IISC EV Charger integration with RB-i.MX6UL

Version: V1.0

Customer: IISC

Project Co-ordination Team

Activity	Name	Email-ID	Company
Prepared By	G V SAIKRISHNA	sai.k@phytec.in	PHYTEC

Table of Contents

1. Overview

This document defines the integration of RB-i.MX6UL SBC with EV Charger (TMS controller Board) and the cloud connectivity between RB-i.MX6UL and AWS cloud. The combination of EV charger with RB-i.MX6UL becomes EV charging station. The charging station has 3 ports where the vehicles are to be charged at different voltage levels (typically 48V, 60V, 72V) and the respective charging port parameters (like voltage and current ratings of each ports etc.) and operations (like ON/OFF etc.) on each port can be remotely controlled using a Web App or a Mobile App and even locally from a LCD Display with touch inputs. And hardware switches at the EV charger circuitry can also be used for turning ON/OFF the respective charging ports.

2. Scope of Work

2.1 Interfacing RB-i.MX6UL with TMS controller

- RB-i.MX6UL and TMS controller shall communicate over a CAN Bus to exchange the charger information and charger operations (like ON/OFF etc.).
- The CAN frames shall be defined and maintained in a proper way for different types of scenarios/messages to distinguish one from the other and the respective error codes also to be defined.

2.2 LCD Display Integration with RB-i.MX6UL

- RB-i.MX6UL shall have a LCD Display (with touch inputs) interfaced over LVDS.
- There shall be an application which can display the charging station information onto the LCD display and can take up the inputs form the display and operate accordingly.
- And the LCD application shall get all the necessary information (charging port details) form the CAN interface and then pushed to display screen.
- The LCD Display screen shall update with latest information even if the operations (to be taken) is received from the Mobile App or Web App and vice versa.
- In case of any error while inputting the data, a pop-up window should be open with the error details
- There shall be a STOP button for each port to stop charging and disconnect the batter at any given time

2.2.1 LCD Display parameters list

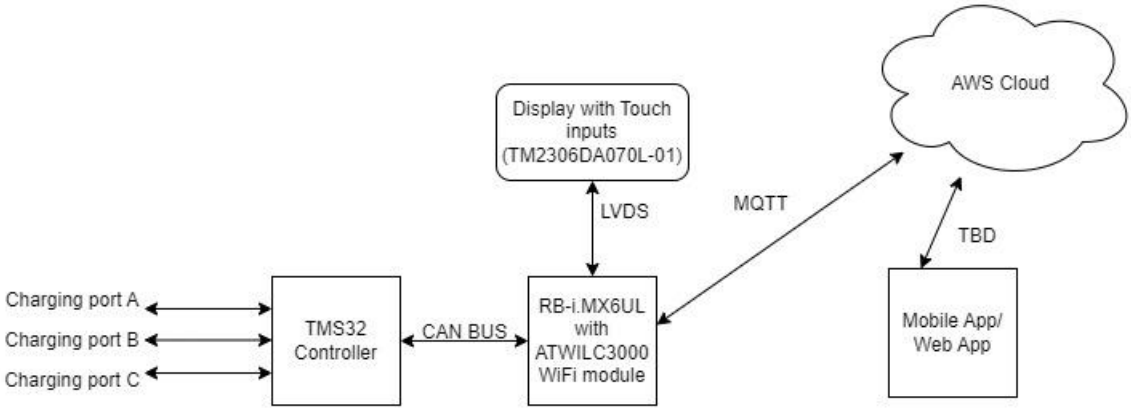
The list of parameters to be shown on the display is as follows:

- Battery voltage and current values from the 3 charging ports.
- Power grid voltage and current values and its availability.
- ON/OFF buttons for selecting the charging ports and input box to provide the respective voltage rating of the battery connected to particular port.

2.3 Cloud connection with RB-i.MX6UL

- RB-i.MX6UL shall connect to the AWS cloud with proper certificates.
- There shall be an application which can have PubSub model to exchange the information between the device and cloud.
- RB-i.MX6UL shall have the Wi-Fi module to connect to the AWS Cloud.
- JSON packet format shall be used for data exchange. And JSON packet should have the proper format for data fields.

