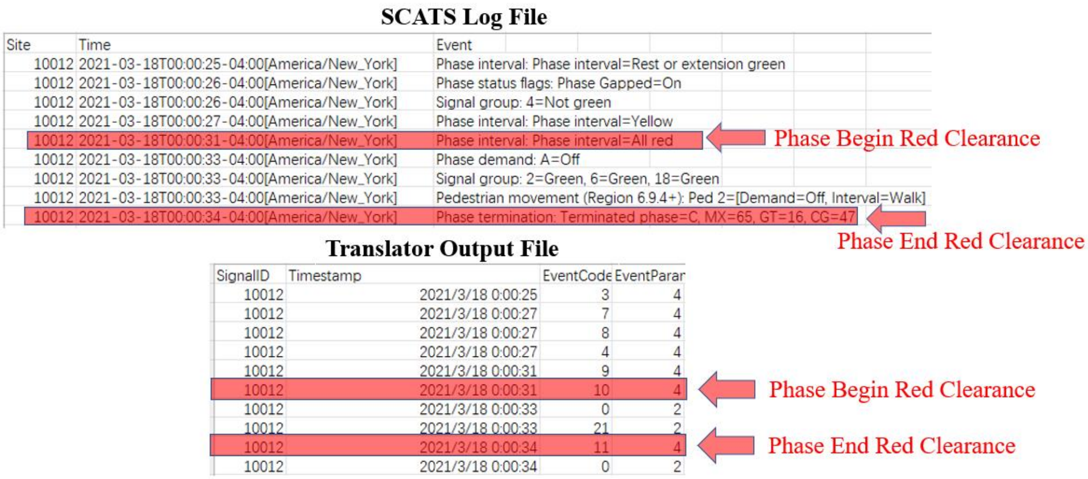
Translator Output File

SignalID	Timestamp	EventCode	EventParam	
10012	2021/3/18 0:00:18	0	4	
10012	2021/3/18 0:00:18	1	4	
10012	2021/3/18 0:00:25	3	4	
10012	2021/3/18 0:00:27	7	4	
10012	2021/3/18 0:00:27	8	4	Phase Begin Yellow Clearance
10012	2021/3/18 0:00:27	4	4	
10012	2021/3/18 0:00:31	9	4	Phase End Yellow Clearance
10012	2021/3/18 0:00:31	10	4	
10012	2021/3/18 0:00:33	0	2	

Figure 56. SCATS Log File and Corresponding Event #8 and Event #9 in Translator Output

Event #10: Phase Begin End Clearance and #11: Phase End Red Clearance

Events #10 and #11 are used to identify the red time interval. The time point of Begin Red Clearance is the timestamp of 'Phase interval: All Red' in the SCATS log file. The time point of End Red Clearance is the end of a whole phase, which is the termination of a phase in the log file. The difference between them is the duration time of the red light. Therefore, for the program logic of events #10 and #11, we only need to search the corresponding SCATS message: 'If log event i is "Phase interval=All Red," Create Phase Begin Red Clearance in translator output.' $t_{m,RedBegin} = t_i^{(SCATS)}$; 'If log event i is "Phase termination," Create Phase End Red Clearance in translator output.' $t_{m,RedEnd} = t_i^{(SCATS)}$



Site	Time	Event
10012	2021-03-18T00:00:25-04:00[America/New_York]	Phase interval: Phase interval=Rest or extension green
10012	2021-03-18T00:00:26-04:00[America/New_York]	Phase status flags: Phase Gapped=On
10012	2021-03-18T00:00:26-04:00[America/New_York]	Signal group: 4=Not green
10012	2021-03-18T00:00:27-04:00[America/New_York]	Phase interval: Phase interval=Yellow
10012	2021-03-18T00:00:31-04:00[America/New_York]	Phase interval: Phase interval=All red
10012	2021-03-18T00:00:33-04:00[America/New_York]	Phase demand: A=Off
10012	2021-03-18T00:00:33-04:00[America/New_York]	Signal group: 2=Green, 6=Green, 18=Green
10012	2021-03-18T00:00:33-04:00[America/New_York]	Pedestrian movement (Region 6.9.4+): Ped 2=[Demand=Off, Interval=Walk]
10012	2021-03-18T00:00:34-04:00[America/New_York]	Phase termination: Terminated phase=C, MX=65, GT=16, CG=47

