**Multi-Modal Intelligent Traffic Signal System Phase III:**

**Deployment Plan**

**May 15, 2018**

**University of Arizona (Lead)**

**University of California PATH Program**

**Table of Contents**

1 Background 2

2 MMITSS Gap Analysis 4

3 MMITSS Development Group Meeting 4

4 MMITSS Enhancements 5

Task 3: Enhancement of Existing MMITSS Prototypes 6

Task 3.1 Update to the SAE J2735 2016 Standard 6

Task 3.2 Develop an API Specification for an RSU Deployed Message Transceiver 7

Task 3.3 Enhancements to the AZ Prototype 7

Task 3.3.1 Improvements to the MMITSS-AZ Code and Documentation (Category 1) 7

Task 3.3.2 Update to the System Performance Measures (Category 2) 8

Task 3.3.3 Add Support for RDE Data Capture (Category 2) 8

Task 3.3.4 Enhance Traffic Control Features to Support Deployment Readiness (Category 3) 8

Task 3.4 Enhancements to the CA Prototype 9

Task 3.4.1 Improvements to the MMITSS-CA Code and Documentation (Category 1) 9

Task 3.4.2 Add Support for 90-Days Field Readiness Testing (Category 2) 9

Task 3.4.3 Add Traffic Controller Interface for Deploying MMITSS-CA with NTCIP Controllers (Category 3) 9

Task 3.4.4 Traffic Control Enhancements to Support Deployment Readiness (Category 4) 9

# Background

This proposal has been developed in response to the request for proposal (RFP) from the Connected Vehicle Pooled Fund Study (CV PFS) to enhance the deployability of the MMITSS prototypes to address needs identified by the Pooled Fund Members, stakeholders, and the MMITSS developers (University of Arizona and California PATH program).

The goal of the Phase III effort is to make deployment readiness enhancements to the MMITSS prototypes that were developed and field tested in Phase II. The readiness enhancements are focused on creating enhanced and “cleaned up”, or mature, application code that is hardware agnostic and interoperable or transferable regardless of the hardware vendors or products. To achieve these goals, the project team (University of Arizona and California PATH program) propose the following enhancements, that are captured in the Task Descriptions below:

* Improve the MMITSS software manuals, including the requirements on the host platforms and compilers, the download and build instructions, software design documentation, and a specification of an API for the RSU DSRC interface that vendors can implement to support interoperability.
* Increase the source code maturity through code reviews with the MMITSS Development Group of stakeholders, including changes that make the source code hardware agnostic.
* Carry out additional development of essential functions for MMITSS through field testing for improvements and fine tuning.
* Identify enhancements required for deployability, including upgrading to the 2016 SAE J2735 message set, and field testing of traffic control enhancements including coordination (for both MMITSS-AZ and MMITSS-CA).

These enhancements are discussed in Section 2.3.

The MMITSS Phase II effort resulted in two distinctly different prototype systems that were field tested. The two systems have distinctive functions that are designed to work with two types of traffic control schemes. The MMITSS-AZ incorporates MMITSS with an adaptive traffic signal control algorithm. The MMITSS-CA is designed as add-on MMITSS features that complement the existing actuated-coordinated traffic signal control.

The MMITSS-AZ system is based on the development of new traffic control algorithms that use connected vehicle data to be adaptive and to serve multiple active priority requests. MMITSS-AZ utilizes NTCIP protocols when communicating with the traffic signal controller. MMITSS-AZ was field tested using Savari RSUs; hence the software utilizes Savari Linux libraries. MMITSS-AZ was also developed to run on standard Linux (Ubuntu 14.04) for use in simulation testing, but the component based architecture allows many of the components to be run on a Linux-based roadside processor (called the MMITSS Roadside Processor in the MMITSS architecture). This alternative deployment has been demonstrated using the Econolite Connected Vehicle Co-Processor board (CVCP) that is a field hardened Linux board.

