Upravljački sustav za autonomne robote s bežičnom i CAN komunikacijom

Shukla, Adamya

JOSIP JURAJ STROSSMAYER UNIVERSITY OF OSIJEK
FACULTY OF ELECTRICAL ENGINEERING, COMPUTING AND INFORMATION TECHNOLOGY

GRADUATE STUDY

CONTROL SYSTEM FOR AUTONOMOUS ROBOTS WITH WIRELESS AND CAN COMMUNICATION

GRADUATION THESIS

ADAMYA SHUKLA

OSIJEK, 2023

FERIT
FACULTY OF ELECTRICAL ENGINEERING, COMPUTER SCIENCE AND INFORMATION TECHNOLOGY OSIJEK

Form D1: Form for appointing members to the Committee for the graduation exam

Osijek, 10 July 2023

To the Committee for final papers and graduation exams

Appointing members to the Committee for the graduation exam

Name and surname:	Adamya Shukla
Study programme, branch:	Graduate University Study Programme in Automotive Computing and Communications
Student's ID number: **Admission date:**	D-3ACC 06.11.2020
Student's personal identification number:	60479426737
Mentor:	izv. prof. dr. sc. Josip Job
Co-mentor:	
Co-mentor from the company:	Per-Lage Gotvall
Head of the Committee:	prof. dr. sc. Marijan Herceg
Member 1 of the Committee:	izv. prof. dr. sc. Josip Job
Member 2 of the Committee:	izv. prof. dr. sc. Ivica Lukić
Title of the Master's thesis:	Control system for autonomous robots with wireless and CAN communication
Scientific branch:	Process Computing (Scientific Field of Computing)
Task of the Master's thesis:	Task of this thesis is to control the motors of Autonomous Transport Robots (ATR) based on the wheel velocity. Input data comes through wireless communication (Wi-Fi or 5G) and communication between the control panel and the motor is through CAN bus. It is necessary to analyze the entire communication within the system in order to find the parameter values for optimal communication. It is also necessary to study the possible oscillatory behavior of the control signal due to effects of response time of the system.
Suggested grade for the written part (Master's thesis):	Excellent (5)
Short explanation of the grade pursuant to the Requirements for evaluating final papers and Master's theses:	Application of knowledge and skills acquired during one's studies: 3 points Achieved results related to the task's complexity: 3 points Clarity and coherence: 3 points Level of independence: 3 levels
Date of the mentor's grade:	10 July 2023

Mentor's signature allowing the student to submit the final version to the Student Administration Office:	The mentor electronically signed the submission of the final version
	Date:

FERIT
FACULTY OF ELECTRICAL ENGINEERING,
COMPUTER SCIENCE AND INFORMATION TECHNOLOGY **OSIJEK**

STATEMENT OF ORIGINALITY

Osijek, 26.09.2023

Name and surname:	Adamya Shukla
Study programme:	Graduate University Study Programme in Automotive Computing and Communications
Student's ID number: **Admission date:**	D-3ACC 06.11.2020
Turnitin similarity rate [%]:	7

I hereby declare that the submitted Master's thesis entitled ***Control system for autonomous robots with wireless and CAN communication*** was done under the mentorship of izv. prof. dr. sc. Josip Job

And the above,

is my own work and that, to the best of my knowledge, it contains no sources or resources other than the ones mentioned and acknowledged. I also declare that the intellectual content of this thesis is the product of my own work, except in parts where I was provided with the assistance of my mentor, co-mentor or other people, which is acknowledged.

Student's signature:

ACKNOWLEDGEMENTS

This Master Thesis has been carried out at Volvo Group Trucks Operations, Gothenburg, Sweden. I would like to acknowledge the constant support of my company supervisor, Per-Lage Götvall, Senior R&D Engineer, Volvo GTO, who has been a constant source of inspiration and guided me throughout this thesis work. His input regarding software architecture and used technologies made me come up with concrete solutions. I, furthermore, like to thank my faculty supervisor, Dr. Josip Job, Associate Professor, FERIT Osijek, for his help, valuable advice, and availability during this thesis work. A big thanks to my team at Volvo GTO which includes Kristofer Bengtsson, Erik Brorsson and Johan Svenningstorp with whom I was able to ask and get help.

ABSTRACT

This thesis presents the development of a control system for Autonomous Transport Robots (ATR) which will be used in the Volvo Factories. The research also evaluates the timing and frequency of the control messages sent over the Controller Area Network (CAN) bus of the ATR. The ATR is a part of a cyber-physical system having control parameters over Wi-Fi and CAN communication.

The ATR has a distributed control system with two controllers, which work simultaneously and interdependently. One controller is the center of the decision-control system, and the other controller handles peripherals, buttons, and sensors. The ATR is developed as a state machine having different operating states. Each operating state has its own tasks to perform and a certain set of state transitions. The ATR uses CAN communication to communicate with different nodes present on the ATR. To provide reliability of the CAN message, an interrupt service routine is developed for both the controllers. This interrupt service routine helps the controller to ensure that all the necessary CAN messages arriving on the CAN bus are received by the controller and no important messages are missed or lost in transmission. For Wi-Fi communication, MQTT Protocol is used to communicate with the edge controller.

The response time of both the controllers as well as the response time of the ATR is calculated in all the ATR states and a constant response time of the ATR for every state is proposed. The frequency of the CAN messages is evaluated and a bus load on the CAN bus is calculated at different baud rates. A suitable baud rate for the CAN bus on the ATR is proposed.

Overall, this thesis contributes to the development of a robust control system for ATR, ensuring reliable communication over both CAN and Wi-Fi interfaces. The analysis of response times and bus loads aids in optimizing the performance and efficiency of the ATR in various operational scenarios.

Keywords: Autonomous Transport Robot, Control System, Frequency Analysis, Timing Analysis, CAN bus, Baud rate, Bus load, State Machine, MQTT Protocol

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

AGV	Automated Guided Vehicle
AMC	ATR main controller
AMR	Autonomous mobile robot
ATR	Autonomous transport robots
BEV	Battery electric vehicle
BLDC	Brushless Direct Current
BMS	Battery management system
CAN	Controller area network
CANH	CAN-High
CANL	CAN-Low
CPS	Cyber physical system
DLC	Data Length Code
EV	Electric Vehicles
FCEV	Fuel cell electric vehicle
FET	Field effect Transistor
GPSS	Generic Photogrammetry based Sensor System
GUI	Graphical user interface
HEV	Hybrid electric vehicle
I/O	Input-Output
IANA	Internet Assigned Numbers Authority
IBM	International Business Machines

ICE	Internal combustion engine
ID	Identifier
IoT	Internet of Things
IPS	In-plane switching
IRQ	Interrupt Request
ISO	International Organization for Standardization
ISR	Interrupt Service Routine
JSON	JavaScript Object Notation
LED	Light emitting diode
M2M	Machine-to-Machine
MQTT	Message Queuing Telemetry Transport
MQTT	Message Queue Telemetry Transport
OASIS	Organization for the Advancement of Structured Information Standards
OSI	Open Systems Interconnection
PDO	Process Data Object
PHEV	Plug-in hybrid electric vehicle
QoS	Quality of Service
ROS	Robot Operating System
SDO	Service Data Object
SM-CAN	Simplex Motion - Controller area network
SoC	State of charge
SoH	State of health
SPI	Serial peripheral interface

TCP / IP	Transmission Control Protocol / Internet Protocol
TFT	Thin Film Transistor
TUB	Top unit box
TUC	Top-Unit Controller
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity

1. INTRODUCTION

According to the reports, the Electric Vehicle share in the European market will be reaching 42 % in 2030. The EV global market share is expected to be 32% of all the vehicles. Even in the EV segment, there are several types of EVs in which the four major types are Battery electric vehicle (BEVs), Hybrid electric vehicle (HEVs), Plug-in hybrid electric vehicle (PHEV) and Fuel cell electric vehicle (FCEV). Considering the shift towards electric vehicles from the conventional ICE vehicles, the production of all types of vehicles will be done in the same factories which earlier only produced conventional vehicles. This shift is forcing the industries to come up with technological advancement to incorporate all kinds of vehicles including conventional and EVs in the same factories, under the same production lines and storage areas. The number of types of vehicle parts in the factories to build several kinds of vehicles are increasing rapidly [1].

To handle such a huge number of different parts for different types of vehicles within the existing infrastructure of production lines and warehouses, a new technology is needed to utilize the existing factory area and infrastructure with having great order of flexibility. There are several Autonomous Mobile Robots (AMRs) and Automated Guided Vehicles (AGVs) which is seen as an approach to solve this problem, but all of them have a major drawback which is very high cost and flexibility of the system which is still not done at the operational level.

To accommodate the production of different vehicles which include different parts at the same production line, a unique approach has been developed by and at Volvo Group, which is an automated vehicle parts delivery system called Generic Photogrammetry based Sensor System (GPSS) is used. The GPSS system ensures a greater flexibility as it can help to produce all variants on the same production line. It is also free to locate the activity anywhere in the factory which is beneficial as no infrastructure is needed to be at a fixed place in the factory rather it can be changed or moved based on the requirement. GPSS approach is a unified environment which includes Autonomous Transport Robots (ATRs) for delivery to production lines and robot arms / humans for kitting and assembly works. Overall GPSS system can help factories to increase efficiency of the process and increase safety by automating repetitive tasks. It has a smooth integration with ATRs, AGVs, forklifts, kitting, and assembly robots.

2. LITERATURE REVIEW

2.1. Problem Statement

The ATRs which are used in the GPSS system are developed by the Volvo Trucks. These ATR should have the capability to have a real time control system to react effectively to the commands provided to it and to efficiently analyze the control parameters for the ATR. These ATRs will be used for transporting parts in the Volvo factories and will be used together with more of the same kind of ATR in a fleet or can be used with other collaborative robots, which makes the real time control of the ATR extremely crucial. To have a real time control, the timing analysis of the ATR is quite significant. The problem that arises in autonomous robots is that the response time of the robot to react to the control messages is so high that it affects the smooth operation of the robot, generating unwanted behavior and uncontrolled movement of the robot which can hinder the safety requirements of the robot in a factory environment. To solve this problem, the analysis of timing and frequency of the control messages and response time of the autonomous robots becomes very critical.