SignalID	Timestamp	EventCode	EventParar
10012	2021/3/18 0:00:25	3	4
10012	2021/3/18 0:00:27	7	4
10012	2021/3/18 0:00:27	8	4
10012	2021/3/18 0:00:27	4	4
10012	2021/3/18 0:00:31	9	4
10012	2021/3/18 0:00:31	10	4
10012	2021/3/18 0:00:33	0	2
10012	2021/3/18 0:00:33	21	2
10012	2021/3/18 0:00:34	11	4
10012	2021/3/18 0:00:34	0	2

Figure 57. SCATS Log File and Corresponding Event #10 and Event #11 in Translator Output

Event #21: Pedestrian Begin Walk and #22: Pedestrian Begin Clearance

Events #21 and #22 identify whether pedestrians are crossing the road. There is a special message in the log file that indicates the Pedestrian movement. In addition, the joint control of demand and walk parameters indicates whether pedestrians start to cross the road. Use clearance to mark whether pedestrians have completed crossing the road. Therefore, the program logic of events #21 and #22 are: Searching for the keyword "Pedestrian movement." Then, if "Demand = Off" and "Interval=Walk," Create Pedestrian Begin Walk-in translator output'. $t_{m,PedBeginWalk} = t_i^{(SCATS)}$; if "Interval=Clearance," Create Pedestrian Begin Clearance in translator output. Including the number of pedestrians and the phase number'. $t_{m,PedBeginClearance} = t_i^{(SCATS)}, N_{m,ped} = n_i^{(SCATS)}$.

SCATS Log File

Site	Time	Event
10012	2021-03-18T00:00:26-04:00[America/New_York]	Signal group: 4=Not green
10012	2021-03-18T00:00:27-04:00[America/New_York]	Phase interval: Phase interval=Yellow
10012	2021-03-18T00:00:31-04:00[America/New_York]	Phase interval: Phase interval=All red
10012	2021-03-18T00:00:33-04:00[America/New_York]	Phase demand: A=Off
10012	2021-03-18T00:00:33-04:00[America/New_York]	Signal group: 2=Green, 6=Green, 18=Green
10012	2021-03-18T00:00:33-04:00[America/New_York]	Pedestrian movement (Region 6.9.4+): Ped 2=[Demand=Off, Interval=Walk]
10012	2021-03-18T00:00:34-04:00[America/New_York]	Phase termination: Terminated phase=C, MX=65, GT=16, CG=47
10012	2021-03-18T00:00:34-04:00[America/New_York]	Alarm timer: Timer value=1
10012	2021-03-18T00:00:34-04:00[America/New_York]	Current phase: Current phase=A, Flags=[1]

Pedestrian Begin Walk

Translator Output File

SignalID	Timestamp	EventCode	EventParam
10012	2021/3/18 0:00:31	10	4
10012	2021/3/18 0:00:33	0	2
10012	2021/3/18 0:00:33	21	2
10012	2021/3/18 0:00:34	11	4
10012	2021/3/18 0:00:34	0	2
10012	2021/3/18 0:00:34	0	6
10012	2021/3/18 0:00:34	1	2
10012	2021/3/18 0:00:34	1	6
10012	2021/3/18 0:00:41	3	2

