

MEDS504L: IN VEHICLE NETWORKING

REPORT

ON

FLEET MANAGEMENT AND TRACKING SYSTEM

Team PROTOCAN

Abishek R – 22MES0011

Ashwin Hariharan R – 22MES0025

Aadhitya S V – 22MES0007

Ezhil Subbia K – 22MES0006

Kevin Abishek S – 22MES0041

Fleet management and tracking system

Abstract

Fleet management has become one of the top business requirements over a better part of a decade. Goods are transported across large distance through various modes of transport. The type of goods varies from simple good to hazardous chemicals and nuclear substances. So, It is very much necessary and important to mange the goods during transportation in order to ensure the safety criteria. The growth of science and technology is phenomenal in all aspects of life. This scientific development could be incorporated for managing the goods during transportation. The vehicles are getting smarter with introduction of various new sensors to the containers and also communication technology is being advanced to offer high speed services. These all could be effectively used for fleet management system. This project aims in utilizing one of the widely used automotive protocol i.e., the CAN protocol for in vehicular communication in management of the fleet. The data which are collected through various sensors are communicated through the CAN bus which is responsible of transmitting the information to the microcontroller which in turn sends it to cloud. The data received in the cloud could be used for analysing various parameters and proper warning mechanism could be implemented and thus effectively managing the fleet.

Introduction

Fleet management covers everything from procurement of vehicles till environment impact, driving pattern and safety compliance. Various loss can occur during transportation of goods which can eventually affect the entire business, for example when perishable items are not maintained in appropriate temperature during its movement then at time of reaching destination it may not be suitable for consumption which can affect the business and incur loss to the buyer. These kind of monetary loss as well as casualties which occurs during transportation can be avoided with proper fleet management system.

A fleet management system is a set of tools and technologies used to track and manage vehicles and other mobile assets. These systems typically include GPS tracking devices, telematics devices, and software that can be used to monitor the location, speed, and status of vehicles in real-time. Some fleet management systems also include features such as route optimization, driver behaviour monitoring, and maintenance tracking. The goal of a fleet management system is to improve efficiency, reduce costs, and increase safety and compliance.

One of the key benefits of fleet management systems is that they can help companies to improve efficiency and reduce costs. For example, by using real-time tracking data, companies can optimize routes for their vehicles, which can help to reduce fuel costs and increase the productivity of their drivers. Additionally, fleet management systems can help to improve safety and compliance by providing alerts when vehicles are being driven outside of safe parameters, and by providing detailed reports on driver behaviour.

Another important aspect of fleet management systems is that they can be used to track and manage maintenance and repairs for vehicles. This can help companies to ensure that their vehicles are always in good working order and to reduce the costs of repairs and downtime.

Overall, a fleet management system is a valuable tool for companies that rely on vehicles and other mobile assets. By providing real-time visibility and control over their fleet, these systems can help companies to improve efficiency, reduce costs, and increase safety and compliance.

Methodology

A. Components Used

Microcontrollers:

1) Arduino Nano - ATmega 328P (used in Node 1):

- 8-bit 16MHz processor
- Least powerful processor of all nodes
- Only used to forward GPS data through CAN bus (GPS has it's own ARM processor for running algorithms)
- 5V logic
- No inbuilt CAN controller

2) Arduino nano 33 BLE sense – UBLOX ARM Cortex M4 (used in Node 2):

- 32-bit 64MHz processor
- Second power processor of all nodes
- Embedded with IMU, temperature, humidity, pressure sensors
- All algorithms for calculating sensor readings and driving patterns are done on edge.
- 3.3V logic
- No inbuilt CAN controller

3) NodeMCU ESP32 - Tensilica Xtensa LX6 (used in Node 3):

- 32-bit 240 MHz dual core processor
- Most powerful processor of all nodes
- Collects data from all nodes and forwards it to dashboard via secured HTTPS API
- Secondary data storage via SPIFFS
- Inbuilt NXP SJA1000 CAN controller

Sensors:

4) LSM9DS1, LPS22HB, HTS221:

- Industrial grade sensors from STMicroelectronics
- Supports both SPI and I2C
- Uses I2C in Arduino nano 33 BLE sense board
- Factory calibrated

