**Multi Modal Intelligent Traffic Signal System（MMITSS）**

**Field Deployment – Installation Manual**

Version 1.0

Mar. 2023

CART Lab, Purdue University

**Table of Contents**

1. Systems Requirements

2. Deployment – Docker Containers

3. Configuration

**Overview**

This documentation is created for the deployment for MMITSS in the city of Ann Arbor for the Smart Intersection Project funded by FHWA. To be specific, MMITSS could also be built from sources or from pre-developed images. This file is used for deploying from pre-developed docker images and run in docker containers.

**Revision History**

|  |  |
| --- | --- |
| **Revision No. (Date)** | **Description** |
| 1.0  (05/20/2023) | 1. Initials on MMITSS Introduction with system structures. 2. Add deployment steps, configuration examples. |

# System Requirements

Some required preparations have to be done before implementing MMITSS into the field.

1. Both MMITSS roadside (MRP) and vehicle-side (VSP) applications are run in an Ubuntu based operation system, the version is 18.04 and 20.04 LTS. The roadside unit (MRP) is running on an x86-based device while the vehicle-side unit (VSP) is running on an arm-based device. (i.e., a laptop for MRP and a Raspberry Pi 4 for VSP respectively)
2. To deploy MMTISS applications in container, docker and supervisor are required to install first. (URL: <https://docs.docker.com/engine/install/ubuntu/>)
3. Docker images need to be extracted from DockerHub and loaded for the field deployment. For Ann Arbor testing, a pre-tested docker image has been uploaded publicly. The following shows the detailed sources for these two images.
   * For roadside applications: cartlabpurdue/mmitss-mrp-x86:AA1.0

<https://hub.docker.com/r/cartlabpurdue/mmitss-mrp-x86>

$ *docker pull cartlabpurdue/mmitss-mrp-x86:AA1.0*

* + For vehicle-side applications: mmitssuarizona/mmitss-vsp-arm:2.1

<https://hub.docker.com/r/mmitssuarizona/mmitss-vsp-arm/tags>

$ *docker pull mmitssuarizona/mmitss-vsp-arm:2.1*

## Configurations

**File 1**: *mmitss-phase3-master-config. json*

This section will introduce the details of modifying configuration files for the deployment of MMTISS. *mmitss-phase3-master-config. json* are the major configuration files for MRP and VSP for the IP addresses and UDP ports. There are two types of ports in this configuration, one is for the inner applications for MMITSS while another is the external port. The following list shows the external port that needs to be configured, where the highlighted are required to modify based on the edging computing devices:

1. For the MRP container, “*HostIp”,* “*SourceDsrcDeviceIp”, “IntersectionName”, “MapPayload”, “IntersectionID”*, *signal controller* “*IpAddress”, “NtcipPort”,* and “*NtcipBackupTime\_sec”, “Vendor”, “TimingPlanMib”*, “*IntersectionReferencePoint*” , “*Priority Parameter*” are required.
   * “*HostIP*”: ipv4 IP addresses for current edge device
   * “*SourceDsrcDeviceIP*”: ipv4 IP addresses for communication device, for MRP container, it should be RSU IP and for VSP, it should be OBU IP.
   * “*IntersectionName*”: name for the current intersection in MRP.
   * “*MapPayload*”: The map payload can be obtained by creating an intersection map using USDOT map tool (<https://webapp.connectedvcs.com/isd/>).
   * “*IntersectionID*”: must be consistent with the ID from map payload.
   * “*IpAddress”*: ipv4 IP address for current signal controller.
   * “*NtcipPort*”: NTCIP communications port, must be consistent with controller settings
   * “*NtcipBackupTime\_sec*”: must be consistent with controller settings.
   * “*Vendor*”: Vendor Name of controller, **for Ann Arbor deployment with SIEMENS controller, set it as “Standardmib”**
   * “*TimingPlanMib”*: path for additional NTCIP mib file. **for Ann Arbor deployment with SIEMENS controller, it is not used.**
   * *“ProrityParameter”*: set the weights of priority for different vehicle types and coordination for traffic operations.
2. For the VSP container, *“HostIp”,* “*SourceDsrcDeviceIp”,* “*VehicleType”* are required. The vehicle type has to be one of following strings:
   * “Transit”
   * “Truck”
   * “EmergencyVehicle”

