A Proposed System for Electric Charging Vehicle Infrastructure

Moreshwar Salpekar

Abstract Electric vehicle development has started in India. The Government of India has announced policies to promote electric vehicles. Electric vehicles are no different from other vehicles except they require charging. The charging is mostly controlled using microcontroller based systems in both vehicle and charging station. The paper proceeds to briefly describe how the design of software and firmware. It also gives the challenges that are perceived in design and briefly describes some reasons for the challenges and issues. It then proposes a design for the system that uses machine learning system to improvise the charging and generate billing accurately. The proposed system will give a generic software stack and how machine learning system will interact with this stack. This is expected to help companies build an efficient software for both charging and billing.

Keywords Electric vehicle · Battery management system · Firmware · Security

1 Introduction

The Government of India announced the FAME policy in 2018 to promote electric vehicles. This was followed by policies by states last one being Delhi which will set up electric vehicle charging stations every 3 km. Electric Vehicle charging infrastructure is already coming up in states all over India. Organisations are developing both vehicles and charging infrastructure both of which are controlled and monitored by firmware with respect to charging and on road operation.

The firmware residing in either charging station infrastructure or electric vehicle is usually modular and is in form of stack. Quite a few parts of the firmware modules or firmware modules like State of Charge Estimation have been patented even but still stack is generic. The stack even includes capability to send data over network for further processing and analysis like billing and data analytics.

M. Salpekar (✉)
Boson Motors Pvt. Ltd., Pune, India

R. K. Pillai et al. (eds.), *ISUW 2019*, Lecture Notes in Electrical Engineering 764,
https://doi.org/10.1007/978-981-16-1299-2_12

Communication networks have evolved to allow huge amount of data to be sent over network. Clouding computing also has evolved now to allow this data to be processed.

Machine Learning residing in the cloud allows data analytics to be carried out on data allowing automation of some operations that earlier required manual involvement.

A generic stack can be developed which incorporates, the necessary functionality while adding data communication with cloud and data analytics so to allow better monitoring and control of the system (vehicle and charging station).

2 The Electric Vehicle Charging Environment

The electric vehicle charging infrastructure firmware can be divided into two parts.

a. Software that resides in electric vehicles and controls and monitors the operation of battery including charging. It consists of Battery Management System, Motor Drive, Electronic Vehicle Charger Controller (EVCC) and Electronic Control Unit (ECU which is the master).
b. Software that resides in charging station and controls monitoring of the charging operation.

The data analytics system residing in the cloud has a software running on a server and caters to the analysis and processing of data.

2.1 The Electric Vehicle Firmware

The block diagram of this is shown in Fig. 1. It resides in the electric vehicle and is responsible for electric vehicle charging and operation. Three subsystems are identified in this firmware

1. Battery Management System
2. Motor Control System
3. Electronic Control Unit (Main Controller).

Each of the above are structured according to the generic software stack is given in Fig. 2 (based on Autosar Classic platform 4.4) [5]

Following vertical software modules relevant to this paper are identified for each of the above. It should be noted that firmware in each of the hardware module confirms to the below vertical module sub-divisions.

1. Control Module
2. Communication Module.

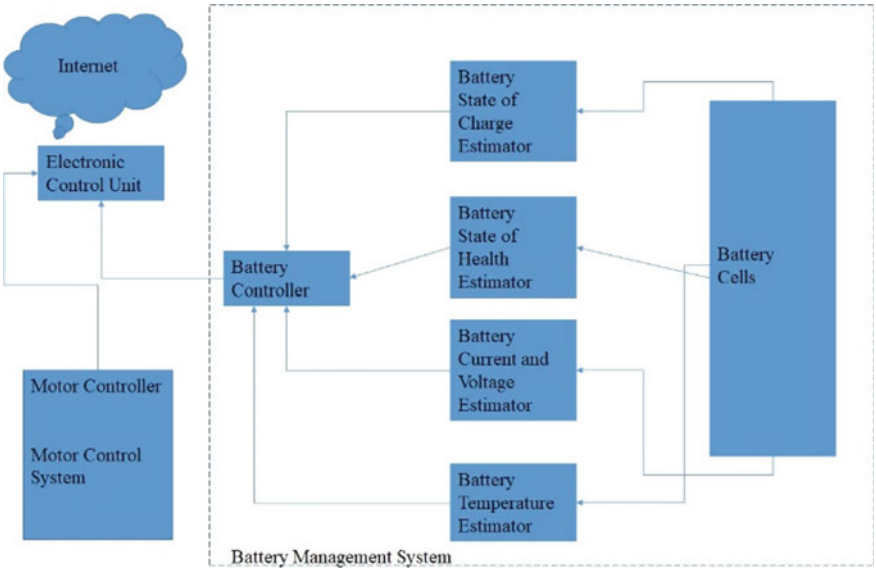

Fig. 1 The electric vehicle subsystems (derived from [1–4])