- Temperature compensated

CAN Controller:

5) MCP2515:

- Standalone CAN controller with SPI interface
- AEC-Q100 qualified chip
- Automotive grade manufactured by MicroChip
- CAN 2.0B

CAN Transceiver:

6) TJA1050:

- High speed CAN transceiver with 1Mbaud speed
- Compatible with the ISO 11898 standard
- Manufactured by NXP

7) SN65HVD230:

- High speed CAN transceiver with 1Mbaud speed
- Compatible with the ISO 11898 standard
- Manufactured by TI
- Compatible with 3.3V logic

GPS:

8) UBLOX NEO 6-M:

- Standalone GPS with inbuilt compute engine
- Cold start
- UART
- AEC-Q100 qualified chip
- ISO 16750 Certified (Road vehicles - Environmental conditions and testing for electrical and electronic equipment)

Power:

8) LM2596:

- Step down 150 KHz synchronous DC-DC buck converter from TI
- 4.5V to 40V

- 3A
- Can withstand constant voltage fluctuations

B. Working

The system is implemented using three nodes where the first and second node transmits the require data and the third node receives and send it over the cloud. The GPS sensor in node 1 sends the corresponding latitude and longitude via the CAN bus. The second node is responsible for sending other crucial features though various sensors. The various features executed are

- Uses GPS tracking to monitor the activity of fleet vehicles and assets.
- Reliable and scalable data collection for vehicle trackers and embedded sensors.
- Advanced and flexible IoT data visualization for both real-time and historical vehicle data.
- Customizable end-user dashboards to share data from the vehicle tracking system with end users and customers
- Driver driving patterns monitoring – sudden acceleration, deceleration, braking, etc.
- Monitoring of critical payload environment – temperature, pressure, humidity, etc

Third node is responsible for receiving all the data transmitted by the other nodes via the CAN bus. This node comprises of a wifi module which then transmits the data over the cloud to the dashboard where further analysis is being performed. The CAN bus is depicted in figure1.

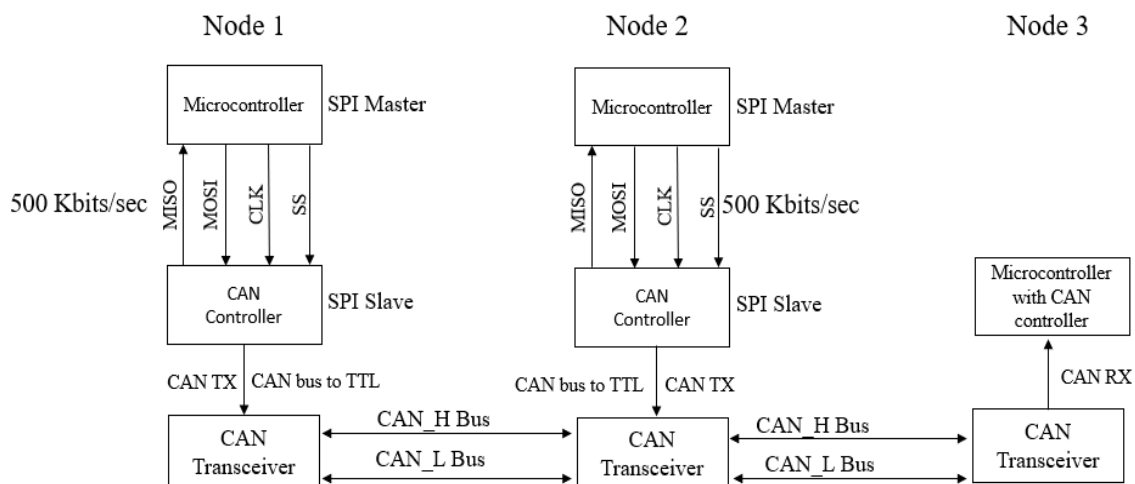


Figure 1: CAN communication

CAN 2.0 B protocol is used as the communication protocol. With CAN 2.0 B specification both 11 bit and the extended 29-bit identifier can be used. With a 11-bit identifier, it is possible to generate 2048 unique CAN frames, which is more than sufficient for this project. Additionally, using a 11-bit identifier also decreases the bandwidth usage of CAN bus. By default, latitude and longitude data has the highest priority frame has the highest priority. If any one of the warning conditions is achieved, the CAN identifier is swapped giving the warning data highest priority temporarily. The identifier is swapped back to original after the warnings subsides.

