**DESIGN AND IMPLEMENTATION OF A TELEMETRY AND DISPLAY SYSTEM FOR MOTORSPORT APPLICATIONS USING CONTROLLER AREA NETWORK COMMUNICATION**

*An Undergraduate Project Report submitted to Manipal University*

*Submitted by*

**ADITYA VIKRAM JINDAL VARUN SACHDEVA**

(Reg. No 180907586) (Reg. No 18932522)

BACHELOR OF TECHNOLOGY

In

ELECTRONICS AND COMMUNICATION ENGINEERING

And

ELECTRONICS AND INSTRUMENTATION ENGINEERING

**MANIPAL INSTITUTE OF TECHNOLOGY**

**INTRODUCTION**

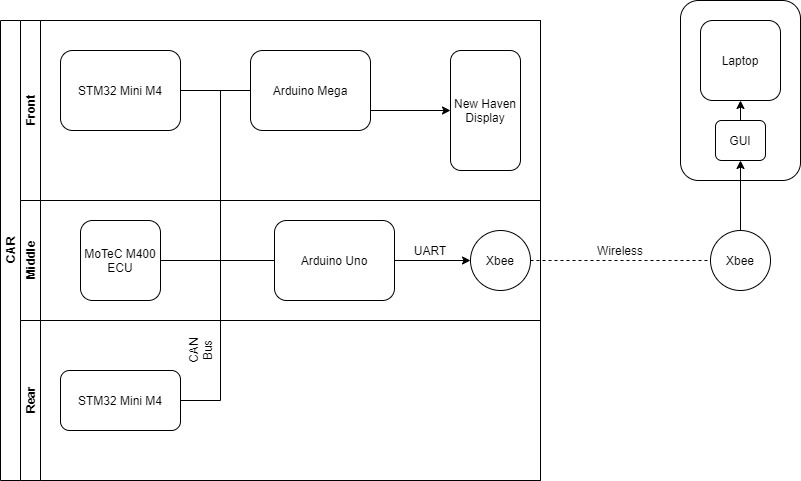
This project report describes the design and working of a Motorsport Data Acquisition System, Live Telemetry System and Display system developed using the Controller Area Network (CAN) communication protocol as the backbone of the system. A Formula One car hosts over a hundred sensors during each of its races. The Data Acquisition System, although does not directly affect the car’s performance, is an indispensable system when it comes to improving and testing designs. Designers can validate their assumptions and calculations, real-time data during testing can be a safety indicator and it provides insight to the driver about the performance of the vehicle.

**OBJECTIVES**

The proposed work is to design and implement a Data Acquisition and Telemetry System to be on-board a race car. The major objectives include

* Establish a modular Controller Area Network bus which runs throughout the vehicle.
* Interfacing multiple nodes to the said CAN bus, each with their own functionalities.
* Transmitting the data of multiple sensors onboard the vehicle.
* Display valid data to the driver while operating the car.
* Wireless data transmission back to host across a long range.

**SYSTEM OUTLINE**



**FIGURE:** System Structure

**THEORY**

This part concentrates on the literature survey involved in designing the system.

**THE NEED FOR DATA ACQUISITON AND TELEMETRY IN MOTORSPORTS**

A design is only as good as the performance of the part in field and in tests, but to judge whether a part/system is behaving like it is supposed to Data from different sensors monitoring different physical parameters are required to validate the design. A Data Acquisition and Telemetry System helps analyse the vehicles behaviour during different tests and set ups.

Simulations can predict a certain parts behaviour but to ensure proper results and safety such a system is a must.

A Telemetry System also helps you analyse your driver’s performance. Looking at real-time data while your driver is on the track can help give invaluable feedback to improve performance.

Lastly, Motorsport is a continuous improvement process. Logged data helps designers avoid the mistakes their predecessors committed or take useful information out of their successes.