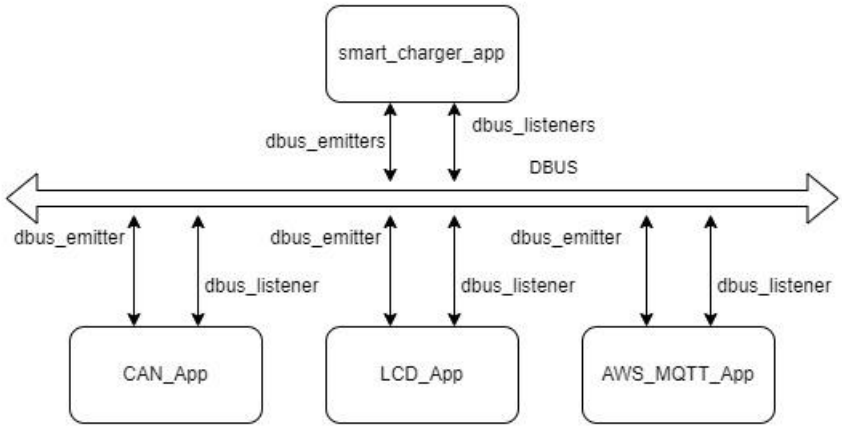
3. Hardware Block diagram



The inter connections among the sub units of the system is as shown above. ATWILC3000 is the Wi-Fi module which can be interfaced with RB-i.MX6UL over SDIO and provides the internet for AWS Cloud connection.

4. Software Architecture

4.1 Software Block Diagram



The above is the high level software block diagram specifying all the main software components of the system. And the detailed explanation is as follows.
In order to maintain the modularity in the software, the entire software is divided into sub components and D-Bus is used as an IPC between the each component for data exchange among them.

D-Bus wrapper Library:
This is a user defined wrapper library consisting of the following D-Bus operations

- Establishing the D-Bus connection/disconnection
- Creating filters for the D-Bus listeners/emitters
- Read/Write from the D-Bus.

CAN_App:
This component is responsible for CAN Bus handling and the respective operations are follows

- Create D-Bus emitter and listener threads
- CAN Bus initialization
- CAN Bus Read/Write operations

As soon as it gets the CAN frame, it does basic analysis like CAN id filtering and for a valid CAN frame, emits a signal with a specific payload and the observer of this signal can act upon.

LCD_App:
This component is responsible for LCD handling (outputting the content, reading the user inputs via a touch panel) and emits/listens the respective D-Bus signals to pass the information to the main application.

AWS_MQTT_App:
This component will take care of the AWS cloud connectivity and publish/subscribe data

Smart_charger_app:
This is the main component of the software which has the decision making capacity with multiple condition checking from the remaining software components and update the information to Cloud, LCD and charging ports.

4.2 CAN frame format

The CAN messages format between the RB-i.MX6UL and TMS controller are as follows.

4.2.1 CAN frame from RB-i.MX6UL to TMS controller

The below are the CAN messages that can be sent from RB-i.MX6UL to TMS controller over CAN Bus

- Turn ON port-A/B/C with some specific voltage request
- Turn OFF port-A/B/C request
- Grid Availability request

B7	B6	B5	B4	B3	B2	B1	B0

B0 – Shall specify whether the given CAN frame is a request or response or a status frame

Value of the B0 byte is
0x10 – Request
0x60 – Positive Response (Adding 0x50 to the command value)
0x11 – Status frame
0x6F – Negative Response for a given Request

B1 – Shall specify the port numbers or Power Grid source and its state (ON/OFF)

Lower Nibble of B1 – Port selection (PA/PB/PC) and Grid
0x01 – PA
0x02 – PB
0x03 – PC
0x04 – Grid (Higher Nibble of B1 is always OFF)
Higher Nibble of B1 – Port State (ON/OFF)
0x01 – ON (Connected)
0x00 – OFF (Disconnected)

B2 – will specify the integral part of port (PA/PB/PC) voltage. In case of Grid Query, this byte is don't care condition

B3 – will specify the fractional part of port (PA/PB/PC) voltage. In case of Grid Query, this byte is don't care condition

B4, B5, B6 and B7 are don't care conditions.