The various threshold criteria for different sensors are depicted in table1.

Table 1. Threshold values for sensors

Parameters	Threshold limit
Temperature	temp> 35.00 C
Humidity	hum> 90.00
Pressure	pres> 150.00
Sudden break	x<-1.00g
Sudden acceleration	x>1.00g
Collision	x>2.00g or y<2.00g
Roll over	x1>200 or y1>200 or z>200 x1<-200 or y1<-200 or z1<-200
Oversteer	y>1.0 g or y<-1.0 g

When any of the sensor values crosses over the above-mentioned threshold limit then warning are raised. This warning mechanism executed in the project through three means, first by a email notification, second by an in app notification and the last by a warning sound. The owner has the access to both the dashboard and the app, while the driver has the access to the app which can be installed onto to the vehicle's existing infotainment system. Various warnings have been displayed in the app which includes temperature, pressure, humidity, Vehicle roll over, sudden acceleration, sudden breaking, EMF warning. For example, when temperature crosses above the threshold an immediate mail and In-App notification would be sent to the owner as well as an alert sound would be turned on and alerts the driver. Similarly, when any parameters vary over the desired limit the appropriate warning would be raised through the alert sound, an email notification and through the indicator in mobile dashboard.

Results and Discussion

Project Demo Link: <https://www.youtube.com/watch?v=n0wHwar0drA>

A total of 7 CAN frames has been sent to the receiver with the total bandwidth of 713 bits/second. A standard CAN 2.0B frame is depicted in figure2. The CAN bus speed was set to 500 kbps. Therefore, the total bandwidth usage is just 0.13% of the allocated bus.

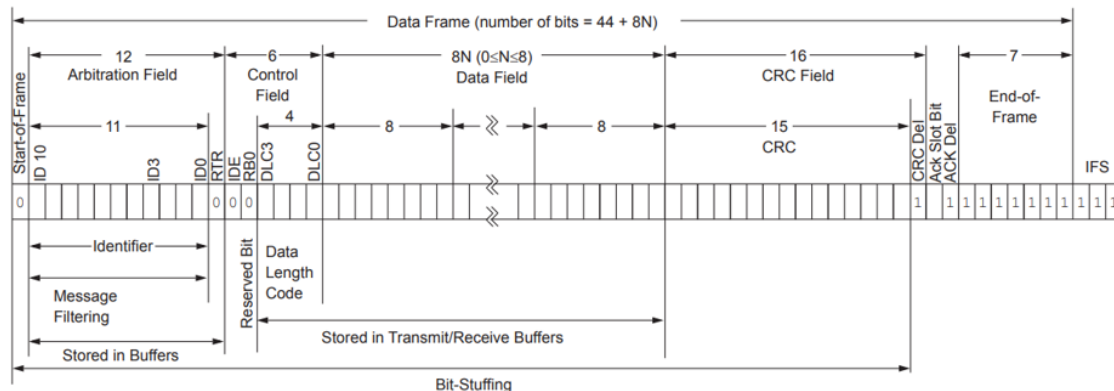


Figure 2. CAN data frame

A web dashboard was created to visualize the data collected from the fleets. The sample dashboard is represented in figure3. The dashboard displays all the data and gets updated every second. There is also a provision to view the historic data collected from the fleet. It contains various indicators to indicate warning, if any value crosses the limit then the indicator turn on and alerts the end user. Similarly, warnings would also be raised in the app installed onto the fleet to alert the drivers.