Pedestrian Begin Walk

Figure 58. SCATS Log File and Corresponding Event #21 in Translator Output

SCATS Log File

Site	Time	Event
10012	2021-03-18T00:02:02-04:00[America/New_York]	Phase termination request: Current phase=B
10012	2021-03-18T00:02:02-04:00[America/New_York]	Phase termination request confirmation: Current phase=A
10012	2021-03-18T00:02:03-04:00[America/New_York]	Phase status flags: Mx Ack=On
10012	2021-03-18T00:02:03-04:00[America/New_York]	Phase termination request: Current phase=B
10012	2021-03-18T00:02:03-04:00[America/New_York]	Signal group: 18=Not green
10012	2021-03-18T00:02:03-04:00[America/New_York]	Pedestrian movement (Region 6.9.4+): Ped 2=[Interval=Clearance 2]
10012	2021-03-18T00:02:04-04:00[America/New_York]	Phase termination request: Current phase=B
10012	2021-03-18T00:02:05-04:00[America/New_York]	Phase termination request: Current phase=B
10012	2021-03-18T00:02:06-04:00[America/New_York]	Phase termination request: Current phase=B

Pedestrian Begin Clearance

Translator Output File

SignalID	Timestamp	EventCode	EventParam
10012	2021/3/18 0:02:01	2	6
10012	2021/3/18 0:02:03	22	6
10012	2021/3/18 0:02:26	7	2
10012	2021/3/18 0:02:26	7	6
10012	2021/3/18 0:02:26	8	2
10012	2021/3/18 0:02:26	8	6
10012	2021/3/18 0:02:26	5	2
10012	2021/3/18 0:02:26	5	6
10012	2021/3/18 0:02:31	2	2

Pedestrian Begin Clearance

Figure 59. SCATS Log File and Corresponding Event #22 in Translator Output

Event #45: Pedestrian Call Registered

Event #45 is used to indicate if there is a need for pedestrians to cross the road. There is a special message in the log file that indicates the Pedestrian movement. In addition, the control of the "Demand" parameter indicates whether pedestrians are waiting to cross the road. Therefore, the program logic of event #45 is: Searching for the keyword "Pedestrian movement." Then, if "Demand = On," Create Pedestrian Call Registered in translator output, including the number of pedestrians and the phase number.' $t_{m,PedCall} = t_i^{(SCATS)}, N_{m,ped} = n_i^{(SCATS)}$

SCATS Log File

Site	Time	Event
10012	2021-03-18T00:02:26-04:00[America/New_York]	Phase interval: Phase interval=Yellow
10012	2021-03-18T00:02:31-04:00[America/New_York]	Phase demand: A=On
10012	2021-03-18T00:02:32-04:00[America/New_York]	Phase interval: Phase interval=All red
10012	2021-03-18T00:02:33-04:00[America/New_York]	Phase demand: B=Off
10012	2021-03-18T00:02:33-04:00[America/New_York]	Signal group: 3=Green
10012	2021-03-18T00:02:33-04:00[America/New_York]	Pedestrian movement (Region 6.9.4+): Ped 2=[Demand=On]
10012	2021-03-18T00:02:34-04:00[America/New_York]	Phase termination: Terminated phase=A, MX=-32, GT=120, CG=67
10012	2021-03-18T00:02:34-04:00[America/New_York]	Alarm timer: Timer value=1
10012	2021-03-18T00:02:34-04:00[America/New_York]	Current phase: Current phase=B, Flags=[0]

Pedestrian Call Registered

Translator Output File

SignalID	Timestamp	EventCode	EventParam
10012	2021/3/18 0:02:32	10	6
10012	2021/3/18 0:02:33	45	6
10012	2021/3/18 0:02:34	11	2
10012	2021/3/18 0:02:34	11	6
10012	2021/3/18 0:02:34	0	3
10012	2021/3/18 0:02:34	1	3
10012	2021/3/18 0:02:41	3	3
10012	2021/3/18 0:02:41	7	3
10012	2021/3/18 0:02:41	8	3

Pedestrian Call Registered

Figure 60. SCATS Log File and Corresponding Event #45 in Translator Output

SCATS EVENT Archiving Programs

Three archiving programs have been developed to archive SCATS files:

1. History File Archiving:

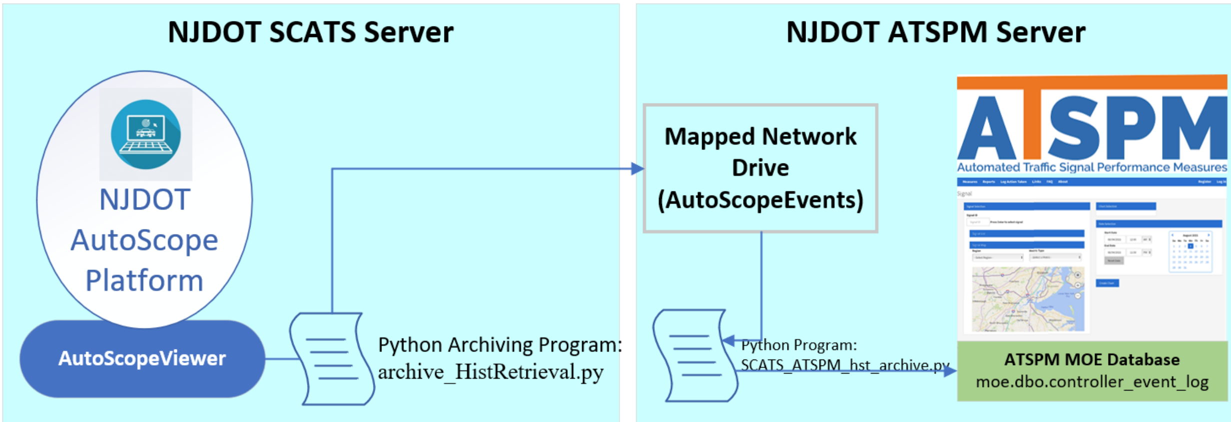


Figure 61. History File Archiving Process

- The SCATS history files are saved on the NJDOT SCATS server in .hst file format.
- The NJDOT ATSPM server maps a shared location to a drive letter on NJDOT SCATS Server, giving far more space than the NJDOT SCATS server would otherwise have access to it.
- The SCATS .hst files were converted into .csv files by running commands with SCATSHistoryViewer.exe and saving the one-year event files onto the shared drive.
- Once the event extraction is completed, we will run the python program to translate the SCATS event file into ATSPM event code and ingest it into ATSPM MOE Database

2. 15-min Live SCATS Records:

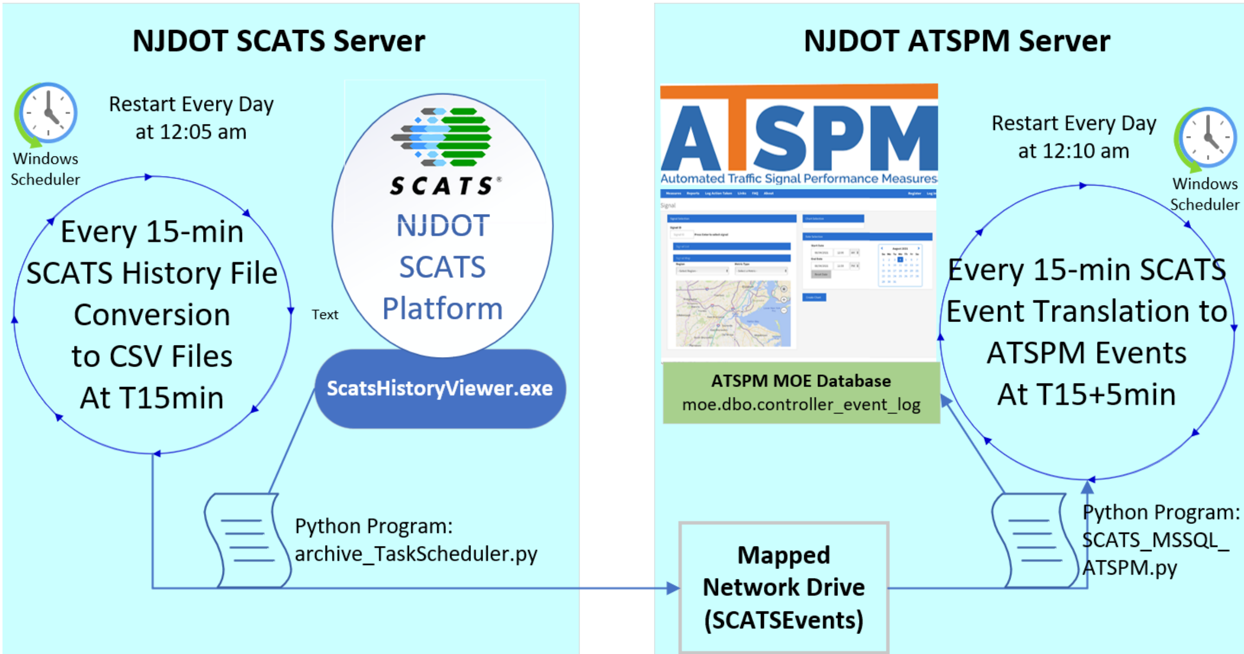


Figure 62. 15-min Live SCATS Records Archiving Process

- 15-min SCATS archiving and translator codes run every 15 minutes by Task Scheduler to update the live signal timing and phase event.
- The archiving program automatically ran at 12:05 AM every day and was repeated at 15-minute intervals. They live 15 minutes SCATS events are stored at the mapped network drive by the NJDOT ATSPM server.
- The translator program runs at 12:10 AM (5 minutes after SCATS Live Archiving Program) every day and repeats at 15-minute intervals to incrementally ingest the signal events into the ATSPM MOE Database.
- The 15-minute signal and timing data will be updated on the ATSPM Website.

3. Autoscope Detection Event:

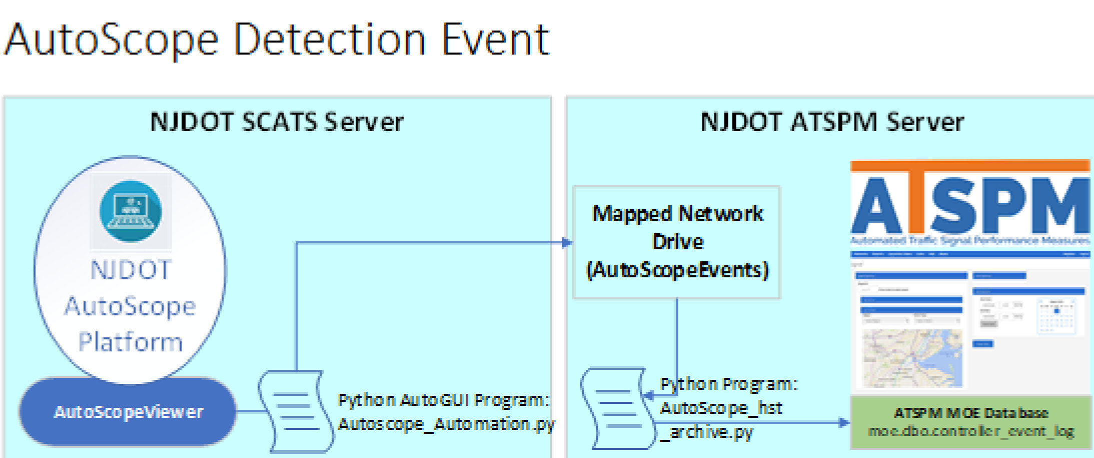


Figure 63. AutoScope Detection Process

- The Autoscope detection event archiving process is implemented using graphical user interface automation since there are no executable command lines to export the Autoscope event data.
- We developed the python program that can control Autoscope software by sending virtual keystrokes and mouse clicks, just as if a person interacts with the Autoscope applications themself.
- The local drive on Autoscope was shared with the ATSPM server. On the ATSPM server, an advanced detector actuation program was used to estimate vehicle arrivals/departures at the advanced detection location.
- The advanced detector events were then loaded into the ATSPM MOE database to generate detection-based performance metrics, such as Purdue Coordination Diagram, Arrival on Red, etc.