**File 2**: *mmitss-coordination-plan.json*

The mmitss-coordination-plan.json file contains 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

]

}

}

]

}

**File 3**: *mmitss-bus-stop-location.json*

The mmitss-bus-stop-location.json file is required only for the transit vehicles. Transit vehicles dwell in the bus stop for boarding and taking off passengers. MMITSS software component (priority-request-generator) is designed to send priority requests 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 (depending 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

}

]

}

**File 4**: *supervisord.conf*

This file is used for managing the applications in MMITSS container. A template file is attached in Github repo in the directory: *AA\_Deploy\_2023/MRP\_config/nojournal/bin* ([*https://github.com/Dylan-Wyl10/SIP-MMITSS.gi*](https://github.com/Dylan-Wyl10/SIP-MMITSS.gi)). For the deployment purposes, there is no need to modify this file.

# Field Deployment

Required files for field deployment include configurations for MRP and VSP, as well as the launch scripts. Purdue Team has published these required files in Git-Hub, and they could be pulled from the command below. For the Ann Arbor deployment, please access the directory: *$HOME/AA\_Deploy\_2023*:

$ *git clone https://github.com/Dylan-Wyl10/SIP-MMITSS.git*

After the finishing the software preparations on docker image, follow the rest steps:

**Step1**: Create configuration files/folder.

It is required to create a configuration folder for MMITSS to access configuration files and restore its running log. The configuration folder needs to follow a given structure in order to successfully run the applications. Both MRP and VSP follow the same structure but including different files. Create a log folder which must be placed in the same directory with *mmitss-phase3-master-config.json* To log the data, specify “Logging”: “True” in the mmitss-phase3-master-config.json file otherwise MMITSS will specify it as “False”. A detailed structure is listed below:

-**ROOT**

-**nojournal**

**-bin**

**-log**

**-** mmitss-phase3-master-config.json (MRP,VSP)

**-** mmitss-bus-stop-location.json (VSP)

**-** mmitss-data-external-clients.json (MRP)

- mmitss-coordination-plan.json (MRP)

- supervisord.conf (MRP, VSP)

Noted that the bold text means the folder directory. The *mmitss-phase3-master-config.json* configuration file contains the IP addresses and UDP ports. The other json files mainly contain the required inputs for MMITSS applications. The supervisord.conf is used for managing the applications running in docker. For Ann Arbor deployment, a configuration template will be provided by Purdue Team in GitHub Repo (<https://github.com/Dylan-Wyl10/SIP-MMITSS/tree/main/AA_Deploy_2023> )

**Step2**: Launch MMITSS container

After putting the configuration files properly in locate device, execute the launch scripts to run MMITSS applications. The scripts are in path: */AA\_Deploy\_2023/launch\_scripts.* For a new device that firstly deploys MMITSS applications, it is required to firstly execute *setup-deployment-environment.sh* to set up the environment variables. Then execute *launch-container.sh* to build MMITSS containers.

A detailed example for running *launch-container.sh* script is showed below:

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 the following information have to be provided. The bold text needs to be figured out based on deployment details.

Full absolute path of MMITSS configuration directory: **<path-to-mmitss>/mrp\_config**

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

Name of container: **vsp\_container**

Specify timezone string: **America/New\_York**

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>*

## Settings for related devices

* + - 1. **Communication devices**

Communication devices, OBU and RSU, are required in the field applications for MMITSS for the information exchange between VSP and MRP. An IMF script needs to be executed for the deployment. Noted that MMITSS has defined a port to receiving the forwarded messages from communication devices (*MessageReceiver: 10002*). So, the IMF script needs to configure its forward address as: *MRP\_IP: 10002.*

MMITSS-VSP will utilize the BSM from OBU for localization, the OBU has an inner application that could send its GPS location with a Basic Safety Message (BSM) following J2735 standard. The scripts to open this application are listed below:

$ *sudo su*

$ *ifconfig cw-mon-tx up*

$ *socat INTERFACE:cw-mon-tx UDP4-SENDTO:<VSP IP>:10002*

* + - 1. **Signal controller**

MMITSS will receive the SPaT mib from signal controller apart from communicating with NTCIP. The receive port is predefined as 6053. So the controller are supposed to send SPaT mib to: <MRP IP>: 6053