**WHY CONTROLLER AREA NETWORK?**

Controller Area Network is a multi-master, message broadcast system that allows a maximum signalling rate of 1 megabit per second(bps). In a CAN network, many short messages like temperature or RPM are broadcast to the entire network, which provides for data consistency in every node of the system. The protocol is a motorsport and commercial industry standard due to its modular and 2-wire physical bus which has led to the replacement of complex wiring harnesses.

Additionally, Controller Area Network sends specialized standard CAN Message formats which allow up to 8 bytes of data per message each with their own 11-bit identifier which decide the message arbitration and prevent message collision.

**DESIGN CONSIDERATIONS**

The main factors that were taken into consideration before picking any component

1. Ability to withstand the harsh motorsport environment
2. Ease of programming and interface
3. CAN compatibility and other communication protocol compatibility.
4. Ability to self-diagnose and repair data errors and physical layer errors.

**SYSTEM DESCRIPTION**

This part elaborates the various components, sensors and processes in the system.

**COMPONENTS**

1. **MoTeC M400 ECU**

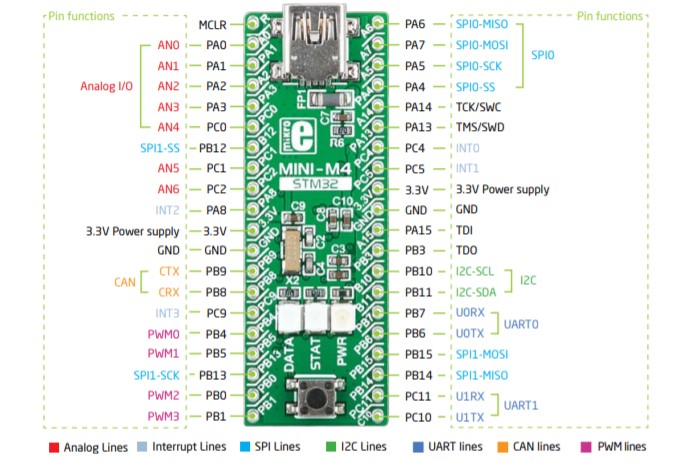
The MotecM400 engine control unit is a part of the Hundred Series of the MoTeC ECU's. It is specifically designed for a four-cylinder engine. It provides multiple features for engine tuning and management and supports CAN protocol. It is connected to various Engine sensors in the car and transmits their data to the CAN bus. A market bought ECU was chosen over a custom made one due to its ease of programmability and driver aids such as launch control and traction control.



**FIGURE:** MoTeC m400 ECU

1. **STM32 Mini M4**

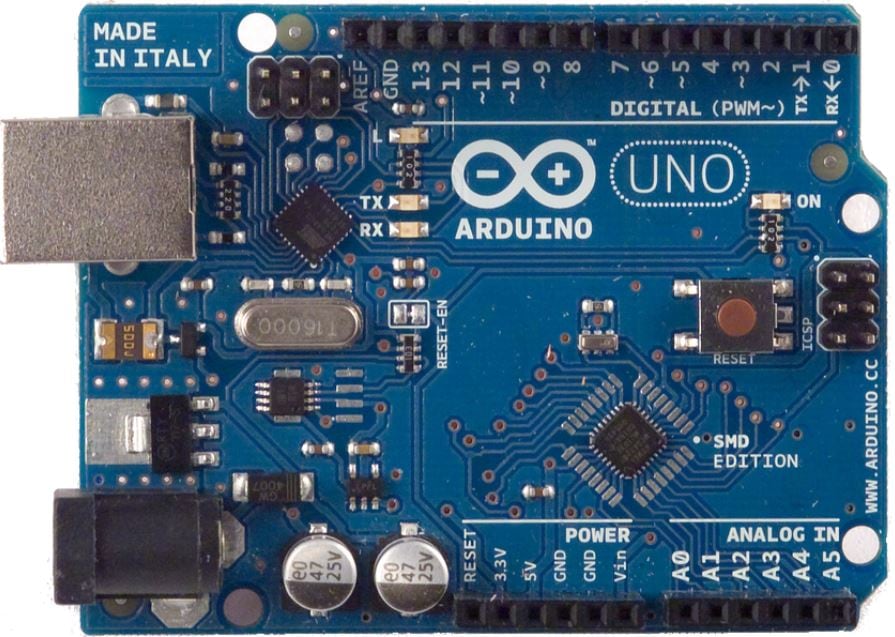
The STM32 Mini M4 is a development Board fitted with STM32f415RG microcontroller powered by ARM cortex. The Mini M4 was chosen because the board comes fitted with 15 Analog Input Pins, an inbuilt CAN Controller and has a very compact form factor. The system Utilises 2 such boards, one positioned in the front and one in the rear of the vehicle, to collect data from all the non-engine sensors (ECU sensors), format them into appropriate CAN messages and transmit them onto the CAN bus.