The MMITSS-CA system is based on improving the existing actuated-coordinated traffic control system, with performance enhancements that utilize connected vehicle data, and providing dynamic priority control that serves multiple active priority requests and considers the prevailing traffic conditions. For example, signal priority and traffic coordination are achieved using the Caltrans 2070 firmware, but the decisions to call priority (with desired green splits for the priority cycles) or to modify a coordination parameter (splits, cycle length, or offset) are made based on data from connected vehicles that allow the MMITSS-CA system to decide the best control strategies. MMITSS-CA utilizes the AB3418 serial communications protocol from a Linux roadside processor (MRP) to the traffic signal controller. The MMITSS-CA system utilizes a central data manager, hosted by the MRP, to collect and distribute information to the other MMITSS MRP software components. The MMITSS-CA team plans to add interface functions for priority and traffic control supporting NTCIP controllers as part of this deployment readiness project.

The MMITSS team has investigated options for integrating the two MMITSS prototypes into a single MMITSS that has options, such as the use of NTCIP or AB3418, use of actuated-coordinated traffic signal control mode from MMITSS-CA, and the adaptive signal control algorithms from MMITSS-AZ. The team determined that traffic control agencies will use either the actuated-coordinated control mode or the adaptive but not both simultaneously; hence there is no need to switch between the two MMITSS schemes. Due to the significant differences in the algorithms and the prototype software, it has been determined that this integration effort would distract, and consume available resources, from the Phase III effort to improve the deployment readiness of each system. Furthermore, as the two systems will be improved throughout the Phase III, it would be difficult to make changes and debug an integrated software while the two system schemes are worked on. Hence, the team we will work simultaneously on two MMITSS prototypes: MMITSS-AZ and MMITSS-CA, and that agencies that choose to adopt MMITSS can consider the individual of merits of each system that best meets their needs.

# MMITSS Gap Analysis

*This section contains a summary of the FHWA Gap Analysis Report. TBD.*

# MMITSS Development Group Meeting

*This section contains a summary of the MDG on May 2-3, 2018*

*The following text needs editing, but was from the SOW.*

The MDG will commence with a two-day kick-off meeting[[1]](#footnote-1) offering detailed presentations of the MMITSS prototypes, code reviews, needs identification and prioritization activities. The MMITSS research team will provide technical support and answers on questions regarding the build process, integration, host platform (e.g. Linux Ubuntu 14.04), standards, source code review (plans) and other topics identified by the MDG members. At the end of the kick-off meeting, a prioritized list of development needs will be identified. Possible and likely approaches will be identified and documented, including the MMITSS research team developing features/improvements, MDG participants/projects undertaking development, or a third-party developer doing the work. The group-defined, prioritized set of needs will be used to develop a plan and cost estimate for the work required to make MMITSS deployment ready.

The MDG activities will continue using a webinar format where source code walkthroughs and reviews will be conducted monthly, or as needed, as part of the effort to “clean up” the code and make it more usable by the MDG stakeholders who are engaged in deploying the code. The sequence of source code reviews will be based on the preference of the MDG stakeholders.

After the two-day kick-off meeting, a regular update schedule will be developed to ensure that the MDG members have access to the current version of MMITSS source code. For example, updates could be provided on a quarterly basis or sooner depending on the nature of the development effort and the ability of the selected team to develop and test associated changes. At a minimum, the OSADP distribution will be updated quarterly. As members of the MDG identify additional issues or needs, the group can hold virtual meetings (WebEx) to discuss solutions, needs, priorities, and development schedules. These virtual meetings will be offered quarterly, or more frequently, as needed.

The findings and activities of the MDG will be documented in the MMITSS Deployment Readiness Plan that will include:

* identification of enhancements needed to make MMITSS deployable,
* development cost estimates for each enhancement,
* prioritization of the identified enhancements by the Pooled Fund Panel, and
* documentation of projects and agencies that are actively engaged in using MMITSS or are considering MMITSS as a solution.

The MMITSS Deployment Readiness Plan will submitted to the Pooled Fund Panel/USDOT Coordinator and support for comment and review. The MMITSS team will brief the Pooled Fund Panel/USDOT Coordinator and support. A final plan will be based on the input received. The Draft MMITSS Development Plan will be submitted within 60-days from the start of the project. The Final Plan will be submitted within 90-days of the start of the project, assuming a 2-week comment period for the Pooled Fund Panel/USDOT Coordinator and support.

