A Secure and Cost-Effective System for Acquiring Traffic Signal Data to Facilitate CAV Operations and Other Use Cases

Manish Kumar Krishne Gowda[1]; Andrew Balmos[2]; Walt Fehr[3]; Richard Ajagu[4]; James V. Krogmeier[5]; Montasir Abbas[6]; and Samuel Labi[7]

[1]Ph.D. Candidate, Elmore Family School of Electrical and Computer Engineering, Purdue Univ., West Lafayette, IN 47907 (corresponding author). ORCID: https://orcid.org/0009-0008-7289-8258. Email: mkrishne@purdue.edu
[2]Ph.D. Candidate, Elmore Family School of Electrical and Computer Engineering, Purdue Univ., West Lafayette, IN 47907. ORCID: https://orcid.org/0000-0001-7208-3974. Email: abalmos@purdue.edu
[3]Principal Technical Advisor, Technology Analysis and Strategy, U.S. DOT Volpe National Transportation Systems Center, Cambridge, MA 02142. Email: walton.fehr@dot.gov
[4]Graduate Research Assistant, Lyles School of Civil and Construction Engineering, Purdue Univ., West Lafayette, IN 47907. Email: rajagu@purdue.edu
[5]Professor, Elmore Family School of Electrical and Computer Engineering, Purdue Univ., West Lafayette, IN 47907. Email: jvk@purdue.edu
[6]Professor, Dept. of Civil Engineering, Virginia Tech, Blacksburg, VA 24061. Email: abbas@vt.edu
[7]Professor, Lyles School of Civil and Construction Engineering, Purdue Univ., West Lafayette, IN 47907. Email: labi@purdue.edu

ABSTRACT

Harvesting real-time data from traffic signal controllers supports connected and automated vehicle (CAV) operations. These benefits, however, require strong end-to-end security and a deployment model that adds minimal cost at each traffic intersection. In this paper, we present a secure, low-cost system that retrieves signal phase and timing (SPaT) data from traffic-signal controllers (TSCs). The system uses two microcontrollers connected through a unidirectional universal asynchronous receiver/transmitter (UART) link, establishing a "data-diode" channel. The first microcontroller interfaces with the TSC, extracts SPaT data, encodes it in base64, and sends it over the one-way UART link. The second microcontroller forwards the encoded data to a remote server via a cellular modem. The data-diode link enforces one-way communication, ensuring security of the TSC. Further, initial estimates suggest that the system achieves a 90% cost reduction per intersection compared to traditional dedicated short-range communication solutions, while assuring reliable data transmission.

INTRODUCTION

Connected and automated vehicles (CAVs) are transforming the future transportation landscape by interconnecting vehicles and roadside transportation infrastructure. They enhance driving experience by enabling vehicles to coordinate their movements, reduce traffic delays by optimizing traffic flow and help to cut down fuel consumption and pollution. Most importantly, they improve traffic safety by alerting the drivers of potential road hazards and reduce the risk of accidents

caused by human error. To support the CAV operations, it is essential for vehicles to receive accurate and real-time information about their surrounding environment, particularly at traffic signal intersections.

One of the important types of traffic-related data needed for CAVs is the signal phase and timing (SPaT) information from the traffic signal controllers (TSC). TSCs are housed in cabinets, typically placed at the side of intersections, and are responsible for managing the traffic light operations at those intersections. The SPaT data from these TSCs provides information about the current and upcoming traffic light statuses for each lane at a traffic intersection. This includes the phase (i.e., red, yellow and green) and the timing of the light in each lane. However, conventional methods for collecting and transmitting SPaT data often rely on expensive computing hardware and short-range communication systems installed at each intersection. While these systems can be effective for data collection and dissemination, they are very costly and difficult to maintain, making large-scale deployment less practical. For example, in the Gwinnett County, Georgia, deployment of just 20 intersections with dedicated short-range communication (DSRC) equipment incurred exorbitant costs of over $450,000, as shown in Table 1 (Gowda et al. 2023). Therefore, there is a need for more affordable but effective solutions to obtain traffic intersection data to facilitate CAV operations.

TABLE 1: Breakdown of the costs for Gwinnett county's DSRC deployment.