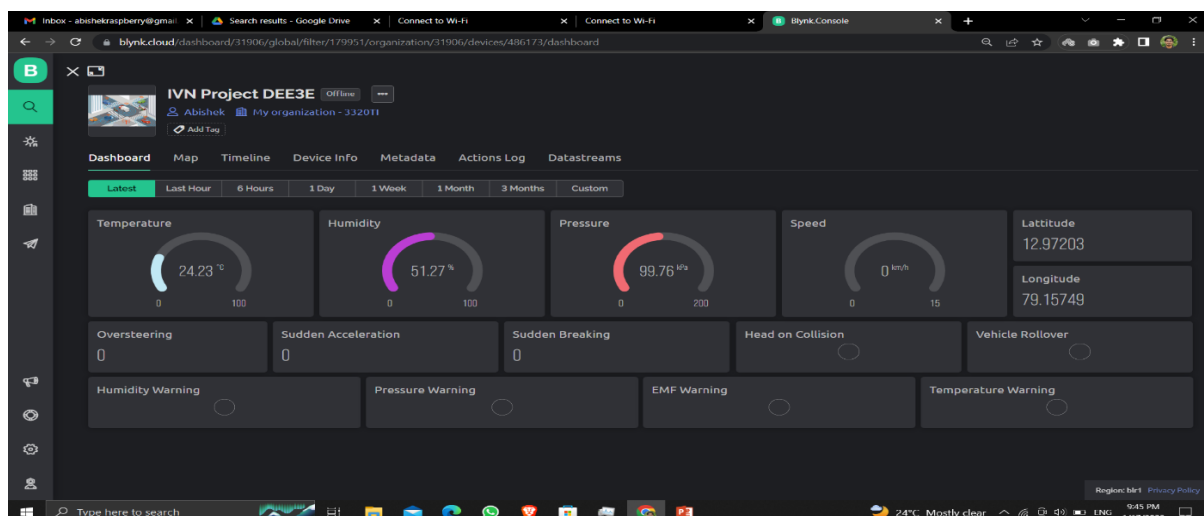


Figure 3. Web dashboard

The dashboard also contains a feature to track the current position of the fleet. The entire journey could be visualized along with the parameters variations over the journey based on the color scale for each parameter. For example, the location of the fleet along with the pressure variation is depicted in the figure 4. Likewise all the parameters could be viewed over the journey so that it helps in finding out at what exact location variations has occurred in the fleet which eventually leads to finding out the root cause of the problem occurred if any.

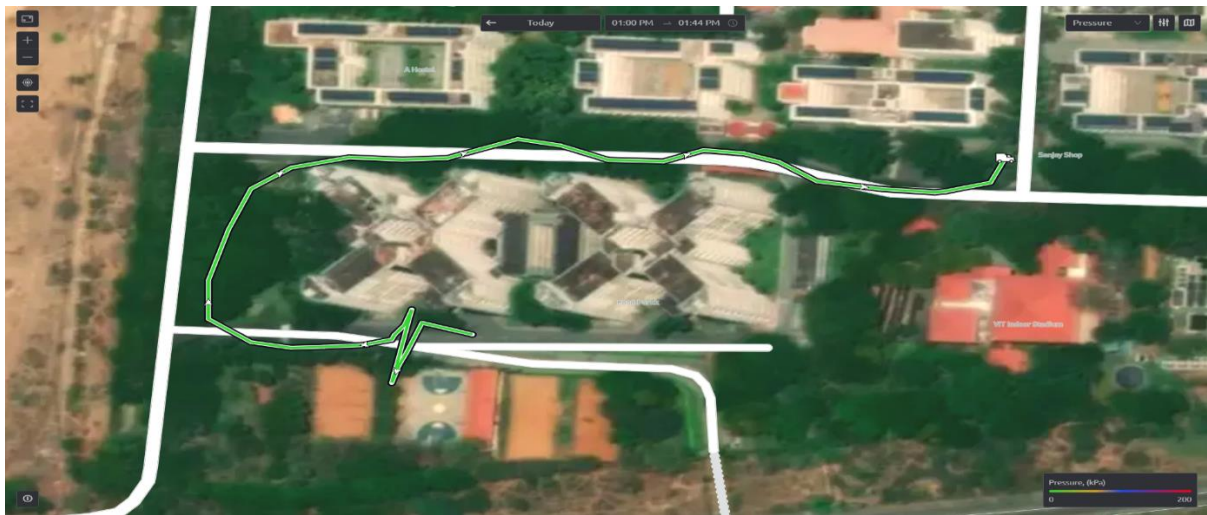


Figure 4. Tracking of fleet

Conclusion

The project is successfully executed with creating a complete fleet management system using CAN protocol. A web dashboard and an application was created for this purpose which helps in visualizing and analyzing data. Various warning mechanisms has been incorporated to the project so that an efficient and robust management of fleets could be performed. This project could be implemented to any vehicles for the sole purpose of management and tracking. Thus, management of fleets which is being one of the crucial operations in transportation is implemented with the help of automotive protocol.

