**Design and Implementation of SAE J1939 and Modbus**

**Communication Protocols for Electric Vehicle**

Paper proposes a CAN bus charger communication protocol based on SAE J1939 standard.

The EV communicates to the charger controller via the battery management system (BMS) using CAN communication.

The EV charger power modules use Modbus communication protocol.

The paper integrates CAN and Modbus to implement communication between the EV BMS and the charger using a cost effective Arduino Uno microcontroller.

By using CAN bus module (MCP2515) and Modbus module (MAX485), the distance between the EV and the charger can be increased while making communication possible.

Communication is finally validated using the PCAN software.

**Introduction**

Coordination between the charger and the EV is essential during pre-charging, charging and post-charging. The EV and charging unit must be connected at the beginning of the charging cycle for identification and authorisation. Other data such as charging time, power flow direction, power availability, energy cost and EV status must also be shared during charging.

**J1939**

This is a high layer CAN-based protocol. J1939 carries the communication between electronic control units (ECUs) and sensory units. Data that is sent during the communication process include vehicle speed, torque, power availability and temperatures.

**SAE J1939 Specifications**

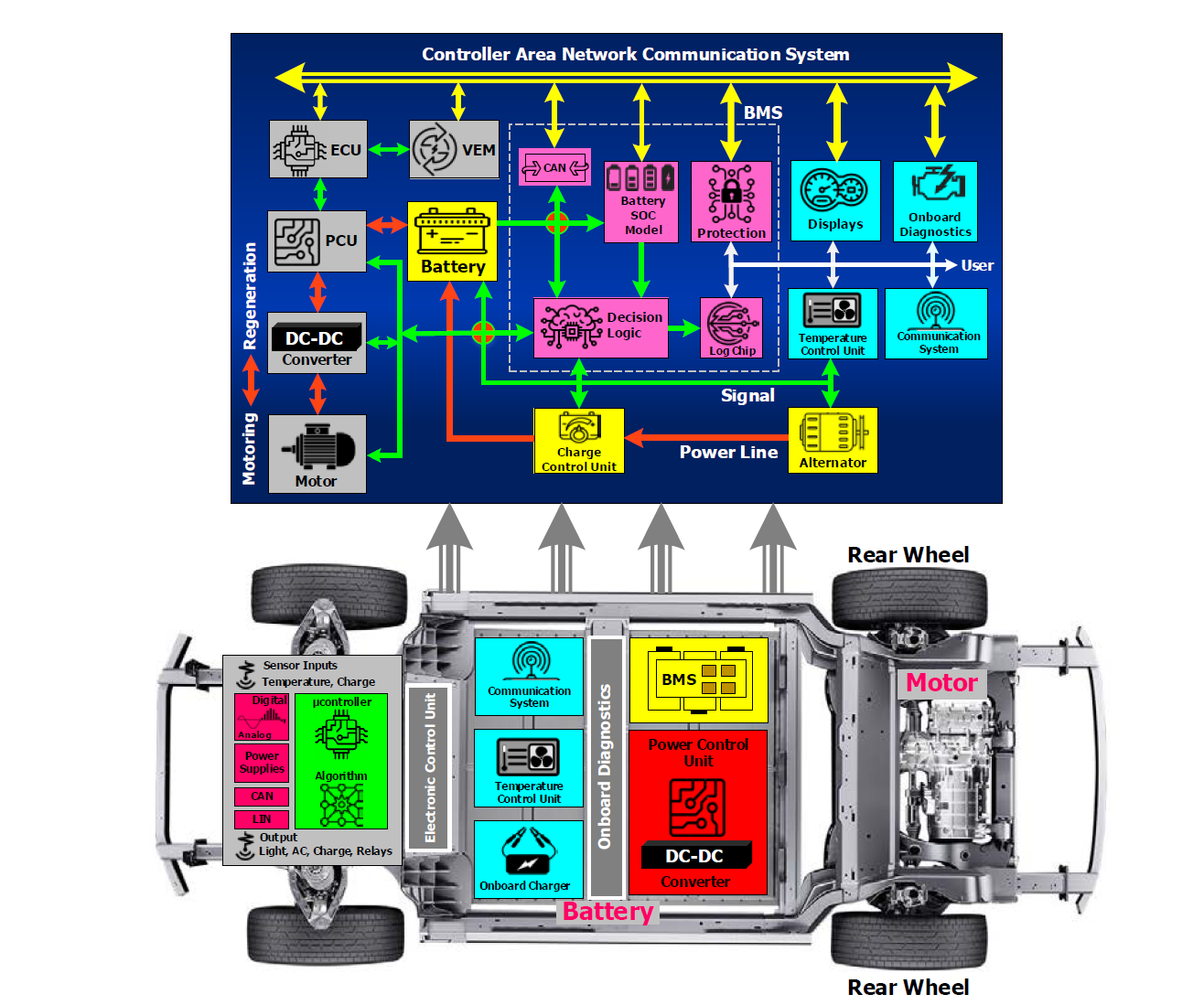
|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Wire | Shielded-Twisted Pair |
| Network Length | 40m |
| Standard Baud Rate | 250 kbps |
| ECUs | Maximum of 30 nodes |
| Controller Applications | Max of 253 |
| Message Lengths | 1758 bytes |
| Additional Supports | Peer-peer broadcast communications  Better data bandwidth  Defines parameter group numbers  Network administration with node IDs  Address requesting process |

CAN communication is not appropriate for network administrations and communications with higher than 8 bytes of data. Hence, SAE 1939 was designed to enable support for unlimited message length as well as network control using node IDs.

Messages are sent over CAN in extended 29-bit format. The BMS and charger controller are connected via CAN. The distance between the EV and the charger is extended by the CAN bus module (MCP2515) and the Modbus module (MAX485). This communication is then verified by PCAN software.

