

**Multi Modal Intelligent Traffic Signal System**

**Field Deployment – Installation Manual**

Revision 0.3

February 08, 2021

**Table of Contents**

1. Purpose of Document 3

2. MMITSS System Architecture 3

3. Systems Requirements 4

4. Deployment – Docker Containers 5

**List of Figure**

Figure 1: MMITSS field deployment architecture 3

**Revision History**

|  |  |
| --- | --- |
| **Revision No. (Date)** | **Description** |
| 0.2  (09/13/2020) | (1) Updated the name of container-launch script  (2) Updated the content of container-launch script |
| 0.3  (02/08/2020) | (1) Added examples of configuration files (mmitss-coordination-plan.json, mmitss-bus-stop-location.json)  (2) Updated the example of the mmitss-phase3-master-config.json file and container-launch script |

# Purpose of Document

This document is an instruction guide for deploying Multi-Modal Intelligent Traffic Signal System (MMITSS) applications in the real world. The document contains the detailed configuration and usage instructions for deploying the MMITSS software components in the docker container.

# MMITSS System Architecture

The physical MMITSS architecture is shown in Figure 1. This basic architecture is applicable at each MMITSS controlled intersection. In a system that consists of several intersections, each intersection would have a roadside unit (RSU), Econolite Connected Vehicle Co-Processor (CVCP) and traffic signal control equipment. The connected vehicle would have an onboard unit OBU and vehicle side co-processor (VSP). OBU is very similar to the RSU, except OBUs are designed specifically for integration with vehicle electronics system.

**Figure 1:** MMITSS field deployment architecture

# Systems Requirements

To deploy MMITSS in the real world following requirements must be met:

1. MMITSS roadside software components are run on the Econolite Connected Vehicle Co-Processor (CVCP) as the MMITSS Roadside Processor (MRP) [This processor was selected because it is field hardened for the environment, but a comparable ARM process, such as a Raspberry Pi, could be used]. The MRP (CVCP) must be networked to the traffic signal controller and the Roadside Unit (RSU). To run MMITSS roadside software components in on the CVCP, install Ubuntu Bionic 18.04.3 LTS operating systems. The operating systems can be installed by following the instructions found in [https://boundarydevices.com/ubuntu-bionic-18-04-3-lts-for-i-mx6-7-boards-august-2019-kernel-4-14-x/#](https://boundarydevices.com/ubuntu-bionic-18-04-3-lts-for-i-mx6-7-boards-august-2019-kernel-4-14-x/)

MMITSS roadside application can be run on the Server or cloud based environment also. MMITSS Arizona team is working on it. In future a guideline will provide how to deploy MMITSS on the Server or cloud.

1. MMITSS vehicle side software components are run on a Raspberry Pi that is connected to the On-Board Unit (OBU). To run MMITSS vehicle software components on the Raspberry Pi, Ubuntu 18.04 Server operating systems must be installed.
2. Install docker and supervisor on the MRP and the vehicle side processor (VSP) arm box.
3. If the arm box doesn’t have the internet access in the field, pull the docker image from Docker Hub and load the docker image (for roadside applications: mmitssuarizona/mmitss-mrp-arm:1.1 or for vehicle applications: mmitssuarizona/mmitss-vsp-arm:1.1)

docker pull <image name>

docker load –i <directory of the image>

If transferring the docker image from different machine, pull the docker image from Docker Hub, save it as a tar ball, and transfer the tar ball to the arm box. Then ssh into the arm box using it’s IPv4 address and load the docker image.

docker pull <image name>

docker save -o <path tar file> <image name>

scp <path to tarfile> armboxname@<IPv4address>:<destination>

ssh <name of the armbox>@<IPv4address>

docker load –i <directory of the image file>

1. If the MMITSS path is not set already, set the MMITSS path in the .bashrc file by executing the following command:

Export /MMITSS\_ROOT=<mmitss directory>

For example, if mmitss is cloned on /home/user directory then the command will be:

Export /MMITSS\_ROOT=/home/user

# Deployment – Docker Containers

To deploy MMITSS software components in the field, follow the following steps.

**Step1:** Create configuration files

It is required to create mmitss-phase3-master-config.json, mmitss-coordination-plan.json, and mmitss-data-external-clients.json configuration files for the MRP container and mmitss-phase3-master-config.json, and mmitss-bus-stop-location.json configuration files for the VSP container. The mmitss-phase3-master-config.json configuration file contain the IP addresses, UDP ports, and other configuration data which are required to establish communication between the MMITSS software components. An example of mmitss-phase3-master-config.json is following:

{

"HostIp": "xxx.xxx.xxx.xxx",

"SourceDsrcDeviceIp": "xxx.xxx.xxx.yyy",

"IntersectionName": "xxx",

"MapPayload":001283fe38083020315abe2149d0eecf1800a0000271c4fcbd028280",

"IntersectionID" : XXXX,

"RegionalID" : 0,

"DataCollectorIP": "xxx.xxx.xxx.xyx",

"HMIControllerIP": "xxx.xxx.xxx.yxx",

"MessageDistributorIP": " xxx.xxx.xxx.zzz ",

"PriorityRequestGeneratorServerIP": "xxx.xxx.xxx.zzz",

"VehicleType" : “Transit”,

"Logging" : "True",

"SRMTimedOutTime" : 10.0,

"ScheduleExecutionBuffer": 1.0,

"SystemPerformanceTimeInterval": 300.0,

"ApplicationPlatform": "roadside",

"PeerDataDecoding": false,

"CoordinationPlanCheckingTimeInterval": 300,

"PortNumber":{

"MessageTransceiver":{

"MessageSender": 10003,

"MessageReceiver": 10002,

"MessageEncoder": 10003,

"MessageDecoder": 10002

},

"MessageDistributor": 5000,

"RsmDecoder": 10006,

"OBUBSMReceiver": 10005,

"HostBsmDecoder": 10005,

"TrajectoryAware": 20001,

"PriorityRequestServer": 20002,

"PrioritySolver": 20003,

"PriorityRequestGenerator": 20004,

"TrafficControllerInterface": 20005,

"TrafficControllerCurrPhaseListener": 20006,

"TrafficControllerTimingPlanSender": 20007,

"PerformanceObserver": 20008,

"HMIController": 20009,

"PrioritySolverToTCIInterface": 20010,

"SignalCoordination": 20011,

"MapSPaTBroadcaster": 6053,

"DsrcImmediateForwarder": 1516,

"PriorityRequestServer\_SendSSM": 50003,

"DataCollector": 30001,

"SnmpEngine": 20020,

"SnmpEngineInterface": 20021,

"PriorityRequestGeneratorServer": 20022,

"TrajectoryAware\_MapEngineInterface": 20023,

"MapEngine": 20024,

"LightSirenStatusManager": 20025,

"PeerToPeerPriority": 20026 "SnmpEngine": 20020,

"SnmpEngineInterface": 20021,

"PriorityRequestGeneratorServer": 20022

},

"psid": {

"map": "8002",

"spat": "8002",

"rsm": "8002",

"srm": "8002",

"ssm": "8002",

"bsm": "20"

},

"msgId": {

"map": "0012",

"spat": "0013",

"rsm": "0021",

"srm\_lower": "001d",

"srm\_upper": "001D",

"ssm\_lower": "001e",

"ssm\_upper": "001E",

"bsm": "0014"

},

"TxChannel": {

"map": 172,

"spat": 172,

"rsm": 172,

"srm": 182,

"ssm": 182,

"bsm": 172

},

"TxMode": {

"map": "CONT",

"spat": "CONT",

"rsm": "CONT",

"srm": "ALT",

"ssm": "ALT",

"bsm": "CONT"

},

"SignalController": {

"IpAddress": "xxx.xxx.xxx.yyy",

"NtcipPort": 501,

"TimingPlanUpdateInterval\_sec": 600,

"NtcipBackupTime\_sec": 300,

"Vendor": "Econolite",

"TimingPlanMib": "/nojournal/bin/EconoliteMib.py",

"InactiveVehPhases": [],

"InactivePedPhases": [],

"SplitPhases": {

"1": 6,

"3": 8,

"5": 2,

"7": 4

},

"PermissiveEnabled": {

"1": true,

"3": true,

"5": true,

"7": true

}

},

"IntersectionReferencePoint": {

"Latitude\_DecimalDegree": 33.82249812,

"Longitude\_DecimalDegree": -112.2315621233,

"Elevation\_Meter": 906

},

"DataTransfer": {

"FtpServerPort": 9090,

"StartTime": {

"Hour": 18,

"Minute": 30

},

"EndTime": {

"Hour": 19,

"Minute": 30

},

"MaxRetries": 3

},

"PriorityParameter": {

"EmergencyVehicleWeight": 1.0,

"EmergencyVehicleSplitPhaseWeight": 0.1,

"TransitWeight": 1.0,

"TruckWeight": 1.0,

"DilemmaZoneRequestWeight": 2.0,

"CoordinationWeight": 0.1

}

}

The mmitss-coordination-plan.json file contain the coordination plan parameter and split data to run priority-based coordination system of MMITSS. An example of mmitss-coordination-plan.json file is following:

{

"IntersectionName": "xxx",

"CoordinationParameters": [

{

"CoordinationPlanName": "AM-Plan",

"CoordinationPatternNo": 1,

"SplitPatternNo": 1,

"CycleLength": 90,

"Offset": 0,

"CoordinationStartTime\_Hour": 6,

"CoordinationStartTime\_Minute": 30,

"CoordinationEndTime\_Hour": 9,

"CoordinationEndTime\_Minute": 30,

"CoordinationSplit": 20.0,

"CoordinatedPhase1": 2,

"CoordinatedPhase2": 6,

"SplitPatternData": {

"PhaseNumber": [

1,

2,

3,

4,

5,

6,

7,

8

],

"Split": [

15,

39,

12,

24,

15,

39,

16,

20

]