This thesis aims to come up with a control system for the ATR and perform timing and frequency analysis of the control messages sent over the CAN bus of the ATR. The bus load on the CAN bus by the CAN messages will also be analyzed in the thesis to come up with a suitable baudrate for the CAN bus of the ATR. This thesis work is carried out with Volvo Group Trucks Operations, Gothenburg, Sweden. The purpose of this ATR is to be used in Volvo Trucks production where it will be used in logistics to transport parts of the vehicles from one place to another autonomously.

2.2. Previous Research and Related Work

With the development of smart logistics systems across various fields, the importance of flexibility in the logistics system seems inevitable as it reduces the setup cost. In the paper [17], the development of an autonomous robotic logistic system is introduced for the use case of meal transporting hospital trolleys to be attached with these robots to carry the food ordered in a medical environment. ROS is used as an operating system for these robots. In the paper [18], a comparative study of various papers is performed by Giuseppe Fragapane et al., in which it says that the control

decentralization is a fundamental strategic decision which mostly depends on the use case of the robotic system.

In the paper [19], Murat Köseoglu et al., have used a distributed control system. It uses two computers which work simultaneously and interdependently. One of the computers is the center of the decision-control system and the other computer is designed to control peripherals and some sensors. Both the computers are connected to each other via a serial connection for communication. This communication is based on USB. The sensory computer sends the sensor data and other parameters of the robot to the decision computer which runs the robotic algorithms to generate resulting parameters which it sends back to the sensory computer to execute the navigation of the robot. The sensors are chosen based on the application of the robot whether it is indoor or outdoor application. In this case, the sensors were chosen based on their indoor application.

3. OVERVIEW OF GPSS SYSTEM

The GPSS project controls a fleet of autonomous transport robots by using a grid of fixed ceiling mounted cameras with a bird eye view of the factory floor, for the purpose of transportation of parts in a dynamic factory environment which can be in collaboration with humans/robotic arms.

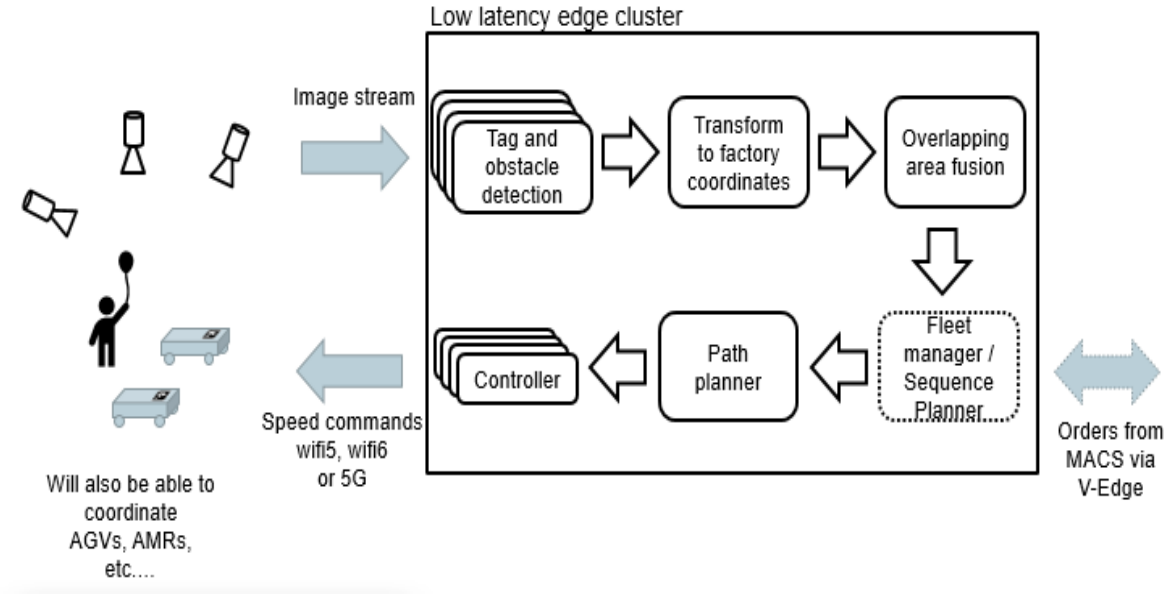

Figure 3.1: Overview of GPSS System

In *Figure 3.1*, the image stream is captured by several cameras and fed to a tag and obstacle detection algorithm which detects the tag placed on autonomous transport robots to get the ATR IDs and detect the obstacles present in the vicinity of the factory area. All these obstacles are mapped to a common factory coordinate which helps the computers to locate the obstacles with respect to the real factory distances. Then, the image is stitched together to remove the overlapping image area of consecutive camera views. This data is now fed to a fleet manager or sequence planner which checks other ATRs in the vicinity and then generates a path for the ATR. Based on the path, a controller generates speed commands for the ATR motors which are sent using wireless technologies, e.g. WiFi-5/6 to the ATR. The ATR control system takes the speed command and forwards it to the motors which provide movement to the ATR.

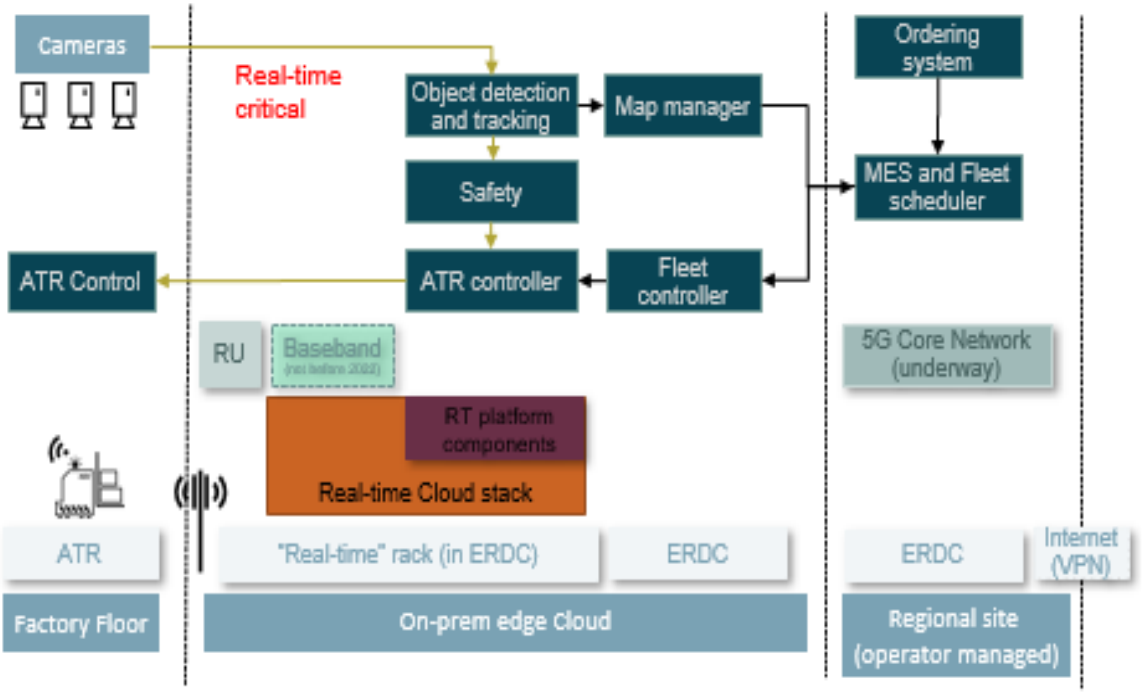

Figure 3.2: Block Diagram of GPSS System

The whole control system is based on three major segments which are operator side control, the edge cloud control, and the control on the factory floor side. The operator side control has an ordering system which is controlled by the operator for scheduling tasks to the ATR. The second segment is edge cloud control which takes care of the tag and object detection and tracking which takes the camera image stream to detect the tag on the ATR to get the information about the ATR and its modules. It also takes care of mapping the obstacles to the factory coordinates. The map manager takes care of it and generates the paths for the ATR based on the obstacles present. The fleet manager algorithm runs in the edge control which takes care of the fleets of the ATR doing their tasks. The ATR controller generates the speed commands for the ATR, and it is sent via WiFi-6 to the specific ATR for the movement over the generated path. The third segment is the control on the factory floor side which consists of physical objects like camera and ATR. The camera is attached on the factory ceiling generating image streams of the factory floor and sending them to the obstacle detection algorithm in the edge cloud control. The speed command from the ATR controller in the edge is received and passed to the motors by the ATR main controller (AMC) which is on the ATR.

3.1. Cyber-Physical System

The Cyber-Physical Systems (CPS) is currently a trending technology introduced in many industries having a close coordination of physical components and computational entities through network communication working together to carry out a process in real time.

3.1.1. Overview of Cyber-Physical System

With the introduction of CPS in Industry 4.0, a lot of new challenges come up, like the need for regulations, design methods, safety, compatibility between software and hardware, transfer of data in real-time, protection against cyber security threats. CPS is developed on components from several engineering disciplines and theories which incorporate embedded systems, control theory, distributed control, sensor fusion and cybernetics.