# MMITSS Enhancements

The MMITSS prototypes represent the results of a research and development effort focused on a new approach to traffic control that utilizes information from connected vehicles, especially passenger vehicles, emergency vehicles, transit, trucks, and pedestrians. The prototypes were developed to the level of maturity required to support field testing, proof-of-concept, and demonstrations. It is acknowledged that they are not ready for deployment and additional development is needed to ensure that they have greater utility to the projects planning to use them and to the future of traffic control. The Task 2 efforts and the MMITSS Developers Group are focused on identification of enhancements needed to advance the deployment readiness of the MMITSS prototypes.

The MMITSS enhancements have been divided into two categories. The first category are the required enhancements that are identified in the original Proposal and Scope of Work. These include:

* Upgrade to the 2016 SAE J2735 Standard
* Development of an API (Interface) to the RSU 4.1 Standard Specification (or other method)
* Improve the software source code and documentation
* Update the System Performance Measures (to support 90-day field testing)
* Develop an interface to the RDE (now the ODE)

The second category of enhancements are “desired” and were discussed in the original project Proposal, Scope of Work, and at the MMITSS Development Group Meeting. These are discussed in Section XXX below.

First, a review of the two prototypes including architecture, design, features and functionality was conducted. On the basis of this review, several common libraries/components have been identified that will be developed to support both the AZ and CA prototypes. In addition, a common component interface approach will be developed that supports allowing any two components to interact to achieve some desired functionality. This common interface approach will be used in both the AZ and CA prototypes to allow exchange of components (if desired) and interoperation between the two software system architectures. [This will be discussed in greater detail in Section XX].

Based on this review, an iterative and incremental approach to development and testing will be used throughout the project.

For example, as the enhancements are made to the code, such as implementing the J2735 2016 messages and the development of a RSU API for the message transceiver will be tested as one build. Then, other improvements and refactoring of the code will be tested. Finally, the traffic control enhancements, such as coordination, will be developed and tested. Using an iterative and incremental approach will help ensure a stable and operating system during the project. The detailed schedule of the build/test cycles will depend on several factors, including the input received from the MDG about improvements and enhancements. Some testing, such as the enhanced traffic control features will likely be tested early in the development process due to the improvements being developed prior to the start of the project.

## Task 3: Enhancement of Existing MMITSS Prototypes

In the development of this plan, a review of the current MMITSS prototypes has resulted in identification of several common libraries/components that can be abstracted from the current code base and then updated/improved in the modified code base and used by both the AZ and CA prototypes. Figure 1 illustrates these common components that will support both the MMITSS-AZ and the MMITSS-CA prototypes, as well as other CV applications that could be developed in the same framework.

Common Libraries/Components

J2735 Message Library, RSU 4.1 Interface, MAP Engine,

NTCIP Library, GPS Library, …

MMITSS – AZ

Components

MMITSS – CA

Components

Other CV App Components

(Work Zone, Ramp Meter, BIM, EVA, RSA, …)

Figure 1. Common MMITSS Libraries/Components

Both prototypes need to upgrade the J2735 messages from the 2009 standard to the 2016 standard. In addition, it is likely that there will be further updates to the SAE J2735 standard and it is desirable to isolate these changes to a single library/component so that future updates will be contained to this library/component. Similarly, an interface to the RSU based on the FHWA 4.1 specification will be developed to isolate the communications between the RSU and the MMITSS Roadside Processor (MRP) to a single component. A common MAP library will be developed that supports critical MAP related functions such as determining if a vehicle is within the coverage of the MAP message, locating a vehicle on the MAP, estimating the distance from the vehicle to the stop bar, etc. A common interface to the traffic signal controller based on NTCIP will also be developed and a GPS interface module will be developed to allow MMITSS components to have access to GPS data where needed.

These common components will be based on the standards that are available to support making applications such as MMITSS hardware agnostic (Phase III Objective), but there are also situations where no standard exists and vendor specific software will have to be developed. For example, there is no OBU interface standard. Where possible, an interface will be defined and developed that can be ported to other vendors devices. However, since the vendor implementations are substantially different, it won’t be possible to ensure that this approach will result in a new interface standard within the scope of this effort.