**FIGURE:** STM32 Mini M4 board

1. **Arduino UNO**

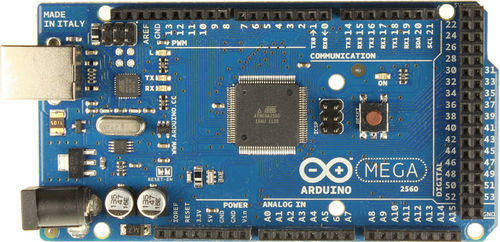
Arduino UNO is a microcontroller board based on the Atmega328P microcontroller. The board has 14 digital I/O pins and 6 Analog input pins. A can controller (MCP2515) is connected to this board via SPI which enables it to communicate through the CAN bus. Its role in the system is to collect data from the CAN bus and transfer it to the XBEE module via UART. It was chosen due to its small form factor, ease of programmability and vast amount of open source libraries.



**FIGURE:** Arduino UNO board

1. **Arduino MEGA**

Arduino MEGA is a microcontroller board based on the Atmega2560 microcontroller. The board has 54 digital I/O pins and 16 Analog input pins. A can controller (MCP2515) is connected to this board via SPI which enables it to communicate through the CAN bus. Its role in the system is to control the display. It receives data from the CAN bus and transfers it to the display using an 8 bit data bus. The Arduino MEGA was selected to control the display due to the availability of open source libraries to code the display and information on how to connect the display to the board.



**FIGURE:** Arduino MEGA board

1. **Xbee Pro 900 Hp**

The Xbee modules are used to transfer data wirelessly from the car to the laptop for live telemetry. They have a frequency band of 902 to 928 MHz and RF data transfer rate of 10Kbps for up to 610m indoor or 15.5km outdoor and 200Kbps for up to 305m indoor and 6.5km outdoor. The Xbee Pro 900 Hp was chosen because of ease of configuration through its software XCTU, long range and small form factor.



**FIGURE:** Xbee pro 900 Hp

1. **Newhaven NHD-240128WG-BTFH-VZ Display**

This display is a 240 x 128 LCD module which is controlled by and receives data from the Arduino MEGA. It was chosen as it has less glare which allows good visibility in daylight.