**EV Energy Management and Charging**

The EV consists of the following units:

* Energy management unit (VEM)
* Charge control unit
* Electronic control unit (ECU)
* Monitoring and communication system

**Vehicle Energy Management with the Power Control Unit**

The vehicle energy management unit (VEM) and power control unit (PCU) are both used to manage energy flow between the battery and the motor drive.

The energy control unit receives instructions from the BMS, vehicle sensors and vehicle operator through the communication bus.

The goal of the VEM is to optimise energy consumption.

1. It computes the electric motor’s power requirement by analysing the user’s acceleration and estimates the available battery charge.
2. The estimated information is then sent to the PCU
3. The PCU provides power to the motor drives using the bi-directional DC-DC power converters.

**ECUs** are used for managing the overall engine operations, and measures parameters such as acceleration, transmission and ambient conditions from vehicle sensors.

* It transmits required details such as
  + Air parameters
  + Volume of fuel pumped
  + Frequency of ignition to other peripheral vehicle equipment to ensure optimal working conditions
* It serves to set and monitor parameters for vehicle operating mode
* It is called the personality module, because it contains all the individual specifications that might be unique to configuration of the customer for programming in the framework
* This module provides for the setting of the desired battery state of charge (SOC) functional level and variables for controlling the electric drive
  + The VEM also provides a memory block to store all the reference data and to collect past data used to track the battery’s health

The **Vehicle Instrumentation module** can receive the data from the CAN bus and show the state of health (SOH) or turn ON warning lights.

The VEM outputs provide the points of reference for setting the battery’s operating conditions or triggering protective circuit action.

A typical RS232 or RS485 serial bus provides the test access to the BMS for testing or setting device parameters and for retrieving battery history.

**Battery Management System**

The BMS has the following functions:

* Determines quantity of energy running from EV batteries with higher precision
* Tracking and control of batteries
* Controls charging and discharging
* Controls cell balance
* Controls connection to high voltage line
* Cooling

The BMS is a microprocessor-based unit that integrates many different submodules and functions.

**Battery SOC Model**

This model is an algorithm for determining behaviour of the battery in response to different internal and external circumstances. The model logs past battery history to predict the state of the battery for service purposes or to estimate the range of the vehicle until it requires a recharge. The distance travelled by the EV can be obtained through the vehicle’s CAN bus.

The battery SOC is calculated by averaging the current flow over a period of time and lessening the value from the fully charged battery rating. The residual charge is calculated using the present SOC, distance travelled and energy expended, based on the latest driving trends.

**The Decision Logic Module**

The Decision Logic (DL) module does the following, in order:

1. Matches the SOC model’s measured or estimated battery parameters to the expected or referred outcome of the module.
2. The DL circuits then supply error signals to either trigger the cell safety behaviour or to be included in specific BMS feedback loops, which push the device to optimal operating point or to separate the battery in optimal operating point or to separate the battery in abnormal operating conditions.
3. These error signals are provided to the battery control unit for further processing

Error signals may be as follows:

* Battery isolation during fault conditions
* Responding to vehicle operating mode changes
* Cell balance-offset defects in individual cells
* Responding to shifts in modes of operation of vehicles
* Binary control and progressive control-limiting overload
* Shift the regenerative braking power into the battery as needed
* Discard unnecessary regenerative braking charges with full powered batteries

**EV Charge Control Unit**

This unit consists mainly of power electronic circuitry that switches the power connection to individual cells. These switches are driven by the control signals from the battery monitoring unit for the charging and discharging process. The process and mode for charging are defined with specific algorithms and conditions.

1. Constant current and constant voltage charging mode

The constant current (CC) of charging changes the output voltage of the charging equipment or the resistance of the battery to keep the current constant at the same level. The current constant is kept the same from start to finish of the charging process. This however, leads to sharp reduction in battery power. The charging voltage (CV) charging method keeps the voltage between the poles of the battery the same.

1. CC-CV switching algorithms

By integrating the two charging methods, charging can be enhanced. Due to the initial higher voltage, the battery can be charged to 90% using CC, and CV could be used to fully charge the vehicle.

1. Multi-step Current Charging Algorithm

This method splits the charging cycle into many stages which are aimed at optimizing the charging efficiency. This is done by using the optimal charging current in each phase. To implement this algorithm, a microcontroller or computer is required.

**Electronic Control Unit (ECU)**

The ECU is the central part of a multi-layer circuit board containing hundred of elements. The communication within the ECU and other parts of the vehicle is essential and security standards of this communications must be met. The protocol that starts controlling in-car communication is CAN. This transmission level sends messages at a rate of 500 kbps. Here are some further ECU components:

* Analog and Digital Converters with Signal Conditioner
  + Vehicle sensor output are typically read using a low level ADC converter module. The analog voltage values usually range from 0-1.5V. The ADC then converts the voltage value into a 10-bit integer. Furthermore, the sensor output must be filtered or “conditioned” to remove noise.
  + Similarly, a low-level DAC module provides a voltage output to operate some components of the EV
* High Level Digital Output
  + The ECU is tasked with controlling the switching operation of the cooling fan. The small amount of power from the ECU processor energizes the digital output transistors, which supply the cooling fan with much more power. Hence a voltage amplifier or driver circuit is required to obtain the high level digital output.

**Monitoring and system communication**

The exchange of information between the nodes, elements and operators within an EV is carried out by the CAN bus system.

* Monitoring system
  + Due to the increase in demand for fast charging capabilities, more EVs are enabling support for fast charging. As such, temperature monitoring systems are put in place to ensure the battery and electronics are safe during charging. Switches such as MOSFETs, IGBTs and SiCs are used in DC-DC converters and inverters to protect onboard power transformers and provide monitoring and stable cooling control.
* Communication system
  + CAN bus is like the nervous system of cars, and enables communication between all parts of the car. All the nodes (ECUs) are connected via the CAN bus, and allow efficient communication without requiring complex dedicated cabling. CAN is like a central networking system, and offers ECUs some advantages via software. For example, the ECU can communicate with the whole system without overloading the controlling device.