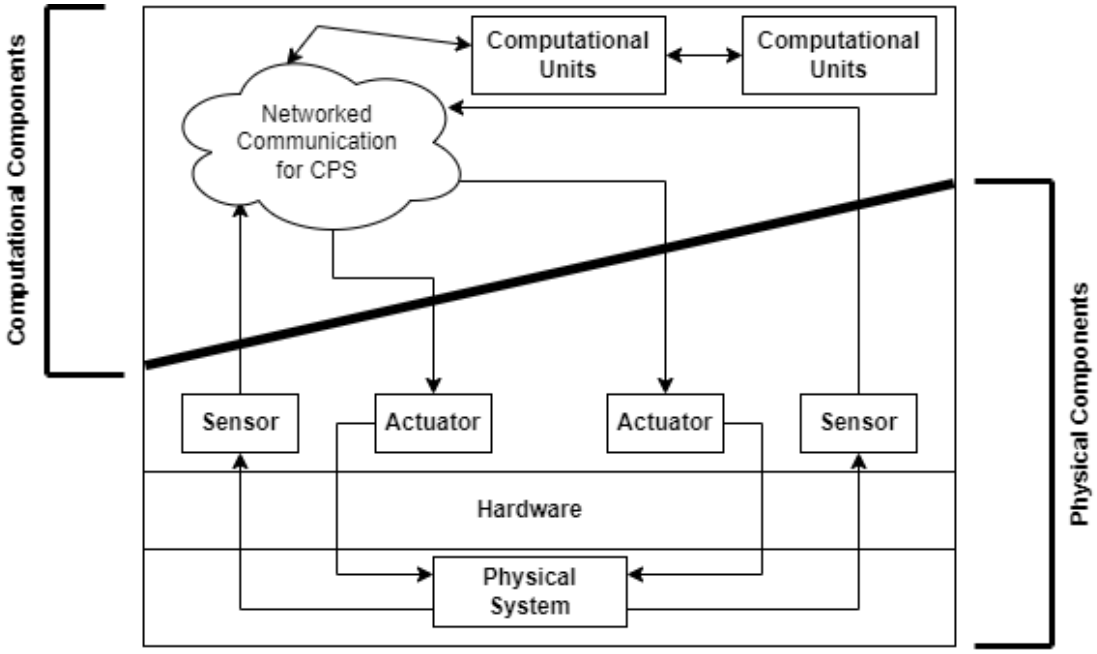

Figure 3.3: Block Diagram of Cyber-Physical System

Figure 3.3 shows a view of CPS. As mentioned before, CPS has two parts working together which are physical and computational components. The physical components consist of a physical system with hardware control, sensors and actuators and other physical elements. The computational / cyber components have networked communication for CPS which handles sensor networks for communication with the computational units and for communication between

computational units and actuators. So, the sensor data is sent to the computational units in the virtual components to perform complex algorithms and provide control parameters for the actuator. Both the physical and computational components work together closely to perform different tasks by the machine.

3.1.2. Cyber-Physical System in the GPSS Set-Up

The GPSS Set-Up is based on the CPS. The physical components present are Arduino and raspberry-pi controllers, sensors, buttons, display, bumper-stop, motors, battery systems which are placed on the ATR whereas the computational component consists of edge-computing with computational units running with different algorithms such as semantic image segmentation algorithms for determining the factory floor and drivable areas for the ATR, object detection and tracking algorithms, fleet management algorithms for smooth operation of ATR fleets, complex calculation of speed and torque commands for running the motors of the ATR. The communication between the physical components like Arduino, battery system, motors, etc. are on Controller Area Network (CAN). The communication between the physical and computational components is done over WiFi-6. The data (from sensors, buttons, etc.) which is needed for performing complex calculations on the edge is sent by the AMC controller on the ATR to the edge and the computed values which are to be sent from edge to ATR are received by AMC controller over the Wi-Fi.

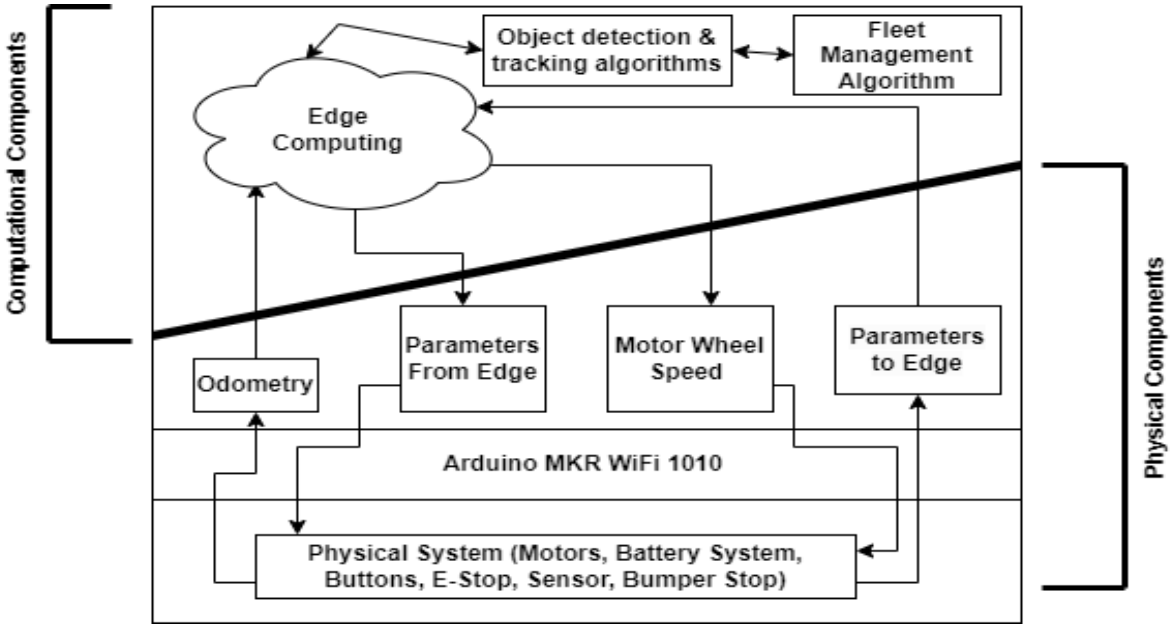

Figure 3.4: Block Diagram of Cyber-Physical System in GPSS

3.2. Distributed Control System with CAN

A distributed control system is a type of control system which is used to monitor and / or control complex processes. It has more than one control unit distributed in a machine where each unit is responsible for a specific part of the process. Distributed control system provides control in real-time with the capability to collect and examine data from actuators, sensors and connected peripherals. The data is analyzed, processed, and shared between different control units, providing a coordinated control of the complete system.

In GPSS ATR, the distributed control system is used as well. The ATR control systems have two controllers coordinating together to provide a smooth operation of the ATR. The first controller is called Top-Unit Controller (TUC) which has all the sensors connected to it via digital and analog pins, it gathers all the required data from different types of buttons, sensors, bumper-stops, emergency stop, joysticks, LEDs and transmits this data to the second controller called ATR Main Controller (AMC) via CAN bus.

The AMC has no direct connection to any sensors, rather it receives the data from the TUC via the CAN Bus, or it receives the data from the edge network via Wi-Fi. AMC is like the brain of the ATR control as all the decision making is done by the AMC. It decides based on the input data what process needs to be executed in that case. AMC has wired connection via CAN bus and wireless connection via Wi-Fi.

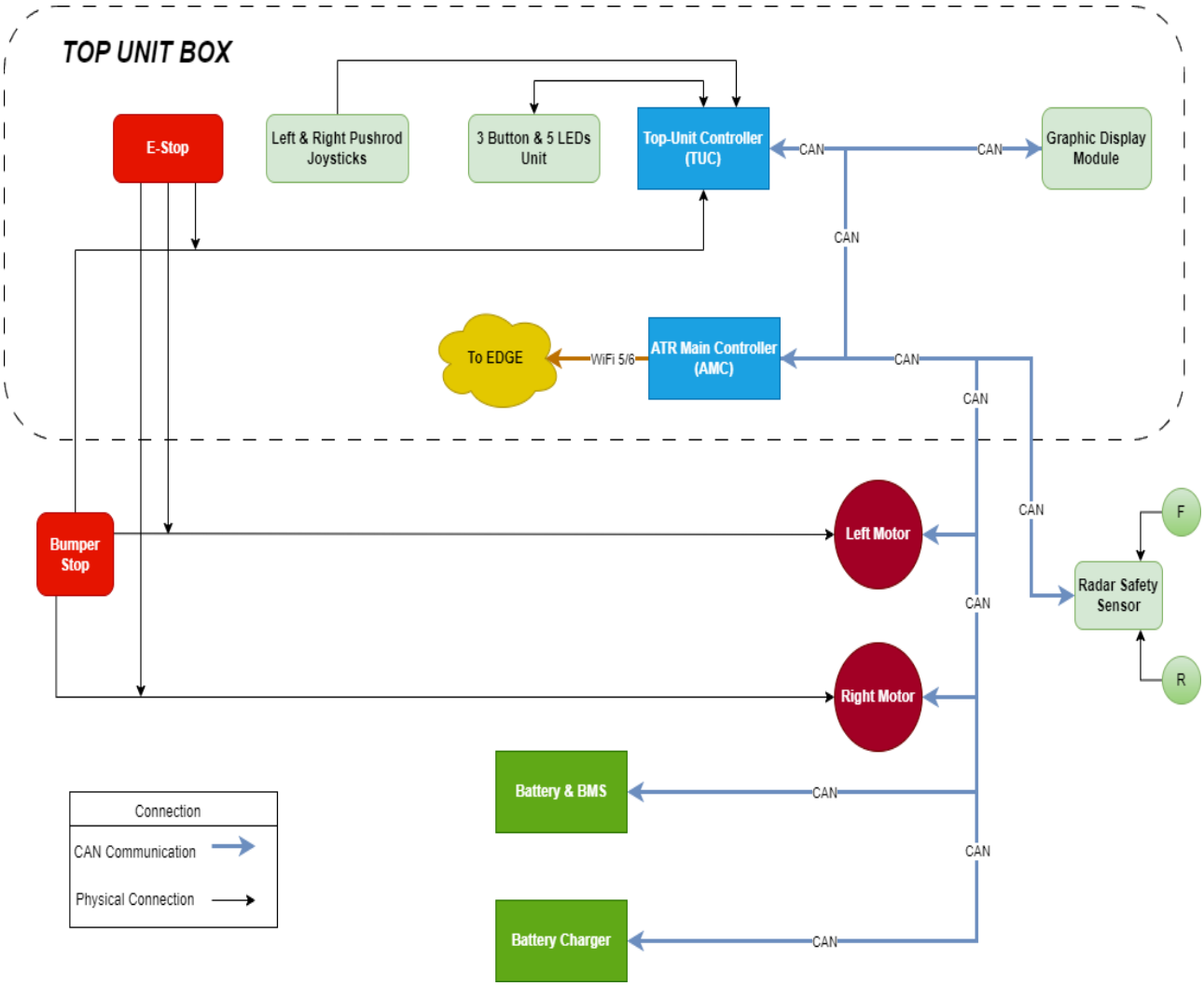

Figure 3.5: Architecture of ATR Control in the GPSS System

4. HARDWARE USED IN THE ATR

The Hardware used in the autonomous transport robots are described below.