(a) Sample Project Field Installations			
Item	**# of units**	**Unit Cost**	**Total Cost**
Roadside unit	20	$5,000/unit	$100,000
Cybersecurity	20	$100/unit	$2,000
Edge processing	20	$350/unit	$7,000
Service (unit = year)	3	$10,000/unit	$30,000
MAP message development	20	$1,500/unit	$30,000
Software (unit=intersection)	20	$7,000/unit	$140,000
TOTAL			**$309,000**
(b) Sample Project Vehicle Installations			
Item	**# of units**	**Unit Cost**	**Total Cost**
On-Board unit	20	$5,000/unit	$100,000
Cybersecurity	20	$100/unit	$2,000
Human Machine Interface	20	$3,000/unit	$60,000
TOTAL			**$162,000**

To address this, we propose a secure, low-cost alternative using two STM microcontrollers and a 4G cellular modem. In this setup, the first microcontroller is connected to the TSC via ethernet and is responsible for retrieving the SPaT data from the controller. This data is then appended with a cyclic redundancy check (CRC) to enable the verification of correct reception and encoded in base64 to ensure reliable transmission of the SPaT data bits. Subsequently, it is transmitted through a unidirectional universal asynchronous receiver/transmitter (UART) interface to the second microcontroller. This simplex communication enforcing one-way data flow is termed the data-diode and it ensures that no commands or data can be sent back to the TSC from external networks.

This eliminates the risk of cyberattacks through the reverse communication channel. The second microcontroller then forwards the processed SPaT data to a remote server in real time using the cell modem, enabling its use in CAV operations. The minimalistic end-to-end workflow plan of the proposed data-diode system, showcasing the TSC, the simplex data transmission scheme and the external server interface, is shown in Figure 1. This data-diode system ensures a reliable, lossless SPaT data transmission from the TSC to the remote server, while significantly reducing the operational cost to about $200 per intersection, making it scalable for wide-scale deployments.

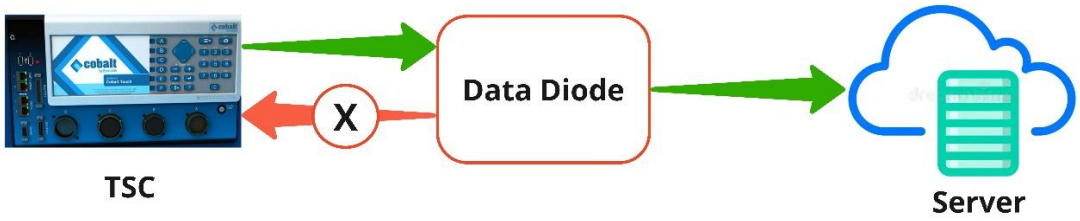

Figure 1. End-to-end workflow of the proposed data-diode system.

METHODOLOGY

The format and timing of SPaT messages broadcast by TSCs are defined by the NTCIP 1202v3 standard (National Transportation Communications for ITS Protocol: Traffic Signal Controller Broadcast Messages) (AASHTO et al. 2019). Each 245-byte message provides information on current and upcoming traffic signal phases and can include adaptive details like time-based lane rules. The timing of signal changes is dynamic, based on inputs from vehicle and pedestrian sensors. The TSC integrates these inputs with pre-programmed timing parameters, updating the traffic lights through a feedback loop to maintain efficient and adaptive traffic flow.

TSCs expose SPaT data via interfaces such as ethernet, allowing third-party systems to deliver real-time traffic updates. However, with traffic controllers considered part of the critical infrastructure of a nation, departments of transportation (DOT) do not allow such direct external access unless it can be guaranteed that the integrity of the TSC will not be compromised. The data-diode ensures that SPaT data is only retrieved from the TSC through a one-way communication link, effectively protecting the controller from cyberattacks.

For ease of understanding, we divide the data-diode architecture into two distinct components:

1. **Controller-side of the data-diode:** This component, comprising of the TSC and a single STM microcontroller, is responsible for extracting the SPaT data from the TSC and reliably transmitting it to the world-side of the data-diode. Its operations include four subtasks:
    i. Configuring the TSC to broadcast the SPaT message
    ii. Retrieving the SPaT message broadcast by the TSC
    iii. Appending the SPaT data with CRC and encoding using base64
    iv. Transmitting the encoded data over to the world-side of the data-diode.
2. **World-side of the data-diode:** This component, comprising of a STM microcontroller and a cell-modem, receives the encoded data from the controller-side of the data-diode and

transmits it to a remote server. It is referred to as the "world-side" as it communicates with server(s) beyond the protected TSC-diode environment. Specifically, its functions include:

i. Receiving the encoded SPaT data from the TSC side of the data-diode
ii. Transmitting the encoded SPaT data to a remote server for network-side processing

The project code is available at: https://github.com/oats-center/data-diode/tree/data_diode_final

1. **Controller-side of the data-diode:** In this section, we outline the four distinct sub-tasks performed on the controller-side of the data-diode.

   i. **Configuring the TSC to broadcast the SPaT Message:** We use Econolite's Cobalt advanced transportation controller (ATC) TSC, which is widely adopted by the DOTs across the United States (Hajbabaie et al. 2020), as shown in Figure 1. The Econolite TSC is equipped with two ethernet ports, ENET1 and ENET2. For our setup, the TSC is configured to transmit SPaT messages via its ENET1 port using simple network management protocol (SNMP) commands, as shown in Figure 2 (Gowda et al. 2023). The STM microcontroller on the controller-side of the data-diode connects to ENET1 and listens for SPaT messages at the configured SERVER_IP. In field deployments, however, DOTs may not allow modification of this pre-configured SERVER_IP. In such cases, the SPaT stream can still be accessed non-intrusively using traffic mirroring or passive forwarding methods, such as port mirroring or a simple network hub, allowing the data-diode to receive SPaT data without altering any existing DOT network configuration (Sanders 2017).