}

},

{

"CoordinationPlanName" :"PM-Plan",

"CoordinationPatternNo": 2,

"SplitPatternNo": 2,

"CycleLength": 90,

"Offset": 10,

"CoordinationStartTime\_Hour": 15,

"CoordinationStartTime\_Minute": 30,

"CoordinationEndTime\_Hour": 19,

"CoordinationEndTime\_Minute": 0,

"CoordinationSplit": 20.0,

"CoordinatedPhase1": 2,

"CoordinatedPhase2": 6,

"SplitPatternData": {

"PhaseNumber": [

1,

2,

3,

4,

5,

6,

7,

8

],

"Split": [

19,

35,

12,

24,

19,

35,

16,

20

]

}

}

]

}

The mmitss-bus-stop-location.json file is required only for the transit vehicles. Transit vehicle dwells in the bus stop for boarding and taking off passengers. MMITSS software component (*priority-request-generator*) is designed to send priority request after passing the bus stop (for transit vehicle). The mmitss-bus-stop-location.json file. It contains the information of the bus stop location for each transit vehicle (depends on the travel route). An example of mmitss-bus-stop-location.json file is following:

{

"NoOfBusStop": 5,

"BusStopInformation": [

{

"IntersectionName": "xx-yy",

"IntersectionID": xy,

"TravelDirection": "EastBound",

"ApproachNo": 3,

"Latitude\_DecimalDegree": 33.142863,

"Longitude\_DecimalDegree": -110.134406,

"Elevation\_Meter": 739

},

{

"IntersectionName": "yy-xx",

"IntersectionID": yx,

"TravelDirection": "EastBound",

"ApproachNo": 5,

"Latitude\_DecimalDegree": 32.250825,

"Longitude\_DecimalDegree": -112.416047,

"Elevation\_Meter": 960

},

{

"IntersectionName": "yy-zz",

"IntersectionID": yz,

"TravelDirection": "WestBound",

"ApproachNo": 1,

"Latitude\_DecimalDegree": 30.289396,

"Longitude\_DecimalDegree": -112.112475,

"Elevation\_Meter": 687

},

{

"IntersectionName": "zz-yy",

"IntersectionID": zy,

"TravelDirection": "WestBound",

"ApproachNo": 1,

"Latitude\_DecimalDegree": 34.346593,

"Longitude\_DecimalDegree": -108.183794,

"Elevation\_Meter": 546

},

{

"IntersectionName": "xx-zz",

"IntersectionID": xz,

"TravelDirection": "WestBound",

"ApproachNo": 6,

"Latitude\_DecimalDegree": 35.9643024,

"Longitude\_DecimalDegree": -114.934988,

"Elevation\_Meter": 839

}

]

}

1. For the MRP container, “*HostIp”,* “*SourceDsrcDeviceIp”, “IntersectionName”, “MapPayload”, “IntersectionID”*, *signal controller* “*IpAddress”, “NtcipPort”,* and “*NtcipBackupTime\_sec”, “Vendor”, “TimingPlanMib”* are required. The “*HostIP”* must match the ip address of the connected vehicle co-processor (CVCP). The *“SourceDsrcDeviceIp”* must be the RSU IP. The map payload can be obtained by creating an intersection map using USDOT map tool (<https://webapp.connectedvcs.com/isd/>).
2. For the VSP container, *“HostIp”,* “*SourceDsrcDeviceIp”,* “*VehicleType”* are required. The vehicle type has to be one of following strings:
   1. “Transit”
   2. “Truck”
   3. “EmergencyVehicle”

1. Create a log folder which must be placed in the same directory structure of the *mmitss-phase3-master-config.json file*. To log the data, specify *“Logging”: “True”* in the *mmitss-phase3-master-config.json file* otherwise specify it as *“False”*.

**Step 2:** Launch scripts

To run the MRP/VSP container the launch-container.sh script is required. The script can be placed in the mmitss directory.

read -p "Full absolute path of MMITSS configuration directory (with no trailing /): " -e config\_path

read -p "Name of container image on the Dockerhub: " container\_image

read -p "Name of container: " container\_name

docker run --privileged -d --restart always -v $config\_path:/nojournal --network host --name $container\_name $container\_image > /dev/null 2>&1 &

**Step 3:** Define the source of configuration files

1. Go to cd <path-to-mmits> directory
2. Create a folder which can be named as intersection name (for MRP container) or vehicle type (for VSP container). Then create a bin folder. In the bin folder place the mmitss configuration files and create log folder. For example-

cd /home/mmitss

mkdir emergency-vehicle

cd emergency-vehicle

mkdir /bin

cd bin

mv /home/mmitss/mmitss-phase3-master-config.json mmitss-phase3-master-config.json

mkdir log

**Step 4:** Run docker container

1. Run the launch-container.sh script to run the docker container. The VSP container can be start by executing the following command:

launch-container.sh

A user interface will appear and following information have to be provided.

Full absolute path of MMITSS configuration directory: <path-to-mmitss>/emergency-vehicle

Name of container image on the Dockerhub: mmitssuarizona/mmitss-vsp-arm:1.1

Name of container: vsp\_container

1. To monitor the containers, execute the following command:

docker container exec –it <container name> /bin/bash

1. To stop all the containers, execute the following command:

docker container stop <container name>

1. To start container the execute the following command:

docker container start <container name>