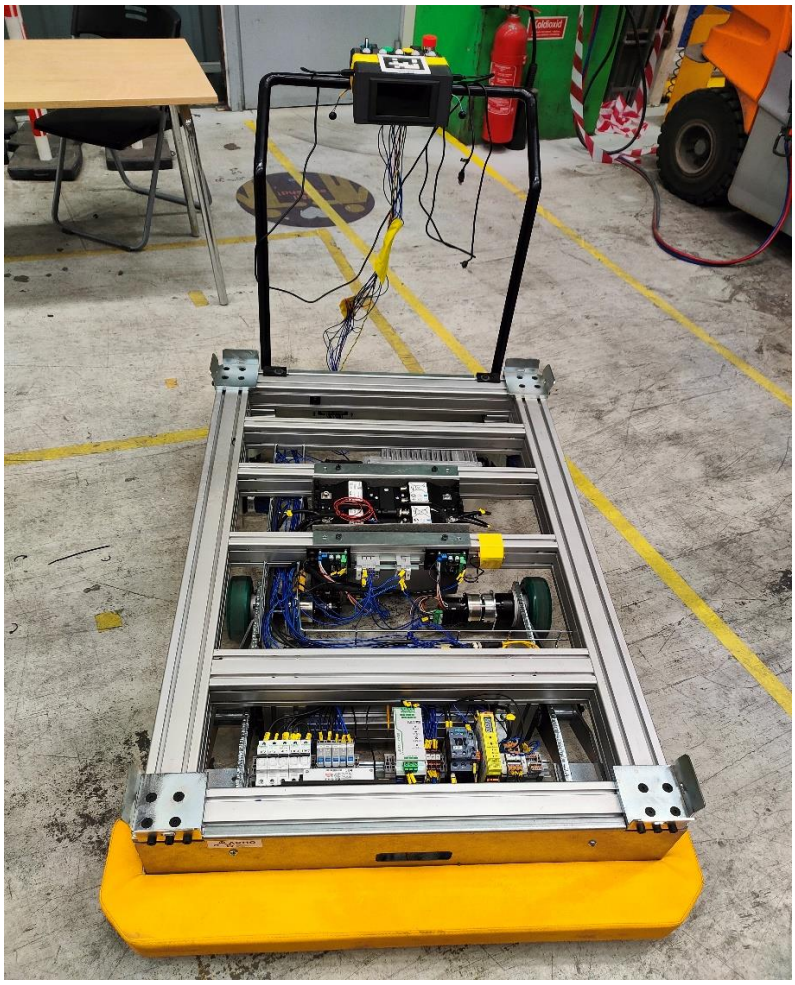

Figure 4.1: Prototype of Volvo's Autonomous Transport Robot (ATR)

4.1. Embedded Controller

The Embedded Controller used for both AMC and TUC are Arduino MKR Wi-Fi 1010. The main processor of the board is a low power Arm Cortex-M0 32-bit SAMD21, like the other boards from MKR family. The Bluetooth and Wi-Fi connectivity is carried out with a module from u-blox, NINA-W10 which is a low power chipset operating in the range of 2.4GHz. The communication is secure by the crypto chip - Microchip ECC508 [5].

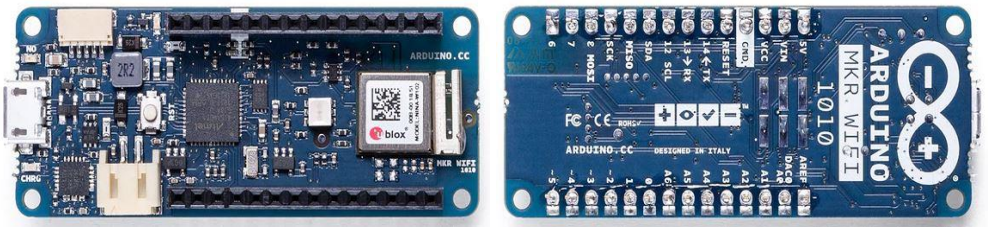

Figure 4.2: Arduino MKR Wi-Fi 1010

4.2. CAN Shield

The CAN shield used for Arduino MKR Wi-Fi 1010 is Arduino MKR CAN Shield which allows interaction between Arduino and the CAN bus. A CAN bus provides the possibility to connect TUC and AMC with motors, batteries with BMS, battery charger, radar module and display. This CAN Shield uses SPI interface to connect with Arduino MKR controller and has Microchip MCP2515 as the CAN controller. It also uses NXP TJA1049 as the transceiver where the screw connectors have input of 7-24v. It includes a switchable onboard termination resistor for CAN bus [6].

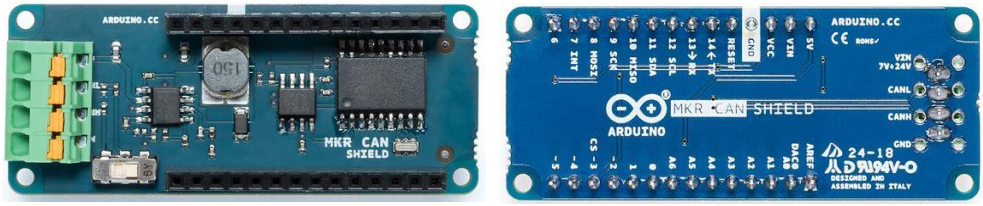

Figure 4.3: Arduino MKR CAN Shield

4.3. Motors

There are two Simplex Motion SH200 BLDC motors. They are in a parallel, but not synchronized set-up and, consequently not either in a master-slave set-up. The motors are of type (type: BXW-04-10H) with a mechanical gearbox with a 1:32 ratio. On the driveline there's a parking/emergency brake mounted. Each motor has a (parking) brake. The parking brake is activated when either power is removed or when the motor has activated the brakes. For the communication, the motors have SM-CAN protocol which is a CAN protocol developed by Simplex Motion AB for their motors which is based on Standard CAN [7].

Figure 4.4: Simplex Motion SH-200 BLDC Motor

4.4. Battery & BMS

The VARTA Easy blade 48V battery system is installed on the ATR. Currently, there are two 48V, 32Ah units installed in parallel. The communication between all batteries in the system is done via the standardized CANopen protocol (PDOs). The batteries have an in-build BMS system. At normal operation, all necessary data is transmitted via PDOs in cyclic intervals. If the application needs more information in addition, more data can be accessed via SDO request to the master battery. The master-slave protocol regularly monitors and adjusts necessary components such as balancing of cells and distribution of temperature at cell as well as system level. Relevant battery data such as SoC, SoH or the cycle status can be called up through the CAN protocol. The battery/batteries are installed according to the VARTA manual. Plus and minus poles are connected, CAN link is connected and there is a termination resistor (120Ω connected at the battery between CANH and CANL) [8].

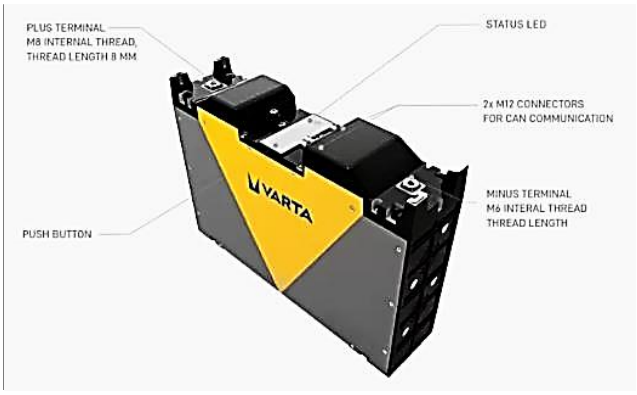

Figure 4.5: VARTA Easy blade 48V battery

4.5. Battery Charger

A Delta-q Easy Charger 48 charger is installed as a permanent On-board charger. The charger is permanently connected to the batteries and the ATR CAN bus. No communication exists between e.g., the AMC or TUC with the Charger. Charger only communicates with the battery master. The communication between all batteries in the system and the charger is done via the standardized CANopen protocol. The battery master also sends the system's requirements for charging to the charging device. The battery master will initialize the charger to the appropriate voltage and current needed to charge the battery system. The voltage and current values requested from the charger will be sent every 100ms. In case of a fully charged battery, the battery will set its "fully charged" flag in the battery information status register and open its charge FET. When all batteries of the system are fully charged, the master sets its "fully charged" flag and sets the charger to a standby value for the next 5 minutes. After this delay, the battery system will shut down [9].

Figure 4.6: Delta-q Easy Charger 48 Charger

4.6. Display

A Winstar 5-inch-high brightness IPS TFT Smart CAN series display is used for GUI of the ATR. The display runs on the CANopen protocol to render display content on the screen and return touch event data with protocol objects. It has Built-in flash memory to store the font and Object Dictionary Data and has a capacitive touch panel. The Smart display can acquire the data that it displays either using the CANopen SDO protocol or using the CANopen PDO protocol [10].

Figure 4.7: Winstar 5-inch-high brightness IPS TFT Smart CAN series Display

4.7. Top Unit Box

There is a Top Unit Box (TUB) which is mounted on the pushrod of the ATR. The TUB consists of 5 LEDs, 1 Emergency-stop button, 2 joysticks, 2 push buttons and 1 bi-stable switch to turn On/Off the ATR. The LEDs are used to communicate the state of the ATR. The TUB also houses the AMC and TUC on it. A bumper stop which is placed in the front of the ATR also is connected with the TUB.

Figure 4.8: Top unit box placed on the pushrod of the ATR

5. STATE MACHINE AND STATE DEFINITIONS OF THE ATR

5.1. State Machine

A State Machine has a finite number of states, in which only a single state is selected as current state at a time, whereas the change of the state is initiated by a condition or an event [11]. A decision-making mechanism can be used to change the states for taking different actions in a particular state. This mechanism should be able to consider different parameters that sums up user and environment status to take the required action according to the state transition rules that have been defined.

A state machine contains a state variable which executes its state, and commands, which changes its state. Each command is executed by some program that modifies the state variables to produce some output. The state variable of the state machine makes a request to initiate a change of the state to execute a state [12].