**Charging modes and level**

The potential negative effects of fast charging is that the unpredictable solid electrolyte interface can accelerate battery ageing, from multiple charging or discharging.

The modes of EV charging are shown below:

1. Mode 1: Domestic socket with an extension cord, without protection
   1. In this mode, the EV is directly connected to the power grid through the domestic outlets which are rated between 5-10 A. The charging socket will therefore share a switchboard feed with the other sockets, so if the total current reaches the safety limit, it automatically trips the breakers and stops the charging process. The electrical system has to follow some guidelines to use mode 1 charging, such as:
      1. It must have an earthing device
      2. It must have earth leakage protection
      3. It must have overload protection
2. Mode 2: Domestic socket with extension cord and protection
   1. This mode of charging connects EVs to the domestic socket outlets, similar to mode 1. In this case however, charging takes place via an available network, with an earthing cable installed. Inside the charger, the cable and sockets are protected. Hence, this approach is generally mor expensive.
3. Mode 3: Slow charging devoted socket with protection
   1. The EV is attached via a specific socket with a dedicated circuit to the electrical system. The safety and control circuitry is found within the wall-mounted charging system. This is the only charging mode that meets the IEC61851 standards.
4. Mode 4: Fast charging devoted socket with protection
   1. EV is connected externally to the power grid. Permanent installation with control and protection features. New pulse loading with improved solid electrolyte interface stability. The new loading method improves battery life by decreasing the chemical reaction in EV batteries.

The charging levels are shown below:

1. Level 1: Low strength
   1. These stations are deployed with single phase power supply and protection circuitry is simple. Contains a floor fault-sensing device and circuit breaker. The charger usually has additional features in addition to the usual BMS functions, such as
      1. Protection interlocks
      2. Isolators

This charger has low power (up to 1.9 kW), low cost and low impact on utility peak demand, in exchange for a very slow charging speed.

1. Level 2: Medium strength
   1. These chargers are designed and developed by communication intelligence to verify and authorize a user in order to begin charging their vehicle. The input power rating for level 2 chargers may be a phase 1/3 supply between 2.5 kW and 19.2 kW. This level charges faster than level 1, and is just as flexible as level 1, being able to charge using various AC sources. This level is more costly than level 1, and creates some impact on peak demand.
2. Level 3: High strength
   1. Similar to level 1 and 2, however it needs heavy duty modules, as it requires an input power of up to a 240kW 3-phase AC supply. Highly efficient and much faster than the previous levels, as well as more costly than the previous ones. It can be set up for public use, and creates huge impact on the peak demand.

**Charger and Vehicle Communication**

The EV charger may be housed onboard or offboard. The onboard charger is built specific to the EV’s battery model and size. The battery is charged by connecting the charger’s power inlet to the dedicated socket.

Offboard chargers require communication with the battery control system, and the communication protocol is independent of the model of EV or battery. Hence, offboard chargers can adapt to various technologies and standards of BMS. It also charges faster than onboard chargers, and is lighter and smaller.

**Charging sequence and control communication**

The battery is charged using either CC, CV or CC-CV charging methods. The vehicle control system is considered as the **master** and the charging control system is considered as the **slave**. The charging process is as follows:

1. The charging control process starts after the charge button initiates ON
2. The charge control system waits for permission from the EV to charge the vehicle. Until the EV has sent this message, the charger will NOT send any charge to the vehicle.
3. The vehicle sends a charging current request via the CAN bus to the charger
4. The output is provided according to the charging current request
5. Once the order value of the vehicle is notified through CAN communication, the system changes its order value within the cycle of 100 ms.