STAKEHOLDER MEETING MINUTES

Time: 2:00-3:30 p.m., December 14th, 2020

Attendees	Department	Email
Peter Jin	Rutgers-CAIT	peter.j.jin@rutgers.edu
Tom Brennan	TCNJ	brennant@tcnj.edu
Kelly McVeigh	NJDOT MSE	kelly.mcveigh@dot.nj.gov
Priscilla Ukpah	NJDOT Research	priscilla.Ukpah@dot.nj.gov
Mohammad Jalayer	Rowan	jalayer@rowan.edu
Deep Patel	Rowan	pateld80@rowan.edu
Terry Zhang	Rutgers	terry.tianya.zhang@rutgers.edu
Ek Phomsavath	FHWA	ekaraj.phomsavath@dot.gov
Jason Simmons	SJTPO	jsimmons@sjtpo.org
Mark Taylor (Derrick)	UDOT	marktaylor@utah.gov
Amy Lopez	Inrix	amy.Lopez@inrix.com
Rick Schuman	Inrix	rick@inrix.com
Mike Massaro	Inrix	mike.Massaro@inrix.com
Richard Cippoletti	NJTPA	rcippoletti@njtpa.org
Eddie Curtis	FHWA	eddie.Curtis@dot.gov
Robert Meyer	Transcore	robert.meyer@transcore.com
Mark Renner	NJDOT	
LaDanya Friday	NJDOT	
Virginia Todd	NJDOT	
Steve Remias	Wayne State University	sremias@wayne.edu
Christopher M. Day	Iowa State University	cmday@iastate.edu
Katie Elliott	SJTPO	
Allen Davis	Georgia DOT	

Introduction – Peter Jin

- Introducing Agenda
- Self-introduction of all attendees.

Project Overview - Peter Jin

- Project background: Objective and Key research problems
- Project Team: NJDOT Research, NJDOT TSM, Rutgers CAIT, TCNJ, Rowan
- Stakeholder Panel

NJDOT Perspectives – Kelly McVeigh

- He provided some background about this research project and its significance in fulfilling the NJDOT needs. He also provided an overview of the current NJDOT process for signal timing.
- He mentioned that NJDOT has a management system developed a couple of years back called classification of arterial system technology. It was essentially a ranking tool, considering congestions and other factors. It results in traditional traffic signal optimization up to adaptive signal. Through system implementation, NJDOT could get a prioritized list of corridors that would lead to project programming. He also mentioned that NJDOT usually uses a consultant on board to perform the work.
- Kelly also mentioned that all these be done through traditional modeling, synchro sim traffic, etc. Then, the project's output would essentially be updating timing plans in New Jersey. He also stated that legal documents describe how a signal should operate. He mentioned that another way would be if NJDOT received a complaint or someone observed an issue, so NJDOT could have some field observation to confirm the issues. A significant issue would lead to a significant project, then NJDOT would put that right into the project programming. It could lead to project delivery, which is a long process, or NJDOT might have some abbreviated project where minimal modeling was performed.
- Kelly mentioned that mainly traditional modeling is required when NJDOT updates timing plans. So that would lead to an updated timing plan. But the critical thing here from the New Jersey standpoint is these updated timing plans are the key to unlocking how you want to update a single timing
- Kelly states that, unlike a real-time traffic signal system, traditional methods can’t show where we are and if there is a need to update a gap time right if an issue with an approach is observed. A real-time traffic signal system is a pretty quick feedback system in a real-time situation. So, NJODT can make the required adjustments to the system parameters and immediately start monitoring it and see how things change.
- He mentioned that it is not much different from an adaptive system regarding how users can update timings, which would be a big win here in New Jersey. Kelly also mentioned that the purpose of this research is to follow the footsteps of Utah or Georgia. He also said that because NJDOT is invested in adaptive signal systems in this state, and there is a lot of existing infrastructure, NJDOT wants to make sure that they are using the unique signal control policy where a user can update gap time split-cycle and cycle length.