A program can sometimes have invalid states, and in conventional software, the variables are scrutinized carefully and changed in such a way that these invalid states don't happen. A state machine is a way of imitating a state, instead of interpreting independent variables, a machine is modeled to handle what states are possible, and when a machine is a given state, what next state is permitted. Understanding state machines is similar to understanding state charts.

5.2. State definitions of the ATR

The ATR acts as a "State machine". It operates in different states or different modes. That is essentially a way to structure the ATR behavior, both from a user perspective as well as for engineering. In this chapter, the States will be described from a user perspective. It can be used as a guide for instructing operators and maintenance staff but is mainly intended for engineering. The ATR states are seen in *Figure 5.1*.

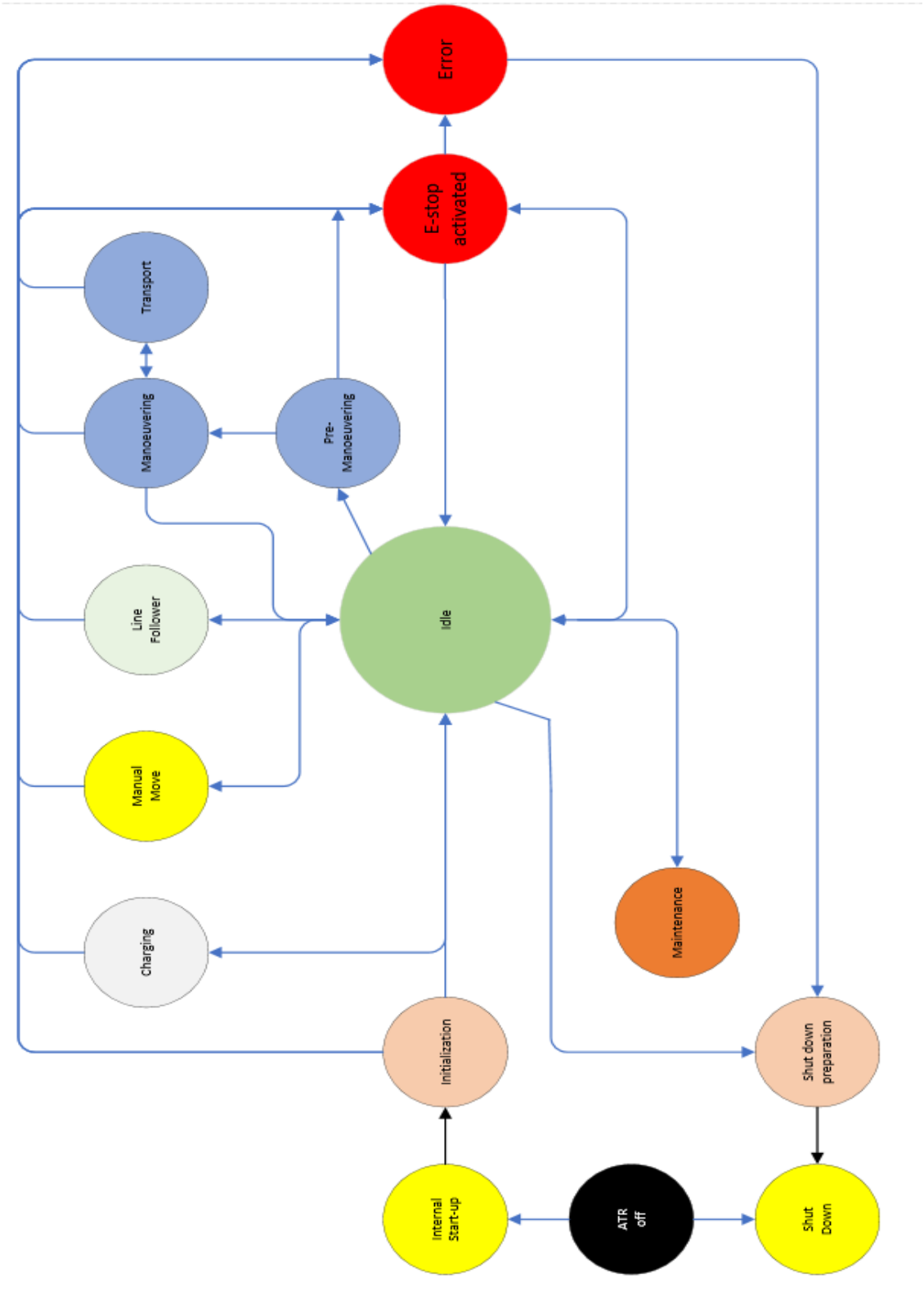

Figure 5.1: ATR States Transition Diagram

There are several ATR States and their priorities (lower value of the priority represents higher state priority) shown in *Table 5.1*.

Table 5.1: ATR States and priority of the states

ATR States	Value (hex)	Value (dec)	Priority	Comment
Start-up	-	-	-	Starting state
Idle	0x10	16	1	-
Manual move	0x11	17	2	-
Pre- Maneuvering	0x12	18	3	-
Maneuvering	0x13	19	3	-
Transport	0x14	20	3	-
Line Follower	0x15	21	3	Introduced later
Charging	0x16	22	1	-
Shutdown Preparation	0x17	23	2	-
Shutdown	0x18	24	-	-
Error	0x19	25	0	-
Emergency Stop Activated	0x20	26	0	-

The state definitions are described below:

5.2.1. Start-up State

The start-up state is the first state when you turn on the ATR where all the starting up of the ATR takes place. This state only runs once during one power cycle while starting up the ATR. The transition back to this state is not allowed. The ATR is ready to be started in this state. CAN bus and MQTT Protocol is initialized in this state.

5.2.2. Idle State

When the Start-up state is completed and the ATR has an established connection to the Edge controller, the ATR enters the Idle state. The ATR brakes are locked and are now ready for use. If an error occurs during the first two states, or if the on-board E-stop is pressed, or if the front bumper stop is affected, the ATR will not go into Idle state, instead ending up in either Error, E-stop activated. This state has a priority 1 which is a medium priority.

When in Idle state, the ATR can transition to charging state, manual move state and Pre-maneuvering state.

5.2.3. Manual Move State

In general, one can only activate the state Manual when the ATR is in state Idle. If an operator presses the button marked “Manual” on the push-rod box, the ATR brakes are released and it can be either pushed manually, or by using the linear joysticks, located on the pushrod box. The joysticks can be used for both forward and reverse control. This state has a priority 2 which is a low priority. If an error occurs during this state, the ATR will go into Error state. If the on-board E-stop or bumper stop is activated, the ATR will go into “E-stop Activated” state. When the ATR is in Manual mode it is not responding to any messages from the Edge controller. i.e., a contact with the Edge controller is not necessary Although, a contact with the Edge controller is needed for going into Idle mode.

5.2.4. Pre-Maneuvering State

This state is activated when the ATR has got an order from the Edge to go in the Pre-Maneuvering state. This state can only be reached from the Idle state. In this state, the ATR needs to wait 3 seconds before moving. It needs to wait 3 seconds, so that the humans working nearby know that ATR is about to move. The brakes are applied to the ATR during this state. Pre-Maneuvering is the first state ATR enters when in autonomous mode. This state has priority 3 which is the lowest priority. If an error occurs during this state, the ATR will go into Error state. If the on-board E-stop or bumper stop is activated, the ATR will go into “E-stop Activated” state. When in Pre-Maneuvering state, the ATR can transition into either Idle state or Maneuvering state.

5.2.5. Maneuvering State

Maneuvering is a state entered from Pre-Maneuvering when the ATR is in autonomous mode. The transition to this state is only initiated by the edge controller. The ATR can achieve a maximum speed of 0.3 m/s in this state. When the ATR is in the Maneuvering state it is allowed to move close to people and fixed installations in the factory. The ATR leaves this state when it either has received its destination or when the circumstances allow the ATR to speed up (i.e., entering Transport mode). This state has priority 3 which is the lowest priority. If an error occurs during this state, the ATR will go into Error state. If the on-board E-stop or bumper stop is activated, the ATR will go

into “E-stop Activated” state. When in Maneuvering state, the ATR can transition into either Idle state or Transport state.

5.2.6. Transport State

The ATR can achieve a maximum speed of 1.5 m/s in this state. The ATR leaves this state when it either gets closer to the destination or when the circumstances force the ATR to slow down. If that happens it enters the “Maneuvering” state. This state has priority 3 which is the lowest priority. If an error occurs during this state, the ATR will go into Error state. If the on-board E-stop or bumper stop or the Virtual stop from the edge controller is activated, the ATR will go into “E-stop Activated” state.

5.2.7. Charging State

The ATR enters the charging state when the ATR is plugged into charging. Charging State can be only entered from Idle state and can only transition to the Idle State. In this state, the brakes are applied, and the ATR stays into the charging state until the ATR is unplugged from the charging. This state has a priority 1 which is a medium priority.

5.2.8. Maintenance State

The ATR enters this state when in Idle state and the Maintenance symbol on the display is pressed. The ATR remains on with brakes applied and the maintenance engineer is able to do preliminary fault seeking and calibration through guidance from the display. E.g., presented on the display are several running data like state of health of the battery, total run time since last maintenance. Also, it is possible to do some simple adjustments like getting real time data from radar sensors. When leaving the maintenance state, the ATR always returns to Idle state.

5.2.9. Error State

The ATR enters the Error State if either an internal Error is detected or if there is an anomaly in the communication with the Edge controller or on the CAN bus. If the ATR goes into Error mode, it stops, and the brakes are applied. There are error codes assigned for the defined possible errors so that it can help the maintenance engineer to rectify the error. The error codes can be seen on the ATR display when in the maintenance state. This state has a priority 0 which has the highest priority. The Error State will be initiated in case of any state transition collision statements.

5.2.10. Emergency Stop (E-Stop) Activated State

The ATR can enter an Emergency stop activated state from any of the previous states except start-up, error, and maintenance state. ATR can enter this state if either the local E-stop button situated on the push-rod box is pressed, or a nearby virtual E-stop is activated by the edge controller, or the Bumper stop situated at the front of the ATR is activated. If the ATR goes into E-stop activated mode, it stops hastily. The ATR brakes are applied and will remain applied until leaving the E-stop state. This state has a priority 0 which has the highest priority.

