

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
AUTOMOTIVE ENGINEERING TECHNOLOGY



HCMUTE

PROJECT

TOPIC: Study about Controller Area Network protocol in automotive system

STUDENT AND STUDENT'S ID:

Đoàn Phong Đạt 20145408

Đỗ Trần Anh Duy 20145406

Phạm Quốc Anh 20145029

SUBJECT: SPECIAL PROJECT

ADVISOR: Vũ Đình Huấn

HO CHI MINH CITY, APRIL 2023

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY AND EDUCATION
AUTOMOTIVE ENGINEERING TECHNOLOGY



HCMUTE

PROJECT

TOPIC: Study about Controller Area Network protocol in automotive system

STUDENT AND STUDENT'S ID:

Đoàn Phong Đạt 20145408

Đỗ Trần Anh Duy 20145406

Phạm Quốc Anh 20145029

SUBJECT: SPECIAL PROJECT

ADVISOR: Vũ Đình Huân

HO CHI MINH CITY, APRIL 2023



THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, April 4, 2023

GRADUATION PROJECT ASSIGNMENT

Student name: Đoàn Phong Đạt

Student ID: 20145408

Student name: Đỗ Trần Anh Duy

Student ID: 20145406

Student name: Phạm Quốc Anh

Student ID: 20145029

Major: Special Project

Class:20145CLA

Advisor: Vũ Đình Huân

Email:

Date of assignment:

Date of submission:

1. Project title: Study about Controller Area Network protocol in automotive system

2. Initial materials provided by the advisor: Vũ Đình Huân

3. Content of the project: _____

4. Final product: _____

CHAIR OF THE PROGRAM

(Sign with full name)

ADVISOR

(Sign with full name)



THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, March 27, 2023

ADVISOR'S EVALUATION SHEET

Student name: Đoàn Phong Đạt Student ID: 20145408

Student name: Đỗ Trần Anh Duy Student ID: 20145406

Student name: Phạm Quốc Anh Student ID: 20145029

Major: Special Project

Project title: Study about Controller Area Network protocol in automotive system

Advisor: Vũ Đình Huân

EVALUATION

1. Content of the project:

.....

.....

.....

2. Strengths:

.....

.....

.....

3. Weaknesses:

.....

.....

.....

4. Approval for oral defense? (*Approved or denied*)

.....

5. Overall evaluation: (Excellent, Good, Fair, Poor)

.....

6. Mark:.....(*in words:.....*)

Ho Chi Minh City, April 4, 2023

ADVISOR

(Sign with full name)



THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom– Happiness

Ho Chi Minh City, April 4, 2023

PRE-DEFENSE EVALUATION SHEET

Student name: Đoàn Phong Đạt Student ID: 20145408

Student name: Đỗ Trần Anh Duy Student ID: 20145406

Student name: Phạm Quốc Anh Student ID: 20145029

Major: Special Project

Project title: Study about Controller Area Network protocol in automotive system

Name of Reviewer:

EVALUATION

1. Content and workload of the project

.....
.....
.....
.....

2. Strengths:

.....
.....
.....

3. Weaknesses:

.....
.....
.....

4. Approval for oral defense? (*Approved or denied*)

.....

5. Overall evaluation: (*Excellent, Good, Fair, Poor*)

.....

6. Mark:.....(*in words:.....*)

Ho Chi Minh City, April 4, 2023

REVIEWER

(Sign with full name)



THE SOCIALIST REPUBLIC OF VIETNAM
Independence – Freedom – Happiness

EVALUATION SHEET OF DEFENSE COMMITTEE MEMBER

Student name: Đoàn Phong Đạt Student ID: 20145408

Student name: Đỗ Trần Anh Duy Student ID: 20145406

Student name: Phạm Quốc Anh Student ID: 20145029

Major: Special Project

Project title: Study about Controller Area Network protocol in automotive system

Name of Defense Committee Member:

.....

EVALUATION

1. Content and workload of the project

.....
.....
.....
.....

2. Strengths:

.....
.....
.....

3. Weaknesses:

.....
.....
.....

4. Overall evaluation: (*Excellent, Good, Fair, Poor*)

.....

5. Mark:.....(*in words:.....*)

Ho Chi Minh City, April 4, 2023

COMMITTEE MEMBER

(Sign with full name)

ACKNOWLEDGEMENTS

First of all, we would like to express our honest and deep gratitude to the college together. All instructors of Ho Chi Minh City University of Technology and Education in common and the instructors in the Faculty of High Quality Training and Faculty of Automotive Engineering Technology in unique gave us a surrounding to find out about well. Teachers have enthusiastically taught, conveyed, and taught us expertise as nicely as precious experiences all through their time at the school.

We are very honored and proud to be college students of Automotive Engineering Technology, Faculty of High Quality Training and Mechanical Engineering. Aerodynamics University of Technical Education Ho Chi Minh City. In the learning process, we have matured, multiplied ourselves more, received extra understanding and journey on the getting-to-know course and organized for the future profession path.

In particular, we would like to thank Mr. Vu Dinh Huan, the one who directly instructs us to implement this topic. During the direction of the project, the trainer was once usually fascinated with monitoring the growth of the assignment and giving guidelines and fundamental training thru growth reviews to make well-timed and fabulous corrections to enhance the consequences of the subject.

We would additionally like to take this chance to thank my friends for their unwavering aid and encouragement at some point during the mission and difficult moments. Your trust in me and our skills capability the world to us, and we should no longer have finished this task barring your support.

Despite excellent efforts, difficulties with substances, and our own confined ability, the thesis can't keep away from shortcomings. Therefore, we seem to be ahead in receiving the education and recommendations of instructors so that we can supplement, enhance their knowledge, and higher serve our future work.

TABLE OF CONTENT

CHAPTER 1: OVERVIEW	1
1.1 Introduction	1
1.2 Research objectives	1
1.3 Research scope	1
CHAPTER 2: THEORETICAL CONTENTS ABOUT CONTROLLER AREA NETWORK.....	2
2.1 History of CAN technology.....	2
2.2 Benefits of CAN in automotive industry	3
2.3 Basic knowledge in vehicle network:	4
2.3.1 What is vehicle network?.....	4
2.3.2 Modules and nodes	4
2.3.3 Types of communication	5
2.3.4 Data transmission flow.....	5
2.4 Study about Open Systems Interconnection Reference Model (OSI Model) ..	7
2.4.1 Open Systems Interconnection Reference Model (OSI Model)	7
2.4.2 Layers in OSI model	8
2.5 Message format of CAN protocol	14
2.6 Principle of self-test of CAN network by Bit Stuffing	15
2.7 Using Pair Twisted in CAN	18
2.8 CAN Network Management	18
2.9 Terminating Resistor on CAN	19
2.10 Study CAN network on Toyota Camry 2007.....	20
CHAPTER 3: Study about the CAN bus on Ford vehicle.....	21
3.1 Study about the Ford Focus 3M5F-10841-B.....	21
3.2 Ford Controller Area Network	29
3.3 Implement “Interfacing MCP2515 CAN Module with Arduino”	31
3.4 Instrument cluster wiring diagram	39
3.5 HS CAN ID and MS CAN ID	43

Conclusion..... 51

Reference..... 51

LIST OF FIGURES AND TABLES

Figure 1.1: Expected Illustration for model

Figure 2.1: History of CAN technology

Figure 2.2: Structure of data transmission flow

Figure 2.3: Communication between 2 CAN nodes

Figure 2.4: OSI model

Figure 2.5: Layers in OSI model

Figure 2.6: Data bits in the physical layer

Figure 2.7: Data Link Layer

Figure 2.8: Network Layer

Figure 2.9: Transport Layer

Figure 2.10: Session Layer

Figure 2.11: Presentation Layer

Figure 2.12: Application Layer

Figure 2.13: Standard frame

Figure 2.14: Extended frame

Figure 2.15: Parity bit

Figure 2.16: Cyclic Redundancy Check

Figure 2.17: Bit stuffing

Figure 2.18: Pair Twisted in CAN

Figure 2.19: Terminal Resistor (120 Ohm, DB9, CAN Bus)

Figure 2.20: Terminal Resistor in CAN communication on cars

Figure 2.21: CAN network on Toyota Camry 2007

Figure 3.1: Ford Focus 3M5F-10841-B instrument cluster

Figure 3.2: Main printed circuit board of Ford Focus 3M5F-10841-B instrument cluster

Figure 3.3: Ford Focus 3M5F-10841-B instrument cluster

Figure 3.4: Structure of SONCEBOZ 6405R421 Stepper Motor Instrument Cluster

Figure 3.5: ELM327 OBD2

Figure 3.6: Circuit diagram for three CAN nodes

Figure 3.7: Instrument cluster wiring diagram

Figure 3.8: Symbols on instrument cluster

Figure 3.9: Power supply and connecting between dashboard and module CAN

Figure 3.10: Active instrument cluster

Figure 3.11: The result on serial monitor

ABBREVIATION

CAN: Controller Area Network

SOF: Start of frame field

CRC: Cyclic Redundancy Check

ACK: Acknowledge

EOF: End of frame field

OSI: Open Systems Interconnection

HS CAN: High Speed Controller Area Networ

MS CAN: Motorola Scalable Controller Area Network

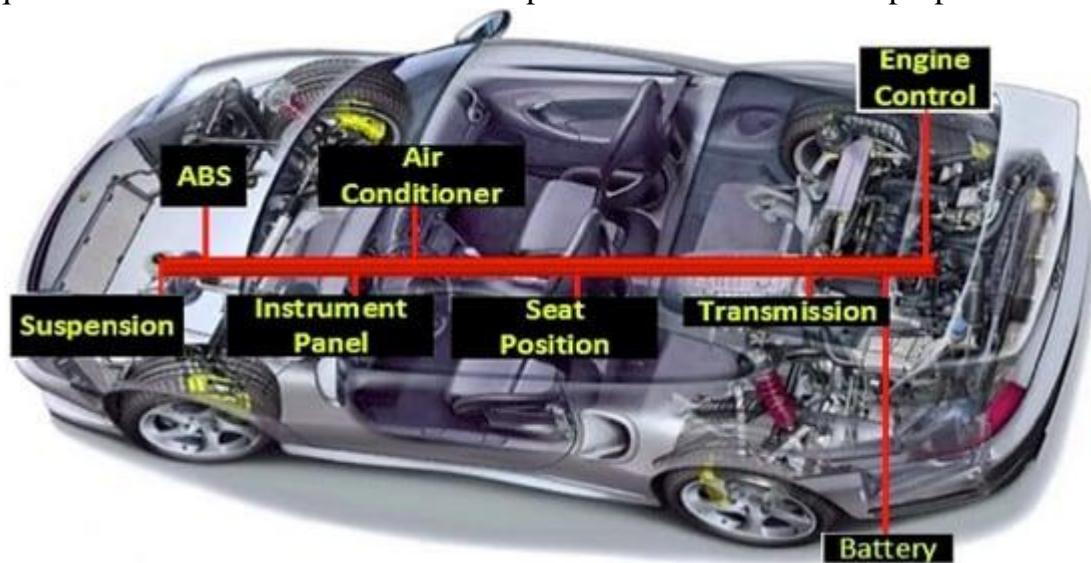
ECU: Electronic Control Unit

CHAPTER 1: OVERVIEW

1.1 Introduction

Controller Area Network (CAN) was designed by Bosch in the mid – 1980s for multiplexing communication between Electronic Control Units (ECUs) in vehicles and therefore for decreasing the overall wire harness which means ECUs communicate via a single CAN system instead of complex analog signal lines – reducing errors, weight, wiring.

In the current trend of technology in the automotive industry, the increasing use of ECUs in vehicles to implement signal recognition, calculation, and sending of appropriate signals to determine the vehicle's working functions most effectively in different situations (Example: Anti-lock Braking System (ABS), Direct Ignition System). In the modern vehicles nowadays may have up to 70 ECUs. Therefore a special communication is required to connect the ECUs and CAN protocol serves the above purpose.



1.2 Research objectives

In this paper, we would like to have a deep knowledge about the Controller Area Network by understanding the importance and role of CAN in vehicles. In addition to this, to have knowledge of theory of CAN that includes structure and working principle. Concurrent we will implement an experimental of CAN by using MCP 2515 CAN module and Arduino board (Use Arduino Integrated Development Environment) and we also study how CAN applied on vehicles such as: Toyota, Ford thereby draw an overview of the role of CAN.

1.3 Research scope

In this paper, we would focus on studying the theory concept of CAN, and besides that, we would also study the CAN on the specific vehicle brand that is FORD at the same time we have studied the Ford Focus 3M5F-10841-B instrument cluster which we have received from the advisor.

More specifically, we will present the methods of studying about CAN:

About the theoretical content about CAN:

- Study about the historical context of CAN and benefits of CAN in automotive industry
- Study about the Basic knowledge in vehicle network
- Study about Open Systems Interconnection Reference Model
- Study CAN network on Toyota Camry 2007
- Beside that, also study several relative information about CAN

About the experiment:

- Expectant components: Arduino Nano Board, Arduino Uno Board, CAN module MCP2515, LCD 16x2 and I2C LCD, DHT11 sensor, Jumper Wires, Breadboard
- Study specifications of using MCP2515 CAN module and Arduino UNO R3 and other devices in above components.
- Study Circuit diagram
- Study about the programming code: CAN transmitter and receiver code
- Estimate the model and results after running the program



Figure 1.1: Expected Illustration for model

About the instrument cluster:

- Study about some component on circuit board
- Study about instrument cluster wiring diagram

CHAPTER 2: THEORETICAL CONTENTS ABOUT CONTROLLER AREA NETWORK

2.1 History of CAN technology

Control Area Network (CAN bus) has a rich history and went through several development stages. The actual development stages within years can be seen below:

- Development of the CAN bus goes back to 1983 when Bosch originally invented Control Area Network which was later codified into ISO 11898-1 standard.
- The CAN protocol was later released to the Society of Automotive Engineers (SAE) in 1986.
- Intel was the first one to introduce the CAN controller chips in 1987, and Phillips joined Intel shortly after that.

- In 1991, Bosch published CAN 2.0 (CAN 2.0A: 11 bit, 2.0B: 29bit).
- CAN bus as an international standard in ISO 11898, was adopted in 1993.
- In 2003, ISO 11898 became a standard series.
- In 2012, Bosch has released the CAN FD 1.0 - flexible data rate.
- In 2015, the CAN FD protocol has become standardized in ISO 11898-1.
- Lastly, the physical CAN layer up to 5Mbit/s has become standardized in ISO 11898-2, in 2016.