Example CAN frame:

- Turn ON port-A with 49.36V
  CAN frame: **7E0#10113124** where 7E0 is the CAN id
- Turn ON port-C with 49.36V
  CAN frame: **7E0#10133124** where 7E0 is the CAN id
- Grid Availability Query
  CAN frame: **7E0#1004** where 7E0 is the CAN id
- Turn OFF port-A
  CAN frame: **7E0#1001** where 7E0 is the CAN id

4.2.2 CAN frame from TMS controller to RB-i.MX6UL

The below are the CAN messages that can be sent from TMS controller to RB-i.MX6UL over CAN Bus

- Response for Turn ON port-A/B/C with some specific voltage request
- Response for Turn OFF port-A/B/C request
- Response for Grid Availability request
- Voltage and Current values of the ports(PA, PB, PC) during the Vehicle is in charging condition at respective port
- Grid availability indication whenever there is a state change in the Grid availability(Grid power source connection/disconnection)
- Vehicle connection/disconnection indication with battery nominal voltage details

B7	B6	B5	B4	B3	B2	B1	B0

B0 – Shall specify whether the given CAN frame is a request or response or status frame or Negative response

Value of the B0 byte is
- 0x10 – Request
- 0x60 – Positive Response (Adding 0x50 to the command value)
- 0x11 – Status frame
- 0x6F – Negative response for a given request

B1 – Shall specify the port numbers and its state (ON/OFF)

Lower Nibble of B1 – Port selection (PA/PB/PC) and Grid
- 0x01 – PA
- 0x02 – PB
- 0x03 – PC
- 0x04 – Grid
- 0x05 – Vehicle detection

Higher Nibble of B1 – Port State (ON/OFF)
- 0x01 – ON (Connected)
- 0x00 – OFF (Disconnected)

In a negative response frame, this byte value will be the same as that of B1 Byte in the given request frame. This is to identify the error code is for which request.

B2 – Shall specify the integral part of port (PA/PB/PC) voltage and Battery nominal voltage (in case of vehicle detection frame). In case of Grid ON condition, B2 shall specify the Half of the Grid voltage integral part. In negative response case and vehicle disconnection case, respective error code will be updated in this byte and rest of the bytes in the frame are don't care conditions.
B3 – Shall specify the fractional part of port (PA/PB/PC) voltage and Battery nominal voltage (in case of vehicle detection frame). In case of Grid ON condition, B3 shall specify the fractional part of the Grid voltage.
B4 – Shall specify the integral part of port (PA/PB/PC) current. In case of Grid ON condition, B4 shall specify the Grid current integral part
B5 – will specify the fractional part of port (PA/PB/PC) current. In case of Grid ON condition, B5 shall specify the fractional part of the Grid current.
B6 and B7 are don't care conditions.

Example of CAN frame:

- Response for Turn ON port-A with some specific voltage
  CAN frame: **7E1#60110D130812**
  where 7E1 is the CAN id of TMS controller. Port V = 13.19V and I = 8.18A
- Response for Turn OFF port-B
  CAN frame: **7E1#6002**
  where 7E1 is the CAN id of TMS controller.
- Negative Response for Turn ON port-C with some specific voltage
  CAN frame: **7E1#6F130A**
  where 7E1 is the CAN id of TMS controller. Request failed and the respective error code = 0x0A
- Reply for Grid Availability Query
  CAN frame: **7E1#6014C72B1E06**
  where 7E1 is the CAN id of TMS controller. Grid V = 398.86V and I = 60.12A
- A voltage and Current value of the ports (PA, PB, and PC) during the Vehicle is in charging condition at port-A
  CAN frame: **7E1#11110D130812**
  where 7E1 is the CAN id of TMS controller. Port V = 13.19V and I = 8.18A
- Grid availability indication whenever there is a state change in the Grid availability (Grid power source connection/disconnection)
  CAN frame: **7E1#1114C72B1E06**
  where 7E1 is the CAN id of TMS controller. Grid is connected and V = 398.86V and I = 60.12A
  CAN frame: **7E1#11041A**
  where 7E1 is the CAN id of TMS controller. Grid is disconnected and respective error code = 0x1A

- Vehicle connection indication with battery nominal voltage details
  CAN frame: **7E1#11150D13**
  where 7E1 is the CAN id of TMS controller. Vehicle is connected and it's battery nominal voltage is V = 13.19V
- Vehicle disconnection indication with respective error code
  CAN frame: **7E1#11052A**
  where 7E1 is the CAN id of TMS controller. Vehicle is disconnected and it's respective error code is 0x2A