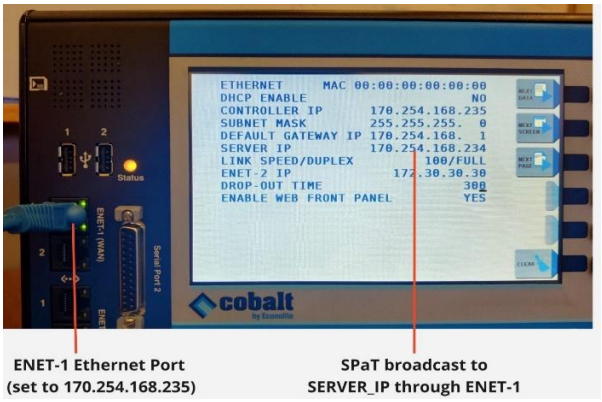

Figure 2. Sample network settings for broadcasting SPaT data on ENET-1.

   ii. **Retrieving the SPaT message broadcast from the TSC:** For the implementation of the data-diode system, an STM32 Nucleo-144 development board was selected as the hardware platform for both the controller-side and world-side units. The board is inexpensive (approximately $25), includes an onboard Ethernet interface, and benefits from strong community and vendor support. Its multiple UART ports allow dedicated channels for the data-diode link, the cellular modem, and debugging. Additionally, the board integrates multiple LED indicators, which are used to provide simple visual feedback on system status and operating conditions.

   The ENET1 ethernet port of the TSC is connected to the STM32 using a standard ethernet cable. The TSC broadcasts SPaT messages over UDP on port 6053 at 100 ms intervals (10

packets per second). To receive these packets, the STM32 is configured to listen for UDP traffic at the SERVER_IP specified during TSC setup, with both devices assigned IP addresses within the same subnet. A UDP interrupt service routine (ISR) on the controller-side STM32 handles incoming SPaT packets and stores them in a circular buffer for non-blocking, lossless capture. A parallel processing task reads from this buffer (Figure 3), appends a CRC, performs base64 encoding, and transmits the resulting payload through the simplex UART channel to the world-side microcontroller, as described in the following sections.

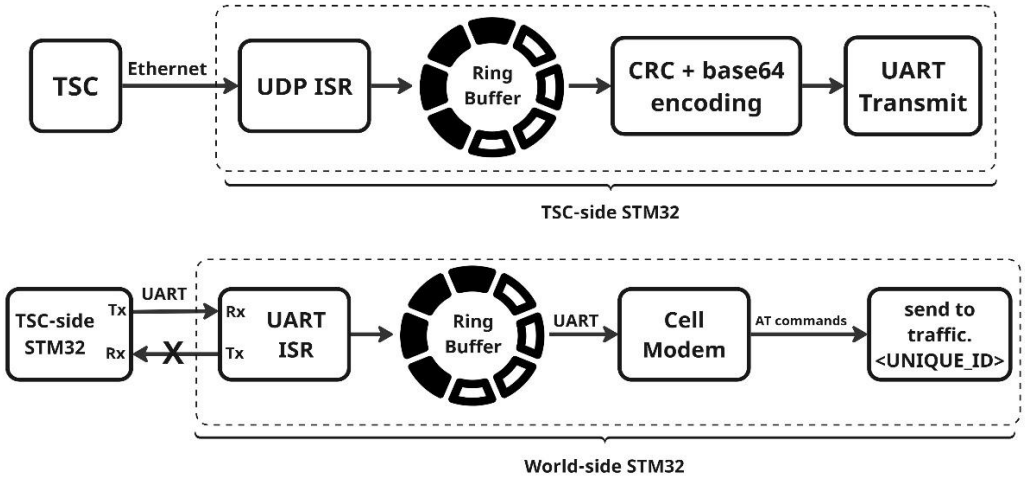

Figure 3. UDP ISR and circular buffer used to capture SPaT packets on the controller-side STM32 (top); UART ISR with circular buffer for processing base64-encoded SPaT packets on the world-side STM32 (bottom).