In Vehicle Networking Project

Fleet Tracking and Monitoring System

Team ProtoCAN

Review 2

Presented By:

Abishek R – 22MES0011

Ashwin Hariharan R – 22MES0025

Aadhitya S V – 22MES0007

Ezhil Subbia K – 22MES0006

Kevin Abishek S – 22MES0041

Features Executed

- Uses GPS tracking to monitor the activity of fleet vehicles and assets.
- Reliable and scalable data collection for vehicle trackers and embedded sensors.
- Advanced and flexible IoT data visualization for both real-time and historical vehicle data.
- Customizable end-user dashboards to share data from the vehicle tracking system with end users and customers
- Driver driving patterns monitoring – sudden acceleration, deceleration, braking, etc.
- Monitoring of critical payload environment – temperature, pressure, humidity, etc

Warning Mechanism Used

- The owner has the access to both the dashboard and the app, while the driver has the access to the app which can be installed onto to the vehicle's existing infotainment system.
- Various warnings have been displayed in the app which includes temperature, pressure, humidity, Vehicle roll over, sudden acceleration, sudden breaking, EMF warning.
- For example, when temperature crosses above the threshold an immediate mail and In-App notification would be sent to the owner as well as an alert sound would be turned on and alerts the driver. Similarly, when any parameters varies over the desired limit the appropriate warning would be raised through the alert sound, an email notification and through the indicator in mobile dashboard.