6. SOFTWARE ARCHITECTURE OF THE ATR

The software architecture of the ATR works as a state machine. The ATR as a whole is seen as a state machine having different states of operation. At a time only execution of one state is permitted. The ATR is started-up and initialization starts.

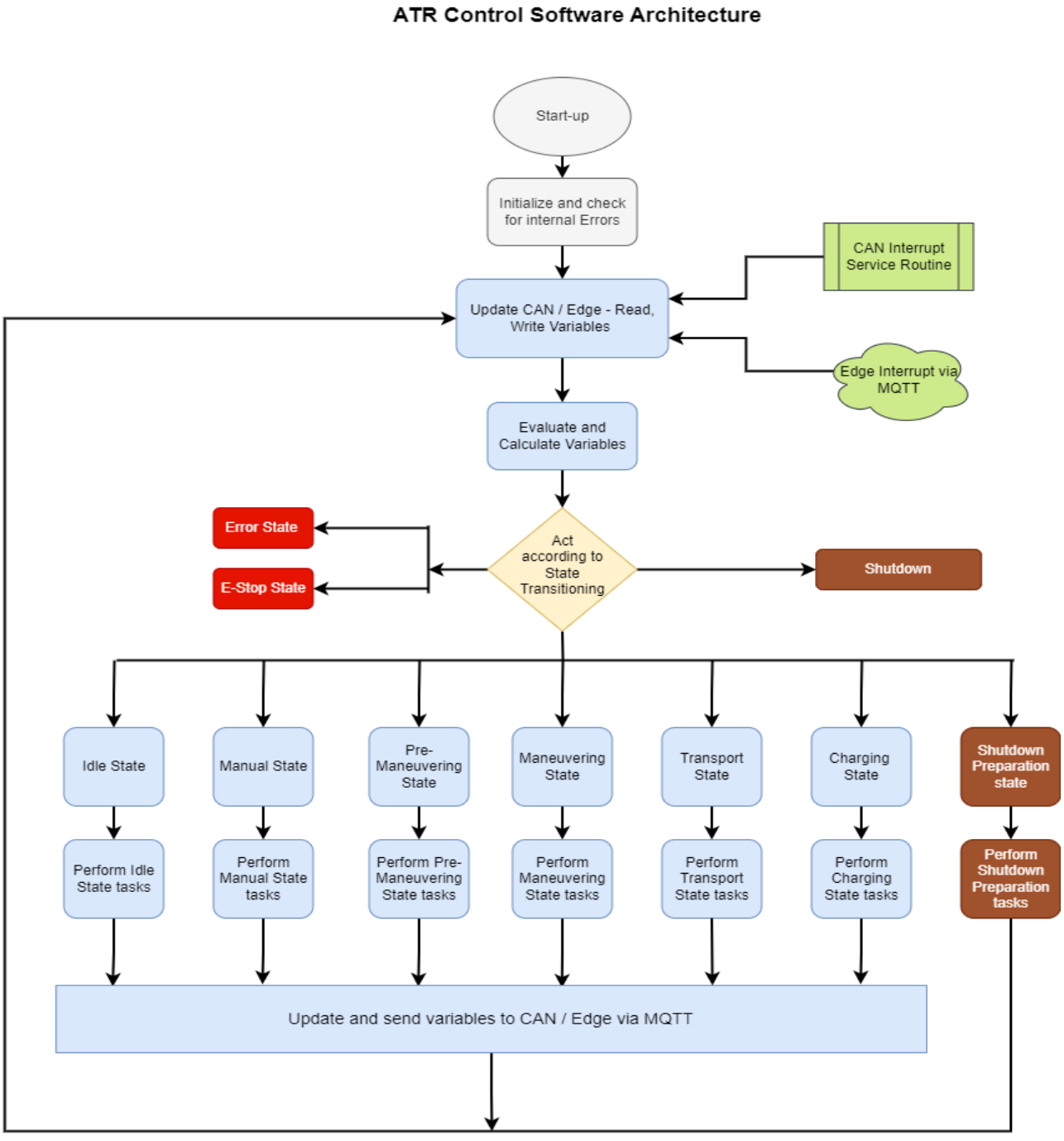

Figure 6.1: Flow Chart of ATR control Software Architecture

During initialization, the ATR sets up connection with the Edge controller through MQTT Protocol via Wi-Fi. ATR sends up initializing data to the edge about the ATR-ID, type of ATR and the parameters related to the specific ATR like measurements of the ATR which will be helpful for the edge to calculate speed commands which will be later send to ATR from edge for driving ATR autonomously. The internal error checks on the ATR are done during initializing which includes checking available nodes on the CAN bus through a heartbeat check function. If even one of the nodes on the CAN bus doesn't respond, the ATR goes into Error state. If all the nodes respond, the ATR internal check is passed. Now, the ATR is ready to receive the data from the edge controller and the CAN bus. The data required to run the ATR such as wheel speed, the request to change ATR state, virtual stop state, charging request and other variables are received on the ATR and are updated and manipulated according to the ATR's requirement. The data are received on an interrupt basis and are updated at the callback. These data are stored in a variable in the ATR and are updated in every loop. The CAN frames are also received on the interrupt with a callback function with an Interrupt Service Routine (ISR). The ISR for the CAN message is shown *Figure 6.2*.

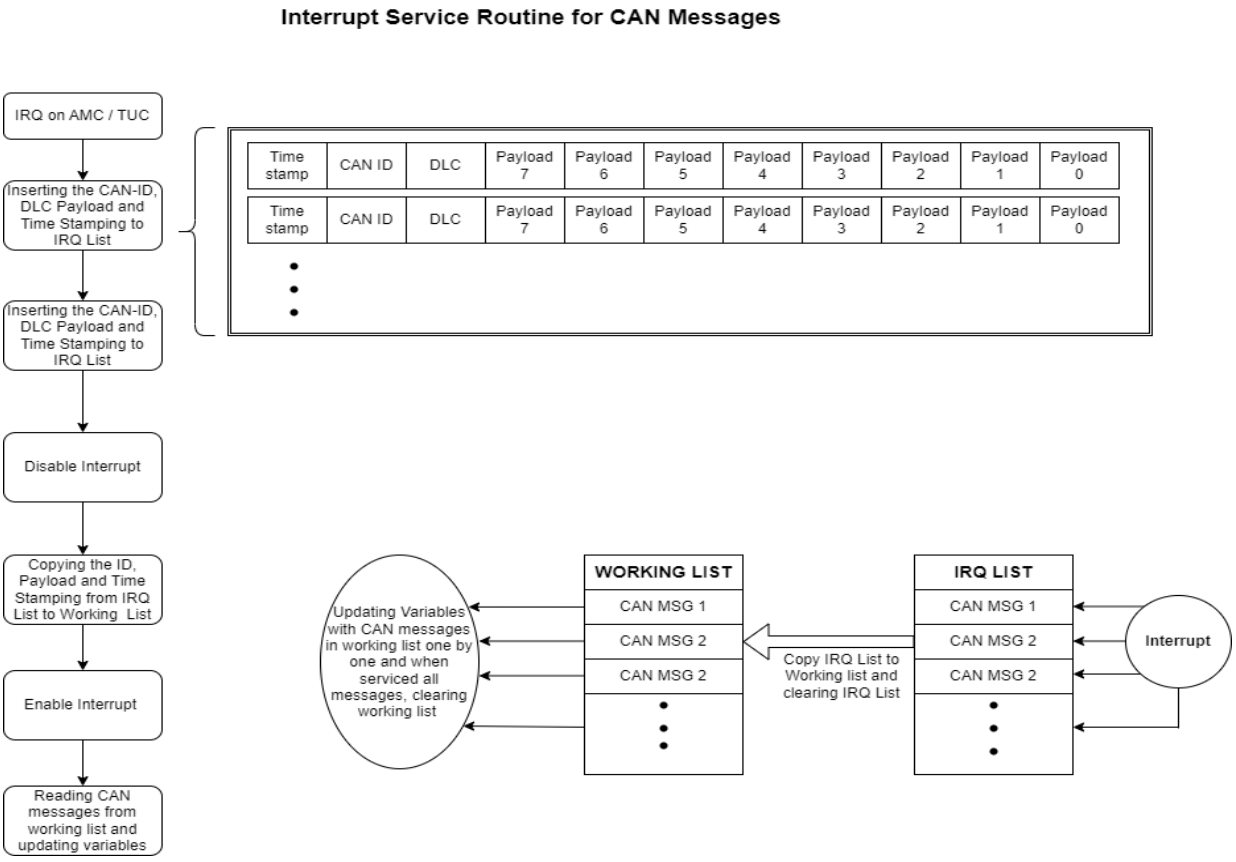

Figure 6.2: Interrupt Service Routine for CAN Messages

The CAN messages are received on an interrupt and each message received is time-stamped with its CAN-ID, DLC and the payload. This message format is created as soon as the ATR receives a CAN message. In the ISR, two lists are created namely IRQ List and Working List. The IRQ list is the list in which the time-stamped CAN frames are put while being on Interrupt. The working list is the list with which the CAN frame payload data is put into the desired variable in the ATR. The working list updates each message in the list by moving pointers on the list so no CAN message is missed. After each loop, the IRQ list consists of several CAN messages. These CAN messages are copied to the working list while detaching interrupt, so no new CAN message arrives in the IRQ list while copying the messages. After copying the messages to the working list, the interrupt is again attached to the IRQ list. Once all the messages from the IRQ list are copied to the working list. Each message from the working list is read and the data is assigned to the relevant variables on the ATR controller. With the help of this ISR for CAN messages, it is secured that no CAN frame is missed, all the frames are serviced, and relevant variables are updated which are necessary for the ATR operation.

Once all the variables are updated from CAN Bus and edge controller via MQTT, with the help of these variables the state transition requirements are evaluated. To transition from one state to another state, there are state transition requirements. These state transition requirements are verified and if the transition requirements are met, the state is changed. If these state transition requirements are not met, the same state is executed again. Each state has separate state transition requirements as only defined states can be transitioned from a particular state. For example, If in the maneuvering state, there will be three transition requirements for transitioning to Idle state or Pre-maneuvering state or to transport state. If none of these transition requirements are met, the ATR stays in the maneuvering state.