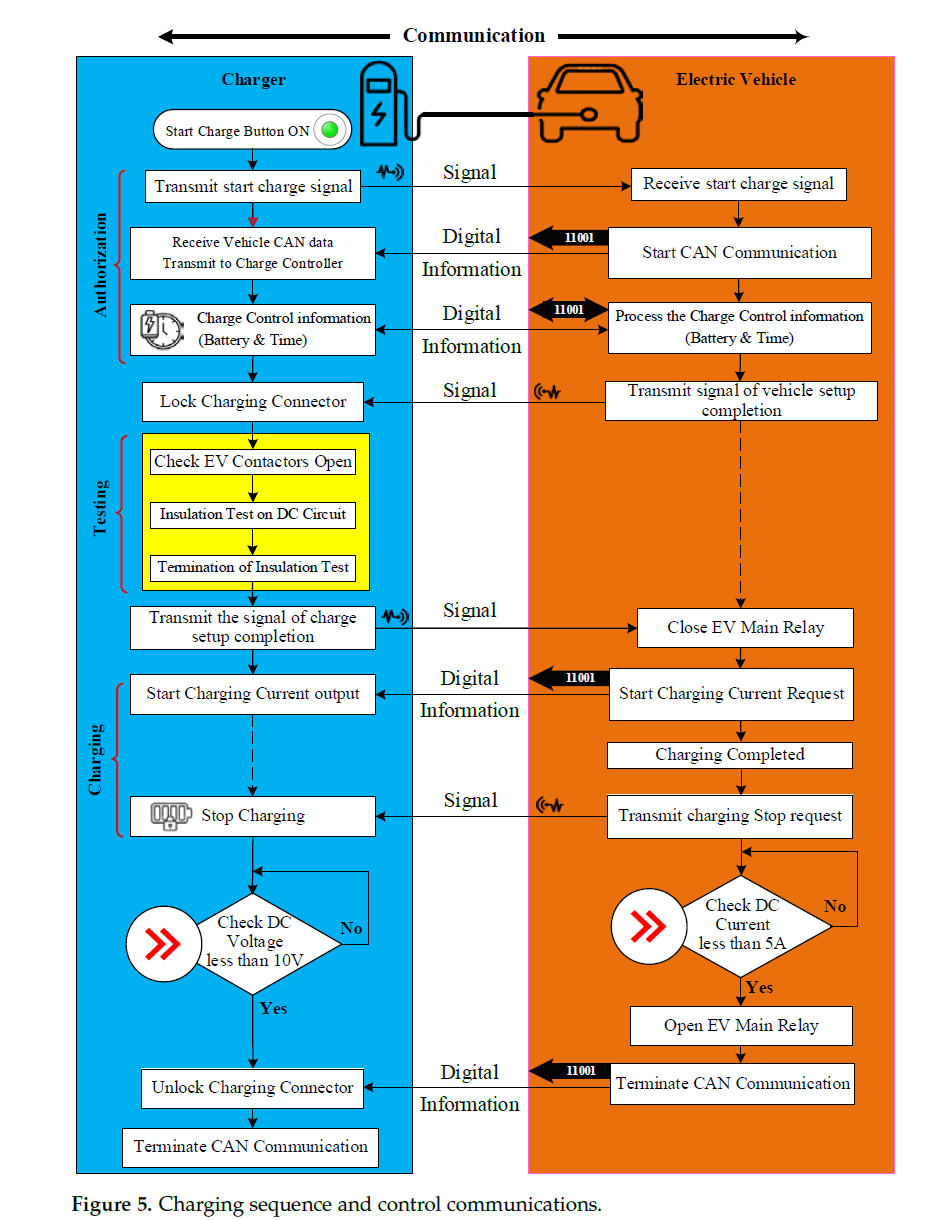
**CAN bus SAEJ1939 protocol**

Vehicle data, such as:

* Torque
* Speed of transmission system
* Engine temperature
* Oil level
* Battery level

Are shared through the CAN interface.

CAN is a serial network system for vehicular communication.



CAN delivers rapid communication between ECUs, real-time sensing and control systems.

**CAN is a two-wired, half-duplex, ultra-speed network which is as flexible as RS232.** This protocol reduces the cost of complicated dual port infrastructure.

CAN bus has its downsides, as it is not ideal for systems involving minimal network management, but more importantly it is not ideal for systems with signals that are larger than 8 bytes of data. As a result, additional software is integrated on the CAN physical layer to create an advanced network system that enables unlimited-length messages.

As such, CAN bus communication is improved using the SAEJ1939 standard, as it is a higher-layer software-based protocol. SAEJ1939 endorses node addresses and can receive data blocks over 8 bytes (limited to 1785 bytes).

A diagram of a can bus

AI-generated content may be incorrect.

SAEJ1939 is set up to use parameter tables, instead of depending on a multitude of protocol features. This protocol does not follow the existing master/slave or client/server architecture as with other CAN protocols. J1939 is a physical layer, and supports both **peer-to-peer** and **broadcast communication with a data rate of 250 kbps.** CAN bus can transmit up to 1765 bytes of data using **Transport Protocol (TP).**

There is a data circuit that facilitates one-on-one communication between the vehicle and charger. The charge control parameters shall be exchanged via this communication data circuit.

A diagram of a computer program

AI-generated content may be incorrect.When programming the system, it is important to take note of the configuration parameter groups and their numbers.

The EV charger can comprehend and supply the necessary energy by analysing CAN messages. This process has a specific format by which the CAN messages are sent and received. As per SAEJ1939, the following regulations must be followed when messages are communicated between the BMS and charger:

* Standard CAN messages must be 8 bytes long
* A message containing more than 8 bytes of data must be sent by a multi-packet message.

There are two types of multi-packet messages: **broadcast announcement messages and connection management.**

Broadcast management: Entire network will receive the messages

Connection management: Establish communication of messages between specific channels.

Connection management messages are used in CAN for security reasons.

**Modbus Communication Protocol**

Modbus is the most popular automation open protocol. It enables computers and other devices to communicate with each other in a common language. Modbus transmits information over serial lines.

In an EV, Modbus enables devices to transmit data to the ECU for comparison between various parameters connected to the same network.

Modbus **Master:** The computer demanding information

Modbus **Slave:** The devices receiving the information

In Modbus, there are up to **247 slaves per 1 master.** Each slave has its own slave address. Modbus messages are sent over interfaces such as **RS232 or RS485.**

Modbus communication uses a **two-wire** system for sending and receiving messages. The master can contact any specific slave, or may send a broadcast message to all the slaves. The slave collects messages and sends the reply message to the master via Modbus. **Slaves return an answer to all message requests sent personally to them, but do not reply to transmitted messages. Slaves do not send messages on their own, and only respond to the master’s message inquiries.**

The Modbus messages, or **Modbus Protocol Storage Unit (PSU)** service request field consists of the following:

* Feature code
* Number of data bytes demanded from the operator

The Modbus memory registers are arranged in 4 basic data reference types, which is identified by a leading number that is used in the computer memory address.

* 0-based reference register to distinct outputs or coils in read/write code
* 1-based reference register reading separates inputs
* 3-based reference register reading input information
* 4-based reference register is to read or write information to output or store them

Function area codes: Used to read or write to and from the slave.