Dashboard

B

×

☰

☼

☰

📄

📌


📢

⚙️

👤

×

🖼️



IVN Project DEE3E Offline ⋮

👤 Abishek

📁 My organization - 3320TI

Add Tag

Dashboard

Map

Timeline

Device Info

Metadata

Actions Log

Datastreams

Latest

Last Hour

6 Hours

1 Day


1 Week

1 Month

3 Months

Custom


Temperature



24.23 °C

0100


Humidity



51.27 %

0100


Pressure



99.76 kPa

0200

Speed



0 km/h

015

Latitude

12.97203

Longitude

79.15749

Oversteering

0


Sudden Acceleration

0


Sudden Breaking

0


Head on Collision




Vehicle Rollover




Humidity Warning




Pressure Warning



EMF Warning




Temperature Warning



Region: blr1

Privacy Policy

Type here to search



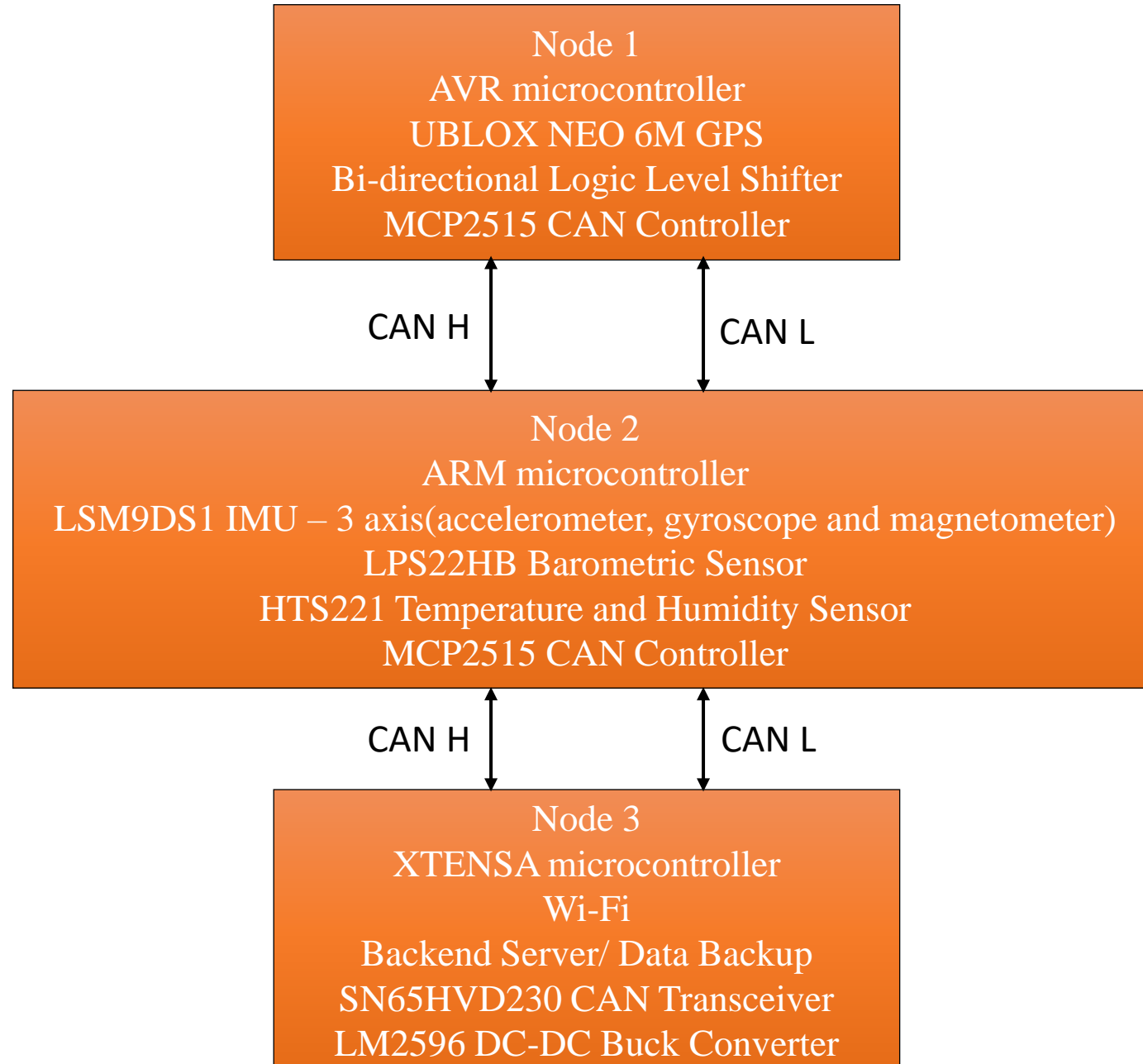
24°C Mostly clear

9:45 PM 1/17/2023

Dashboard



CAN Nodes



Components Justification

Microcontrollers:

1) Arduino Nano - ATmega 328P (used in Node 1):

- 8-bit 16MHz processor
- Least powerful processor of all nodes
- Only used to forward GPS data through CAN bus (GPS has it's own ARM processor for running algorithms)
- 5V logic
- No inbuilt CAN controller

2) Arduino nano 33 BLE sense – UBLOX ARM Cortex M4 (used in Node 2):

- 32-bit 64MHz processor
- Second power processor of all nodes
- Embedded with IMU, temperature, humidity, pressure sensors
- All algorithms for calculating sensor readings and driving patterns are done on edge.
- 3.3V logic
- No inbuilt CAN controller

3) NodeMCU ESP32 - Tensilica Xtensa LX6 (used in Node 3):

- 32-bit 240 MHz dual core processor
- Most powerful processor of all nodes
- Collects data from all nodes and forwards it to dashboard via secured HTTPS API
- Secondary data storage via SPIFFS
- Inbuilt NXP SJA1000 CAN controller

Components Justification

Sensors:

4) LSM9DS1, LPS22HB, HTS221:

- Industrial grade sensors from STMicroelectronics
- Supports both SPI and I2C
- Uses I2C in Arduino nano 33 BLE sense board
- Factory calibrated
- Temperature compensated

CAN Controller:

5) MCP2515:

- Standalone CAN controller with SPI interface
- AEC-Q100 qualified chip
- Automotive grade manufactured by MicroChip
- CAN 2.0B

CAN Transceiver:

6) TJA1050:

- High speed CAN transceiver with 1Mbaud speed
- Compatible with the ISO 11898 standard
- Manufactured by NXP