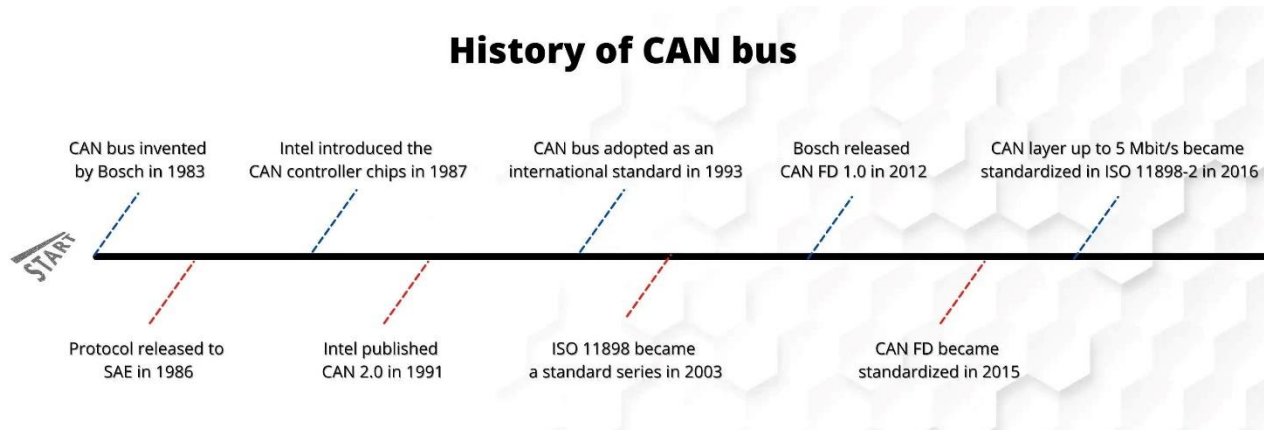


Figure 2.1: History of CAN technology

2.2 Benefits of CAN in automotive industry

CAN (Controller Area Network) is a communication protocol used in the automotive industry to enable communication between various electronic control units (ECUs) in a vehicle. Here are some of the benefits of CAN in the automotive industry:

- **Efficient communication:** CAN is a high-speed, efficient communication protocol that allows various ECUs to communicate with each other quickly and accurately. This is particularly important in modern vehicles, where there are many ECUs controlling different systems such as the engine, transmission, and brakes.
- **Reduced wiring:** By using CAN, the amount of wiring required in a vehicle is reduced. This is because CAN allows multiple ECUs to communicate over a single bus, reducing the need for multiple wires.
- **Real-time communication:** CAN is designed to provide real-time communication, which is essential in automotive applications where accurate and timely data is critical. For example, in a modern engine management system, the fuel injection system needs to be precisely timed to ensure the best possible performance and fuel economy.
- **Robustness:** CAN is a robust protocol that can operate in harsh automotive environments. The protocol is designed to tolerate electrical noise, voltage fluctuations, and other environmental factors that can impact communication.
- **Scalability:** CAN is a scalable protocol that can be used in a wide range of applications. For example, it can be used in small, low-cost vehicles as well as in high-end luxury vehicles with complex control systems.
- **Speed:** Currently defined by two physical layers - High-Speed CAN (CAN L) and Low-Speed CAN (CAN L), both with their advantages and disadvantages.

- Flexibility: CAN bus protocol is well-known as a message-based protocol, meaning nodes can easily be added or removed without performing any updates on the system. This makes it easy for engineers to integrate new electronic devices without significant programming and modify it to the user's requirements.

2.3 Basic knowledge in vehicle network

2.3.1 What is vehicle network?

Vehicle networking refers to the communication between various electronic components and systems within a vehicle. It is the exchange of information between systems such as the engine control module, transmission control module, anti-lock braking system, powertrain, infotainment system, and other electronic systems that are present in modern vehicles. Beside that Vehicle networking is specialized communication that connects the internal components of vehicles, helping vehicles communicate their locations and receive real-time information to prevent accidents and crashes.

Almost all vehicles have similar internal parts. They include:

- Electronic control units (ECUs)
- Sensors
- Engine control system
- Actuators

The quantity of these components can differ from vehicle to vehicle. For instance, in contemporary electric vehicles, there can be roughly 70 ECUs, which can exchange up to 2500 electronic signals amidst their various components.

ECUs, sensors, and actuators work together to establish a solid in-vehicle networking system. The ECUs receive input from the sensors, which help drivers identify and solve many recurring and potential car issues by monitoring the vehicle's speed, fuel temperature, tire pressure, etc.

2.3.2 Modules and nodes

A module is an electronic control unit (ECU) that controls a specific function or subsystem in a vehicle. For example, an engine control module (ECM) is a module that manages the engine's performance, while a transmission control module (TCM) controls the transmission system.

A node is a point of connection in a network where data is transmitted and received. In vehicle networking, each module is a node, and they communicate with each other through a network protocol. The network protocol specifies the rules and procedures for communication between the nodes.

2.3.3 Types of protocol in a vehicle

There are various types of protocols used in vehicle networking to ensure reliable and secure communication between different electronic components and systems. The protocols used for in-vehicle networks include CAN, Local Interconnect Network (LIN), FlexRay, Ethernet, and Media Oriented Systems Transport (MOST):

Local Interconnect Network (LIN): LIN bus is a supplement to CAN bus and is a low-speed, single-wire communication protocol used for communication between less critical components in a vehicle, such as door locks, window controls, and interior lighting.

FlexRay: is a unique time-triggered protocol that provides options for deterministic data that arrives in a predictable time frame. FlexRay is a serial communication technology that is used in particular for data communication in very safety-critical use areas in the automobile.

Ethernet: is a critical part of the in-vehicle networks that support modern vehicle applications. It satisfies high bandwidth requirements for safety applications, autonomous driving, and multimedia applications. It has a high 100 Mbps speed, which is 100 times quicker than CAN.

Media Oriented Systems Transport (MOST): is the de-facto standard for high-bandwidth automotive multimedia networking. It can be used to connect multiple consumer devices, via optical or electrical physical layers, directly to one another or in a network configuration. As a synchronous network, MOST technology provides excellent Quality of Service (QOS) and seamless connectivity for audio/video streaming.

In this paper, we just focus on **Controller Area Network Protocol (CAN)**, CAN is set of two electrical wires in the car network (CAN_Low and CAN_High), where the information is sent to and from ECUs. The network that allows ECUs to communicate is called Controller Area Network (CAN).

2.3.4 Data transmission flow

Before talking about the data transmission, we will study about what is a CAN node and structure of CAN node:

CAN (Controller Area Network) node refers to a device or subsystem that is connected to a CAN bus, which is a type of network used in vehicles, industrial automation, and other applications to allow different electronic devices to communicate with each other. A CAN node can be a sensor, actuator, control module, or any other device that is capable of sending or receiving messages over the CAN bus. Each CAN node has a unique identifier, or CAN ID, which is used to distinguish it from other nodes on the network. By exchanging messages over the CAN bus, CAN nodes can share information and work together to perform a range of functions in a variety of applications.

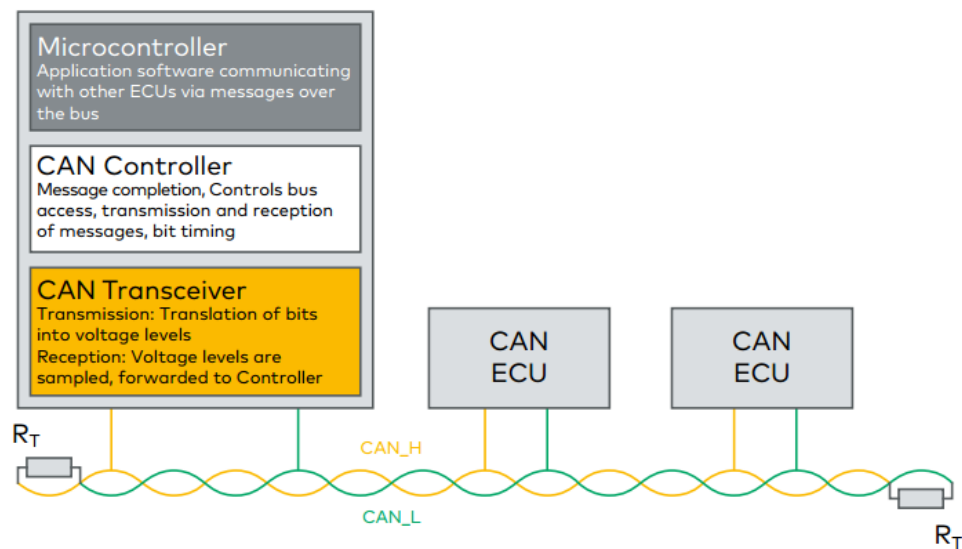


Figure 2.2: Structure of data transmission flow

A CAN node has a microcontroller, a CAN controller and a CAN transceiver, which is connected to the CAN bus. The microcontroller is a part of transport layer and the CAN controller is the data link layer and the CAN transceiver is the physical layer. The microcontroller deals with the software part and the actual data, the CAN controller and the CAN transceiver are the hardware component

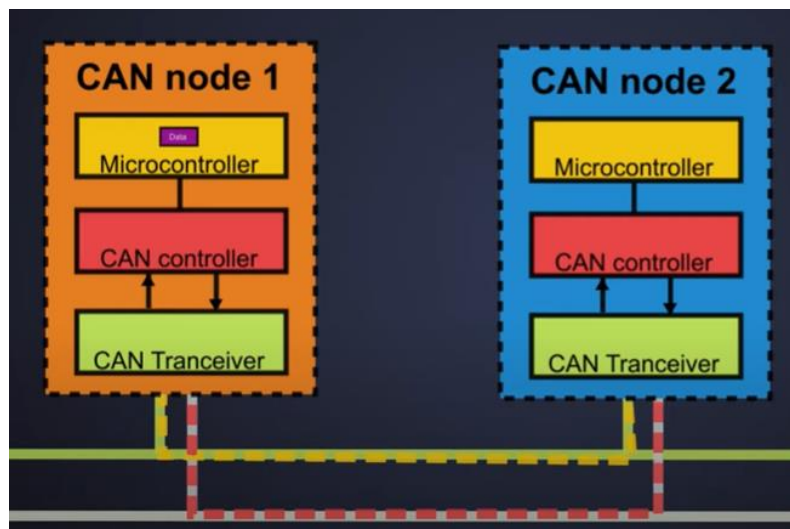


Figure 2.3: Communication between 2 CAN nodes

To clear the concept of data transmission, let's say node 1 wants to send some data to node 2 so the microcontroller will prepare the data which is supposed to be sent to another node, that data will go to the CAN controller which is preparing this data in a particular format like adding the proper message identification and formulating the CAN frame and now this data will be in the form of single-ended signal. The CAN controller will send this frame to the CAN transceiver and the CAN transceiver will convert this message into a differential signalling format and after that the data frame is sent on the bus and through the CAN bus the data frame from node 1 will be sent to node 2.

In the node 2 the CAN transceiver present inside the node will decode this data from differential signalling format to 0 and 1 which is readable to the CAN controller. The CAN controller will interpret this message and send the actual data to the microcontroller. Base on that the node will decide wheather this information is useful or not.

2.4 Study about Open Systems Interconnection Reference Model (OSI Model)

2.4.1 Open Systems Interconnection Reference Model (OSI Model)

OSI model or Open Systems Interconnection Reference Model is a theoretical framework that provides one way of thinking about networking. The OSI model separate the network communication between two devices on a network into seven abstractions layer.

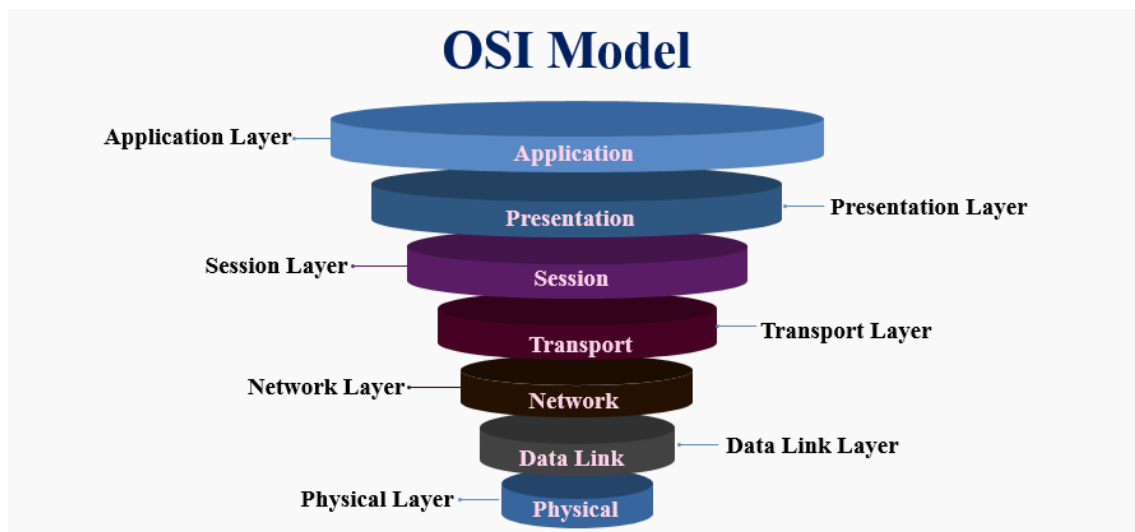


Figure 2.4: OSI model

The OSI is devided into two broad layers: upper layers and lower layers

The upper layers deals with the application-related layer subject, for instance: user interface, data representation, encryption. The upper layers are applied in software and are closest to the end user. The upper layers comprise these following layers:

- Application Layer
- Presentation Layer
- Session Layer

The lower layer of the OSI model deals with the data transport issues. The lower layers comprise these following layers:

- Transport Layer
- Network Layer
- Data Link Layer
- Physical layer

2.4.2 Layers in OSI model

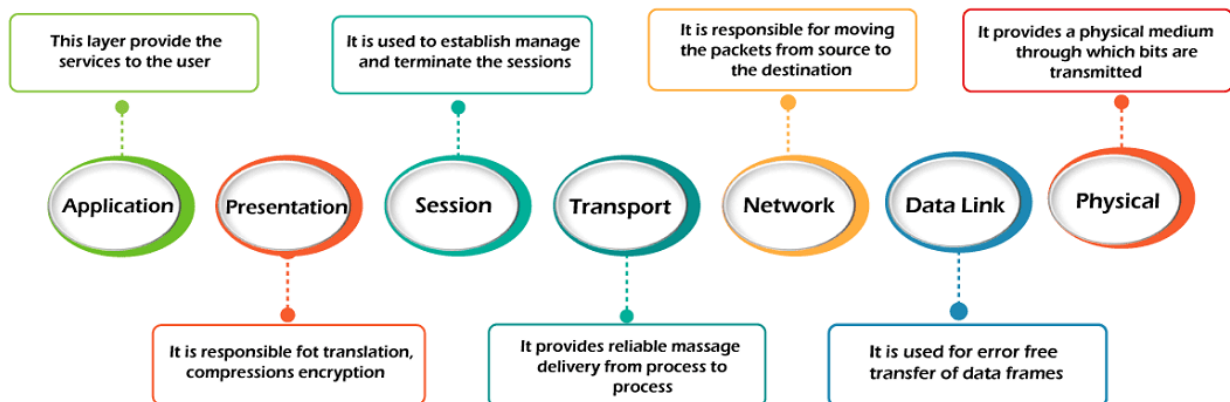


Figure 2.5: Layers in OSI model

2.4.2.1 Physical layer

The physical layer is the lowest layer of the OSI reference model. It is responsible for the actual physical connection between the devices. The physical layer contains information in the form of bits. It is responsible for transmitting individual bits from one node to the next. When receiving data, this layer will get the signal received and convert it into 0s and 1s and send them to the Data Link layer, which will put the frame back together.