The request message contains the following:

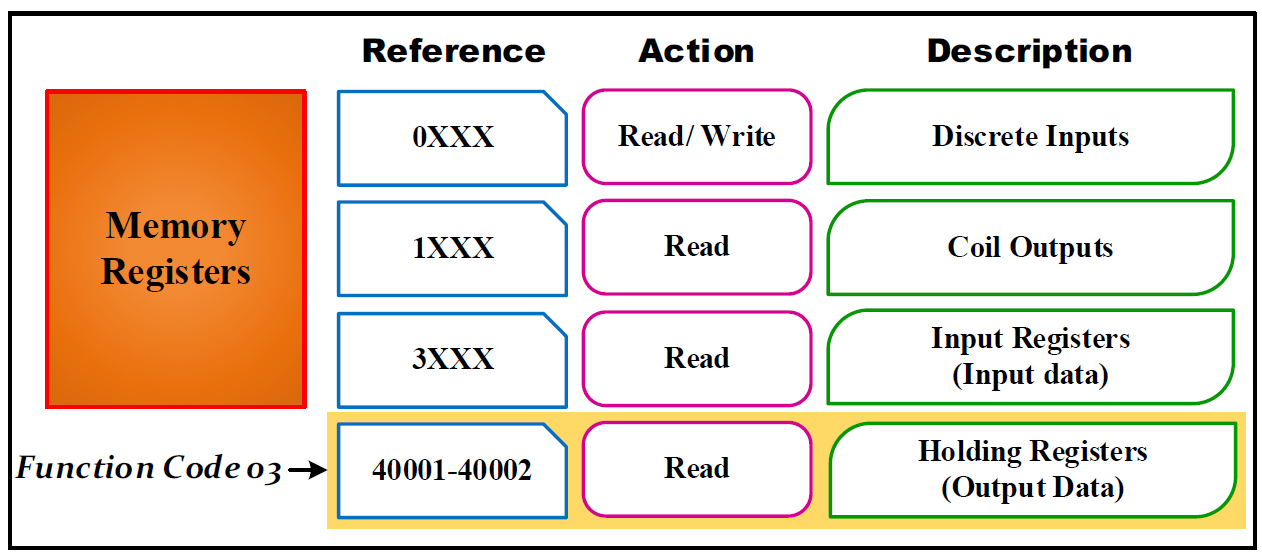
* The feature code of 03
* The beginning address of HI and LO bytes of address 0000
* The count number of addresses to be read from the slave

Register HI and LO bytes of count value 0002 specify the beginning register and number of registers to be read from the slave.

All the data types from the driving relays are named, such as:

* Coil: a single bit physical output
* Discrete input/communication: a single bit physical input

Message function code field: Contains one byte that informs the slave of what action to take. Valid function codes range from 1 – 255, but not all function codes will apply to a specific slave.

The following figure shows the subset of the regular functions of Modbus:

Master request data field: Gives the slave additional details for performing the operation as specified by the master’s request in the function code. Additional information in the master request data field includes:

* Address of the slave map register
* Number of registers that should be requested
* Written data from the master.

The usual response from the slave is a mere repeat of the master’s original function code, given that there are no issues or problems associated with fulfilling the master’s request. However, the error response of the slaves is a bit different. A simple error response involves returning a code equal to the original function code, except for the most important logic bit, which is set to 1. A custom code can be added which notifies the master of the nature of the error/problem that occurred, and this custom code can be added to the answer message data field.

**Implementing EV charging communication system**

**Charging communication system using SAEJ1939 CAN protocol**

Using SAEJ1939 protocol overlay with CAN, which includes a series of uniform messages and transfer rules extend to a wide variety of vehicles, is excellent from a data logging point of view. The SAEJ1939 protocol includes some other complex operations, such as:

* Emails
* Communication with a multi pack
* Mulitplexing

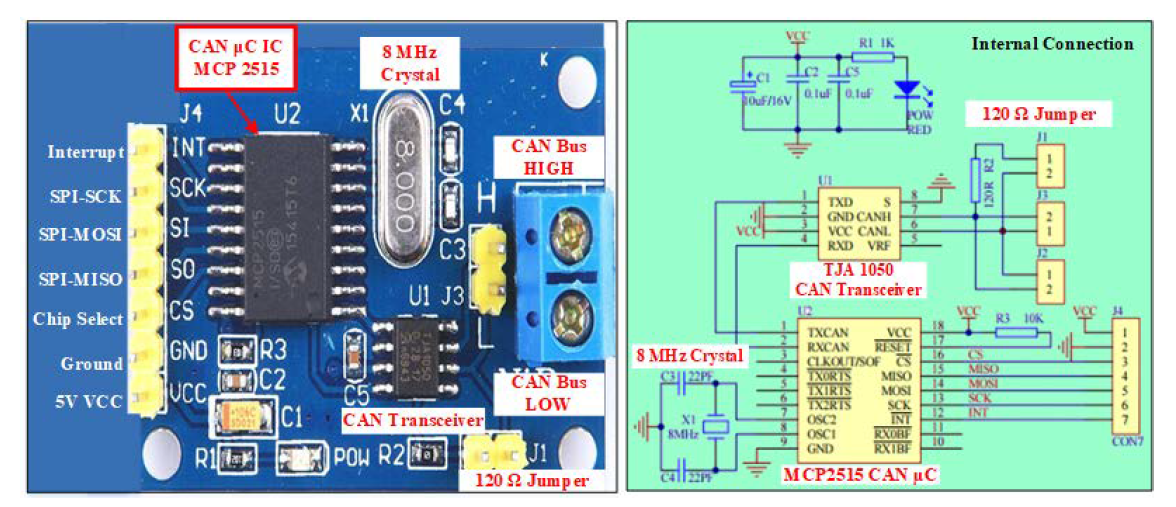
Standalone SAEJ1939 data loggers based on CAN bus enable data capture for over weeks or months of driving.

**Goal of the research paper: Integrate the Arduino controller and the SAEJ1939 protocol stack.**

The stack library for the ARD1939 protocol is available as a free download.

**CAN shield**

The CAN shield consists of 2 parts: the MCP2515 CAN controller IC and the TJA1050 transceiver.

* MCP2515 CAN controller IC
  + Standalone CAN controller
  + Built-in SPI interface for microcontroller communication
  + The key structure which consists of 3 components:
    - CAN board
      * Capable of sending and receiving messages via CAN bus
    - Control logic
      * Manages MCP2515 setup and service by interfacing all of its lines
    - SPI block
      * Responsible for client SPI communication
* TJA1050 transceiver IC
  + Interface between MCP2515 IC and the actual CAN bus

**Arduino and CAN communication**

Arduino Uno specifications:

* Atmel AVR (8-bit) microprocessor
* 32 kB memory chips
* 1 kB EEPROM
* 2 kB RAM

The CPU uses registers to temporarily store information.