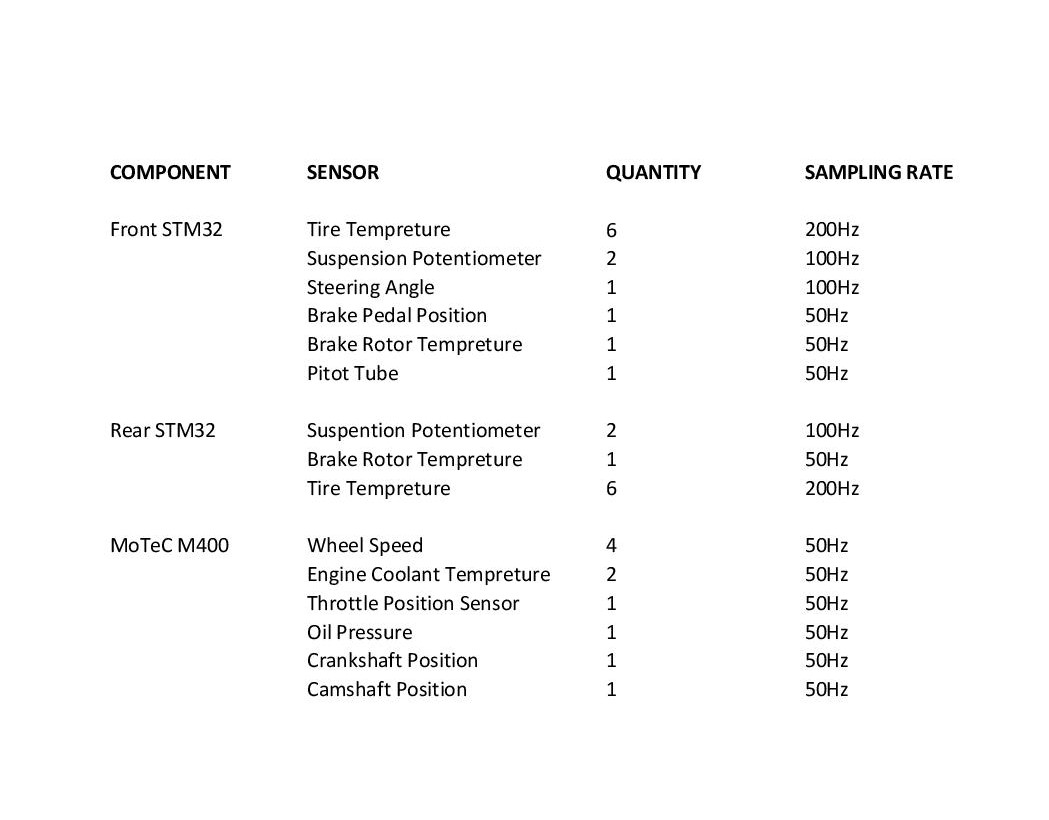
**FIGURE:** Newhaven NHD-240128WG-BTFH-VZ Display

Front and back

**SENSORS**

Motorsport/Automotive grade sensors were picked for the system. Temperature considerations, ruggedness, accuracy, and ease of interfacing with the different components helped us narrow down products. Ratio metric voltage output products were preferred.

As stated before, formula one cars have over a hundred sensors during each race, but due to design and monetary restrictions our Telemetry System consists of the following sensors which are crucial for the design and performance upgradation.

****

1. Tire Temperature sensor- Three tire temperature sensors are mounted on a PCB which is attached at certain height above the wheel. These are thermopile IR sensors which provide contact less temperature measurement. The sensors are mounted such that they measure they temperature of the temperature of one third part of the tire namely the middle, inner and outer. This data is used to validate the camber of the tire. The more the temperature of a part means that it is scrubbing more with the surface.
2. Suspension Potentiometer- Linear Potentiometers with a stroke length of 100mm. Data used to validate the vehicle model, derive wheel travel and loads acting on it.
3. Steering Angle- Angular Potentiometer. Data used to validate steering design.
4. Brake Pedal Position Sensor- Potentiometer based sensor. The data is used for safety circuits.
5. Brake Rotor Temperature sensor- Infrared Temperature Sensor from Texense. Used for thermal brake temperatures.
6. Wheel Speed sensor- Hall effect-based sensors used to calculate ground speed, used during launch control and traction control.
7. Engine Coolant Temperature sensor- Negative temperature co-efficient thermistors, one placed upstream and one downstream in the cooling path, helps in monitoring safe engine temperature and design of new radiators.

1. Throttle Position Sensor-This is a potentiometer-based sensor which is used to measure the amount of throttle pressed and this data is used by the ECU to determine the air intake in the engine so that it can adjust the amount of fuel injected to maintain the air-fuel ratio.
2. Oil Pressure sensor- Used to measure the pressure of the lubricant to engine to ensure safe running of the engine.
3. Crankshaft Position Sensor- Electromagnetic Sensor which calculates position of pistons relative to the Crankshafts angular position, helps in adjusting ignition timing.
4. Camshaft Position Sensor-Electromagnetic Sensor which calculates position of inlets relative to the Camshafts angular position, helps in adjusting ignition timing.