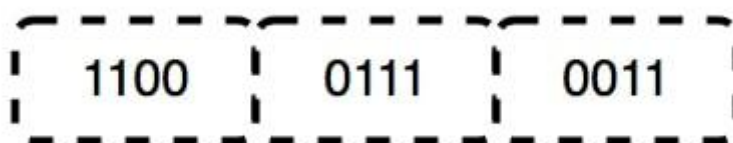


Figure 2.6: Data bits in the physical layer

Functions of physical layer:

Bit synchronization: The physical layer provides the synchronization of the bits by providing a clock. This clock controls both sender and receiver thus providing synchronization at the bit level.

Bit rate control: The Physical layer also defines the transmission rate such as the number of bits sent per second.

Physical topologies: Physical layer specifies how the different devices/nodes are arranged in a network such as bus, star, or mesh topology.

Transmission mode: Physical layer also defines how the data flows between the two connected devices. The various transmission modes possible are Simplex, half-duplex and full-duplex.

2.4.2.2 Data Link Layer

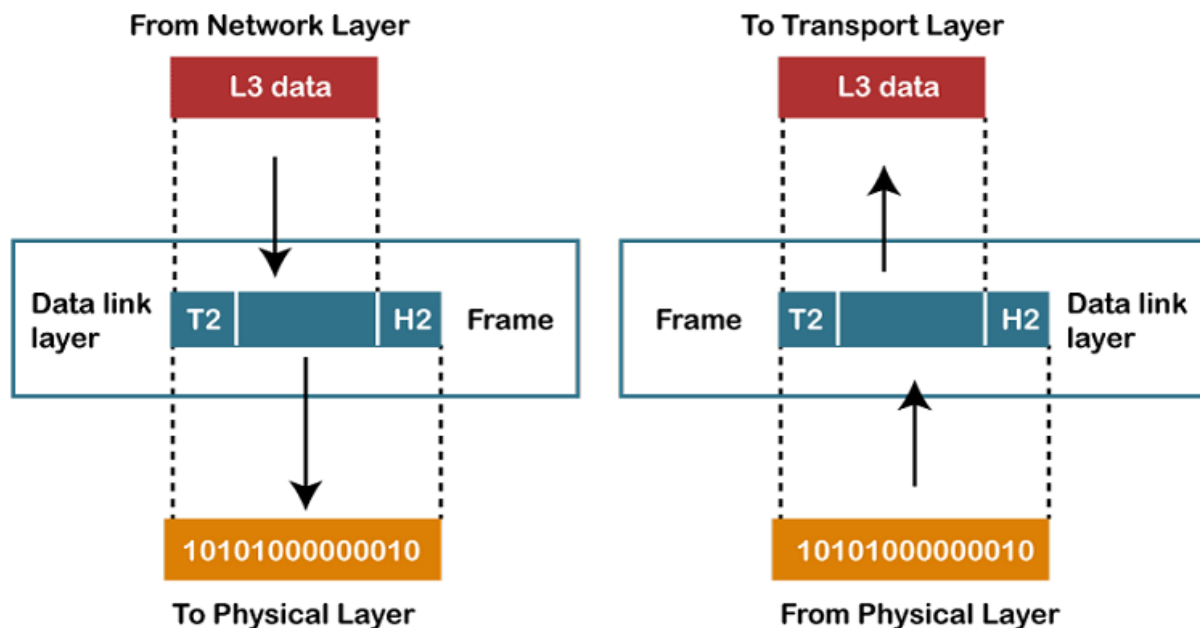


Figure 2.7: Data Link Layer

The Data Link layer, which is the second layer in the OSI (Open Systems Interconnection) model, is responsible for the reliable transfer of data between network nodes. Its primary function is to ensure the accurate transmission of information while also encoding, decoding, and organizing incoming and outgoing data. This is considered the most complex layer of the OSI model as it hides all the underlying complexities of the hardware from the other above layers.

Data Link Layer contains two sub-layers:

Logical Link Control:

It is responsible for transferring the packets to the Network layer of the receiver that is receiving.

It identifies the address of the network layer protocol from the header.

It also provides flow control.

Media Access Control:

A Media access control layer is a link between the Logical Link Control layer and the network's physical layer.

It is used for transferring the packets over the network.

Functions of Data Link Layers:

Framing: Framing is a function of the data link layer. It provides a way for a sender to transmit a set of bits that are meaningful to the receiver. This can be accomplished by attaching special bit patterns to the beginning and end of the frame.

Physical addressing: After creating frames, the Data link layer adds physical addresses (MAC addresses) of the sender and/or receiver in the header of each frame.

Error control: The data link layer provides the mechanism of error control in which it detects and retransmits damaged or lost frames.

Flow Control: The data rate must be constant on both sides else the data may get corrupted thus, flow control coordinates the amount of data that can be sent before receiving an acknowledgment.

Access control: When a single communication channel is shared by multiple devices, the MAC sub-layer of the data link layer helps to determine which device has control over the channel at a given time.

2.4.2.3 Network Layer

The network layer works for the transmission of data from one host to the other located in different networks. It also takes care of packet routing for example: selection of the shortest path to transmit the packet, from the number of routes available. The sender and receiver's IP addresses are placed in the header by the network layer.

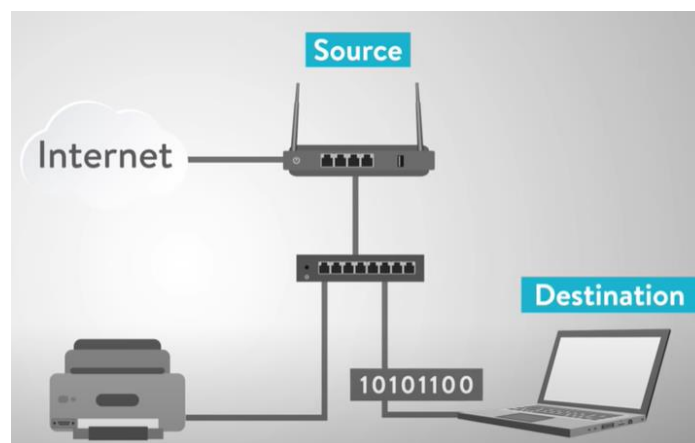


Figure 2.8: Network Layer

Functions of Network Layer:

Routing: The network layer protocols determine which route is suitable from source to destination.

Internetworking: An internetworking is the main responsibility of the network layer. It provides a logical connection between different devices.

Addressing: A Network layer adds the source and destination address to the header of the frame. Addressing is used to identify the device on the internet.

Packetizing: A Network Layer receives the packets from the upper layer and converts them into packets. This process is known as Packetizing. It is achieved by internet protocol (IP).

2.4.2.4 Transport Layer

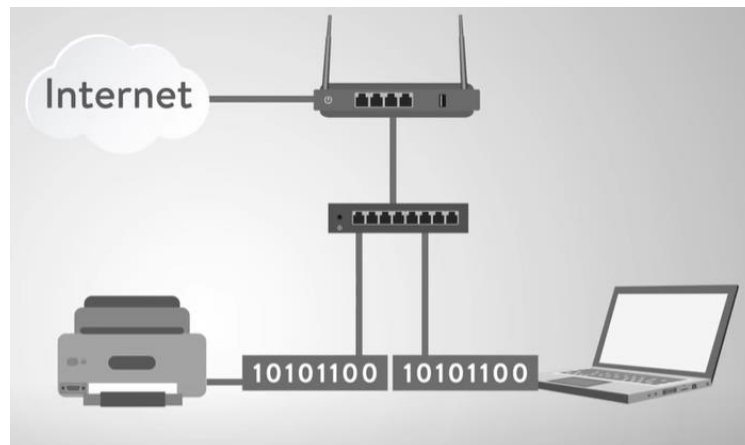


Figure 2.9: Transport Layer

The Transport layer is a Layer 4 ensures that messages are transmitted in the order in which they are sent and there is no duplication of data.

The main responsibility of the transport layer is to transfer the data completely.

It receives the data from the upper layer and converts them into smaller units known as segments.

This layer can be termed as an end-to-end layer as it provides a point-to-point connection between source and destination to deliver the data reliably.

There are two layers that used in the transport layer:

Transmission Control Protocol

Transmission is a standard protocol that allows the systems to communicate over the internet. As data is transmitted via a TCP connection, the TCP protocol breaks it down into smaller units called segments. These segments traverse the internet using various routes, leading to their arrival at the destination in different orders. However, the Transmission Control Protocol (TCP) at the receiving end rearranges the segments into the correct order.

User Datagram Protocol

UDP provides a mechanism to detect corrupt data in packets, but it does not attempt to solve other problems that arise with packets, such as lost or out of order packets. That's why UDP is sometimes known as the Unreliable Data Protocol.

Functions of transport layer:

Segmentation and reassembly: Upon receiving a message from the upper layer, the transport layer divides it into several segments, each of which is assigned a unique sequence number for identification purposes. Once the message reaches its destination, the transport layer reassembles the segments in the correct order based on their respective sequence numbers.

Flow control: The transport layer regulates the flow of data between the sender and receiver to prevent the receiver from being overwhelmed with too much data.

Service-point addressing: Due to the ability of computers to run multiple programs concurrently, data transmission occurs not only between computers but also between different processes within the same computer. To facilitate this, the transport layer adds a header to the data that includes a service-point address or port address. While the network layer is responsible for transmitting data between computers, the transport layer is tasked with ensuring that messages are correctly delivered to their intended processes.

Error control: In addition to its other functions, the transport layer is responsible for error control. Unlike error control at the link layer, which is limited to individual links, error control at the transport layer is end-to-end. The sender's transport layer verifies that messages are delivered to the destination without errors.

2.4.2.5 Session Layer

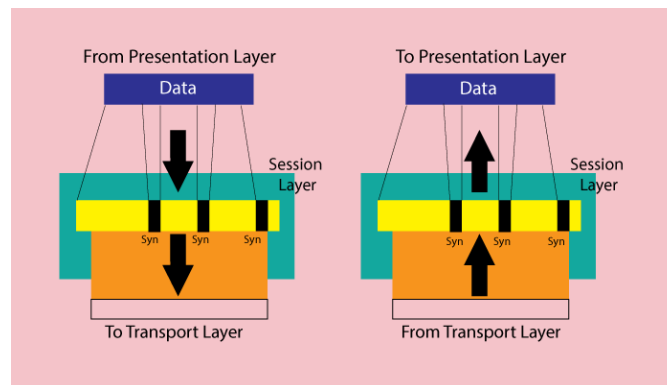


Figure 2.10: Session Layer

The Session layer is used to establish, maintain and synchronizes the interaction between communicating devices.

Functions of session layer:

Session establishment, maintenance, and termination: The layer allows the two processes to establish, use and terminate a connection.

Synchronization: The session layer incorporates checkpoints into the data transmission process to ensure that if any errors occur during transmission, the data can be retransmitted from the checkpoint. This mechanism is commonly referred to as synchronization and recovery.

Dialog control: The session layer allows two systems to start communication with each other in half-duplex or full-duplex.

2.4.2.6 Presentation Layer

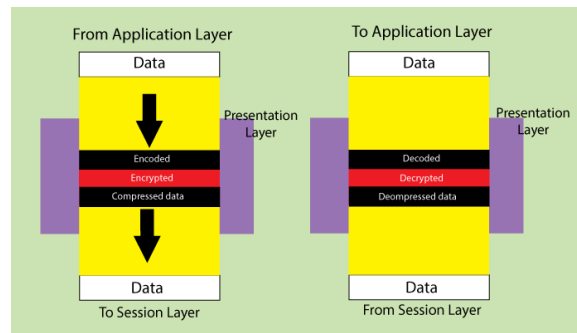


Figure 2.11: Presentation Layer

The Presentation layer is the sixth layer in the OSI (Open Systems Interconnection) model. It is responsible for ensuring that the data sent by one system is readable by another system, regardless of their different data representations, character sets, or encoding schemes. That is why the Presentation layer is also known as the syntax layer.

Functions of presentation layer:

Translation: When two systems exchange information, they may use different encoding methods such as character strings or numerical formats. This can lead to interoperability issues, as different computers may not understand each other's data formats. However, the Presentation layer in the OSI model addresses this problem by converting the data from the sender's format into a standardized format that both systems can interpret, and then converting it back into the receiver's format at the other end. In summary, the Presentation layer ensures that data is exchanged accurately and efficiently between systems with different encoding methods.

Encryption: The Presentation layer provides encryption and decryption services to ensure the confidentiality and integrity of data transmitted over the network.

Compression: Data compression is a process of compressing the data, for instance: it reduces the number of bits to be transmitted. Data compression is very important in multimedia such as text, audio, video.

2.4.2.7 Application Layer

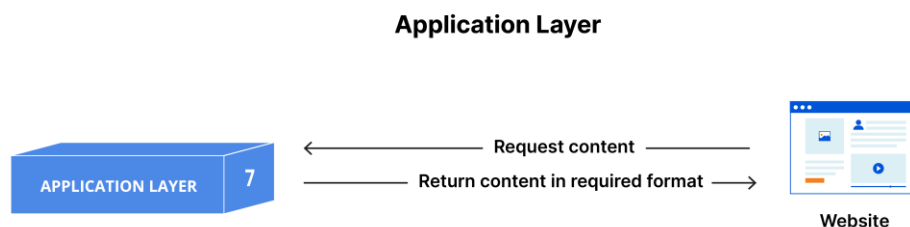


Figure 2.12: Application Layer

The Application layer is the topmost layer of the OSI (Open Systems Interconnection) model. It is responsible for providing end-user services and applications that communicate with the network.

Functions of Application layer:

File transfer, access, and management (FTAM): An application layer allows a user to access the files in a remote computer, to retrieve the files from a computer and to manage the files in a remote computer.

Mail services: An application layer provides the facility for email forwarding and storage.

Directory services: An application provides the distributed database sources and is used to provide that global information about various objects.

2.5 Message format of CAN protocol

In the CAN (Controller Area Network) protocol, message framing is a technique used to define the boundaries of a message or data packet. Messages in CAN are sent in a format called frames. A frame is defined structure, carrying meaningful sequence of bit or bytes of data within the network.

CAN frames can be classified into two types: standard and extended frames, based on the identifier fields used. A standard CAN frame has an 11-bit identifier field, while an extended frame has a 29-bit identifier field.

2.5.1 Standard frame

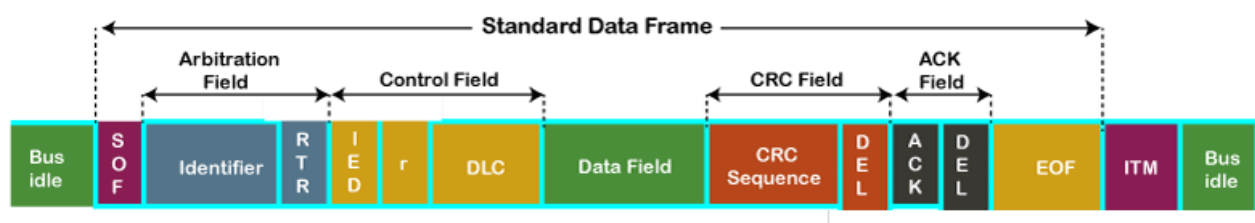


Figure 2.13: Standard frame

Start of Frame (SOF) field: This is a dominant bit that signals the start of the frame.