Function registers are used to program and control the following:

* Timers
* Serial ports
* I/O ports
* Other hardware peripherals

Each port is controlled by 3 registers in Arduino:

1. The **DDR** register determines whether a pin is an output or an input.
2. The PIN register reads the status of the pins set to input using the PINMODE function
3. The port register determines whether the pin is set to high or low

**Baud rate: The number of bits per second that the UART can transmit or receive.**

CAN bus baud rate is up to 1 Mbps, but SAEJ1939 sets a 250 kbps limit. Both the charger and the BMS should operate in the same baud rate to ensure proper communication.

**PCAN-View software is used for viewing, transmitting and recording CAN data traffic.**

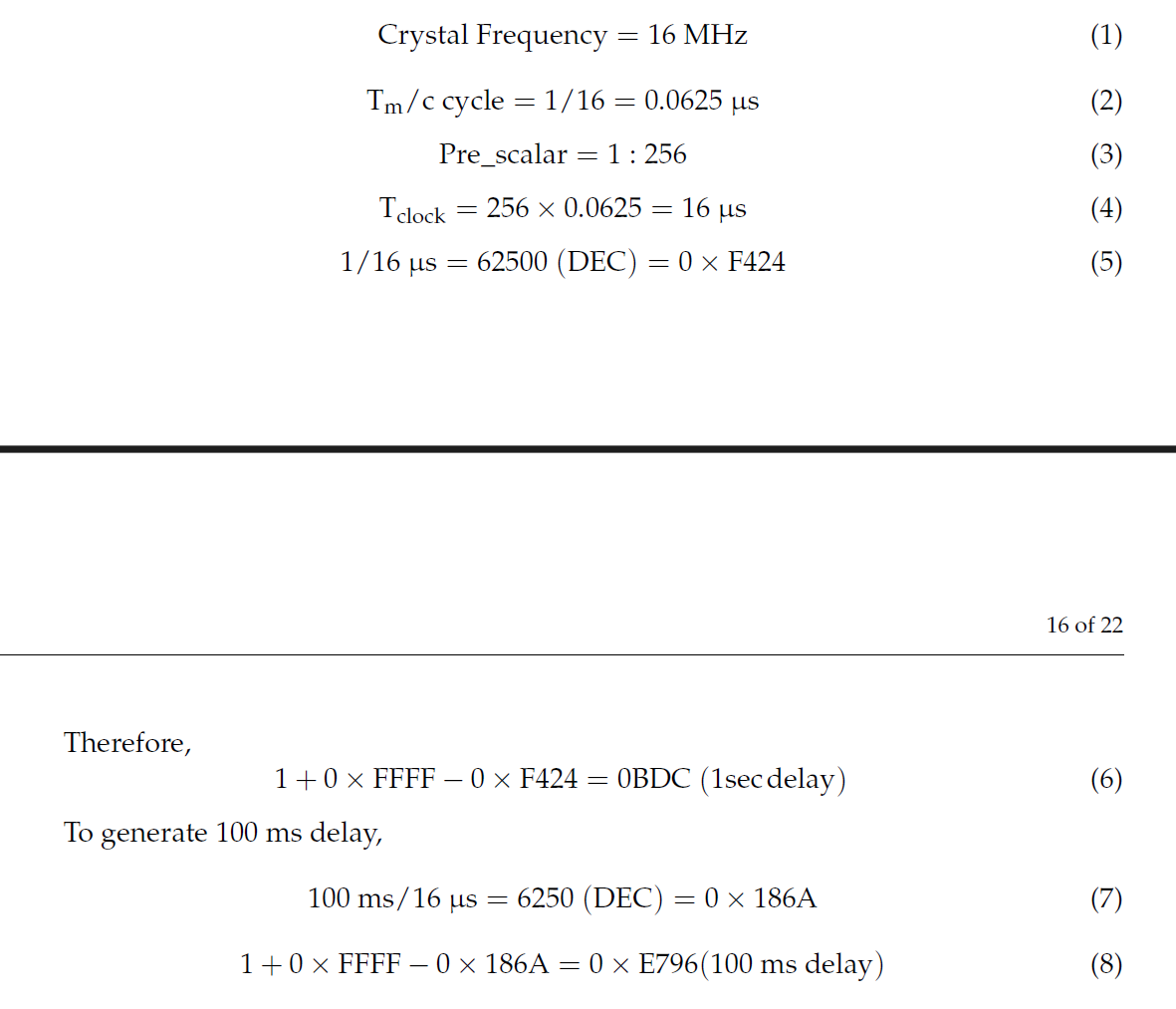
**Delay Generation Logic**

For effective communication, an appropriate time interval between two consecutive messages must be calculated and implemented.

The timer method is used to apply a delay in the program. There are **three** counter registers in Arduino Uno (Timer-0, Timer-1, Timer-2).

The **PCAN device** is used for tracking and decoding CAN data and for sending and receiving CAN letters.

The 16-bit Timer-1 is used, the calculations for the timer is given below:



**PCAN View Software**

Communication between the BMS and the battery charger is extracted and analysed by the PCAN View software. It is the software that is used for the tracking, transfer and storage of CAN data traffic.