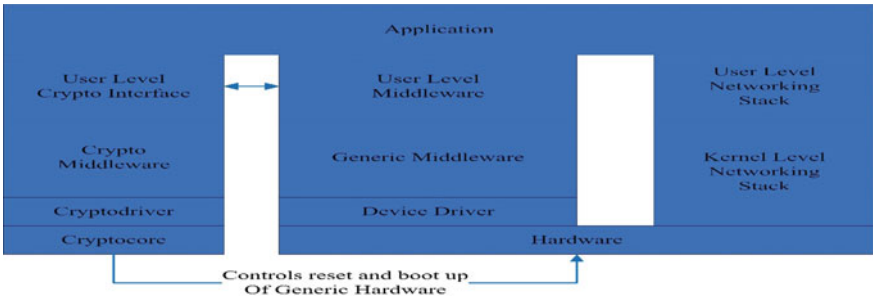

Fig. 2 Software stack (derived from [5])

Each of them is controlled by a thread, with appropriate interrupts to handle incoming and outgoing data. The threads communicate with each other using messages and shared memory.

Each of this is given in sections below.

2.1.1 Control Module

This module is responsible for controlling the hardware it is responsible for. It includes hardware initialization and getting measurements of parameters from

system. Each subsystem has its own control module. The following control modules are identified

1. BMS Control Module: Battery Management System (BMS) has responsibility to control and monitor battery. The battery is comprised of cells connected in series. Therefore, it collects data from various cells in form of cell voltages and temperatures. It then computes total voltages total current that can be given out by battery. Optionally, it may compute State of Charge (SoC) and State of Health (SoH) of battery. It may also perform cell balancing and report balancing status. Thus the BMS firmware reports Battery Voltage, Battery Current, Battery Temperature, and Optionally SoC, SoH and Cell Balancing status (see [6] for more information).
2. Display Control Module: It is responsible for controlling LCD display and get user inputs from LCD
3. Motor Control Module: It controls operation of Motor that provides torque and controls rotation (measured by RPM)
4. Charge Control Module: It is used to control electric vehicle charger. It has its own communication module to communicate with charging station control module
5. Master Control Module: it controls and coordinates activities including configuration of other modules. It also collects data and passes to network for transmission to cloud.

2.1.2 Communication Module

It is responsible for communicating parameters to other subsystems. This module also sends data to the cloud for data analytics and get its response (only from main controller).

It may be divided into following

1. Bus Communication Module: it is used to transmit data over the bus e.g. CAN bus to send data or serial bus.
2. Network Communication Module: it is used to send data to cloud for analytics. It may use LPWAN or any other network stack. It is present only in the master control module.

2.2 The Charging Station Firmware

The Charging station firmware can also be vertically divided into.

1. Control Module
2. Communication Module.

2.2.1 Charging Station Control Module

It is responsible for getting data related to charging station. For charging station firmware, parameters are charging current, charging voltage, charging indicator, time of charging and charger temperature. It may optionally report maximum voltage and maximum current. If more than one vehicle can be charged, each charging point has different firmware.

Charging Station Communication Module

It is responsible for communicating parameters to other subsystems. This module also sends data to the cloud for data analytics and get its response (only from main controller). It also communicates with charger control module in electric vehicle if required.

The communication uses

1. Serial Peripheral Interface and/or UART to get data from charging station subsystems
2. Use a LPWAN to send data to cloud for analytics.

3 Addressing the Security Concern

Security needs to be addressed in design phase and not as afterthought. The security is proposed to be compliant with [7]. The security is added in the following manner

1. Boot up software security
2. Run time software security
3. Security in software download
4. Key Management.

3.1 Boot Up Software Security

A Crypto core is added to system for boot up along with secure RAM. The software and keys (non-fused) are stored in secure EEPROM in encrypted form. The crypto core boots starts and verifies the software (using SHA-4 hash, see [8]). Alternately Authentication Encryption (see [9, 10]) may be used. After successful verification, control is passed to two stage bootloader which finally boots up the kernel and application. The security for run time security is given below.

3.2 Run Time Software Security

This is ensured by the following

1. A Secure RAM is provided. This RAM is accessible only in specified operating mode. No debugger access is available to this RAM unless specified instructions are given.
2. Debugging through debugger is disabled in field. Debug can be done only by using prints and that too through kernel log or on console.
3. Software does not access secure EEPROM for anything than keys, if required, during runtime.
4. Even the new software is in this RAM before being flashed to secure EEPROM.
5. The downloaded software is always checked against the Secure HASH transmitted signed by the sender. The signature match and hash) are checked using crypto core (or authenticated encryption check). This is discussed in next section.