New State initiation can be done from either the specific ATR, or from the Edge controller. Edge controller acknowledges on all state changes but do also initiate the following transitions:

- Idle State → Pre-maneuvering State
- Pre-maneuvering State → Maneuvering State
- Maneuvering State → Transport State
- Pre-maneuvering State → Idle State

- Maneuvering State → Idle State
- Maneuvering State → Pre-maneuvering State
- Transport State → Maneuvering State

All other transitions are initiated by the ATR, but they are acknowledged by the edge controller and the edge needs to accept the state change. When a new state transition is identified locally on the ATR, Next State is defined in the ATR according to transition conditions and a state change request (Next State) is sent to the Edge. The Edge controller acknowledges by sending Current State value (that is the previous Next State value) back to the ATR. Once acknowledgement is received from the edge, the ATR changes its state. It performs the tasks required to be executed in that specific state. For example, In Maneuvering state, the new motor control parameters, relevant mode of the motor and the speed commands for the wheels are sent to the motors. Once the state specific tasks are executed, the updated variables during this loop are sent on the CAN bus and to the edge controller via MQTT from the ATR. Once again, the next loop continues from reading and updating variables sent by edge controller and CAN bus.

6.1. Software Architecture of the ATR Main Controller (AMC)

The software architecture of AMC is based on a state machine. The AMC's embedded computer is Arduino MKR Wi-Fi 1010 used with a CAN shield. The AMC is connected with both CAN bus as well as Wi-Fi. When the ATR is started-up, the AMC starts initializing communication with CAN bus and setting up MQTT protocol over the Wi-Fi. Once communication is set-up, the CAN frames are received by using interrupt service routine of CAN messages shown in *Figure 6.2.* These CAN frames are read and populated to AMC specific variables. For example, A message with state of buttons is received from the TUC, these states of buttons are transferred into an AMC specific variable like Manual button or Ready button or E-stop with states true or false based on whether the button is pressed or not.

Similarly, the data sent by the edge is received via MQTT protocol over Wi-Fi. These data are combined in a form of a topic and are published to the broker. The AMC subscribes to the specific topic to receive data. Arduino JSON data format is used by the AMC to get the data via MQTT. The data are stored in the AMC specific variables. Based on these variables, the state transition

requirements are evaluated. The state change can be initiated by the Edge controller as well as locally by evaluating the transition requirements. After evaluating the transition requirements, the ATR next state is selected. If the ATR next state is the same as the current state, the ATR remains in the same state but if the ATR next state is not the same as the ATR current state, a state synchronizing message is sent to the edge to acknowledge the state change.

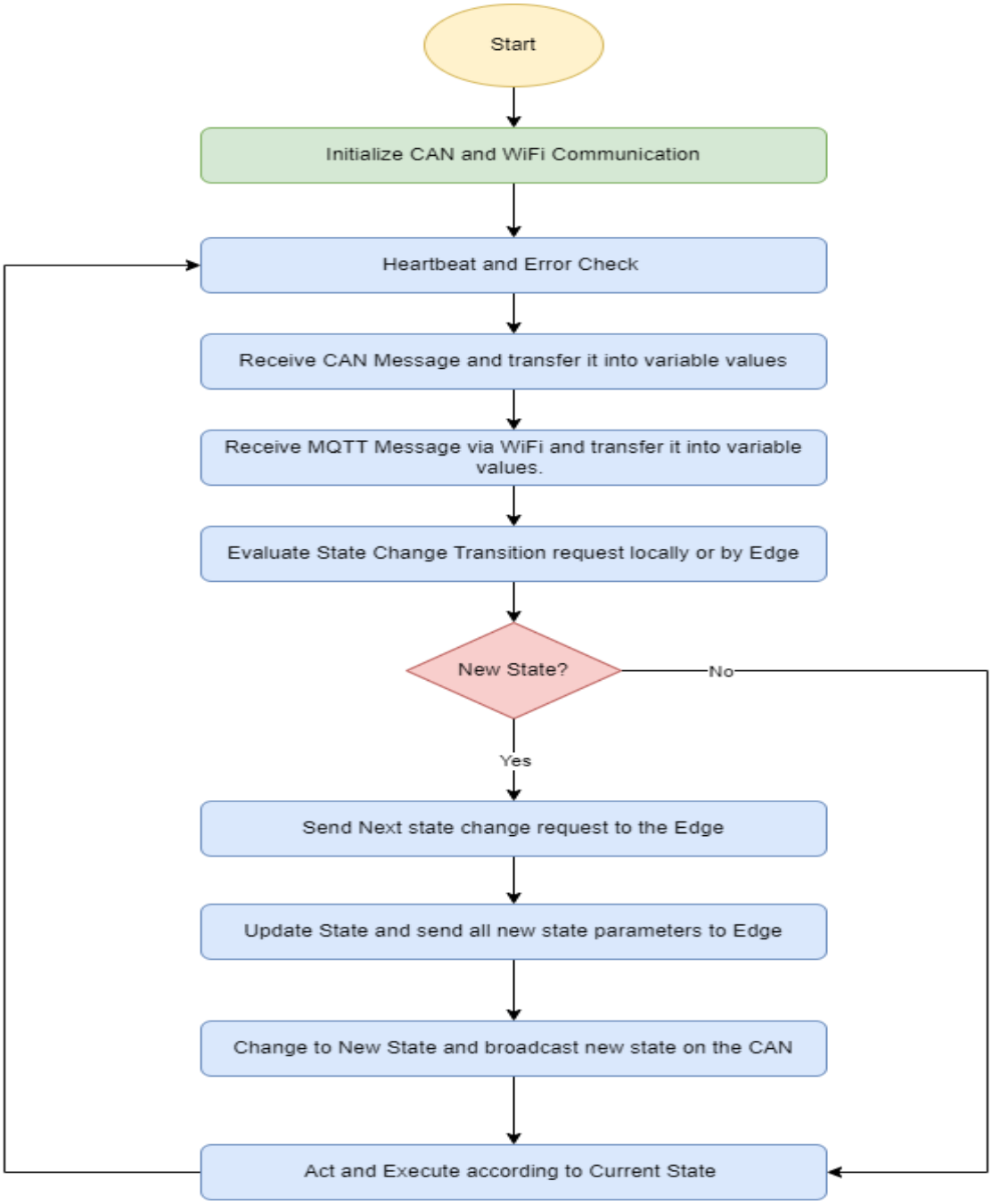

Figure 6.3: Flow Chart of AMC Software Architecture

When the edge acknowledges the state change initiated by the ATR, the ATR changes its current state to the ATR next state. If the acknowledge is not received, the ATR waits for the edge to send the acknowledge and only after receiving it, changes its state. As soon as the ATR state is changed, an ATR state message is sent on the CAN bus carrying the current state of the ATR and an order for TUC to change its state also similar to AMC. This is done to ensure AMC and TUC are synchronized as they must have the same state at a particular time.

After AMC and TUC are synchronized with the same current states, the specific-state tasks in the AMC are performed and executed. Then, the MQTT messages are sent on the topic to the broker with the relevant data for the edge controller. The CAN messages are also framed to be sent on the CAN bus with the updated value from the loop. The AMC after this continues back to the receiving of CAN and MQTT messages for the next loop.

A heartbeat check is designed which runs in the background and keeps track if all the nodes are alive. The heartbeat messages generally have only one byte data frame which means its DLC is one. This byte provides the status of the CAN nodes. The CAN nodes could be in Pre-operational state ($7F_{16}$), Operational state (05_{16}), or Stopped state (04_{16}). In Operational state, all CAN functions are provided. The heartbeat messages are checked continuously throughout the run-time of the AMC, if any of the heartbeat messages is not received for 3 seconds consecutively, an error on that node is flagged with an error code. Even if any of the nodes send the message that the node has stopped working that means it is in stopped state (04_{16}), an error on that node is also flagged with another error code.

The heartbeat messages are received by the AMC from the TUC, both the motors, the BMS and the display. The charger heartbeat is also visible when the charger is connected to the ATR. This heartbeat check is performed in every loop cycle, and it helps in determining internal error on the CAN bus. The error on the ATR initiates the Error state and the ATR is stopped and requires maintenance. The error code for that particular error is assigned so it can be provided to the maintenance engineer to ease the fault seeking and debugging of the error.

6.2. Software Architecture of the Top Unit Controller (TUC)

The TUC is also seen as a state machine. The TUC is like the sensory organ of the ATR which only cares about the external interfaces with the controller. The software architecture of TUC is similar to the AMC but with different functionalities. The TUC has Arduino MKR Wi-Fi 1010 as the embedded controller. The TUC is only connected to a CAN bus through CAN shield and has no connection to the edge controller via Wi-Fi. TUC only communicates to the AMC through the CAN bus. The TUC is connected to all the input-output peripherals through its I/O pins. The Bumper-stop, E-stop, Ready button, Manual Button, Joysticks, ON/OFF Button and indicating LEDs, all are connected to TUC through the I/O pins.