iii. **Appending the SPaT data with CRC and encoding using base64 :** To protect the integrity of the SPaT data, a 32-bit CRC is appended to every packet received from the TSC. The CRC is computed from the packet's contents and allows the receiver to detect bit flips or other transmission errors. When the remote server receives the data, it recomputes the CRC and verifies that the packet matches what was originally transmitted by the TSC. Furthermore, since the combined CRC + SPaT packet contains non-printable binary bytes, it cannot be sent directly over the simplex UART link or through the AT-command-based cellular modem. To make the data UART-safe, the entire payload is encoded using base64, which converts binary data into printable ASCII characters with relatively low overhead. Although this adds some expansion, base64 remains more compact than alternatives like base32 or hexadecimal. Together, CRC integrity protection and base64 encoding provide a secure and transmission-friendly data format for the end-to-end data-diode pipeline.

iv. **Transmitting the encoded data over to the world-side of the data-diode:** The one-way communication link between the controller-side and world-side STM32 boards is created by connecting only the Tx pin of the controller-side UART to the Rx pin of the world-side UART, with no return connection. This physically enforces unidirectional data flow, allowing the SPaT data to be sent outward while preventing any data from being sent back toward the TSC. Because UART pins have fixed transmit/receive roles that cannot be reversed in software, the absence of the return Tx-Rx link guarantees that no command,

packet, or signal can flow back to the controller-side microcontroller. This simple hardware configuration forms the core of the data-diode design, ensuring strict one-way communication and protecting the TSC from any external or malicious inputs, as shown in Figure 1.

2. **World-side of the data-diode:** In this section, we outline the two distinct sub-tasks performed on the controller-side of the data-diode

   i. **Receiving the encoded SPaT data from the TSC side of the data-diode:** The world-side STM32 receives the base64-encoded SPaT packets through the Rx pin of its UART interface. Any UART port on the STM32 Nucleo-144 can be configured for this purpose, as long as it matches the baud rate of the controller-side transmitter. In our setup, the controller-side Tx (UART7, PE8) is connected to the world-side Rx (UART7, PE7), but any Tx-Rx pairing is valid as long as the reverse link is not connected, preserving the one-way data-diode architecture.

   Incoming UART bytes are handled using another interrupt-driven circular buffer, as shown in Figure 3. The UART ISR pushes each incoming base64-encoded packet into this buffer, while a parallel modem-client routine reads from the buffer and forwards packets to the remote server through the 4G cellular modem. The buffer holds up to 20 packets, enough for about two seconds of data, ensuring smooth operation during brief modem outages. If the buffer overflows during longer connectivity losses, older packets are discarded to preserve the real-time nature of the system. Since each SPaT packet includes a timestamp, the remote server can detect gaps or delays and handle longer outages accordingly.

   ii. **Transmitting the encoded SPaT data to a remote server for real-time access and network-side processing**: The world-side STM32 transmits the base64-encoded SPaT data to a remote server via a 4G cellular modem (SIM7600X), enabling real-time network-side processing. Unlike the unidirectional UART used for the data-diode, this UART connection is bidirectional to support data transmission and reception of acknowledgments from the server. The modem-client running on the STM32 manages the TCP connection using AT commands, handling tasks such as defining the packet context, opening the TCP socket, sending/receiving messages, and responding to PING messages to maintain an active connection. In the event of temporary network failures, the modem-client automatically reconnects, ensuring persistent data delivery.

   Each data-diode is assigned a unique 12-byte identifier (UNIQUE_ID) that tags all transmitted SPaT packets, allowing the server to distinguish intersections and organize subscriptions using subject-based messaging (e.g., traffic.<UNIQUE_ID>). Connectivity errors are reported to a dedicated subject (connect.<UNIQUE_ID>) for monitoring and diagnostics. LED indicators on the STM32 provide visual feedback on system status and modem connectivity, while the integrated code handles packet reception, transmission, and LED signaling. This messaging system ensures secure, reliable, and real-time SPaT data delivery from multiple intersections to a centralized network server.

Figure 4 illustrates the end-to-end interconnection of the controller-side STM32, world-side STM32, and cellular modem. To facilitate monitoring, the world-side STM32 features LED indicators that provide a visual summary of the system status: LED 1 blinks during startup, LED 3 remains red until the modem client establishes a TCP connection, and LED 2 illuminates blue once SPaT data transmission is active. The figure also shows the labeled LEDs on the STM32 Nucleo board, alongside the final v1.0 enclosure (which will be presented later in 'results' section).