**DATA ACQUIRING**

The process on Data Acquiring is divided between the ECU and the Mini M4 microcontrollers, each of them host ADC channels with number of analog input pins and since most of the sensors used are analog sensors, any addition of sensors is as easy as configuring one more new microcontrollers onto the CAN bus with the new sensors attached to it.

Different sensors demand different sampling rates. To address this each microcontroller is programmed to read and transmit and specific time intervals, and to ensure data consistency a message order is followed.

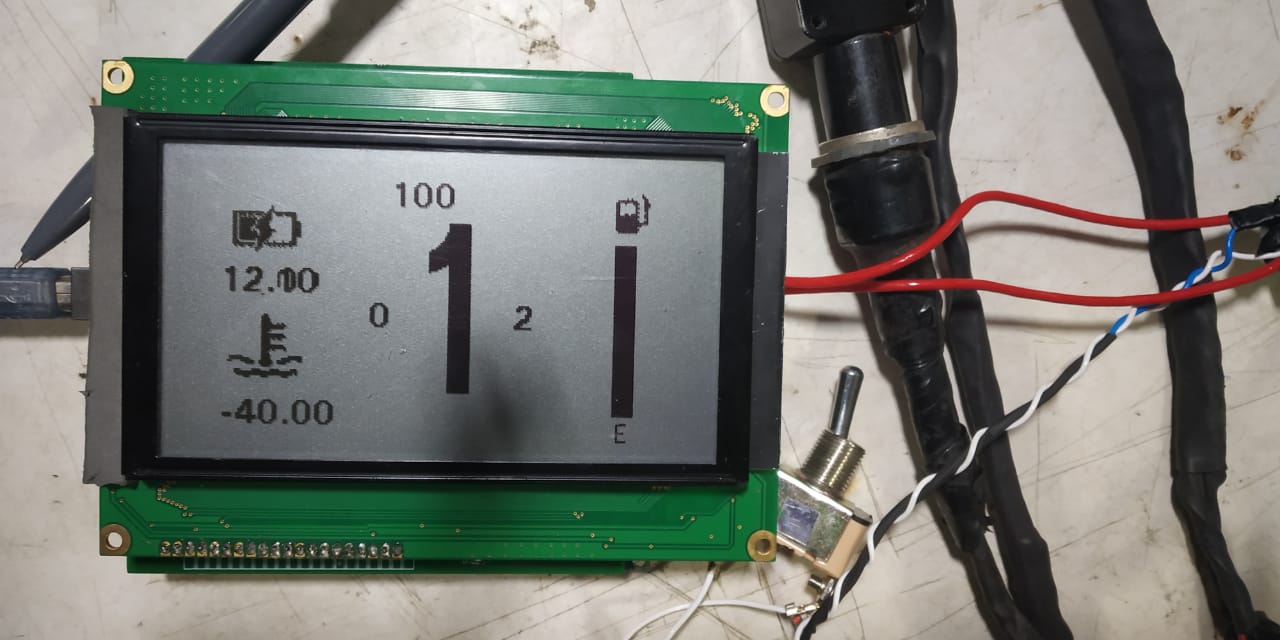
Appropriate filters were designed and placed to eliminate any sort of high frequency hardware noise.

**WIRELESS DATA TRANSMISSION**

Seeing live data while the car is running ensures safety of both the car and the driver and also enables the team members to see how the components are performing in real time. The designed telemetry system uses a pair of Xbee Pro 900 Hp modules to establish communication between the car and a laptop. The Xbee module placed on the car receives data from the Arduino UNO through UART. It then broadcasts this data to the other the Xbee connected to the laptop. A software was designed using Matlab App Designer to display various data received in an organized way.