- Kelly mentioned that we are in the process of putting together the actual architecture for our ATSPM system so we can download the software and start running it in production as opposed to the beta version that we already had.

ATSPM Platform Details – Peter Jin

- Peter provided an overview of the ATSPM system and mentioned that there are over 2,500 traffic signals to maintain in New Jersey; among those, over 300 signals are uncontrolled traffic signal systems, and 76 signals are adaptive traffic signal systems. He mentioned that some initial maps were identified in this project and potentially will be expanded in 2021 as the team will have a final count of all the adaptive signals.
- Peter stated that some critical challenges are following the standard ATSPM deployment procedure requiring higher resolution controllers, adding data probe, and FTP configuration at signal boxes. He mentioned that we have standardized event data in the adaptive single control systems. The ATSPM system is also open-ended action, so users decide how the performance is measured. Peter mentioned that the performance metrics for ATSPM are relatively simple. Still, the data generated from the ATSPM technologies can potentially be fed into the ATSPM to generate more comprehensive performance metrics.
- Peter discussed that the original ATSPM framework relies on point-to-point center communications and high-resolution controllers, and this system is also agnostic to the controller type. He also provided detailed information about the SCATS signal event conversation for various events, including Gap-Out, Max-Out, Force-Off, Phase-on, Green-Begin, PedestrianBeginWalk, and PedestrianCallRegistration. Peter also compared the standard language in the SCATS log file and the ATSPM standard code for various events. He also explained the SCATS Autoscope/Wavetronix detection event translator and how it can be integrated into the ATSPM system.
- Peter mentioned that the research team conducted a lot of testing throughout the first and second phases of the project. He mentioned that the team identified several locations, including Route 18 and US 1, and developed some performance measures.

SCATS ATSPMs Results-Tom Brennan

- Tom mentioned that the team took the existing system and pulled it into the SCATS data. He mentioned that the data could not directly get from the controllers, and it should be converted to be used in the ATSPM system.
- Tom shows the results of the Purdue Phase Diagram at Carnegie center Road intersection-US 1 using the high-resolution traffic signal controller. The graph depicted several events, including the force-off, gap-out, or max-out events for each traffic signal phase at the intersection over 24 hours.
- Tom also demonstrated the results of the Purdue Coordination Diagram that is used to evaluate the operation of coordinated signals and identify the signal timing parameters, similar to EKG monitoring. He also discussed the results of the split monitor for the SCATS system at the US 1 and Carnegie center Road intersection. He mentioned that the split monitor shows the amount of split time used by various

phases at the intersection. Tom discussed the results of pedestrian delay at the intersection, which is a significant parameter to evaluate one perspective of the safety level of an intersection. He mentioned that the longer the pedestrian waits, the more likely a pedestrian will violate the traffic signal at risk.

- Tom also showed an NJDOT ATSPM 2.0 platform demo at Carnegie intersection from July 30th to 31st, 2020. He showed some performance measures such as Purdue Phase Termination, Purdue Coordination Diagram, Arrival on Red, Split Monitor, and Pedestrian Delay.