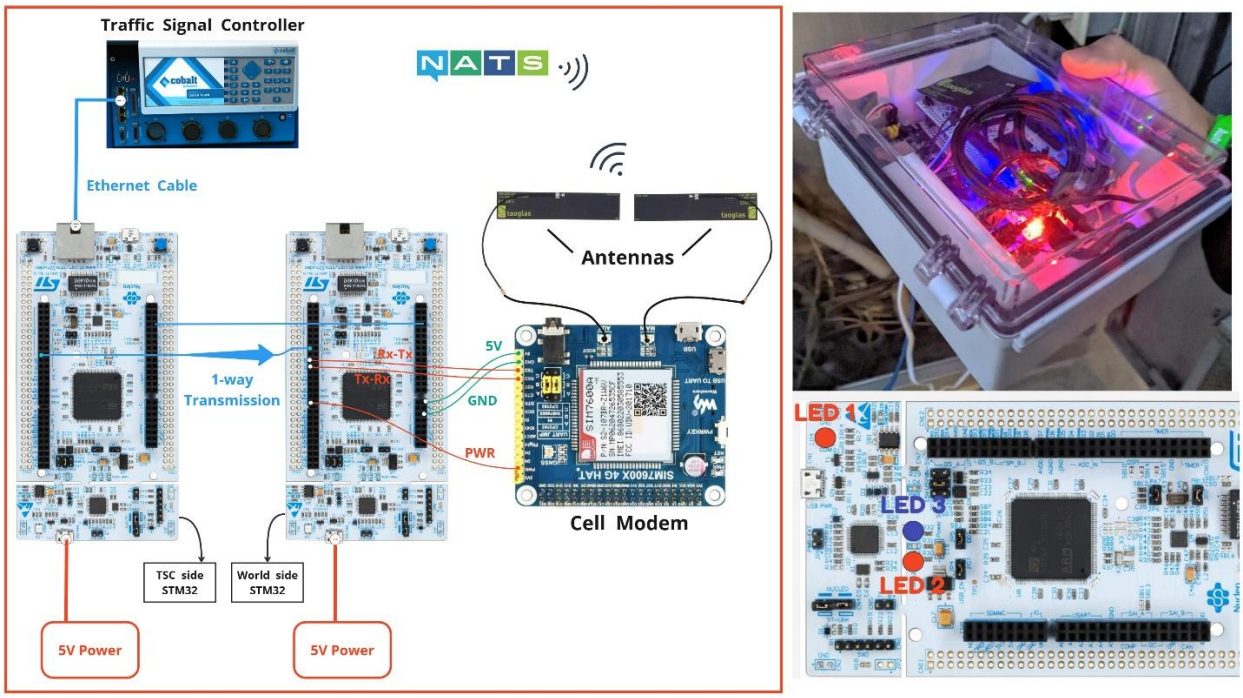

Figure 4. Interconnection of the data-diode components (left). The right side shows data-diode enclosure with LED lights indicating the operating status (top), and the world-side STM32 LED status indicators (bottom).

RESULTS

SPaT data transmission by the data-diode and it's reception on the server side: When the data-diode is connected to the TSC and SPaT data is being retrieved and actively transmitted to the remote server, subscribing to the subject pattern traffic.* on the remote server allows for real-time monitoring of the base64-encoded SPaT data stream. This is illustrated in Figure 5. A single data-diode is connected to a TSC using the ethernet cable (the other end of the ethernet cable is internally connected to the ethernet port of the TSC-side STM32), emulating one intersection. The base64-encoded SPaT data is continuously received at 100ms intervals on the subject

traffic.383135333431511600340003C. Here, 0x383135333431511600340003C is the 12-byte unique identifier of the world-side STM32, as previously discussed