Identifier field: This is an 11-bit field that identifies the message and determines its priority. The first bit of the identifier field indicates the type of identifier used: a dominant bit for standard identifier or a recessive bit for extended identifier.

Remote Transmission Request (RTR) field: This is a single bit that indicates whether the message is a data frame or a remote frame. In a remote frame, this bit is set to recessive.

Control field includes identifier extension, data length code, data field.

The IDE (Identifier Extension) bit in the control field of a CAN (Controller Area Network) frame is used to determine the type of identifier being used. A dominant IDE bit indicates that the frame is using an 11-bit standard identifier, while a recessive IDE bit indicates that the frame is using a 29-bit extended identifier.

Data length field: defines the data length in a data field. It is of 4 bits.

Data field: The data field can contain upto 8 bytes.

Cyclic Redundancy Check (CRC): In a data frame of the CAN (Controller Area Network) protocol, a 15-bit cyclic redundancy check (CRC) field is included to detect any data corruption that may occur during transmission. Prior to sending the data frame, the sender computes the CRC and includes it in the frame. The receiver also computes the CRC upon receiving the frame and compares it with the CRC received from the sender. If the computed CRC does not match the received CRC, the receiver generates an error indicating that the data may have been corrupted during transmission.

Acknowledge (ACK) field: This is a recessive bit that signals that the frame has been received correctly.

End of Frame (EOF) field: This is a recessive bit that signals the end of the frame.

2.5.2 Extended frame

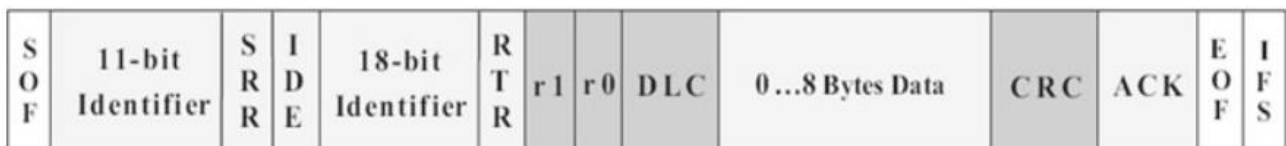


Figure 2.14: Extended frame

An extended frame is a type of message frame that uses a 29-bit identifier to identify the message. Extended frames are used when the standard 11-bit identifier is not sufficient to uniquely identify the message, or when more information needs to be transmitted in the identifier field.

Extended frames in the CAN protocol are typically used in applications where a large number of nodes need to communicate on the same network, such as in industrial automation, automotive, and aerospace applications. The use of extended frames allows for a higher level of specificity and information density in the messages being transmitted, while still maintaining the reliability and efficiency of the CAN protocol.

2.6 Principle of self-test of CAN network by Bit Stuffing

2.6.1 Parity bit

A parity bit is a check bit, which is added to a block of data for error detection purposes. It is used to validate the integrity of the data. The value of the parity bit is assigned either 0 or 1 that makes the number of 1s in the message block either even or odd depending upon the type of parity. Parity check is suitable for single bit error detection only.

The two types of parity checking are:

Even Parity: Here the total number of bits in the message is made even.

Odd Parity: Here the total number of bits in the message is made odd.

Error Detection by Adding Parity Bit

During the frame creation process at the sender's end, the number of 1s in the frame is counted, and the parity bit is added accordingly.

In case of even parity: If number of 1s is even, parity bit value is 0. If number of 1s is odd, parity bit value is 1.

In case of odd parity: If number of 1s is odd, parity bit value is 0. If number of 1s is even, parity bit value is 1.

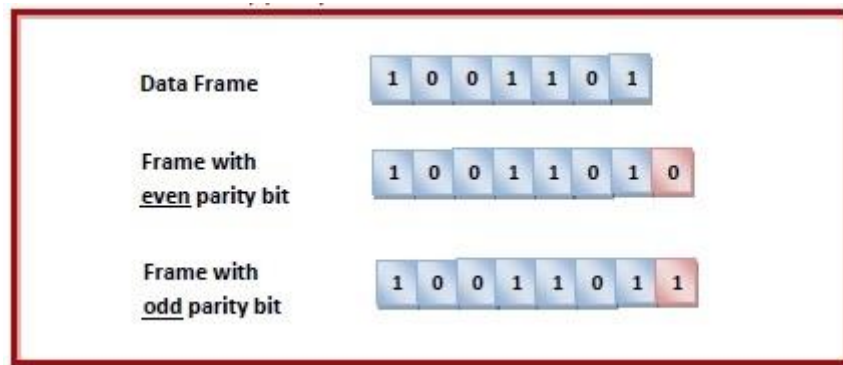


Figure 2.15: Parity bit

At the receiver's end, upon receiving a frame, the number of 1s in the frame is counted. For even parity checks, if the count of 1s is even, the frame is accepted. If the count of 1s is odd, the frame is rejected. For odd parity checks, if the count of 1s is odd, the frame is accepted. If the count of 1s is even, the frame is rejected.

2.6.2 Cyclic Redundancy Check

The Cyclic Redundancy Checks (CRC) is the most powerful method for Error-Detection and Correction. It is given as a kbit message and the transmitter creates an $(n - k)$ bit sequence called frame check sequence. The out coming frame, including n bits, is precisely divisible by some fixed number.

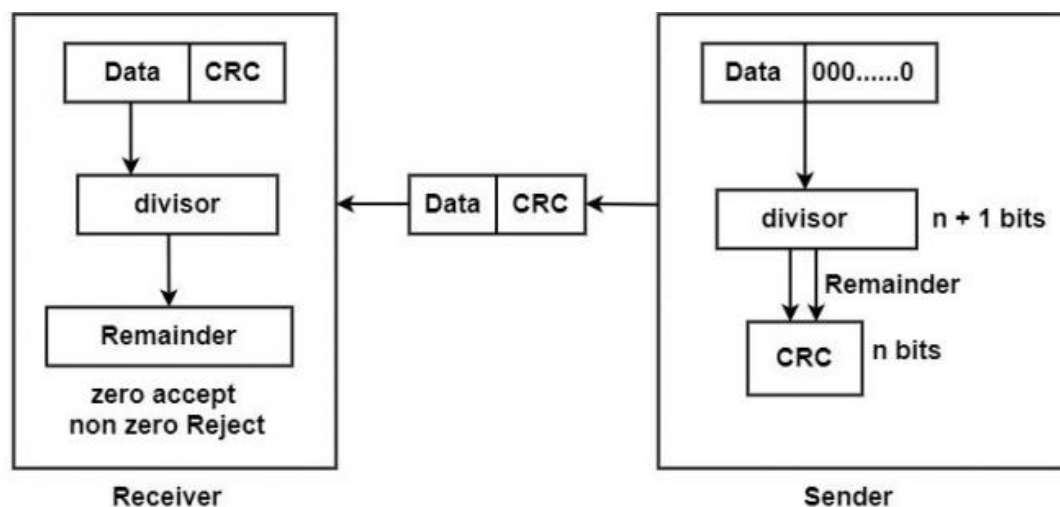


Figure 2.16: Cyclic Redundancy Check

Process of Cyclic Redundancy Check:

A string of n 0s is added to the data unit. The number n is one smaller than the number of bits in the fixed divisor.

The new data unit is divided by a divisor utilizing a procedure known as binary division; the remainder appearing from the division is CRC.

The CRC of n bits interpreted in phase 2 restores the added 0s at the end of the data unit.

2.6.3 Bit stuffing

The data link layer is responsible for something called Framing, which is the division of stream of bits from network layer into manageable units (called frames). Frames could be of fixed size or variable size. In variable-size framing, we need a way to define the end of the frame and the beginning of the next frame.

Bit stuffing is the insertion of non information bits into data. Note that stuffed bits should not be confused with overhead bits (Overhead bits are non-data bits that are necessary for transmission)

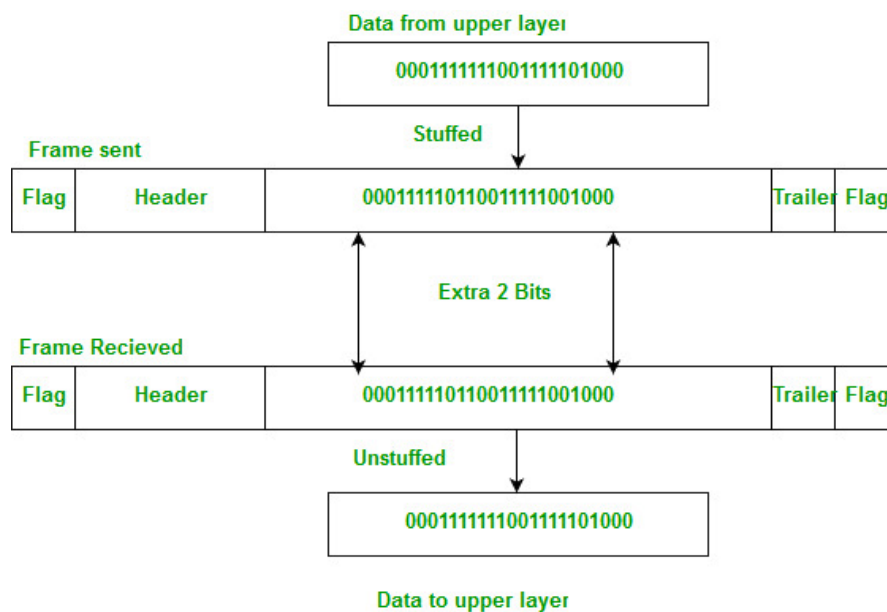


Figure 2.17: Bit stuffing

Functions of bit stuffing:

- synchronize several channels before multiplexing

The primary function of bit stuffing is to ensure that there are no consecutive identical bits in a data stream, which can cause synchronization problems in some communication protocols. Bit stuffing achieves this by inserting an extra non-informational bit into the data stream after every five consecutive bits of the same value. This process helps to maintain the integrity of the data and ensures that the receiver can correctly interpret the data being transmitted.

- rate-match two single channels to each other
- run length limited coding

2.7 Using Pair Twisted in CAN

Like electrical systems in cars, the CAN network is also affected by electromagnetic interference (EMI) from the surrounding environment (EMI caused by the current running through the wires). There are several methods to reduce this interference, and one of them is to use twisted pair cables to transmit signals. The CAN network mainly uses 2 wires (except for the LIN network), and the signal on the 2 wires is a differential bus signal, meaning that the voltage signal on the CAN Low and CAN High wires are symmetrical, so the twisting of the pair will eliminate any magnetic field (if any) generated by each wire.

So it will reduce the mutual interference between wires, and with other electrical systems. In addition, twisting the wires also ensures a constant impedance (120Ω), which helps to transmit signals effectively. The twist also keeps the two wires close together, protecting the signal from external electric fields by ensuring that both wires have the same level of common-mode voltage noise, eliminating the influence of noise.

Finally, the degree of twist helps the wire pair to resist interference from electromagnetic fields. The limited distance between the two wires helps to minimize contact with the magnetic field, along with the twist ensuring that the current flows in the opposite direction in the next twist, neutralizing the effect of the magnetic field.

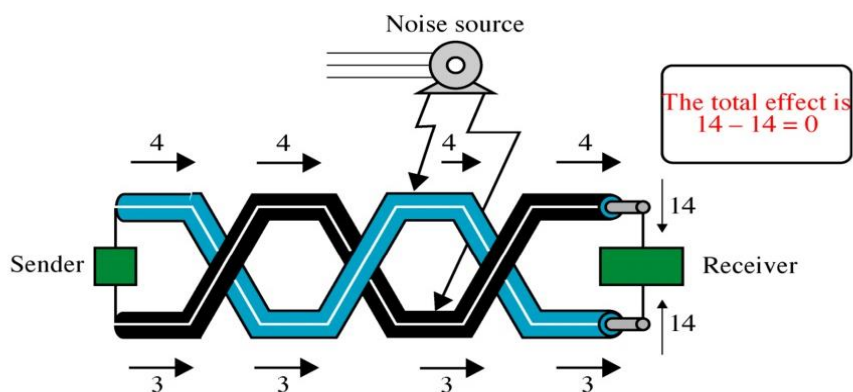


Figure 2.18: Pair Twisted in CAN

2.8 CAN Network Management

CAN Network Management is a module which is present in service layer of AUTOSAR and CAN Network Management is responsible for initiating wakeup and shutdown of CAN bus and synchronisation of all nodes in a cluster.

In an CAN node there are two types of frames mainly: Application frames, Network Management frame.

One CAN cluster can have many nodes. If a node needs to inform other nodes that it needs bus communication that it shall transmit periodic Network Management frames and if a node does not need bus communication it shall transmit no Network Management frame.

There are multiple functions that are handled by the CanNM module. The main purpose of CanNM is to co-ordinate the transition between the CAN network Normal operation and sleep operation mode. In addition to the main core functionality, it provides the service to detect all the nodes present in a particular network. Even if it also can detect that either all the nodes are ready to go to sleep mode or not. These kinds of functionality can be configurable by the AUTOSAR tools available like Vector Davinci Configuration Tool.

2.9 Terminating Resistor on CAN

A CAN bus terminator can be used for termination of any high speed (ISO 11898-2) CAN bus system. The 120 Ohm terminating resistor is setup between pin 2 (CAN low) and pin 7 (CAN high). In general, ISO 11898-2 CAN networks must be terminated at each end using 120 Ohm terminal resistors.



Figure 2.19: Terminal Resistor (120 Ohm, DB9, CAN Bus)

This CAN bus termination resistor is a D-sub 9 pin male connector & female socket. It contains a 120 Ohm terminal resistor between the CAN High / CAN Low pins.

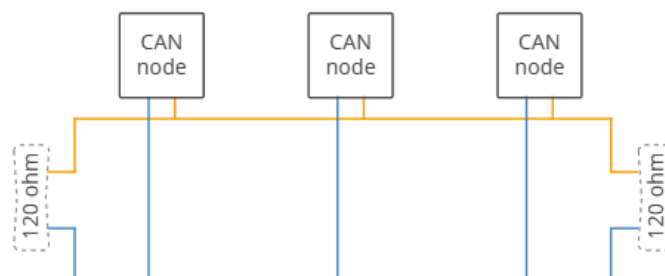


Figure 2.20: Terminal Resistor in CAN communication on cars

In a CAN Bus system, the terminal resistor is typically placed at the two ends of the bus, and it helps to reduce signal reflections and improve signal quality. The resistor value is usually 120 Ohms, which is the characteristic impedance of the CAN Bus.

Function of terminal resistor in CAN bus systems:

In CAN bus systems, terminal resistors are necessary because the communication flow is bidirectional. The termination resistors placed at each end of the bus absorb the energy of the CAN signal, preventing it from reflecting back from the cable ends. Reflections could cause interference and potentially damage the signals, so the resistors ensure that the communication is reliable and free of interference.