4.3 Cloud connectivity details

AWS Cloud side:

Subscribing topic: device_to_cloud
Publishing topic: cloud_to_device
MQTT message: <JSON STRING>
AWS endpoint: a29rjcslolshy0-ats.iot.us-east-1.amazonaws.com
AWS_MQTT_PORT: 8883

AWS Client side:

Subscribing topic: cloud_to_device
Publishing topic: device_to_cloud
MQTT message: <JSON STRING>
AWS endpoint: a29rjcslolshy0-ats.iot.us-east-1.amazonaws.com
AWS_MQTT_PORT: 8883

The required certificates are as follows:

root-CA.crt
switch.cert.pem
switch.private.key

4.4 JSON packet format

The JSON packet format between Device (RB-i.MX6UL) and AWS cloud is as follows

4.4.1 JSON object from RB-i.MX6UL to AWS cloud

The below are the JSON objects that can be sent from RB-i.MX6UL to AWS cloud

- Response for Turn ON port-A/B/C with some specific voltage request
- Response for Turn OFF port-A/B/C request
- Response for Grid Availability request
- Voltage and Current values of the ports(PA, PB, PC) during the Vehicle is in charging condition at respective port
- Grid availability indication whenever there is a state change in the Grid availability(Grid power source connection/disconnection)
- Vehicle connection/disconnection indication with battery nominal voltage details

4.4.2 JSON object from AWS cloud to RB-i.MX6UL

The below are the JSON messages that can be sent from AWS cloud to RB-i.MX6UL

- Turn ON port-A/B/C with some specific voltage request
- Turn OFF port-A/B/C request
- Grid Availability request

4.4.3 JSON object structure details

The below is the JSON object structure specifying all the fields and their respective values. And the below table brief each field name, data type and description. The same JSON shall be used for sending to LCD module as well to update the same information on the display.

JSON object structure:

```
{
        "ev_station": {
                "status": "active/inactive",
                "cause": 0,
                "id": 12345,
                "event": {
                        "type": "request/response/status/error/check_grid/vehicle_detection",
                        "trigger_from": "cloud/ev_station/lcd",
                        "is_request": true/false,
                        "is_err": true/false,
                        "err_code": 0,
                        "request": {
                                "port_id": 456789,
                                "action": "set/get/reset",
                                "voltage": 48.45
                        },
                        "reponse": {
                                "port_id": 456789,
                                "is_avail": true/fasle,
                                "voltage": 48.45,
                                "current": 2.10
                        }
                }
        }
}
```

Field Name	Field Type	Description	Possible value
ev_station	JSON Object	This shall specify the respective EV station details (request, response etc.)	A valid JSON object
status	string	This shall describe the EV station condition based on the grid source/charger connector availability	active/inactive
cause	integer	In case of EV station in active condition, this shall have the respective error code	0(active)/non-zero(inactive)
id	integer	This is the unique id of the EV station	A non-zero number
event	JSON Object	This shall specify the details about the triggered event	A valid JSON object
type	string	The type of the event that triggered	request/response/status/ error/check_grid/vehicle_detection
trigger_from	string	It is the source from where the event got triggered	cloud/ev_station/lcd
is_request	boolean	Shall specify whether the given trigger is a request or not	true/false

is_err	boolean	This shall specify whether there is an error in the response of a given request or a system error event	true/false
err_code	integer	Respective error code in case of error	A Non-zero integer
request	JSON object	This shall specify the request with necessary inputs	A valid JSON object
port_id	integer	This shall specify the respective charging port where the vehicle is being connected for charging	A non-zero integer
action	string	This specifies the action to be taken on the respective charging port	set/get/reset
voltage	float	The voltage of the vehicle battery to which it has to be charged	A float value within the range of battery ratings
response	JSON object	This shall specify the response of a given request or a status or vehicle detection event from the EV station	A valid JSON object
port_id	integer	Response or status or vehicle detection event generation charing port	A non-zero integer
voltage	float	Battery voltage of the corresponding vehicle connected at this specified port	A non-zero float value
current	float	Battery current of the corresponding vehicle connected at this specified port	A non-zero float value