This approach has implications to the improvements to the source code and documentation as well. Currently, the objects (or structures) that represent the messages that are used in MMITSS (including the BSM, MAP, SPaT, SRM, and SSM) are used in many of the software components, but they have been modified in each of the components for local use. This impact was an artifact of the development process where individuals (e.g. PhD students and team members) developed and tested the different components with only a loose attempt at coordination since the work was focused on new algorithm development and each of the developers added new data to the objects as needed. Hence, the development of a common J2735 message library will also require the development of an interface that allows MMITSS components to get and set message data from a set of common objects that are used throughout the other MMITSS components. These common objects will be the building blocks for each of the MMITSS components and will be utilized in the code improvements (refactoring).

An iterative and incremental approach will be used in the development. First, the common libraries/components will be developed. The interfaces to these components will be defined through a detailed review of each component that will utilize the interface. This review will identify the objects used in each component and required in the interface. The common component interface will be designed and used to implement the communication between the components. Once these common libraries/components are developed and tested, they will be used in the improvements and refactoring of each of the MMITSS components that implement the traffic control algorithms.

### Task 3.1 Update to the SAE J2735 2016 Standard

The Phase II effort utilized the SAE J2735 2009 standard and the MMITSS team participated in making improvements to the standard to support the MMITSS operation. There are significant changes to the 2016 version of the standard including a new message frame for every message, a new MAP message specification, a new Signal Status Message (SSM) that is based on the MMITSS Phase II priority use case, and the use of UPER encoding of the message payload. The MAP message is used in several MMITSS components to determine if a vehicle is in the roadway or off the roadway, such as in an adjacent parking lot. Hence, changing the MAP representation and functions in MMITSS is a significant change.

<<Kun is defining how this will be developed>>

### Task 3.2 Develop an API Specification for an RSU Deployed Message Transceiver

In the Phase II effort, the MMITSS prototypes utilized Savari specific DSRC WAVE stack functions to transmit and receive BER encoded messages. To support the goal of a hardware agnostic solution, an API specification will be developed, based on the DSRC Roadside Unit (RSU) Specifications Document v4.1[[2]](#footnote-2) message format and requirements, that will support use of RSUs from multiple vendors that conform to the 4.1 specification.

Initial testing has shown that this approach will work, but different vendors have implemented the RSU 4.1 interface differently. For example, one vendor requires that the security certificate header be included in the properly encoded J2735 message, but another vendor assumes that this header was removed. This has caused interoperability issues.

<<possible approach>> - since we know that Savari wants the header and Cohda and others do not, can we have a child class for Savari devices (requires configuration support) that handles this difference. If the device is not a Savari device, then it just leaves it off>>

### Task 3.3 Development of a Common Component Interface

Currently, the MMITSS-AZ and MMITSS-CA prototypes both used sockets to allow the software components to communicate with each other. The two prototypes are based on slightly different architectures, but they are both based on the concept of components sharing data to accomplish the desired traffic control algorithms. The MMITSS-CA prototype uses a common data manager component where all data is available to any component at any time. The MMITSS-AZ prototype assumes that any component can communicate with any other component to get or provide data when it is needed. While architecturally different, they both use the approach of getting and providing data between components.

A common communication approach (sockets) with serialized communications and a common set of data objects will be designed and developed that will allow any of the MMITSS-AZ components to communicate with MMITSS-CA components and vice versa. Serialization is the process of packing the data in an object (or data structure) into a bit stream that can be transmitted over a communications network. This same mechanism can be used if software components are run on the same computer or on different computers (distributed).

Currently, MMITSS-AZ uses sockets with data elements defined based on the J2735 (2009) messages and data elements with additional data elements defined as needed. Serialization is done using an ASN.1 encoding approach or manually packing messages into the bit stream. MMITSS-CA uses internally defined data objects (structures) and manually packing of the bit stream. Harmonizing the data objects (elements) and using an open source serialization library, such as Boost[[3]](#footnote-3) libraries, will theoretically allow the different components to interoperate. It should be noted that the two prototypes use some different data objects, such as the MMITSS-AZ vehicle trajectories, that might not allow complete integration, but this harmonization and common approach will allow some level of integration and some future integration as desired.

XXX

### Task 3.4 Enhancements to the AZ Prototype

This section discusses enhancements to the MMITSS-AZ prototype to ensure the prototype is ready for deployment. These enhancements will be tested in the Task 4 Field Readiness Test. It is assumed that the Task 2 effort will identify, prioritize, and estimate the required level of effort of additional enhancements identified by the MDG and Pooled Fund Panel. The discussion below is the MMITSS team’s suggestions for enhancements.