2.10 Study CAN network on Toyota Camry 2007

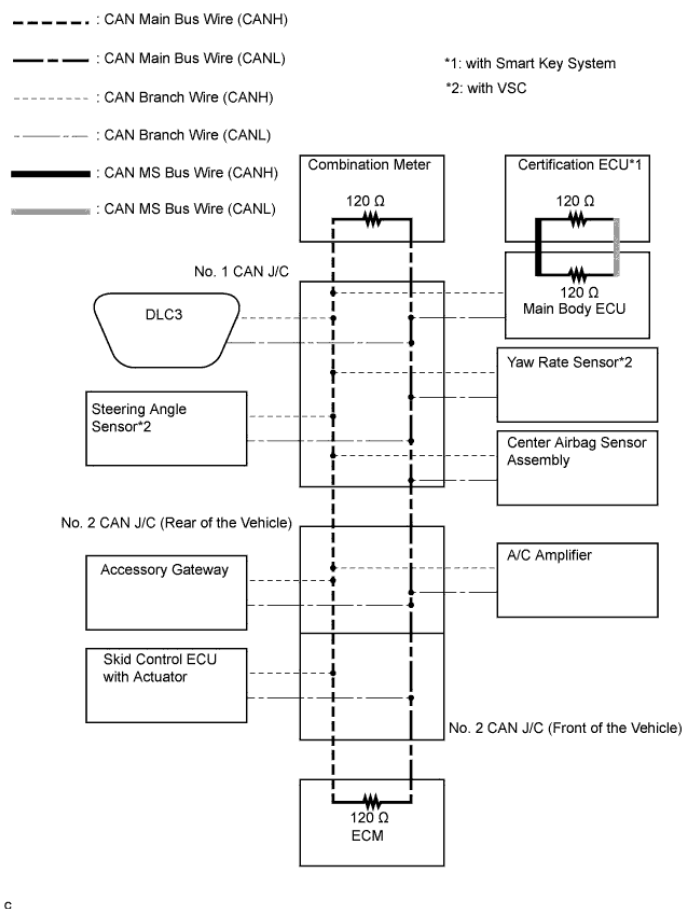


Figure 2.21: CAN network on Toyota Camry 2007

Two different CAN buses are used. The CAN buses are classified into two types based on typical communication speed.

The HS-CAN bus is a high-speed communication bus that is used for powertrain, chassis, and some body electrical communication. The HS-CAN bus is referred to as the “CAN bus” and it operates at speeds of approximately 500 kbps. Terminating resistors for the HS-CAN bus are located in the ECM and combination meter.

The MS-CAN bus is a medium-speed communication bus that is used for body electrical communication. The MS-CAN bus is referred as the “MS bus” and it operates at speeds of approximately 250 kbps. Terminating resistors for the MS-CAN bus are located in the main body ECU and the certification ECU. The resistance of the MS-CAN bus can not be measured from the DLC3 connector.

CHAPTER 3: Study about the CAN bus on Ford vehicle

3.1 Study about the Ford Focus 3M5F-10841-B instrument cluster



Figure 3.1: Ford Focus 3M5F-10841-B instrument cluster

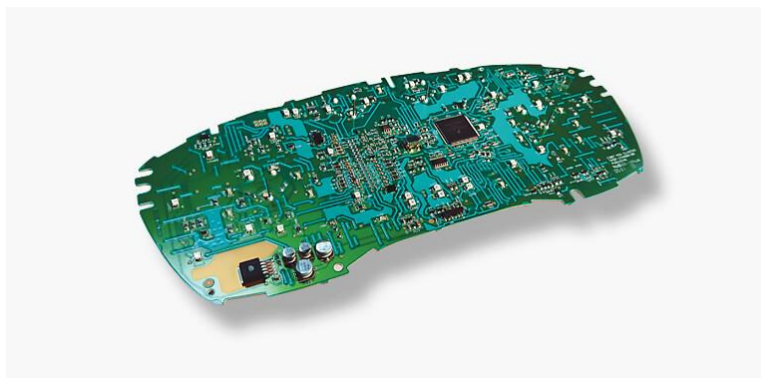


Figure 3.2: Main printed circuit board of Ford Focus 3M5F-10841-B instrument cluster

3.1.1 Some general details about Speedometer Ford Focus 3M5F-10841-B

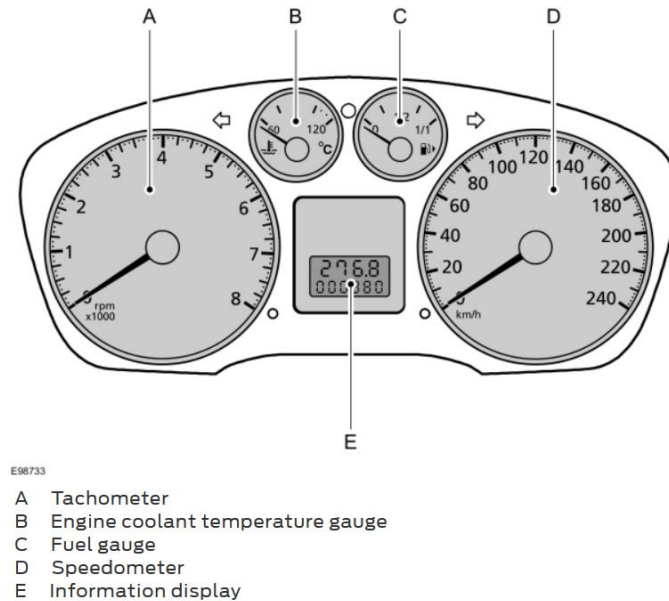


Figure 3.3: Ford Focus 3M5F-10841-B instrument cluster

- The needles of the gauges move because of stepper motors, which receive their signals from the processor. These signals usually originate from various CAN messages that were sent, sometimes more than a hundred times per second.
- Looking at the plug in more detail, it immediately becomes clear that no less than 32 pins are used. The reason that this speedometer uses many pin when CAN is also used and all CAN messages sent through two channels because this speedometer is connected to multiple components. Next to the regular connections for Vcontact, Vbattery and mass, the cluster also receives separate signals that were not sent through CAN, for example:
 - One of those signals that are sent through a separate channel, is a resistance signal coming from the electronic accelerator pedal. This signal is received by an independent pin and converted to a CAN message, enabling the signal to become useful for the Powertrain Control Module (PCM). Therefore, the PCM is completely dependent of the instrument cluster: when the cluster fails, there is no signal from the accelerator pedal. The driver can press the accelerator pedal as much as the driver want, but nothing will happen...
 - Another pin is specially reserved for managing the cluster lighting. This lighting is fully comprised of LED lights and obtains power from the lighting switch. Once the lighting (e.g. headlights) is switched on, the cluster lighting also turns on. The dimming function is controlled by the lighting switch itself.
 - There are 3 pins in use that can be connected to the mass once the menu in the cluster is operated by using the buttons on the steering column switch. Controlling these buttons produces a short signal to the processor that in turn changes the menu on the display.



Figure 3.4: Structure of SONCEBOZ 6405R421 Stepper Motor Instrument Cluster

3.1.2 Study about some component on main circuit board

3.1.2.1 YU3F-18B921-CA

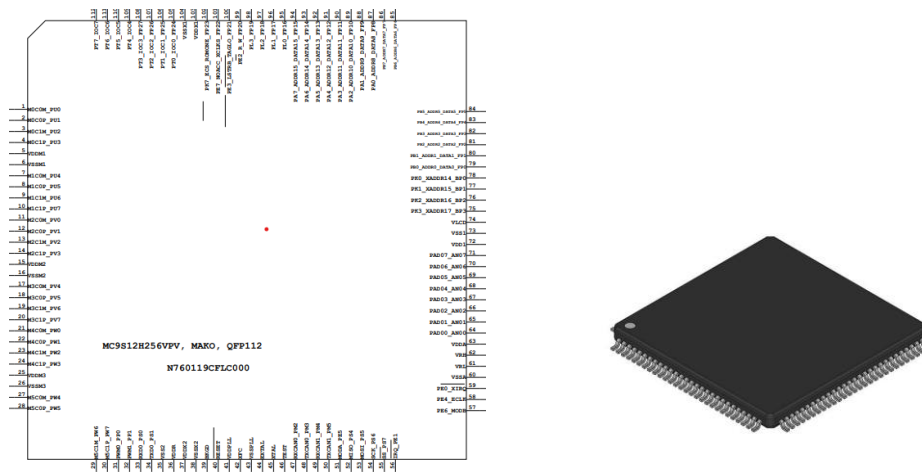
CONNECTOR, 6PIN, LCD		
M_LCD_SCL	<SCL>	1 -J3
LCD_RES	<POR>	2 -J3
M_LCD_SDA	<SDA>	3 -J3
M_LCD_POWER	<VDD>	4 -J3
	<VSS>	5 -J3
LCD_CAP	<VLCD>	6 -J3
YU3F-18B921-CA		
LCD CONNECTOR		



An LCD connector on an instrument cluster likely refers to the connector that connects the LCD screen on the instrument cluster to the device's internal circuitry. The LCD connector is typically a small rectangular port with several pins or contacts.

The connector is responsible for transmitting data and power signals between the LCD screen and the device's circuitry, allowing the screen to display information accurately.

3.1.2.2 MC9S12H256VPV (IC1)



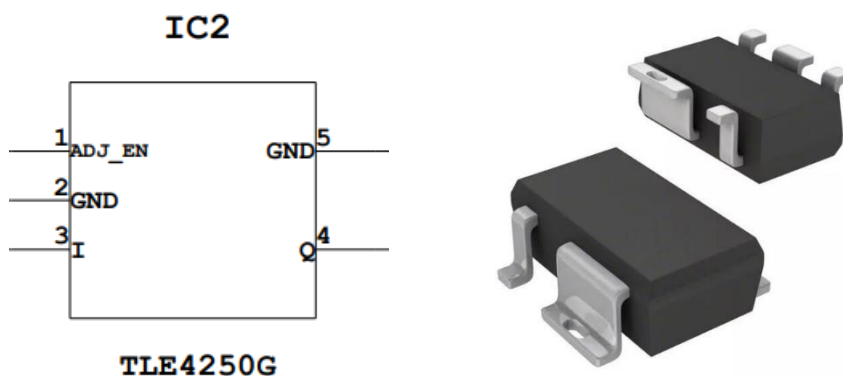
MC9S12H256VPV is a 16-bit flash microcontroller from NXP Semiconductors, formerly known as Freescale Semiconductor. It is based on the HCS12 (High-performance Computing System) architecture and is designed for use in automotive and industrial applications.

The MC9S12H256VPV microcontroller features a 16-bit central processing unit (CPU) with a clock speed of up to 25MHz, 256KB of on-chip flash memory, and 8KB of RAM. It also includes a range of peripherals, such as analog-to-digital converters (ADCs), timers, and communication interfaces like CAN, SCI, and SPI.

One notable feature of the MC9S12H256VPV microcontroller is its on-chip flash memory, which provides non-volatile storage for program code and data. This allows for easy reprogramming of the microcontroller in the field, without the need for external programming hardware.

The MC9S12H256VPV microcontroller is commonly used in automotive and industrial applications requiring high-performance computing, real-time control, and communication capabilities. It is well-suited for applications such as engine control, motor control, and instrumentation, as well as communication and networking in industrial automation systems.

3.1.2.3 TLE4250G (IC2)



(Low-drop voltage tracker)

Pin No.	Symbol	Function
1 A D J		Adjust/Enable input; connect to the reference voltage via ext. resistor or micro-controller port; high active input
2 G N D		Ground; internally connected to pin 5
3I		Input voltage
4Q		Output voltage; must be blocked by a capacitor $C_Q \geq 1 \mu\text{F}, 2 \Omega \leq \text{ESR} \leq 7 \Omega$
5 G N D		Ground

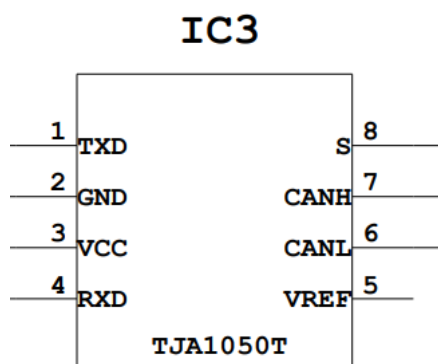
TLE4250G is a voltage regulator IC (integrated circuit) from Infineon Technologies AG. It is a low dropout linear voltage regulator designed for use in automotive applications.

The TLE4250G voltage regulator provides a fixed output voltage of 5 volts and is capable of supplying a maximum output current of 150 milliamperes (mA). It has a low dropout voltage of just 100 millivolts (mV) at full load, which helps to minimize power dissipation and improve efficiency.

The TLE4250G voltage regulator also includes a range of protection features, including over-temperature shutdown, over-current protection, and reverse polarity protection. These features help to ensure safe and reliable operation of the voltage regulator in harsh automotive environments.

The TLE4250G voltage regulator is commonly used in automotive applications such as electronic control units (ECUs), dashboard displays, and lighting systems, where a stable and regulated power supply is required. It is also suitable for use in various industrial and commercial applications where a low dropout voltage regulator is needed.

3.1.2.4 TJA1050T (IC3)



(High speed CAN transceiver)

SYMBOL	PIN	DESCRIPTION
TXD	1	transmit data input; reads in data from the CAN controller to the bus line drivers
GND	2	ground
V _{cc}	3	supply voltage
RXD	4	receive data output; reads out data from the bus lines to the CAN controller
V _{ref}	5	reference voltage output
CANL	6	LOW-level CAN bus line
CANH	7	HIGH-level CAN bus line
S	8	select input for high speed mode/silent mode

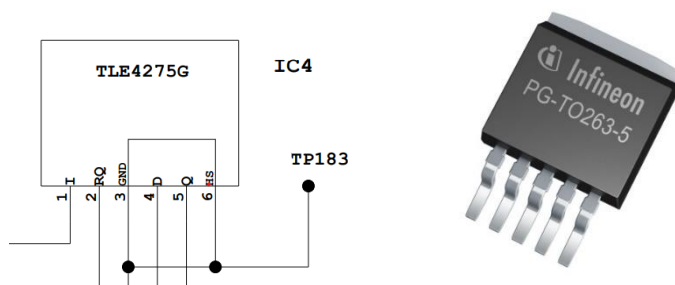
TJA1050T is a transceiver IC (integrated circuit) from NXP Semiconductors, formerly known as Philips Semiconductors. It is a high-speed CAN (Controller Area Network) transceiver designed for use in automotive and industrial applications.

The TJA1050T transceiver is designed to provide reliable communication between electronic control units (ECUs) in a CAN bus network. It is capable of transmitting and receiving data at speeds of up to 1 megabit per second (Mbps), making it suitable for high-speed applications.

The TJA1050T transceiver features a differential receiver with a wide common-mode range, which helps to ensure reliable operation in noisy automotive and industrial environments. It also includes a range of protection features, such as overvoltage protection, undervoltage protection, and thermal shutdown, to ensure safe and reliable operation of the transceiver.

The TJA1050T transceiver is commonly used in automotive applications such as engine control, transmission control, and body electronics, as well as in industrial automation systems and other applications where reliable and high-speed communication is required. It is a widely used and well-established component in the automotive and industrial electronics industry.

3.1.2.5 TLE4275G (IC4)



(5-V Low-drop voltage regulator)

Pin No.	Symbol	Function
1	I	Input; block to ground directly at the IC by a ceramic capacitor.
2	RO	Reset Output; open collector output
3	GND	Ground; Pin 3 internally connected to heatsink
4	D	Reset Delay; connect capacitor to GND for setting delay time
5	Q	Output; block to ground with a $\geq 22 \mu\text{F}$ capacitor, ESR < 5 Ω at 10 kHz.