Discussion

- Eddie Curtis (FHWA) asked if this project intends to overlay ATSPM on facilities equipped with adaptive control? Or will it be possible to deploy ATSPM on facilities not equipped with adaptive control?
- Kelly McVeigh (NJDOT) responded that NJDOT wants a stand-alone ATSPM system so they can include non-adaptive signals. The Translator would sit on the same server and run-on adaptive logs, and the end-users wouldn't know if the signal is adaptive or not. He also mentioned that NJDOT doesn't want separate interfaces for adaptive SPMs vs. traditional ATSPMs.
- Eddie Curtis (FHWA) asked if the signals equipped with SCATS don't conform to the data enumerations. The Translator produces the information required to produce the measures and stores them in the database for retrieval, so no one knows the difference, and Kelly McVeigh (NJDOT) confirmed this.
- Eddie Curtis (FHWA) asked if the process for updating timing directives will allow the flexibility to "tinker" with timings to discover solutions for problems, e.g., gap/extension times, detector delays, fixed/floating force offs...
- Kelly McVeigh (NJDOT) answered that that is what I am hoping. We have been able to update language in our adaptive timing directives, so I would like to take that and make it for "ATSPM signals" for non-adaptive signals that are tied into the ATSPM system
- Mark Taylor (UDOT) asked if SCATS data is 1/10th second resolution, and Peter Jin (CAIT) responded that it currently is at the second level, the same as detector data. There can be multiple events in the exact second.
- Kelly McVeigh (NJDOT) mentioned that a lot of the effort takes available data. The most abundant available data that we have is through our adaptive systems, and it can be provided in this format by translating it. He also mentioned that Georgia DOT had done something towards the higher level of adaptive systems.
- Mark Taylor (UDOT) mentioned that many jurisdictions with adaptive systems do not have an excellent way to measure what is happening. He said putting everything on a playing field and looking at the adaptive system versus the time-of-day coordination plans. He also mentioned that most traffic signals in Utah are not fixed time, and in fact, they are activated systems except in Salt Lake City in the CBD area, where they have some fixed time in two sections. Mark also mentioned that around 2000 intersections are deployed in the ATSPM system.
- Christopher Day (Iowa State University) provided a few thoughts; since the team is looking at adaptive systems, that would be interesting if there was any way to extract any of its internal decision-making and apply it to some of the performance

measure diagrams. He also mentioned that, for example, we are just showing the plan in the coordination diagram, and the plan is unknown. Still, there can be a way to extract some decisions, such as green times during a specific time of day period. It probably decides whether to coordinate or not to coordinate might be an exciting overlay.

- Robert Meyer (Transcore) mentioned that he forwarded some documentation to Kelly (NJDOT) regarding how the team could use it to go directly into those files. Still, a graphical user interface already shows the green times for each phase throughout the day.
- Thomas Brennan (TCNJ) mentioned that for an intersection, maybe even a movement, a higher resolution type of data, the team might be able to indicate better how vehicles are moving through the system. Tom stated that right now, the one-minute increment gives us an idea of the location, the midpoint of a segment, but perhaps the team can start to outline a more specific type of spatial layout for intersections.
- Mark Taylor (UDOT) mentioned that one of the first metrics they created at the ATSPM website was the cumulative frequency diagrams.
- Thomas Brennan (TCNJ) mentioned those diagrams are compelling graphs that they can show you the before and after relatively quickly change.
- Peter Jin (CAIT) stated that the team is looking to see how they can better integrate and enable some of the hidden treasure code. He also mentioned that most coders are written in C sharp language, and some scripting language is used to create some results.
- Mark Taylor (UDOT) mentioned that the ATSMP source code is open-source when others enhance the source code and get to it. He stated that they would hope that the team would push it back through GitHub through a pull request. The developers pull everything in that does not break anything. So, if there is a breaking in the website or the coding, they will not pull it in. But if it does not break anything that they will go, you can go ahead and pull it in. He also mentioned that if the coding is written in other languages such as Python that does not fit well with the ATSPM platform, they cannot pull it unless there are links to it.
- Peter Jin (CAIT) mentioned that the team would have a complete deployment and some of the added functionality to the ATSPM system by the next stakeholder meeting. He also stated that by using China data, the team would try to develop a more comprehensive dataset to validate the results.

Next Steps

- Peter Jin (CAIT) outlined the research plans for the next quarter.
- The meeting was adjourned.