[Note: All category 2 and 3 improvements will be approved by the CV PFS before development starts.]

The enhancements to the MMITSS-AZ prototype are grouped into three categories: Category 1- improvements to the MMITSS code and documentation, Category 2 - MMITSS prototype system enhancements to support the 90-day field readiness testing (Task 4) and Category 3 - traffic control enhancements that operational agencies will require/desire for deployment.

### Task 3.4.1 Improvements to the MMITSS-AZ Code and Documentation (Category 1)

This task will focus on improving the MMITSS-AZ source code, software build manual, installation manual, and software design documentation. As described in Task 2.6, monthly webinars will be held with the MDG to conduct reviews of the source code, including modifications made under this project. Each review will focus on one of the software components, including the detailed design documentation (revised). The MMITSS-AZ installation manual will be updated and one agency will be ask to review and attempt to set up an intersection in MMITSS. The source code and documentation will be maintained and updated in the OSADP repository.

### Task 3.4.2 Update to the System Performance Measures (Category 2)

The system performance measures will be updated to collect data to support the Task 4.2 – 90 Day Readiness Testing, including collection of system uptime (i.e. System Availability), number of messages (vehicles) processed, and other system performance data identified to support evaluation of the system performance over the 90-day readiness test.

### Task 3.4.3 Add Support for RDE Data Capture (Category 2)

The Research Data Exchange (RDE) is operated by FHWA to provide data sets for researchers to access for the purpose of supporting research, analysis, application development and support. Adding features to support data collection for reporting to the RDE will require that the desired data format be defined and the current logging functions modified to produce data logs in the defined format. The level of effort required for this development will depend on the required format, but some initial effort has been estimated for the purpose of budgeting. The MMITSS team will provide data logging/streaming that can be captured and uploaded to the RDE.

### Task 3.4.4 Enhance Traffic Control Features to Support Deployment Readiness (Category 3)

Category 3 enhancements include features needed to support additional use cases such as integration between coordination and priority, section level priority, and the N-level priority policy, as well as new functionality identified in Task 2. Since the March 2015 MMITSS-AZ Field Test, progress has been made on several of these items under support from other projects. For example, Mehdi Zamanipour has been working on the integration of coordination and priority while serving as an FHWA Intern and NAS Fellow at the Turner-Fairbank Transportation Research Laboratory. His work is currently being tested using simulation and limited field testing. His work needs to be integrated into the MMITSS AZ code base, tested, and installed in the Arizona Test Bed for field testing and evaluation. The Utah Department of Transportation (UDOT) is supporting the addition of peer-to-peer communication of priority request to extend the headway (range) of request to allow the priority control algorithms to have additional time to plan better accommodation of priority requests. This enhancement will support section level priority (route priority) for emergency vehicles, transit and trucks. Both of these enhancements will be included in the Arizona MMITSS code base and updated to the OSADP once they have been developed, simulation tested, and field tested before the completion of this MMITSS Phase III task. This effort is assumed to be included in this project for no additional development cost, but will be included in the field testing Task 4.1.

Additional traffic control enhancements may be identified by the MMITSS Developers Group in the Task 2 effort. The cost and impact on the overall project budget will be reviewed with the Pooled Fund Panel when they are identified and cost estimates prepared.

### Task 3.5 Enhancements to the CA Prototype

The enhancements to the MMITSS-CA prototype are grouped into four categories: 1) improvements to the MMITSS code and documentation, 2) MMITSS prototype system enhancements to support the 90-day field readiness testing (Task 4), 3) add traffic controller interface for deploying MMITSS-CA with NTCIP controllers, and 4) traffic control enhancements that operational agencies will require/desire for deployment.

[Note: All category 2, 3 and 4 improvements will be approved by the CV PFS before development starts.]

### Task 3.5.1 Improvements to the MMITSS-CA Code and Documentation (Category 1)

This task will focus on improving the MMITSS-CA source code, software build manual and automake scripts, installation manual, and software design documentation. The MMITSS Team will participate in the webinars with the MDG to conduct reviews of the source code, update build manual/scripts and installation manual. The source code and documentation will be maintained and updated in the OSADP repository.