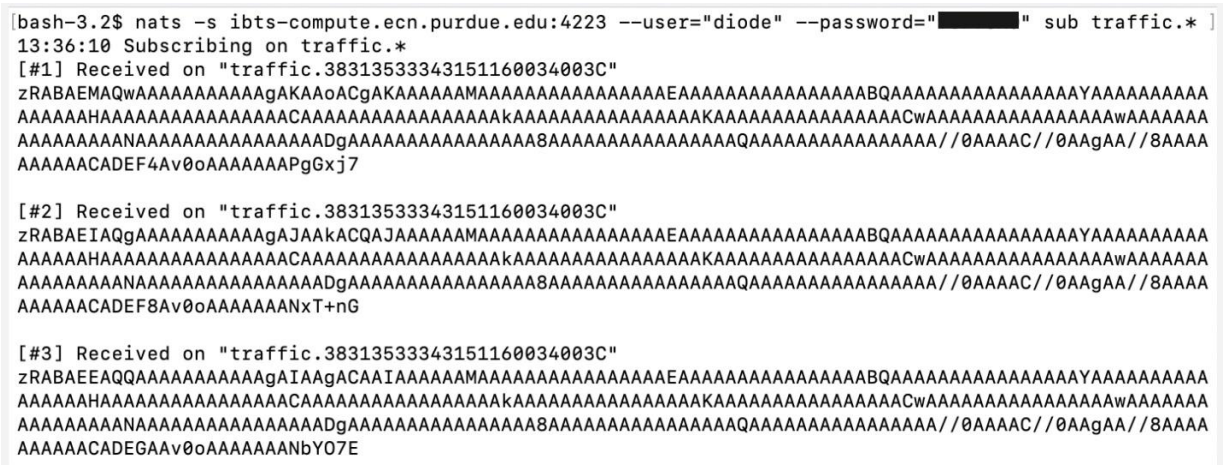

Figure 5. Subscribing to the subject "traffic.*" on the server side

As an example, to visualize the encoding process, consider the base64-encoded SPaT packet #2 at the server, i.e,
"zRABAEIAQgAAAAAAAAAAgAJAAkACQAJAAAAAAMAAAAAAAAAAAAAAAAE
AAAAAAAAAAAAAAAABQAAAAAAAAAAAAAAAAYAAAAAAAAAAAAAAAHAA
AAAAAAAAAAAAAACAAAAAAAAAAAAAAAAAkAAAAAAAAAAAAAAAAKAAAA
AAAAAAAAAAAACwAAAAAAAAAAAAAAAwAAAAAAAAAAAAAANAAAAAA
AAAAAAAAAADgAAAAAAAAAAAAAAA8AAAAAAAAAAAAAAAAQAAAAAAA
AAAAAAAA//0AAAAC//0AAgAA//8AAAAAAAAAACADEF8Av0oAAAAAANxT+nG"

Decoding this base64 packet yields a 249-byte sequence, represented in hexadecimal as –
"CD10010042004200000000000000000200090009000900090000000030000000000000000000
0000004000000000000000000000000005000000000000000000000000060000000000000000000
0000007000000000000000000000000080000000000000000000000009000000000000000000
0000000A000000000000000000000000B000000000000000000000000C000000000000000000
000000000D000000000000000000000000E000000000000000000000000F000000000000000000
0000000010000000000000000000000000FFFD00000002FFFD00020000FFFF000000000000
00002003105F00BF4A000000000003714FE9C6"

The last 4 bytes of this hexadecimal sequence (0x71,0x4F,0xE9 & 0xC6), interpreted as 0xC6E94F71 in big-endian format, represent the CRC-32 checksum appended by the TSC-side STM32. This allows verification of any transmission errors at the server side. The remaining 245 bytes represent SPaT data formatted according to the NTCIP 1202v3 standard. This payload can be parsed according to the byte-map structure as defined in (AASHTO et al. 2019) to extract the status of all signal lights at the intersection in real-time.

The encoding efficiency of crc+base64 can be defined as:

$$Efficiency = \frac{Size\ of\ original\ SPaT\ data\ packet\ (bytes, without\ CRC)}{Size\ of\ base64\ -\ encoded\ SPaT\ data\ with\ CRC\ (bytes)}$$

Given :

- The original SPaT data broadcast by TSC = 245 bytes
- Base64-encoded sequence = 332 characters (each base64 character is 1 byte when stored as ASCII)

Thus, we get the efficiency as,

$$Efficiency = \frac{245}{332} \approx 0.74 = 74\%$$

This is more efficient compared to hexadecimal ASCII encoding or base32 encoding. In hexadecimal ASCII encoding, each byte encodes as two ASCII characters (e.g., byte 0xCD becomes 'C' and 'D'). Thus efficiency reduces to 245/498 ≈ 0.49 = 49%. In base32, the 249 byte CRC appended SPaT packet is encoded to a 400 character sequence, yielding an efficiency of 245/400 ≈ 0.61 = 61%. Therefore, base64 encoding not only ensures compatibility with UART and modem AT commands by converting the raw SPaT byte stream of the TSC to a fully printable ASCII string, but also offers significantly superior efficiency than other popular alternatives, such as the hexadecimal and base32 encoding.

Cost analysis of the data-diode components and enclosure: Given the significant cost of existing deployments, as discussed previously, it is important to analyze the cost breakdown of the proposed data-diode system to evaluate the feasibility of its large-scale deployment. In all, the data-diode system consists of two STM32 Nucleo-144 boards, a cellular modem, two antennas, an ethernet cable, two power cables for the STM32 boards, and seven jumper wires for the interconnections shown in Figure 4. Two versions of enclosures were designed to house the STM32 boards, the cellular modem and the two antennas. The first version (v1.0) used a waterproof Polycase® box and included a 5V USB hub inside to power the STM32 boards and the modem. The hub's power cable and the ethernet cable connected to the TSC side STM32 was routed outside the box, making it easy to install and use at a TSC cabinet.

The second version (v2.0) of the data-diode is more compact and eliminated internal wiring by placing all components, i.e. the two STM32 boards and the cell-modem on a single custom printed circuit board (PCB). Similar to v1.0 enclosure, only the power cable that is used to power the PCB and ethernet cable connected to the TSC side STM32 was routed out of the v2.0 box. The PCB design files (KiCAD) and 3D-printed enclosure CAD files are available in the GitHub repository. Both enclosures were designed to fit within a standard TSC cabinet. The two enclosures, v1.0 and v2.0 are shown in Figure 6.