3.3 Software Download

The software needs to be updated in field for maintenance which includes feature change and bug fixes. The generic protocol is

1. The software to be downloaded is encrypted with AES 128 bit key (see [11]). Authenticated encryption is proposed to be used (see [9, 10]). Secure Hash Standard (SHS) [8] can also be used but two different algorithms may need further consideration in terms of crypto core processing and memory.
2. The upgrader sends the software and its hash signed with key.
3. The software is downloaded into RAM and also the key.
4. The crypto core is then requested to verify the software.
5. Upon verification, software is written to secure EEPROM.
6. The new software runs on next boot.

3.4 Key Management

Key Management is proposed to be conformant to [12]. The keys required and processing is given below.

3.4.1 The Keys

The following keys are proposed to be present and used

1. Signature key which is used to compute secure hash of software (it is used for software verification).
2. Software OTA decryption key: used to decrypt software sent over the air.
3. Software encryption/decryption (SE/D) key: The software is stored encrypted in secure ROM using this. This is a symmetric key i.e. same key used for encryption and decryption.
4. Encrypter Key: used to encrypt above keys. This is symmetric key.

3.4.2 Key Update

The basic process for update is always same and is as follows:

1. The sender sends request to update key and request is encrypted using previous key and signed by sender.
2. The request is validated using crypto core.
3. The recipient, on successful verification, sends the acknowledgment.
4. The sender sends new key encrypted and signed.
5. The keys is decrypted and verified.
6. The key is stored in secure EEPROM upon successful verification and acknowledgement sent to sender.

Each of key requires more steps after basic process

1. For Signature key: the new hash of software must be sent and hash verified with new key before key is stored.
2. For SE/D key: It must be possible to decrypt the existing software in secure EEPROM using this key else following steps must be done.
   a. The full software is copied from secure EEPROM and copied to RAM
   b. The software is encrypted using new key and stored in secure EEPROM.
3. For Encrypter key: the other two keys must be stored afresh in RAM after encrypting with new key.

4 Conclusion and Further Work

A full software system architecture and high level design is given for electric vehicle charging infrastructure is given. This is an initial architecture. More refinements and partitioning will be present in the architecture. Further, the software also has to comply with ISO 26262 [13], Autosar [14], Coding guidelines such as MISRA [15] also need to be followed to implement a full working system. The communication modules also need a full design thought like choosing which communication protocol to be followed. Usually, Controller Area Network (CAN) [16] is used for communicating for automotive parts so it is expected to be used in communication between

motor, BMS and master controller. However, there is a flexibility that other protocols may be used. It is left to implementer to work with actual protocols, hardware components, etc.

References

1. Xing Y, Ma EWM, Tsui KL, Pecht M (2011) Battery management systems in electric and hybrid vehicles. Energies 4:1840–1857. https://doi.org/10.3390/en4111840
2. Renesas (2018) Battery management system tutorial. Retrieved from www.renesas.com/doc/whitepapers/battery-management/battery-management-system-tutorial.pdf on 25th Dec 2018
3. Hu R (2011) Battery management system for electric vehicle applications. Electronic Theses and Dissertations. Paper 5007. Retrieved through https://scholar.uwindsor.ca/etd
4. Kettles D (2015) Electric vehicle charging technology analysis and standards. FSEC Report Number: FSEC-CR-1996-15, U.S. Department of Transportation's University Transportation Centers Program
5. Autosar (2018) General specification of basic software modules AUTOSAR CP Release 4.4.0, 31st Oct 2018. Retrieved from www.autosar.org
6. Meissner E, Richter G (2003) Battery monitoring and electrical energy management precondition for future vehicle electric power systems. J Power Sources 116:79–98
7. Information Technology Laboratory National Institute of Standards and Technology, FIPS PUB 140-2 Security requirements for cryptographic modules, federal information processing standards publication, 25 May 2001
8. Information Technology Laboratory National Institute of Standards and Technology, FIPS PUB 180-4, Secure hash standard (SHS), federal information processing standards publication, Aug 2015. https://doi.org/10.6028/NIST.FIPS.180-4
9. Dworkin M (2007) NIST Special Publication 800-38D, recommendation for block cipher modes of operation: galois/counter mode (GCM) and GMAC
10. Dworkin M (2001) NIST Special Publication 800-38A, recommendation for block cipher modes of operation methods and techniques
11. Daemen J, Rijmen V (2001) Federal information processing standards publication 197. Announcing the Advanced Encryption Standard (AES), 26 Nov 2001
12. Barker E, Roginsky A NIST special publication 800-133 recommendation for cryptographic key generation. https://doi.org/10.6028/NIST.SP.800-133
13. International Standards Organisation, ISO 26262 Road vehicles—functional safety—all parts, latest versions. www.iso.org
14. Autosar. www.autosar.org
15. The Motor Industry Software Reliability Association (2012) MISRA-C: 2012 guidelines for the use of the C language in critical systems
16. International Standards Organisation, ISO 11898-Road vehicles—controller area network (CAN)—all parts, latest versions, www.iso.org