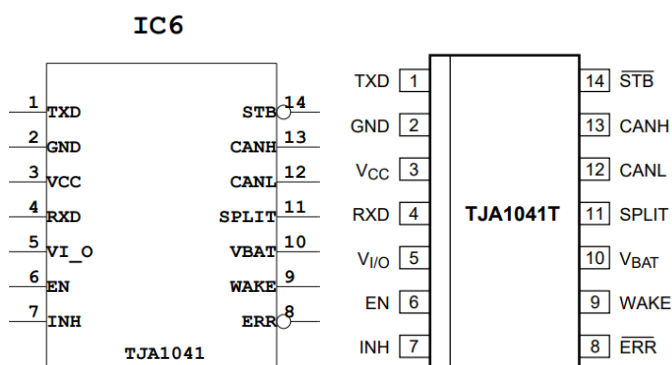
TLE4275G is a low-dropout linear voltage regulator IC (integrated circuit) from Infineon Technologies AG. It is designed to provide a stable and regulated output voltage for a wide range of automotive and industrial applications.

The TLE4275G voltage regulator is capable of providing a fixed output voltage of 5 volts with a maximum output current of 450 milliamperes (mA). It features a low dropout voltage of only 40 millivolts (mV) at full load, which helps to minimize power dissipation and improve efficiency.

One notable feature of the TLE4275G voltage regulator is its high accuracy and low output voltage noise, which makes it well-suited for use in sensitive electronic applications that require a stable and noise-free power supply. The voltage regulator also includes a range of protection features, such as over-temperature shutdown, over-current protection, and reverse polarity protection, to ensure safe and reliable operation in harsh environments.

The TLE4275G voltage regulator is commonly used in automotive applications such as electronic control units (ECUs), dashboard displays, and lighting systems, as well as in various industrial and commercial applications where a low dropout voltage regulator is needed.

3.1.2.6 TJA1041 (IC6)



(High speed CAN transceiver)

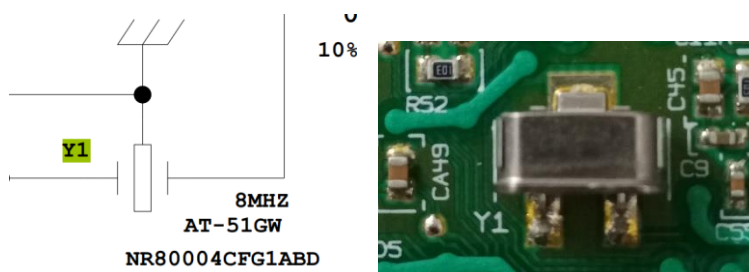
Symbol	Pin	Description
TXD	1	transmit data input
GND	2	ground
V _{CC}	3	transceiver supply voltage input
RXD	4	receive data output; reads out data from the bus lines
V _{I/O}	5	I/O-level adapter voltage input
EN	6	enable control input
INH	7	inhibit output for switching external voltage regulators
$\overline{\text{ERR}}$	8	error and power-on indication output (active LOW)
WAKE	9	local wake-up input
V _{BAT}	10	battery voltage input
SPLIT	11	common-mode stabilization output
CANL	12	LOW-level CAN bus line
CANH	13	HIGH-level CAN bus line
$\overline{\text{STB}}$	14	standby control input (active LOW)

TJA1041 is a transceiver chip used in Controller Area Network (CAN) bus systems. It is designed to provide reliable communication between different nodes in a CAN network, allowing them to exchange data and control signals.

The TJA1041 chip is manufactured by NXP Semiconductors, a company that specializes in the design and production of semiconductor components for a range of industries. The TJA1041 chip is commonly used in automotive applications, such as in the communication between various electronic control units (ECUs) in a car.

The TJA1041 chip is designed to operate in harsh environments, with a wide operating temperature range and robust protection features to ensure reliable operation in the presence of electrical noise and other disturbances. It supports both high-speed CAN (up to 1 Mbit/s) and low-speed CAN (up to 125 kbit/s) communication, making it a versatile and widely used component in CAN bus systems.

3.1.2.7 8 MHz crystal resonator



(Crystal resonator)

An 8 MHz crystal resonator is an electronic component that is designed to provide a stable and precise frequency of 8 MHz. It is commonly used in a variety of electronic applications, such as in microcontrollers, oscillators, and timing circuits.

3.2 Ford Controller Area Network

3.2.1 Standard Corporate Protocol (SCP)

The SCP bus is a 2 wire twisted pair network conforming to the J1850 standard (meaning that it can be used for Generic Scan Tool communications). SCP was the most common protocol on Ford applications until the proliferation of CAN (HSCAN and MSCAN). For network and "no communication" faults, always check wiring diagrams, since the modules communicating on SCP (as opposed to ISO 9141), vary considerably between models and years.

The SCP bus communicates in the 0-5 Volts range, and is wired to the Generic DLC at Pins 2 and 10.

3.2.2 Generic DLC

The DLC or Data Link Connector is an OBD II, 16 pin, standard connector used in all automobiles manufactured since 1996. In today's vehicles, it's in constant communication with the CAN, or Computer Area Network. It provides a standard connection for automotive technicians to tap into and diagnose different onboard computers.

Many Ford, Lincoln, Mercury and Mazda models are equipped with MSCAN bus, in addition to HSCAN bus. ELM327 is supported HSCAN bus, MSCAN bus is not supported by the stock OBD2 ELM327 adapter, because MSCAN bus is a Ford protocol specific solution and on non-OBD2 pins.

The ELM327 OBD2 adapter supported HSCAN bus from stock, because HSCAN bus completely with OBD2 standards. It is support pins 6 and 14. With MSCAN bus occupies pins 3 and 11.

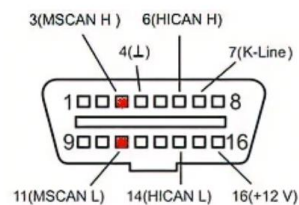


Figure 3.5: ELM327 OBD2

In addition to supporting pins 6 and 14 for the HSCAN bus and pins 3 and 11 for the MSCAN bus, the ELM327 OBD2 adapter also supports other pins on the OBD2 connector. The specific pins supported by the adapter may vary depending on the adapter model and manufacturer, but common pins supported by OBD2 adapters include:

- Pin 4 (Chassis Ground): This pin provides a ground connection to the vehicle's chassis.
- Pin 5 (Signal Ground): This pin provides a ground connection for the OBD2 signal.
- Pin 7 (K-Line): This pin is used for communication with the engine control module (ECM) on some vehicles.
- Pin 10 (J1850 Bus+): This pin is used for communication with some Ford and GM vehicles.

- Pin 15 (L-Line): This pin is used for communication with the body control module (BCM) on some vehicles.
- Pin 16 (Power): This pin provides power to the OBD2 adapter, typically from the vehicle's battery or accessory power.

3.2.3 J1850 Standard

The SAE J1850 Standard had been a recommended practice for seven years before being officially adopted by the Society of Automotive Engineers, (SAE), as the standard protocol for Class B in-vehicle networks on February 1, of 1994. Today, J1850 is implemented in a variety of production vehicles for diagnostics and data sharing purposes.

3.2.4 ISO 9141

The ISO 9141 protocol is a network with series or parallel legs that connect each of the modules on the network. It is usually a single wire network, and is connected to the Generic DLC at Pin 7. The ISO 9141 protocol is not used for intermodule communications. It is only a diagnostic protocol. The modules on this network can only communicate with the Scan Tool, and not with each other.

The ISO 9141 bus voltage is 12 Volts at rest, and modules pull the signal low to communicate.

3.2.5 Controller Area Network

Ford used CAN C and CAN B, which they refer to as HSCAN and MSCAN. Both use 2 wires for redundancy and both are 0-1 Volt signals riding on a 2.5 Volt bias. Each protocol has positive and negative circuits. MSCAN+ and HSCAN+ cycle from 2.5-3.5 Volts; MSCAN- and HSCAN- cycle from 2.5-1.5 Volts.

HSCAN is wired to the Generic DLC at Pins 6 and 14, and MSCAN is wired to Pins 3 and 11. If an application has MSCAN but no connection to the DLC, then the data is being converted to HSCAN through a module acting as a gateway.

3.2.6 Example of Controller Area Network on 2010-2014 Ford Mustangs:

The 2010-2014 Mustangs have 2 CAN networks HSCAN and MSCAN. They are twisted pair networks run throughout the car:

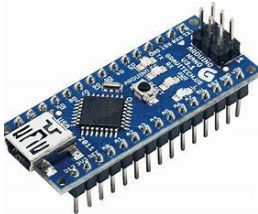
- HSCAN runs at 500Kbps. It is the fastest of the two networks, and all mission critical systems communicate here. Here are some modules on this network:
 - PCM - Powertrain Control Module - 0x7E0
 - ABS - Anti Lock Brake System - 0x760
 - IPC - Instrument Panel Cluster - 0x720
 - PSCM - Power Steering Control Module - 0x730
 - RCM - Restraints Control Module - 0x737
- MSCAN runs 125kbs. It is the slowest of the two networks, and most non mission critical items reside here such as infotainment. Here are some modules on this network:
 - IPC - Instrument Panel Cluster
 - BCM - Body Control Module
 - ACM - Audio Control Module



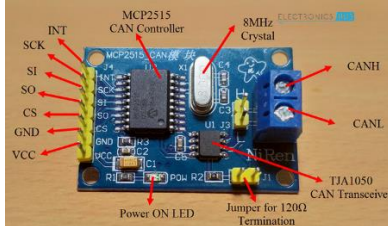
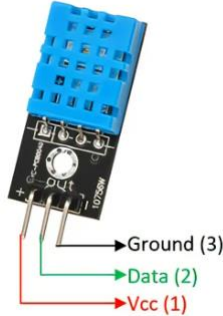
- APIM - Accessory Protocol Interface Module
- FCIM - Front Controls interface module
- FDIM - Front Display Interface Module
- GPSM - Global Positioning System Module

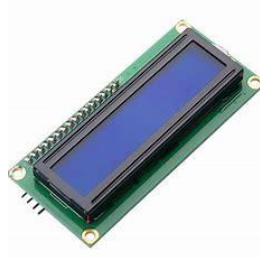

At some times though information from the HSCAN bus needs to be relayed on the MSCAN bus. The Instrument cluster serves the functions of translating and re transmitting data from the HSCAN bus to MSCAN both. It is connected to both networks.

3.3 Implement “Interfacing MCP2515 CAN Module with Arduino”

3.3.1 Components and specifications/Schematics:

Number	Name	Component	Specification/Schematics																																															
1	Arduino Nano Board		<table><tr><td rowspan="2">Board</td><td>Name</td><td>Arduino® Nano</td></tr><tr><td>SKU</td><td>A000005</td></tr><tr><td>Microcontroller</td><td colspan="2">ATmega328</td></tr><tr><td>USB connector</td><td colspan="2">Mini-B USB</td></tr><tr><td rowspan="4">Pins</td><td>Built-in LED Pin</td><td>13</td></tr><tr><td>Digital I/O Pins</td><td>14</td></tr><tr><td>Analog input pins</td><td>8</td></tr><tr><td>PWM pins</td><td>6</td></tr><tr><td rowspan="3">Communication</td><td>UART</td><td>RX/TX</td></tr><tr><td>I2C</td><td>A4 (SDA), A5 (SCL)</td></tr><tr><td>SPI</td><td>D11 (COPI), D12 (CIPO), D13 (SCK). Use any GPIO for Chip Select (CS).</td></tr><tr><td rowspan="3">Power</td><td>I/O Voltage</td><td>5V</td></tr><tr><td>Input voltage (nominal)</td><td>7-12V</td></tr><tr><td>DC Current per I/O Pin</td><td>20 mA</td></tr><tr><td>Clock speed</td><td>Processor</td><td>ATmega328 16 MHz</td></tr><tr><td>Memory</td><td>ATmega328P</td><td>2KB SRAM, 32KB flash 1KB EEPROM</td></tr><tr><td rowspan="3">Dimensions</td><td>Weight</td><td>5gr</td></tr><tr><td>Width</td><td>18 mm</td></tr><tr><td>Length</td><td>45 mm</td></tr></table>	Board	Name	Arduino® Nano	SKU	A000005	Microcontroller	ATmega328		USB connector	Mini-B USB		Pins	Built-in LED Pin	13	Digital I/O Pins	14	Analog input pins	8	PWM pins	6	Communication	UART	RX/TX	I2C	A4 (SDA), A5 (SCL)	SPI	D11 (COPI), D12 (CIPO), D13 (SCK). Use any GPIO for Chip Select (CS).	Power	I/O Voltage	5V	Input voltage (nominal)	7-12V	DC Current per I/O Pin	20 mA	Clock speed	Processor	ATmega328 16 MHz	Memory	ATmega328P	2KB SRAM, 32KB flash 1KB EEPROM	Dimensions	Weight	5gr	Width	18 mm	Length	45 mm
			Board		Name	Arduino® Nano																																												
				SKU	A000005																																													
			Microcontroller	ATmega328																																														
			USB connector	Mini-B USB																																														
			Pins	Built-in LED Pin	13																																													
				Digital I/O Pins	14																																													
				Analog input pins	8																																													
				PWM pins	6																																													
			Communication	UART	RX/TX																																													
I2C	A4 (SDA), A5 (SCL)																																																	
SPI	D11 (COPI), D12 (CIPO), D13 (SCK). Use any GPIO for Chip Select (CS).																																																	
Power	I/O Voltage	5V																																																
	Input voltage (nominal)	7-12V																																																
	DC Current per I/O Pin	20 mA																																																
Clock speed	Processor	ATmega328 16 MHz																																																
Memory	ATmega328P	2KB SRAM, 32KB flash 1KB EEPROM																																																
Dimensions	Weight	5gr																																																
	Width	18 mm																																																
	Length	45 mm																																																