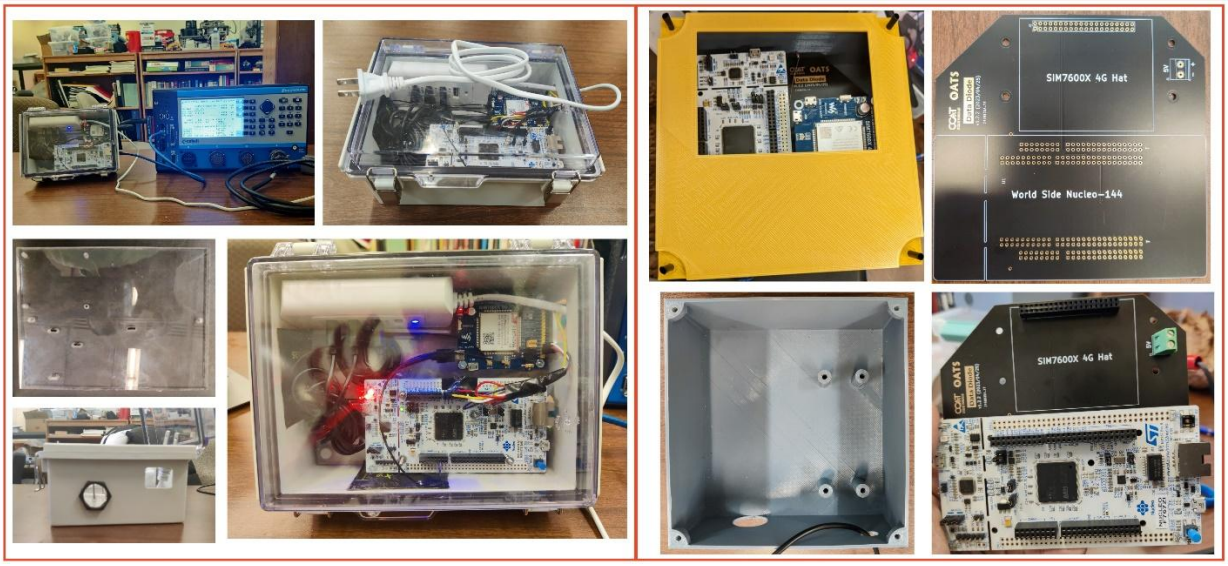

Figure 6. Data-diode enclosure version 1.0 (left) and version 2.0 (right)

The estimated cost of the system hardware (not considering the cost of the cables) is evaluated in Table 2. Version 2.0 of the data-diode enclosure is a slightly more cost-effective solution compared to version 1.0. While both versions are affordable in terms of initial costs, their cost-advantages become even more pronounced when scaled for larger deployments due to economies of scale. Notably, the hardware cost of the proposed data-diode system is substantially lower than that observed in the Gwinnett County deployment.

TABLE 2: Cost breakdown of components used in data-diode enclosure versions 1.0 & 2.0

Component	Unit × Price	Total Cost ($)
STM32	2 × 24	46
Cell Modem	1 × 65	65
Antennas	2 × 10	20
Polycase Box (V1.0)	1 × 40	40
Mounting Plate (V1.0)	1 × 20	20
USB Hub (V1.0)	1 × 10	10
PCB (V2.0)	1 × 10	10
3D Printed Box (V2.0)	1 × 20	20
Total (Version 1.0)	–	**201**
Total (Version 2.0)	–	**161**

Cellular Data Usage Estimation: While the hardware costs were analyzed in the previous subsection, we now evaluate the cellular data usage requirements for the data-diode in this section. To estimate the daily cellular data requirement per intersection, with 10 SPaT messages per second, with each message being 332 bytes in size (after crc and base64 encoding). The estimated total data size transmitted per month are calculated as follows:

TABLE 3: Estimated daily and monthly cellular data consumption for data-diode operation

Metric	Remarks/Calculation	Value
Data Rate per Second	10 × 332 bytes	3,320 bytes/s
Data per Day	3,320 × 86,400 seconds	286,848,000 bytes
Data in megabytes per Day	286,848,000 ÷ (1024 × 1024)	273.5 MB
Data per month:	273.5 × 30	8,205 MB
Data per Month	8,205 ÷ 1024	8.01 GB

Thus, each deployed unit requires approximately 8 GB of cellular data per month, per intersection, for continuous transmission. Similar to how the hardware costs are manageable, the monthly data costs are also reasonable, likely around $10 per month per unit, making large-scale data-diode deployment economically feasible.