**DISPLAY**

A Newhaven NHD-240128WG-BTFH-VZ 240 x 128 LCD display is placed on the dash of the car. It is controlled by the Arduino MEGA board which receives data through the CAN bus and transfers it to the display. The display shows information like gear position, rpm which are critical for the driver for optimum performance on track. Information like oil pressure, fuel level tells the drivers the current state of the car so that they can adjust their driving accordingly. The data transfer between the board and the display takes place through an 8-bit parallel data transfer.



**FIGURE:** Working display showing relevant data

**GUI**

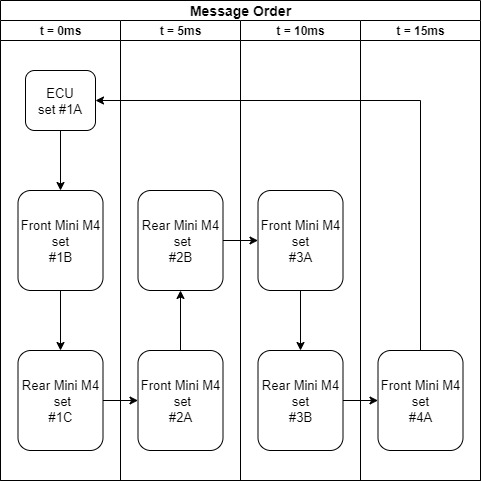
A graphical user interface was designed to using Matlab App Designer to represent the data received from the Xbee module as clearly and possible. Data such as engine coolant temperature and oil pressure so that the team can ensure that all systems are running nominally and there is no issue with the car. Throttle position and RPM is also displayed along with fuel level and battery voltage. The throttle position along with the RPM is used to check the responsiveness of the engine to driver input.

**MESSAGE ORDER AND FORMATING**

As Controller Area Network only allows short messages to be broadcast at a time, to ensure data consistency a specific message order was designed and each message was formatted to deliver the exact data required, this was done with consideration of the specific sampling rate of each sensor and with the ease of logging in mind.

The message order is described in the figure below.

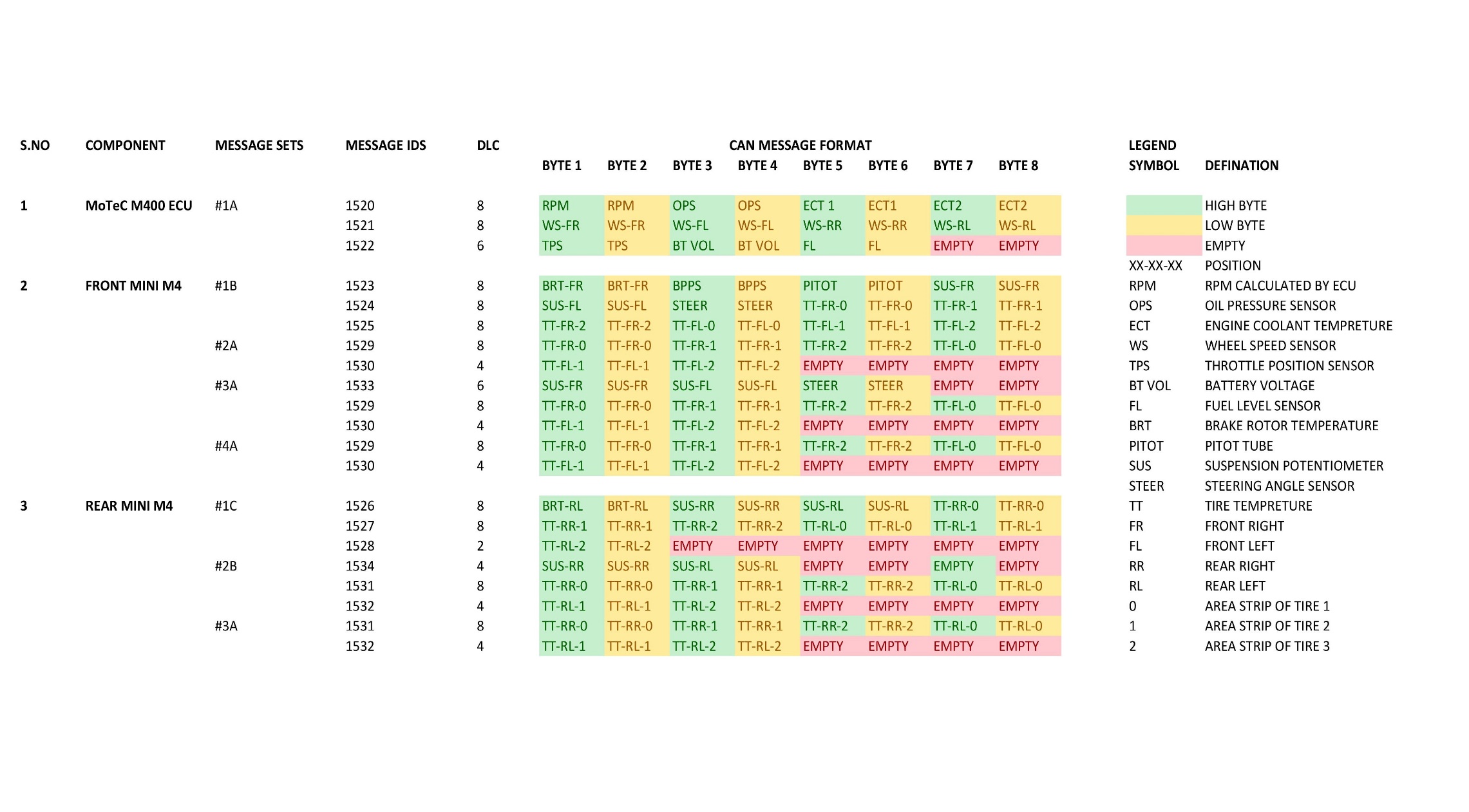
Each message cycle is initiated by the ECU and follows the specified order as illustrated below, each message set is comprised of a group of messages, until all messages have been sent in a set the next set is not sent, this is done to ensure data consistency while logging and further transmitting.