2	Arduino UNO Board		<table><tr><td>Microcontroller</td><td>ATmega328P</td></tr><tr><td>Operating Voltage</td><td>5V</td></tr><tr><td>Input Voltage (recommended)</td><td>7-12V</td></tr><tr><td>Input Voltage (limit)</td><td>6-20V</td></tr><tr><td>Digital I/O Pins</td><td>14 (of which 6 provide PWM output)</td></tr><tr><td>PWM Digital I/O Pins</td><td>6</td></tr><tr><td>Analog Input Pins</td><td>6</td></tr><tr><td>DC Current per I/O Pin</td><td>20 mA</td></tr><tr><td>DC Current for 3.3V Pin</td><td>50 mA</td></tr><tr><td>Flash Memory</td><td>32 KB (ATmega328P) of which 0.5 KB used by bootloader</td></tr><tr><td>SRAM</td><td>2 KB (ATmega328P)</td></tr><tr><td>EEPROM</td><td>1 KB (ATmega328P)</td></tr><tr><td>Clock Speed</td><td>16 MHz</td></tr><tr><td>LED_BUILTIN</td><td>13</td></tr><tr><td>Length</td><td>68.6 mm</td></tr><tr><td>Width</td><td>53.4 mm</td></tr><tr><td>Weight</td><td>25 g</td></tr></table>	Microcontroller	ATmega328P	Operating Voltage	5V	Input Voltage (recommended)	7-12V	Input Voltage (limit)	6-20V	Digital I/O Pins	14 (of which 6 provide PWM output)	PWM Digital I/O Pins	6	Analog Input Pins	6	DC Current per I/O Pin	20 mA	DC Current for 3.3V Pin	50 mA	Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader	SRAM	2 KB (ATmega328P)	EEPROM	1 KB (ATmega328P)	Clock Speed	16 MHz	LED_BUILTIN	13	Length	68.6 mm	Width	53.4 mm	Weight	25 g
Microcontroller	ATmega328P																																				
Operating Voltage	5V																																				
Input Voltage (recommended)	7-12V																																				
Input Voltage (limit)	6-20V																																				
Digital I/O Pins	14 (of which 6 provide PWM output)																																				
PWM Digital I/O Pins	6																																				
Analog Input Pins	6																																				
DC Current per I/O Pin	20 mA																																				
DC Current for 3.3V Pin	50 mA																																				
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader																																				
SRAM	2 KB (ATmega328P)																																				
EEPROM	1 KB (ATmega328P)																																				
Clock Speed	16 MHz																																				
LED_BUILTIN	13																																				
Length	68.6 mm																																				
Width	53.4 mm																																				
Weight	25 g																																				
3	CAN Module MCP2515																																				
4	DHT11–Temperature and Humidity Sensor		<table><tr><td>1</td><td>Vcc</td><td>Power supply 3.5V to 5.5V</td></tr><tr><td>2</td><td>Data</td><td>Outputs both Temperature and Humidity through serial Data</td></tr><tr><td>3</td><td>Ground</td><td>Connected to the ground of the circuit</td></tr></table>	1	Vcc	Power supply 3.5V to 5.5V	2	Data	Outputs both Temperature and Humidity through serial Data	3	Ground	Connected to the ground of the circuit																									
1	Vcc	Power supply 3.5V to 5.5V																																			
2	Data	Outputs both Temperature and Humidity through serial Data																																			
3	Ground	Connected to the ground of the circuit																																			

5	LCD Display		<table> <tr> <th>Item</th> <th>Standard Value</th> </tr> <tr> <td>Display Type</td> <td>16characters × 2 lines</td> </tr> <tr> <td>LCD Type</td> <td>STN. POSITIVE (Y-G) ,TRANSFLECTIVE</td> </tr> <tr> <td>Driver Condition</td> <td>LCD Module : 1/16Duty , 1/5Bias</td> </tr> <tr> <td>Viewing Direction</td> <td>6 O'clock</td> </tr> <tr> <td>Backlight Type</td> <td>SIDE Yellow-Green</td> </tr> <tr> <td>Interface</td> <td>8-bit MPU interface</td> </tr> <tr> <td>Driver IC</td> <td>SPLC780D</td> </tr> </table>	Item	Standard Value	Display Type	16characters × 2 lines	LCD Type	STN. POSITIVE (Y-G) ,TRANSFLECTIVE	Driver Condition	LCD Module : 1/16Duty , 1/5Bias	Viewing Direction	6 O'clock	Backlight Type	SIDE Yellow-Green	Interface	8-bit MPU interface	Driver IC	SPLC780D
Item	Standard Value																		
Display Type	16characters × 2 lines																		
LCD Type	STN. POSITIVE (Y-G) ,TRANSFLECTIVE																		
Driver Condition	LCD Module : 1/16Duty , 1/5Bias																		
Viewing Direction	6 O'clock																		
Backlight Type	SIDE Yellow-Green																		
Interface	8-bit MPU interface																		
Driver IC	SPLC780D																		
6	RTC DS1307																		

3.3.2 Circuit diagram

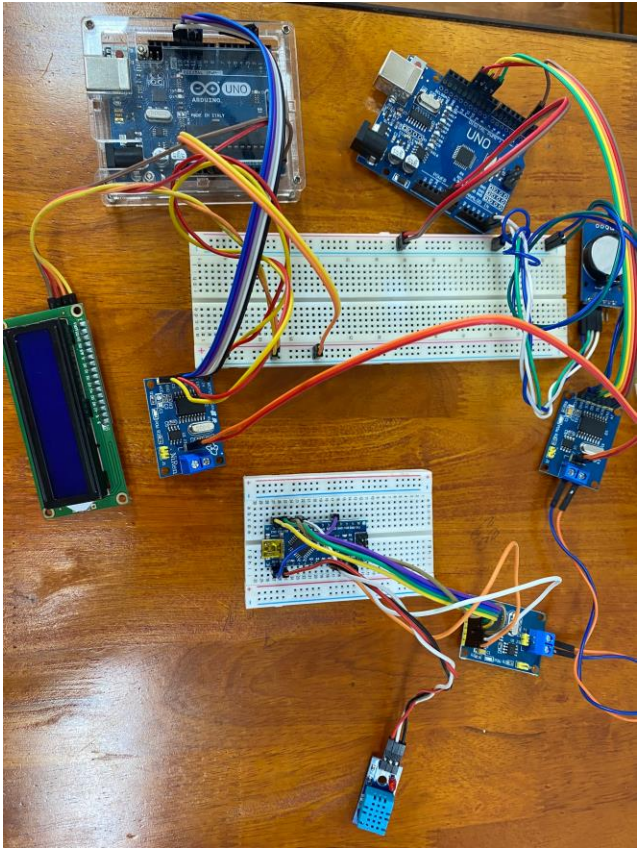


Figure 3.6: Circuit diagram for three CAN nodes

3.3.3 CAN Transmitter code and CAN Receiver code

3.3.3.1 CAN Transmitter for DHT11-Temperature and Humidity Sensor

```
#include <SPI.h>
#include <mcp2515.h>
#include <DHT.h>

#define DHTPIN 8
#define DHTTYPE DHT11
struct can_frame canMsg;
MCP2515 mcp2515(10);
DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  while (!Serial);
  Serial.begin(115200);
  SPI.begin();
  dht.begin();
  mcp2515.reset();
  mcp2515.setBaudrate(CAN_500KBPS, MCP_16MHZ);
  mcp2515.setNormalMode();
}

void loop()
{
  int h = dht.readHumidity();
  int t = dht.readTemperature();
  canMsg.can_id = 0x036;
  canMsg.can_dlc = 8;
  canMsg.data[6] = h;
  canMsg.data[7] = t;
  mcp2515.sendMessage(&canMsg);
  delay(1000);
}
```

Explain the code:

The code begins with including the necessary libraries: SPI.h for communication with the MCP2515, mcp2515.h for controlling the MCP2515, and DHT.h for communicating with the DHT11 sensor.

Next, some constants are defined: DHTPIN for the digital pin connected to the DHT11 sensor, DHTTYPE for the type of the sensor, and a struct can_frame for defining the CAN message to be sent.

In the setup() function, the serial communication is initialized, SPI is started, the DHT11 sensor is initialized, the MCP2515 is reset, the bit rate of the CAN bus is set to 500 kbps, and the MCP2515 is set to normal mode.

In the loop() function, the temperature and humidity readings are obtained from the DHT11 sensor using the readTemperature() and readHumidity() functions and stored in variables t and h, respectively. Then, the can_id and can_dlc fields of the can_frame struct are set to 0x072 and 8, respectively. The data[6] and data[7] fields of the can_frame struct are set to the humidity and temperature readings, respectively. Finally, the message is sent using the sendMessage() function of the MCP2515 object and a delay of 1000 milliseconds is added before the next loop iteration.

3.3.3.2 CAN Transmitter for RTC DS1307

```
#include <Wire.h>
#include "RTCLib.h"
#include <SPI.h>
#include <mcp2515.h>

RTC_DS3231 rtc;
struct can_frame canMsg;
MCP2515 mcp2515(10);

void setup()
{
  while (!Serial);
  Serial.begin(115200);
  Wire.begin();
  rtc.begin();
  SPI.begin();
  mcp2515.reset();
  mcp2515.setBtrate(CAN_500KBPS, MCP_16MHZ);
  mcp2515.setNormalMode();
}

void loop()
{
  DateTime now = rtc.now();
  canMsg.can_id = 0x072;
  canMsg.can_dlc = 8;
  canMsg.data[0] = now.hour();
```

```

canMsg.data[1] = now.minute();
canMsg.data[2] = now.second();
canMsg.data[3] = now.day();
canMsg.data[4] = now.month();
canMsg.data[5] = now.year() % 100;

```

```

    mcp2515.sendMessage(&canMsg);
}

```

Explain the code:

The code begins by including the necessary libraries for the RTC module, the MCP2515 CAN bus transceiver module, and the Wire and SPI communication protocols.

In the setup function, the serial communication is initiated, and the RTC module, Wire library, and SPI are also initialized. Then, the MCP2515 module is reset, and its bitrate is set to 500 kbps with a 16MHz clock. Finally, the MCP2515 module is set to normal mode, which allows it to send and receive messages on the CAN network.

In the loop function, the current date and time are obtained from the RTC module using the now variable. A struct can_frame variable canMsg is defined, which contains the data and ID to be transmitted over the CAN network. In this example, the message ID is set to 0x072, and the data length code (DLC) is set to 8 bytes. The data in the message contains the hour, minute, second, day, month, and year.

The last line of the loop function sends the canMsg message over the CAN network using the sendMessage function of the MCP2515 module.

3.3.3.3 CAN Receiver code

```

#include <SPI.h>
#include <mcp2515.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <Wire.h>
#include <RTCLib.h>

#define CS_PIN 10

struct can_frame canMsg;

MCP2515 mcp2515(CS_PIN);

#define LCD_ADDR 0x27
#define LCD_COLS 16
#define LCD_ROWS 2
LiquidCrystal_I2C lcd(LCD_ADDR, LCD_COLS, LCD_ROWS);

```



```

#define DHTPIN 8
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

RTC_DS3231 rtc;

void setup()
{
  while (!Serial);
  Serial.begin(115200);
  SPI.begin();
  lcd.init();
  lcd.clear();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("CANBUS TUTORIAL");
  delay(3000);
  lcd.clear();

  mcp2515.reset();
  mcp2515.setBitrate(CAN_500KBPS, MCP_16MHZ);
  mcp2515.setNormalMode();

  rtc.begin();
}

void loop()
{
  receiveMessages();
  displayData();
}

void receiveMessages()
{
  while (mcp2515.readMessage(&canMsg) == MCP2515::ERROR_OK) {
    if (canMsg.can_id == 0x036) {

      int humidity = canMsg.data[6];
      int temperature = canMsg.data[7];

      Serial.print("Received Data from DHT11: Humidity=");
      Serial.print(humidity);
      Serial.print(" Temperature=");
      Serial.println(temperature);

      lcd.setCursor(0, 0);
      lcd.print("Humi:");
      lcd.print(humidity);
    }
  }
}

```

```

    lcd.setCursor(9, 0);
    lcd.print("Temp:");
    lcd.print(temperature);
} else if (canMsg.can_id == 0x072) {

    int hour = canMsg.data[0];
    int minute = canMsg.data[1];
    int second = canMsg.data[2];

    Serial.print("Received Data from DS3231: ");
    Serial.print(hour);
    Serial.print(":");
    Serial.print(minute);
    Serial.print(":");
    Serial.println(second);

    lcd.setCursor(0, 1);
    lcd.print("Time: ");
    lcd.print(hour);
    lcd.print(":");
    lcd.print(minute);
    lcd.print(":");
    lcd.print(second);
}
}
}

```

Explain the code:

The code begins with including the necessary libraries: `SPI.h` for communication with the MCP2515, `mcp2515.h` for controlling the MCP2515, `LiquidCrystal_I2C.h` for controlling the LCD display, and `RTCLib.h` for communicating with the RTC module.

Next, some variables are defined: `lcd` for controlling the LCD display, `rtc` for controlling the RTC module, and a struct `can_frame` for storing the received CAN message.

In the `setup()` function, the serial communication is initialized, SPI is started, the LCD display is initialized and turned on, the message "CANBUS TUTORIAL" is displayed on the LCD for 3 seconds, the MCP2515 is reset, the bit rate of the CAN bus is set to 500 kbps, and the MCP2515 is set to normal mode. Finally, the RTC module is initialized.

In the `loop()` function, the `readMessage()` function of the MCP2515 object is used to read a CAN message. If a message is received successfully, the data from the message is extracted and stored in corresponding variables. The humidity and temperature readings are stored in variables `x` and `y`, respectively. The hour, minute, and second of the current time are obtained from the RTC module using the `now()` function of the `DateTime` object.

3.4 Instrument cluster wiring diagram



-

Techometer

Here are some the role of the tachometer:

- 40

is overheating, the tachometer can be used to see if the engine is running too high in RPMs.

- Monitor engine performance. The tachometer can be used to monitor engine performance over time. This can be helpful for tracking things like fuel economy and engine wear and tear.

ABS warning lamp



If it illuminates when you are driving, this indicates a malfunction. You will continue to have normal braking (without ABS). Have the system checked by a properly trained technician as soon as possible.

Frost warning lamp



It will illuminate and glow orange when the outside air temperature is between 4°C (39°F) and 1°C (34°F). It will glow red when the temperature is below 1°C (34°F).

Airbag warning lamp



If it illuminates when you are driving, this indicates a malfunction. Have the system checked by a properly trained technician.

Brake system lamp



It illuminates when the parking brake is engaged.



WARNING: Reduce your speed gradually and stop your vehicle as soon as it is safe to do so. Use your brakes with care.

If it illuminates when you are driving, check that the parking brake is not engaged. If the parking brake is not engaged, this indicates a malfunction. Have the system checked by a properly trained technician immediately.

Cruise control indicator



It will illuminate when you have set a speed using the cruise control system.

Direction indicators



Flashes during operation. A sudden increase in the rate of flashing warns of a failed indicator bulb. See [Changing a Bulb](#).

Doors open warning lamp



Illuminates when the ignition is switched on and remains on if any door, the bonnet or the luggage compartment is not closed properly.

Engine warning lamp



If it illuminates with the engine running, this indicates a malfunction. If it flashes when you are driving, **reduce the speed of your vehicle immediately**. If it continues to flash, avoid heavy acceleration or deceleration. Have the system checked by a properly trained technician immediately.



CAUTION: If the engine warning lamp illuminates in conjunction with a message, have the system checked as soon as possible.

Front fog lamp indicator



It will illuminate when you switch the front fog lamps on.

Glow plug indicator



See Starting a diesel engine .

Headlamp indicator



It will illuminate when you switch the headlamp dipped beam or the side and tail lamps on.

Ignition warning lamp



If it illuminates when you are driving, this indicates a malfunction. Switch off all unnecessary electrical equipment. Have the system checked by a properly trained technician immediately.

Low fuel level warning lamp



If it illuminates, refuel as soon as possible.

Main beam indicator



It will illuminate when you switch the headlamp main beam on. It will flash when you use the headlamp flasher.

Message indicator



It will illuminate when a new message is stored in the information display. See Information messages .

Oil pressure warning lamp



If it stays on after starting or illuminates when driving, this indicates a malfunction. Stop your vehicle as soon as it is safe to do so and switch the engine off. Check the engine oil level. See Engine oil check .