CONCLUSION

In this paper, we proposed a low-cost and secure system for retrieving SPaT data from the TSC. The system employs a dual-microcontroller approach: the first microcontroller interfaces with the TSC via ethernet, retrieves the SPaT data, appends a CRC for data integrity check at the server side, encodes the data packet using base64 scheme and transmits the encoded data to the second microcontroller through a unidirectional UART link. The second microcontroller continuously processes the incoming data on the unidirectional UART interface and forwards the data packet to a remote server using a cellular modem. The unidirectional UART link between the two microcontrollers ensures that the data flows only from the TSC towards the server, and never in the reverse direction. This is because the physical connection itself is one-way, effectively isolating the TSC and the controller-side microcontroller from any incoming data or commands, thereby ensuring the overall security of the TSC. In addition, the circular buffers implemented on each microcontroller ensure efficient FIFO-based processing of the SPaT data packets without loss or delay.

The proposed data-diode approach reduces the cost per intersection to about $200, in contrast to the $23,500 per intersection incurred during the DSRC deployment in Gwinnett County (Gowda et al. 2023), while still maintaining secure and reliable data acquisition from the TSC. The SPaT data collected at the server can enable numerous applications, including providing real-time traffic signal updates to drivers (Gowda et al. 2024), offering location based services to pedestrians (Kumar et al. 2025a; Kumar et al. 2025b), supporting machine learning-based optimization strategies (Gowda et al. 2025), facilitating CAV and driverless operations, improving traffic simulations, and more. The developed device can also be used to rapidly and securely disseminate vital traffic signal information to service and emergency workers and other end users. It can also help small towns and rural or isolated areas with limited resources gain greater access to their traffic signal information for various use cases, thereby ensuring that they are not left out in the nationwide evolution towards smart and sustainable communities. Overall, the research product is expected to promote affordable, connected, cyber-secure and smart transportation systems.

ACKNOWLEDGEMENTS

Funding for this research was provided by the Center for Connected and Automated Transportation under Grant No. 69A3551747105 of the U.S. Department of Transportation, Office of the Assistant Secretary for Research and Technology (OST-R), University Transportation Centers Program. The authors acknowledge the support provided by the following team members: the City of Owosso Mayor Christopher Eveleth, Messrs. Mark Goodfriend, David Acton (President of The Transformation Network, Inc.), Tim Johnson (T-Mobile), Sandy Klausner (Cubicon Inc.), and Michigan DOT Engineers (Messrs. Nathan Bouvy, Phillip Travis, Hilary Owen, Scott Holzhei, Ryan Schian, Ross Venable, and James Kwapiszewski), David Hong of VirginiaTech, and Dr. Darcy Bullock of Purdue.

REFERENCES

AASHTO, ITE, and NEMA. 2019. National transportation communications for ITS protocol object definitions for actuated traffic signal controller (ASC) units. NTCIP 1202 v03.03. Washington, DC: AASHTO, ITE, NEMA.

Gowda, M., W. Fehr, A. Balmos, R. Ajagu, J. Krogmeier, M. Abbas, and S. Labi. 2023. *Economical acquisition of intersection data to facilitate CAV operations*. Paper 45. Center for Connected and Automated Transportation. https://doi.org/10.5703/1288284317731.

Gowda, M., W. Fehr, A. Balmos, R. Ajagu, D. Hong, J. Krogmeier, M. Abbas, and S. Labi. 2024. *Economical acquisition of intersection data to facilitate CAV operations phase II – implementation*. Paper 55. Center for Connected and Automated Transportation. https://doi.org/10.5703/1288284317851.

Gowda, M. K. K., A. Balmos, S. Boonam, and J. V. Krogmeier. 2025. "Enhancing pavement sensor data acquisition for AI-driven transportation research." *arXiv preprint arXiv:2502.14222*. https://arxiv.org/abs/2502.14222.

Hajbabaie, A., S. M. A. Bin Al Islam, M. Tajalli, R. Mohebifard, et al. 2020. *Preparing for traffic signal operations in a multi-modal connected and autonomous vehicle environment*. Olympia, WA: Washington State Department of Transportation.

Kumar, M., T.-H. Chou, B. Lee, N. Michelusi, D. J. Love, and J. V. Krogmeier. 2025a. "Hybrid fingerprint-based positioning in cell-free massive MIMO systems." *arXiv preprint arXiv:2502.02512*. https://arxiv.org/abs/2502.02512.

Kumar, M., T.-H. Chou, B. Lee, N. Michelusi, D. J. Love, Y. Zhang, and J. V. Krogmeier. 2025b. "Distributed machine learning approach for low-latency localization in cell-free massive MIMO systems." *arXiv preprint arXiv:2507.14216*. https://arxiv.org/abs/2507.14216.

Sanders, C. 2017. *Practical packet analysis: Using Wireshark to solve real-world network problems. San Francisco, CA: No Starch Press*