**FIGURE:** Message Sending Order

Each message in a set is formatted individually with data from specific sensors, this done independently by Each Node in the message cycle. These nodes first read from the sensors via the ADC channels then each sensor data is divided into 2 bytes of data, the high byte and the low byte, then these bytes are formatted into their respective CAN messages.

The messages are formatted in the order illustrated below.

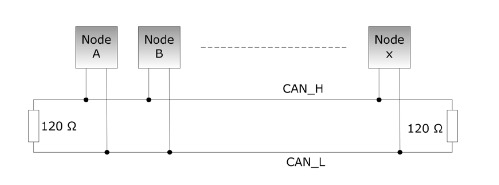


**CAN BUS SPECIFICATIONS**

The High-Speed ISO 11898:2003 Standard CAN was chosen as it specifies for a maximum signalling rate of 125Kbps to 1 Mbps with a bus length of 40 m with a maximum recommendation of 30 nodes. This standard provides an 11-bit identifier which allows for a max number of 2048 of unique message identifiers.

To facilitate the requirement of fast and uniform data transfer a bus type network topology was chosen, furthermore with a bus topology it becomes easier to attach and remove nodes from the network.

Two 120ohm resistors were chosen with relation to ISO 11898:2003 standard as bus terminators this was done as to match the characteristic impedance of the line to prevent signal reflections.

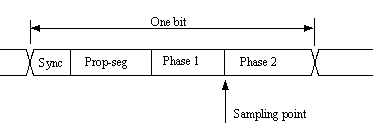


**FIGURE:** Standard CAN bus topology

**CAN BIT TIMING**

CAN Bit Timing is defined as the nominal bit rate of a CAN message. Each time taken for a single bit in a can message can be divided into four separate non-overlapping time segments called SYNC\_SEG, PROP\_SEG, PHASE\_SEG1 and PHASE\_SEG2. These are illustrated in the Figure below.