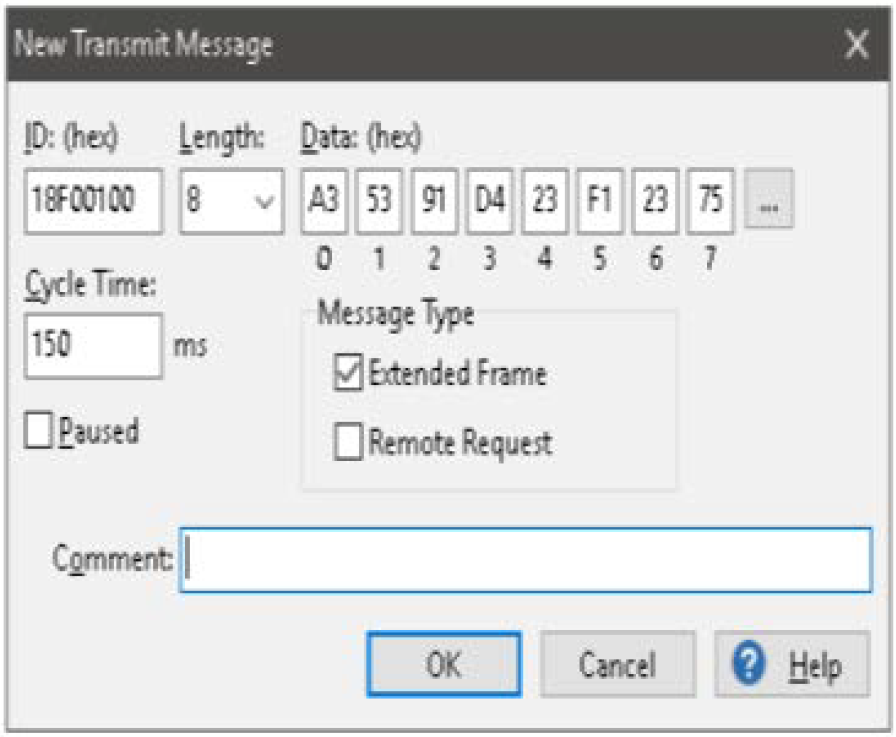
The messages are transmitted manually and regularly at a user-defined bit rate.

Device bus errors and memory overflows are reflected in the CAN hardware during operation. The data traffic can be registered and saved using the “trace” feature.

The **PCAN PC Hardware Device,** that comes with PCAN View, enables fast initialisation.

The **Link Dialogue** lists all available PEAK CAN interfaces. The user can access all device features, hardware-specific parameters, as well as information after choosing the bit rate.

Below is a screenshot of the PCAN View software: New Transmit Message Dialog Box

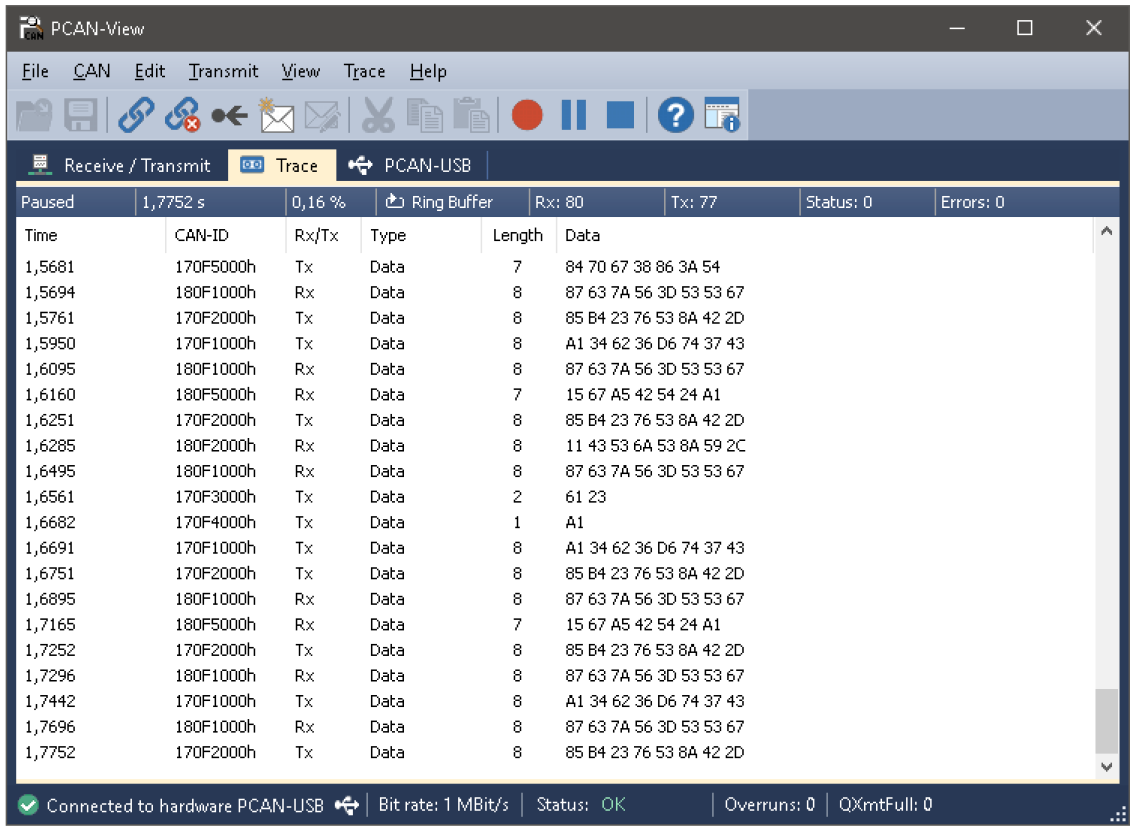


**Received CAN Message:** This message has a 29-bit CAN ID, and has a source address of 0x00 and a destination address of 0x20. The message is 8 bytes long,

**Transmitted CAN Message:** Also contains a 29-bit CAN ID, with a source address starting from 0x00 and a destination address of 0x10.

**NOTE:** The CAN standard contains the two lowest tiers in a seven layered OSI model – the physical and data link layer. As a result, it provides the ability to connect small packets on the CAN bus, but nothing more.