Power steering warning lamp



It illuminates to indicate a malfunction of the power steering system. Full steering will be maintained but you will need to exert greater force on the steering wheel. Have the system checked by a properly trained technician as soon as possible.

Rear fog lamp indicator



It will illuminate when you switch the rear fog lamps on.

Seat belt reminder



See Seat belt reminder .

Shift indicator



It will illuminate to inform you that shifting to a higher gear may give better fuel economy and lower CO2 emissions. It will not illuminate during periods of high acceleration, braking or when the clutch pedal is pressed.

Stability control (ESP) indicator



While driving, it flashes during activation of the system. After switching on the ignition, if it does not illuminate or illuminates continuously while driving, this indicates a malfunction. During a malfunction, the system switches off. Have the system checked by a properly trained technician as soon as possible.

If you switch ESP off, the warning lamp will illuminate. The lamp will go out when you switch the system back on or when you switch the ignition off.

3.5 HS CAN ID and MS CAN ID

3.5.1 Code to read CAN ID

```
#include <SPI.h>
#include <mcp2515.h>
struct can_frame canMsg;
MCP2515 mcp2515(10);
void setup() {
  Serial.begin(115200);
  mcp2515.reset();
  mcp2515.setBaudrate(CAN_500KBPS);
  mcp2515.setNormalMode();

  Serial.println("----- CAN Read -----");
  Serial.println("ID  DLC  DATA");
}
void loop()
{
  if (mcp2515.readMessage(&canMsg) == MCP2515::ERROR_OK)
  {
    Serial.print(canMsg.can_id, HEX);
    Serial.print(" ");
    Serial.print(canMsg.can_dlc, HEX);
    Serial.print(" ");

    for (int i = 0; i<canMsg.can_dlc; i++)
    {
      Serial.print(canMsg.data[i],HEX);
      Serial.print(" ");
    }
    Serial.println();
  }
}
```


Explain the code:

The code begins with including the necessary libraries: SPI.h for communication with the MCP2515 and mcp2515.h for controlling the MCP2515. A struct can_frame is defined for storing the received CAN message, and an MCP2515 object is created.

In the setup() function, the serial communication is initialized, the MCP2515 is reset, the bit rate of the CAN bus is set to 500 kbps, and the MCP2515 is set to normal mode. The message "----- CAN Read -----" and "ID DLC DATA" are printed on the serial monitor.

In the loop() function, the readMessage() function of the MCP2515 object is used to read a CAN message. If a message is received successfully, its contents are displayed on the serial monitor using the print() function of the Serial object. The message ID, DLC (data length code), and data bytes are printed in hexadecimal format, separated by spaces. A for loop is used to print each data byte in the message. Finally, a new line is printed to separate each message.

3.5.2 Hardware settings

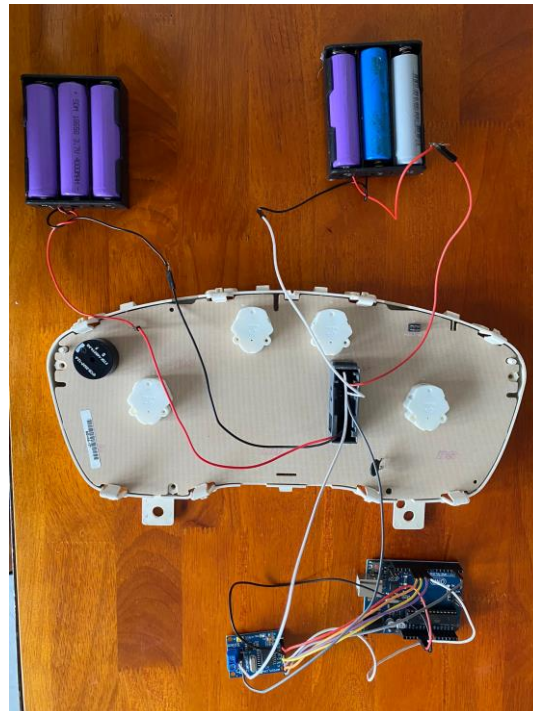


Figure 3.9: Power supply and connecting between dashboard and module CAN



Figure 3.10: Active instrument cluster

3.5.3 Ford Focus 3M5F-10841-B instrument cluster'S HS CAN ID

HS CAN MESSAGES
404 0
0 0
400 0
0 0
9E3FF08F 0
400 0
4 0
91FC8080 0
9FFFFFFF 0
400 0
500 0
0 0
0 0
9FFF83FE 0
38 0
0 0
0 0
400001FC 3 FF FF FF
9E17C000 0
7FF 0
0 0
700 0
9F17FC00 0
0 0
90028000 0
9F1FF800 0
9FFFFFFF 0 6 0 0 0 0 0
5FF 0
9E3FE0FF 8 0 0 0 0 0 0
400 0

0 0
7E0 0
9FFEFFFF 8 0 0 0 0 0 0 0
9FE3FEBF 8 0 0 0 0 0 0 0
400 0
40000404 7 FF FF FF FF FF FF FF
3E 0
400 0
0 0
3E 0
0 0
0 0
7C0 0
9FD7F000 0

3.5.4 Ford Focus 3M5F-10841-B instrument cluster'S MS CAN ID

MS CAN MESSAGES
0 0
9FFFFFFF 0
7C4 0
404 0
400 0
0 0
400 0
0 0
404 0
9FFFFFFF 8 0 0 0 0 0 0 0
9FFFFFFF 8 0 0 0 0 0 0 0
0 0
9FFFFFFF 7 0 0 0 0 0 0 0
0 0
500 0
0 0
9C7FE08F 0
0 0
400 0
7FF 0
700 0
4 0
400 0
9FFFFFFF 3 0 0 0
4 0
784 0
400 0
9FFC8FFC 0
9FD7F883 8 0 0 0 0 0 0 0
0 0

9FFFFFFF 8 0 0 0 0 0 0 0
0 0
0 0
4 0
400 0
0 0
9F7FF8BF 8 0 0 0 0 0 0 0
907F803F 0
7F 0
1FF 0
0 0
0 0
D012E01F 8 FF FF FF CF FF FF FF FF
4 0

3.5.5 FORD FOCUS 2018'S MS CAN ID

MS CAN messages
0 7 0 F 7 F F 1 F F
0 7 0 3 7 F F F F
0 7 3 7 1 F F 1 F F 1 F
18 7 0 3 7 F F F F
18 7 1 1 7 7 F 7 F
0 0
0 0
0 0
0 0
0 0
0 0
0 3 0 0 1 F
0 0
0 0
0 0
0 0
78 0
0 0
8 0
0 0
18 0
38 0
0 0
40000000 0
0 0
78 0
81E30F00 0
40000000 0
87E70000 3 3 F 0 3

87E70003 7 3F 0 3 7 F F F
C7E71F00 0
C7E40307 7 7F 7F 7F 3 1F 3F 3F
40000060 0
C7E73F3F 0
81E31F00 0
81E30F00 0
81E30F00 0
81E30F00 0
18 7 F F F F F F F
0 0
40000000 0
0 7 1 3 7 3 F 7 F
8 7 1 3 7 F F F F
0 7 1 3 7 7 7 F F
0 7 1 3 7 7 F 7 F
8 7 3 7 7 F F F F
0 7 3 7 7 F F F F
8 7 1 7 7 F F F F
18 7 3 7 7 F F F F
8 7 3 7 F F F F F
40000018 3 3 F F

3.5.6 FORD FOCUS 2018'S HS CAN ID

HS CAN messages (engine is on)
4000052D 8 FF FF FF FF FF FF FF FF
0 0
4000052D 8 FF FF FF FF FF FF FF FF
4000052D 8 FF FF FF FF FF FF FF FF
40000607 0
40000000 0
9FFFFFF80 0
40000000 0
40000000 0
0 0
9FFFFFF00 0
0 0
0 0
40000000 0
9FFFFFFE 0
0 0
9FFF0000 0
801FFFFFF 0
DFFFFFFE 8 FF FA F0 F0 E0 E0 E0 E0

HS CAN messages (engine is off)
77F 4 A6 AF BB BD
DFF79F80 0
DFFFFFFC 0
DFFFFFF80 3 FF FF FF
40000504 0
40000404 0
4 0
DFFFAFAA 1 FF
404 0
90128080 0
404 0
404 0
0 7 1 3 7 7 F F F
5FC 0
404 0
9FFFAF95 8 FF BF AA FF FF AF AB BF
40000404 0
DFF78A82 0
97FFFFFF 0

3.5.7 Code to transmit CAN ID into the dashboard

```
#include <SPI.h>
#include <mcp_can.h>

MCP_CAN CAN(10);

void setup() {
  Serial.begin(9600);
  while (!Serial) { } // Wait for Serial Monitor to open
  SPI.begin();
  if (CAN.begin(MCP_ANY, CAN_500KBPS, MCP_16MHZ) == CAN_OK) {
    Serial.println("MCP2515 Initialized Successfully!");
    CAN.setMode(MCP_NORMAL);
  } else {
    Serial.println("Error Initializing MCP2515...");
  }
}

void loop() {

  unsigned char data1[8] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
  unsigned short canId1 = 0x4000052D;
  byte sndStat1 = CAN.sendMsgBuf(canId1, 0, 8, data1);
  if (sndStat1 == CAN_OK) {
    Serial.println("Message Sent Successfully - CAN ID 0x4000052D!");
  } else {
    Serial.println("Error Sending Message - CAN ID 0x4000052D...");
  }
  delay(1000);
}
```

Explain the code:

The sketch starts by including the necessary libraries: SPI.h and mcp_can.h. The SPI library is used to communicate with the MCP2515 CAN controller, and the mcp_can library provides an easy-to-use interface for sending and receiving messages on the CAN bus.

In the setup function, the sketch initializes the Serial communication at a baud rate of 9600 and waits for the Serial Monitor to open. It then initializes the SPI communication and checks whether the MCP2515 CAN controller was successfully initialized with the MCP_ANY argument, which allows the controller to listen to any CAN bus message. The CAN_500KBPS argument sets the CAN bus speed to 500 kbps, and the MCP_16MHZ argument sets the clock speed of the MCP2515 to 16 MHz. If the initialization is successful, the sketch sets the CAN controller to normal mode using the CAN.setMode(MCP_NORMAL) function. If there is an error during initialization, the sketch outputs an error message.

In the loop function, the sketch sends a CAN message with an 8-byte data payload and a CAN ID of 0x4000052D. The sendMsgBuf function of the MCP_CAN library is used to send the message, and it returns a byte value indicating whether the message was sent successfully. If the message is sent successfully, the sketch outputs a success message to the Serial Monitor. If there is an error sending the message, the sketch outputs an error message.

The sketch then delays for 1 second using the delay function before repeating the process of sending the CAN message. This creates a loop that sends the same CAN message every second.

3.5.8 The result on Arduino IDE

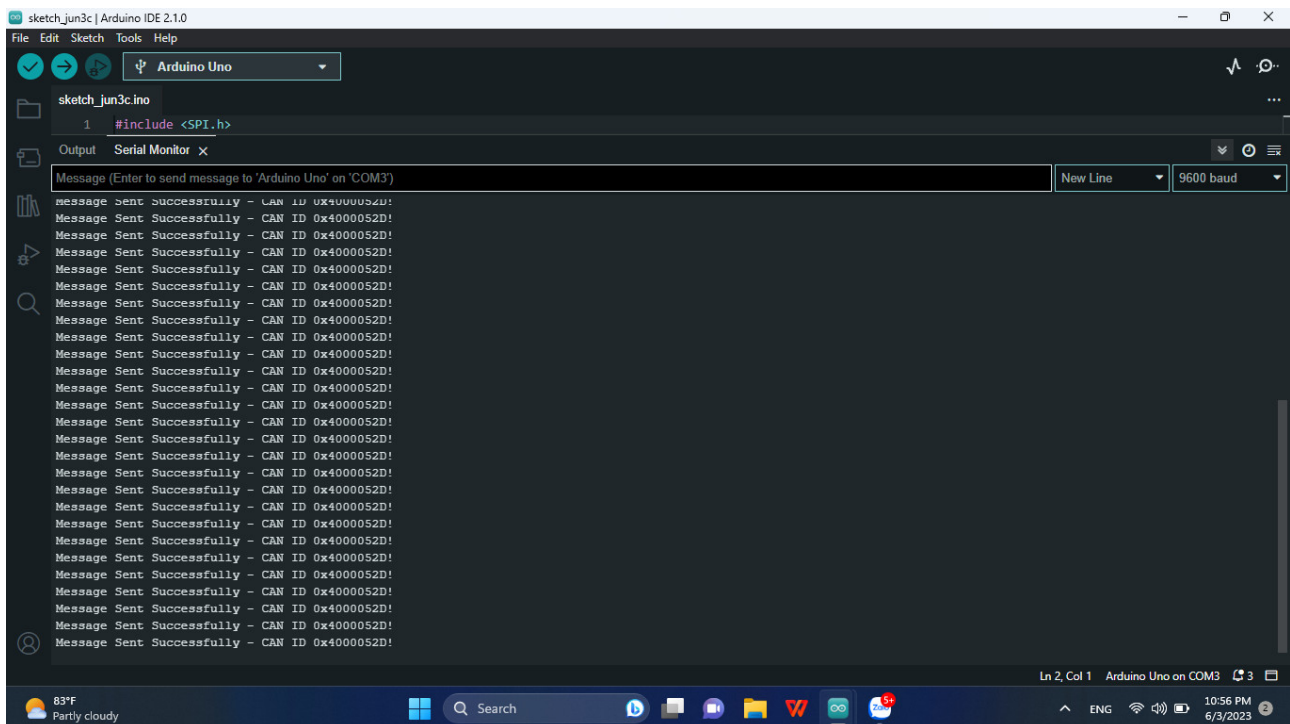


Figure 3.11: The result on serial monitor

Conclusion

We have successfully accomplished our initial goal of studying the structure and principles of the CAN network, its practical applications, and conducting data transmission experiments using the CAN MCP2515 module. After referring to various sources of literature, we immediately applied the theories we just learned to the task of setting IDs for messages transmitted on the CAN network.

Through the research on "Study about the CAN Network protocol in Automotive system," we have gained a better understanding of the concept of communication networks, specifically the CAN network in automobiles, which is a widely applied communication trend. The CAN network serves as a foundation for us to further explore its applications in industrial environments, where interconnected machines utilize the CAN bus for communication.

Reference

- [1] CRC Press, 1st edition (December 20, 2008), Automotive Embedded Systems Handbook (Industrial information technology)
- [2] Dominique Paret (11 May 2007), Multiplexed Networks for Embedded Systems: CAN, LIN, Flexray, Safe-by-Wire...
- [3] Karl Henrik Johansson, Martin Torngren, and Lars Nielsen, (2005), Vehicle Applications of Controller Area Network
- [4] CSS ELECTRONICS (2003), CAN Bus Explained – A Simple Intro, 02/03/2023, From< <https://www.csselectronics.com/pages/can-bus-simple-intro-tutorial>>
- [5] Nicolas Navet, Yeqiong Song, Françoise Simonot-Lion, and Cédric Wilwert, (2005), Trends in Automotive Communication Systems.
- [6] Olaf Pfeiffer, Andrew Ayre and Christian Keydel, (November 2003), Embedded Networking with CAN and CANopen. "This book offers a deep dive into the theory and practice of CAN"