Each segment holds a purpose in the grand scheme of the CAN bus, the SYNC\_SEG is set to allow all clocks of the nodes to synchronise, PROP\_SEG is the propagation delay segment it exists to delay the earliest possible sample of the bit by a node until the transmitted bit values from all the transmitting nodes have reached all of the nodes, the PHASE\_SEG1 and PHASE\_SEG2 segments are there to ensure the clocks stay in sync and between them the sample point is taken.

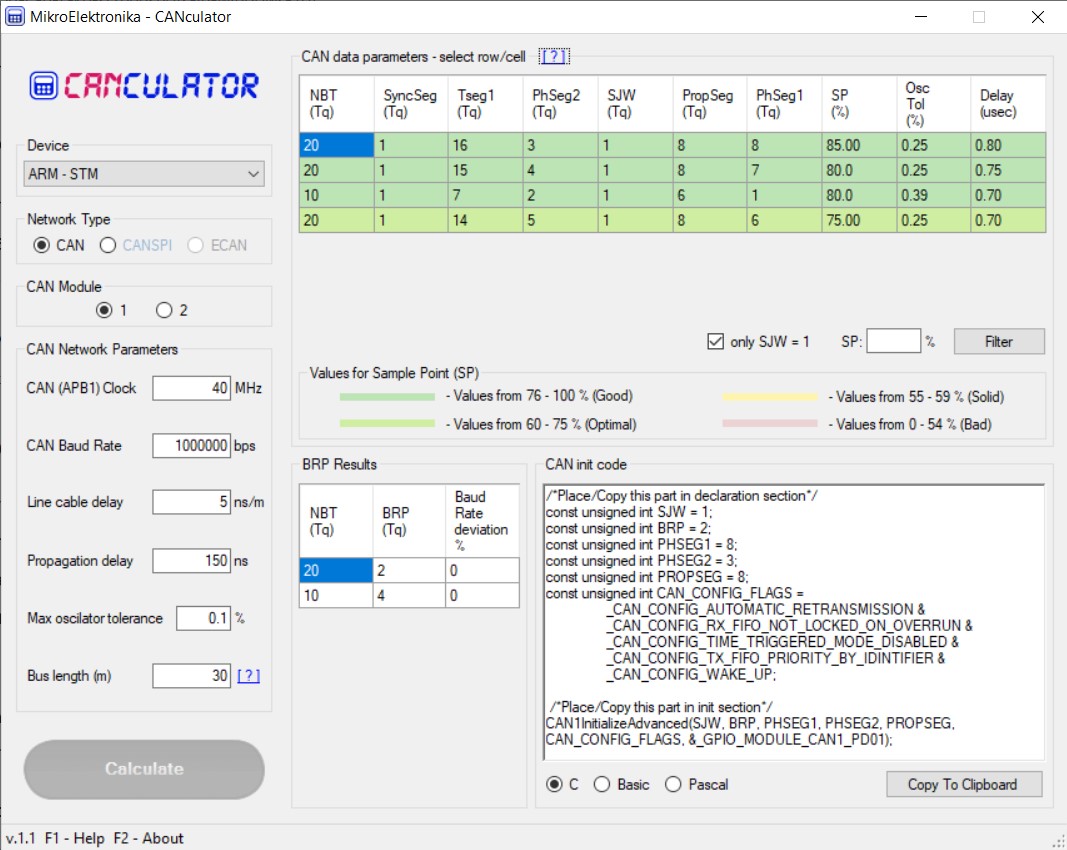


**FIGURE:** CAN bit segments

Each of these segments is an integer multiple of a unit of time called a Time Quantum, tQ. The duration of a Time Quantum is equal to the period of the CAN system clock, which is derived from the microcontroller (MCU) system clock or oscillator by way of a programmable prescaler, called the Baud Rate Prescaler.

Too set the nature of CAN messages, each of these segments were calculated using open source CAN Calculator, see figure below, with regards to the required baud rate, bus length and propagation delay.

This process was done only for the STM32 microcontrollers and could not be done for the MoTeC m400 ecu as it does not allow this customization.



**FIGURE:** CAN calculator