Components Justification

6) SN65HVD230:

- High speed CAN transceiver with 1Mbaud speed
- Compatible with the ISO 11898 standard
- Manufactured by TI
- Compatible with 3.3V logic

GPS:

7) UBLOX NEO 6-M:

- Standalone GPS with inbuilt compute engine
- Cold start
- UART
- AEC-Q100 qualified chip
- ISO 16750 Certified (Road vehicles - Environmental conditions and testing for electrical and electronic equipment)

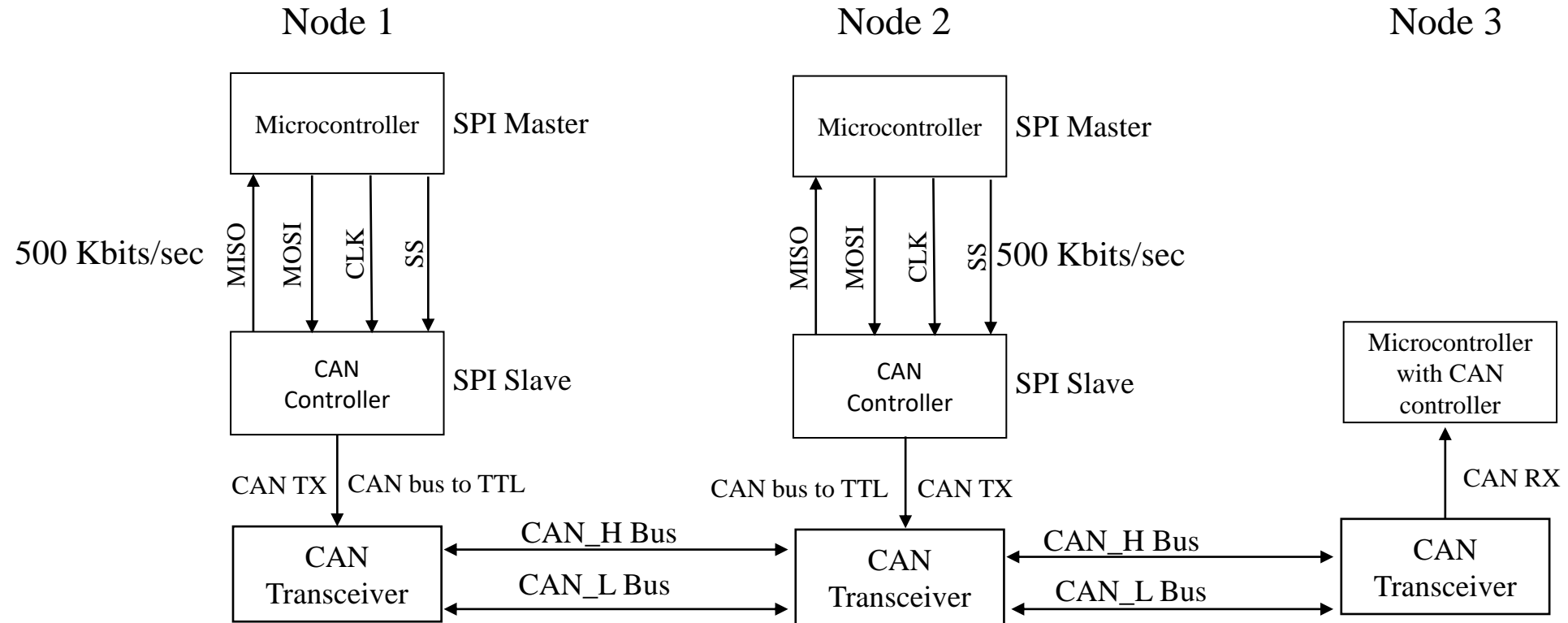
Power:

8) LM2596:

- Step down 150 KHz synchronous DC-DC buck converter from TI
- 4.5V to 40V
- 3A
- Can withstand constant voltage fluctuations