The screenshot of the PCAN View data logger shows some received and transmitted messages:



**Modbus Communication System with Arduino Microcontroller**

The EV charger converts the AC-DC and DC-DC converters to ensure supply of pure DC energy to the EV.

* AC-DC converter: Converts AC supply to a pulsating DC supply
* DC-DC converter: Converts pulsating DC to pure DC supply

Within EV charging, communication is required to supply controlled energy to drives and to avoid faults in the system. The **Modbus** protocol is used to establish communication between the power modules.

Commands such as:

* Voltage Demand
* Current Demand
* Power Demand

Can be sent to the power modules.

**RS485 Serial data transmission standard –** foundation of the Modbus RTU protocol.

**MAX485 module –** Used to interact with the Arduino UNO microcontroller.

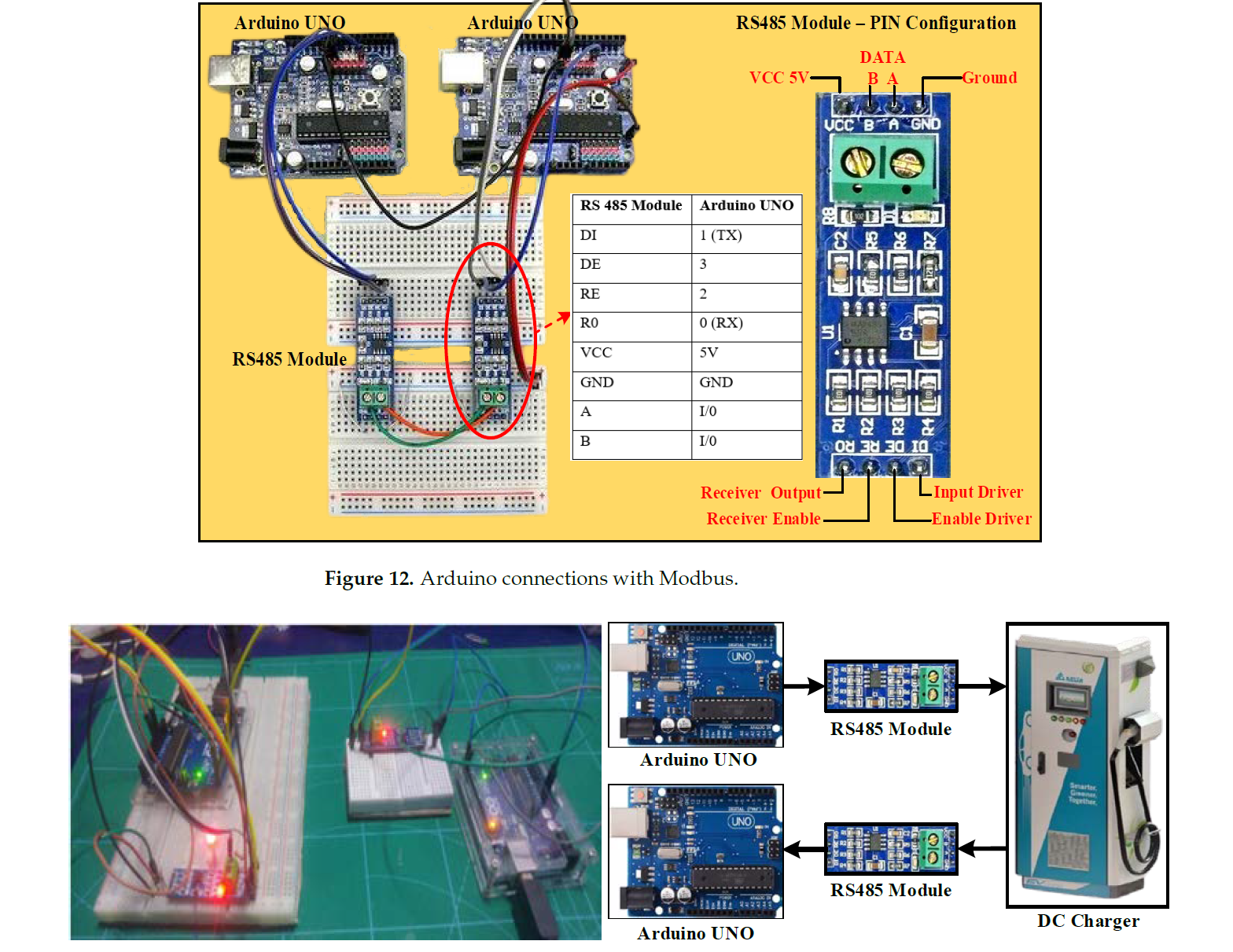
The MAX485 and RS485 modules transfers TTL signal to RS485 for long range, sequential communication. It is however sensitive to high data rate error. To overcome this, digital communication networks use the EIA-485 standard for sending messages over long distances and in environments with electrical noise. Furthermore, multiple receivers may be connected to such a network in a linear, multi-drop configuration.

The EV charger converters send and receive data from the Arduino using the RS485 network.

**Features of the RS485:**

* Industry standard data transfer protocol
* Allows transmission of data from up to 32 computers
* Max data rate of 10Mbps
* Cable length of up to 1.2 km
* Designed for (non-isolated) office and industrial applications
* Half-duplex communication

To make the system full-duplex, two Arduino UNO microcontrollers, with two MAX485 modules, are used, as shown below:



**Data Communication**

The RS485 and Arduino microcontrollers communicate with each other using HEX code. Data is converted into HEX code to send the voltage and current as per demands. The value “01” is used to turn the module ON, and the data for voltage and current are shown:

Voltage Demand:

500V x 10 = 5000 (DEC) = 1388 (HEX)

Current Demand:

10 A = 0A (HEX)

These values need to be written to holding registers. The function code “10” (HEX) is used to store the values accordingly.

At first, the charging request is sent to the charger communication system to write the data. Upon receipt of the request message, the data are read from the charger.