### Task 3.5.2 Add Support for 90-Days Field Readiness Testing (Category 2)

The system performance measures will be updated to collect data to support the Task 4.2 – 90 Day Readiness Testing including collection of system uptime (e.g. Availability), number of messages (vehicles) processed, and other system performance data identified to support evaluation of the system performance over the 90-day readiness test. The data capture features will be added to support data collection for reporting to the RDE. The MMITSS Team will provide data logging/streaming that can be captured and uploaded to the RDE.

### Task 3.5.3 Add Traffic Controller Interface for Deploying MMITSS-CA with NTCIP Controllers (Category 3)

In MMITSS-CA, the decisions for priority control (desired green split for priority cycles, when and how to place priority call to the traffic controller, etc.) and for coordination control (desired cycle length, splits, and/or offsets, etc.) are made in the MRP based on data from connected vehicles. The control commands are sent to the Caltrans 2070 controller using AB3418 serial communications protocol. The MRP receives signal status raw data from the 2070 controller via AB3418 protocol and encodes SPaT messages. This task focuses on adding an interface to NTCIP controllers using NTCIP objects for ASC to support deploying MMITSS-CA over NTCIP controllers. The existing NTCIP objects for sending control parameters such as cycle length, splits, offsets, force-off points, will be added in parallel to AB3418 messages over Caltrans 2070 controllers.

### Task 3.5.4 Traffic Control Enhancements to Support Deployment Readiness (Category 4)

Enhancements include adding traffic control features to support section level priority control, N-level priority, and integration with coordination control.

In the current (Phase II) MMITSS-CA, TSP is implemented by using the 2070 controller’s built-in priority control logic. The MRP (MMITSS Roadside Processor) receives Signal Request Messages (SRMs) and Basic Safety Messages (BSMs) from transit vehicles, prioritizes the requests for priority, selects the priority request to serve, and sends the request to the 2070 controller to grant the priority. Caltrans division of traffic operations is modifying the 2070 control software to allow dynamically change force-off points for priority control. The modification shall be in place for MMITSS Phase III, which allows more intelligent priority and traffic control, i.e., determination of phase splits based on connected vehicle data and system detector data in real time.

#### Task 3.5.4.1 Section Level Priority Control

In MMITSS Phase II, signal priority control is managed at the intersection level, without the consideration of coordinating the signal priority control along a section of signals. A vehicle that obtained priority at an upstream intersection may have to stop at a downstream intersection, resulting in possible interruption of signal progression and no net travel time saving for the vehicle. The section-level priority control coordinates the priority strategies among a section of signals based on projected movements of requested vehicles, signal status and estimated queue length of each intersection, thereby gaining net travel time savings for priority eligible vehicles and reducing side impact on general traffic. Santa Clara Valley Transportation Authority (VTA) operates both the BRT line and local bus lines along the California test network, which provide the environments for testing N-Level Priority Policy, and evaluate priority benefits on BRT buses vs. local buses.

#### Task 3.5.4.2 Integration with Coordination Control

Improved traffic coordination or progression also benefits the priority eligible vehicles when they are travelling with major platoons. Signals in the California test network are operated under the actuated-coordinated control mode with three time-of-day timing plans. Signal coordination parameters, including cycle length, phase sequence, split and offset, are programed based on historical field survey data and/or field observations, and does not reflect the dynamic and changing of traffic flow and speed. During peak hours, queue spillback often occurs at closely spaced intersections. With the enhanced queue length estimation, detection of vehicle trajectories (e.g., VTA buses and other connected vehicles), and count and occupancy data from system detectors, the coordination timing plan parameters will be dynamically adjusted to adapt to the prevailing traffic conditions and requests for priority.

1. The MMITSS team proposed to hold this meeting at the Turner-Fairbank Highway Research Center in McLean, VA so that as many participants as possible can attend. [↑](#footnote-ref-1)
2. FHWA, DSRC Roadside Unit (RSU) Specifications Document v4.1, DTFH61-12-D-00020, October 31, 2016. [↑](#footnote-ref-2)
3. Provide a link to the boost libraries on the web [↑](#footnote-ref-3)