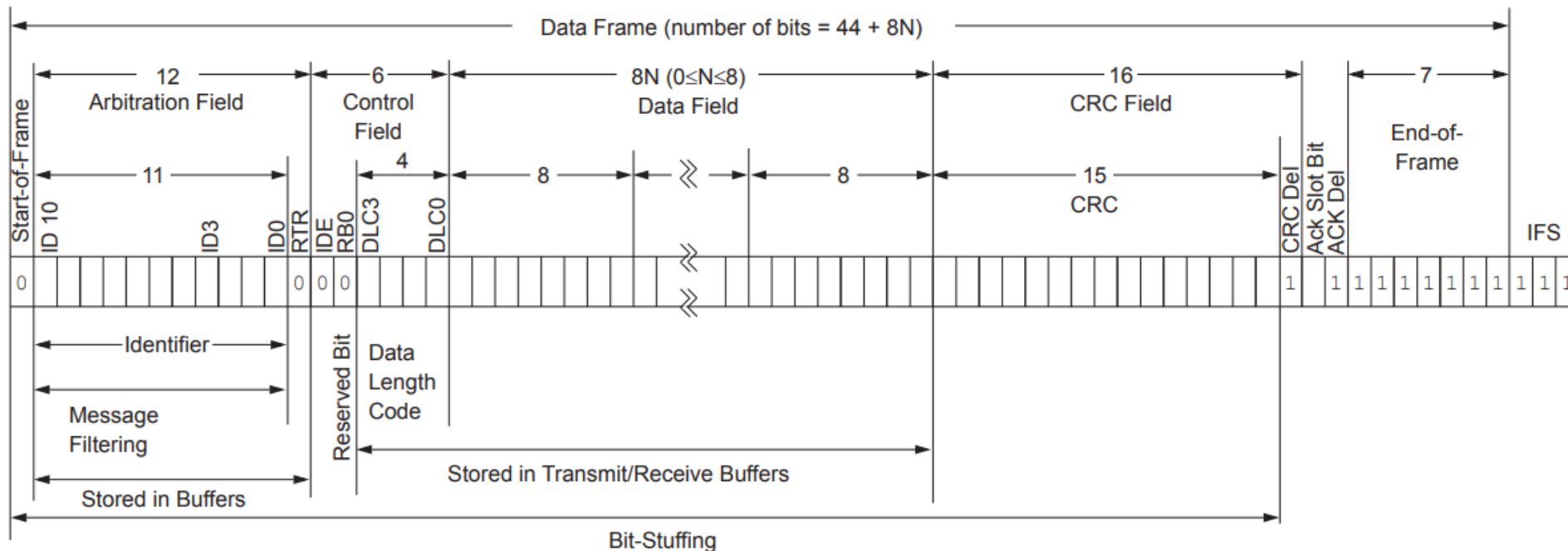


Block diagram for CAN communication



Why CAN 2.0B with 11 bit identifier

- With CAN 2.0B specification both 11 bit and the extended 29 bit identifier can be used.
- With a 11-bit identifier, it is possible to generate 2048 unique CAN frames, which is more than sufficient for this project.
- Additionally, using a 11-bit identifier also decreases the bandwidth usage of CAN bus. (7 frames)



Bandwidth Usage

Node 1:

3 CAN frames – 301 bits/sec (8,8,4)

Node 2:

4 CAN frames – 412 bits/sec (8,8,8,4)

CAN bus speed set – 500 kbps

Total Bandwidth usage – 713 bits/sec

Total bandwidth utilization – 0.13%

Rest of the bandwidth will be used by other vehicular communication

Microcontroller Architecture Issues

Microcontroller INT datatype handling:

AVR – 2 bytes

ARM – 4 bytes

XTENSA – 3 bytes (signed int by default)

CAN controller can transmit data but the receiver cannot understand the data due to size difference

Microcontroller CHAR datatype handling:

AVR – 1 byte

ARM – 1 byte

XTENSA – 1 byte

Data is transferred via character array

Our own proprietary operating system will deliver an extraordinary customer experience

MB.OS

Compelling luxury experience
Speed of execution
Higher customer loyalty

Scalable, lower variable costs
Lower complexity
Recurrent revenues

CAN Frame Priority

- By default latitude and longitude data has the highest priority.
- If any one of the warning conditions is achieved, the CAN identifier is swapped giving the warning data highest priority temporarily.
- The identifier is swapped back to original after the warnings subsides.

Thank You