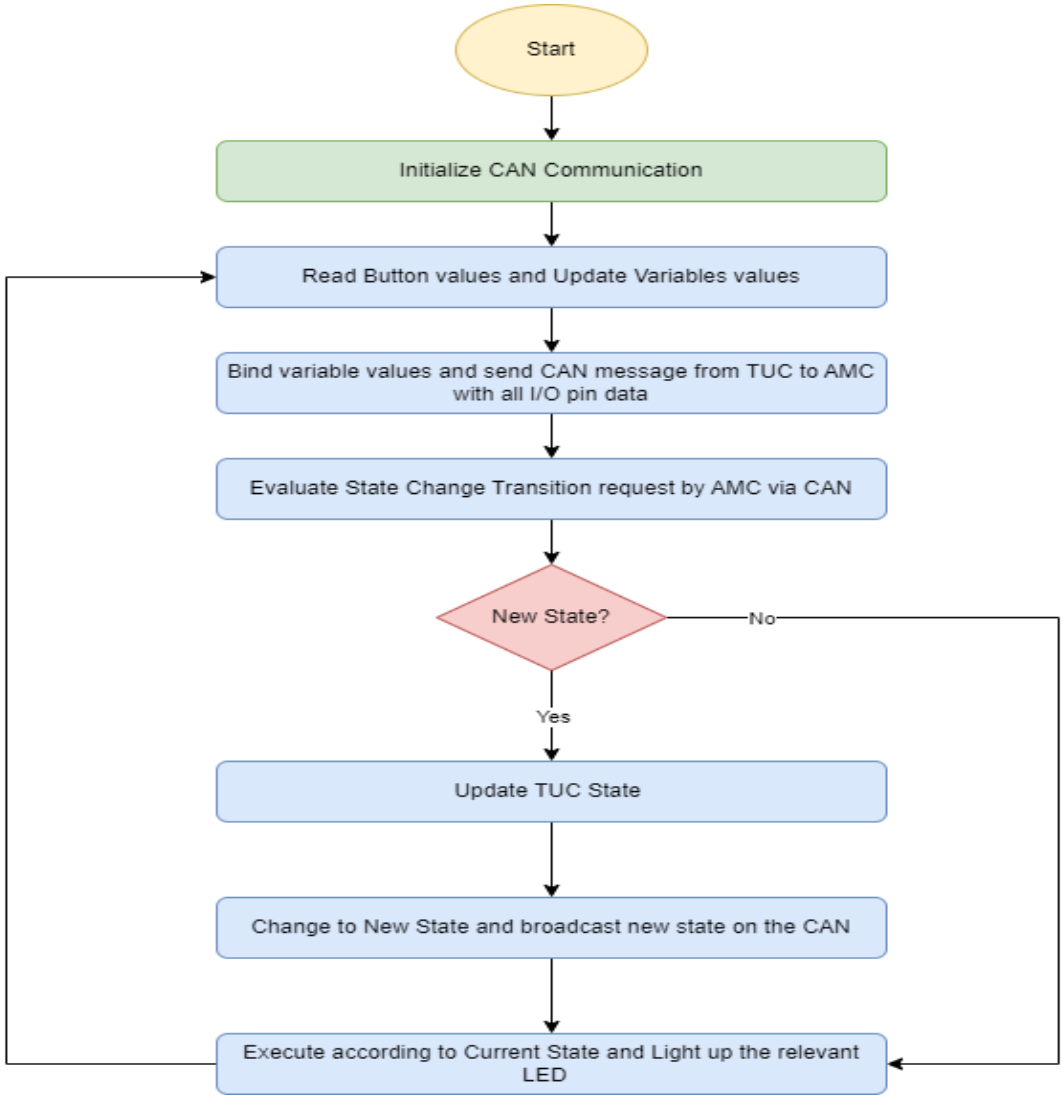

Figure 6.4: Flow Chart of TUC Software Architecture

When the ATR is started-up, the TUC starts initializing the communication on the CAN bus. After the CAN communication is set-up, the TUC starts polling and reading the I/O pins of the Arduino to get the attached button / peripheral state and this state is updated in specific variables. These different variables with I/O pin states are bound together to create a CAN frame namely TUC heartbeat message and this frame is sent on the CAN bus to be received by the AMC where AMC evaluates ATR state based on these I/O pin state values. As the AMC evaluates its next state, the AMC sends a message to TUC to change its state. This message contains the next state the TUC needs to go in.

The TUC does not decide its own state, rather the AMC directs TUC to change its state to be synchronized with the ATR state as a whole unit. As this state message is received from the AMC, TUC changes its state as well. The TUC then updates its state and binds this state variable and sends it back to the AMC within the same TUC heartbeat message. Based on the new state, the tasks defined in the particular state are executed. For example, If the TUC goes into Manual State, the yellow LED needs to be turned on and rest all other LEDs are to be turned off. After this the loop continues with reading the I/O pins for any change in the button states placed on the top-unit box.

The AMC and TUC work together to provide a complete function to the ATR. The ATR has most of its control communication via CAN bus and a lot of calculations and algorithms are run on the edge controller to provide us with commands to run the ATR. Although all the major control of the ATR stays locally on its own controllers.

7. ATR COMMUNICATION

7.1. CAN Communication

The Controller Area Network bus also known as CAN bus is a serial communication protocol widely used in automotive, rail transportation, aerospace, and other industry to allow controllers and other devices (nodes) to communicate with each other without a central computer controlling the communication. It was developed by Robert Bosch, a German company in the late 1980's which later eventually became an international standard – ISO 11898 [13]. It is currently one of the most widely used field buses. CAN bus has the advantage to operate reliably in harsh environments such as variation in temperature or large noises caused by electromagnetic instability. The protocol is also fault-tolerant and has advantages of error handling and error detection mechanism. CAN is a priority-based bus which supports real-time communication. Every CAN message on the bus is assigned a priority and the message transmission follows a deterministic order decided by the priorities given to the messages [14].

The CAN bus is a two-wired bus consisting of twisted pairs of wires that carry power as well as data. It uses a differential signal naming CANH and CANL to transmit data, which means it can transmit data at high speeds with low signal noise. CAN uses the OSI model to transfer data among nodes connected in a network. Every node like AMC, TUC, Battery, Motor, display, and others has a controller which is a small and low-cost computer. Messages in CAN are sent in a format called CAN frames. A frame is a defined structure, carrying a sequence of meaningful bits of data on the bus.

In GPSS, Standard CAN is used. The standard CAN and SM-CAN both use standard CAN frames to communicate with other nodes. CAN communication is used on the ATR to communicate between AMC, TUC, motors, battery, charger, and display.

Figure 7.1: Standard CAN Frame Format

The standard CAN frame uses a 11-Bit Identifier also known as CAN-ID which is unique to every single CAN frame. No two CAN frames should have the same CAN-ID, otherwise error occurs on CAN bus. The CAN-ID includes priority of the CAN frame, lower the CAN-ID higher the priority of the message. CAN frame can carry 8 bytes of data in one message. The number of bytes to be used in the message is selected by DLC which could be a maximum of 8 bytes. CAN consists of error frames in case any node is communicating error on the bus.

7.2. ATR CAN Nodes

There are several ATR CAN Messages which are used for communication between all the CAN nodes on the ATR. There are eight CAN nodes on the CAN bus which are shown below with the Node-IDs. All these CAN nodes are on the same CAN bus and hence, all the nodes have their unique Node-IDs.

Table 7.1: ATR CAN Nodes with Node-IDs

Name	Abbreviation	Node ID (Dec)	Node ID (hex)
Battery Management System	BMS	27	0x1B
ATR Main Controller	AMC	40	0x28
ATR Top Unit Controller	TUC	50	0x32
Right Motor Controller	RMC	61	0x3D
Left Motor Controller	LMC	62	0x3E
Radar Sensor Controller	RSC	70	0x46
Display Unit	DU	80	0x50
Battery Charger	BCC	100	0x64

7.3. CAN Messages from the Battery Management System

7.3.1. BMS Master Voltage Current Message

The General description:

BMS Master Voltage Current message is sent from BMS on the CAN Bus. It is received by AMC. It contains information regarding Master Voltage and Master Current values. This message's CAN-ID is 0x19B and Node-ID is 27. The DLC for this message is 8.

Name	CAN-ID	Node-ID	DLC	Data		Event
				Master Voltage (Max. Battery Voltage)	Master Current (sum of all modules)	
bms_master_Vl_msg	0x19B	27	8	uint32 [mV]	int32 [mA]	1 sec

Figure 7.2: BMS Master Voltage Current Message Structure

Repetition rate: 1 second

Arbitration field:

- Bit 0-10 = 00110011011
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Battery information status register
- Battery warning status register

7.3.2. BMS Temperature Design Capacity Message

The General description:

BMS Temperature Design Capacity message is sent from BMS on the CAN Bus. It is received by AMC. It contains information regarding battery temperatures and master design capacity values. This message's CAN-ID is 0x29B and Node-ID is 27. The DLC for this message is 8.

Name	CAN-ID	Node-ID	DLC	Data			Event
				Max. Battery FET Temperature	Max. Battery Cell Temperature	Master Design Capacity (sum of all modules)	
bms_temp_des_cap_msg	0x29B	27	8	int16 [0.1°C]	int16 [0.1°C]	uint32 [mAh]	1 sec

Figure 7.3: BMS Temperature Design Capacity Message Structure

Repetition rate: 1 second

Arbitration field:

- Bit 0-10 = 01010011011
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Max. Battery FET Temperature
- Max. Battery Cell Temperature
- Master Design Capacity

7.3.3. BMS Charge Capacity Message

The General description:

BMS Charge Capacity message is sent from BMS on the CAN Bus. It is received by AMC. It contains information regarding Charge Capacity values. This message's CAN-ID is 0x39B and Node-ID is 27. The DLC for this message is 8.

Name	CAN-ID	Node-ID	DLC	Data		Event
				Master Full Charge Capacity (sum of all modules)	Master Remaining Capacity (sum of all modules)	
bms_charge_cap_msg	0x39B	27	8	uint32 [mAh]	uint32 [mAh]	1 sec

Figure 7.4: BMS Charge Capacity Message Structure

Repetition rate: 1 second

Arbitration field:

- Bit 0-10 = 01110011011
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Master Full Charge Capacity
- Master Remaining Capacity

7.3.4. BMS Status Message

The General description:

BMS Status message is sent from BMS on the CAN Bus. It is received by AMC. It contains information regarding flag and status values of the BMS. This message's CAN-ID is 0x49B and Node-ID is 27. The DLC for this message is 8.

Name	CAN-ID	Node-ID	DLC	Data				Event
				Master Information Status Register	Master Warning Status Register	Master Error Status Register	Master Charge Control Status Register	
bms_status_msg	0x49B	27	8	uint16	uint16	uint16	uint16	200 ms

Figure 7.5: BMS Status Message Structure

Repetition rate: 200 ms

Arbitration field:

- Bit 0-10 = 10010011011
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Master Information Status Register
- Master Warning Status Register
- Master Error Status Register
- Master Charge Control Status Register

7.3.5. BMS Charging Message

The General description:

BMS Charging message is sent from BMS on the CAN Bus. It is received by Charger. It contains information regarding charge control and battery status values of the BMS. This message's CAN-ID is 0x264 and Node-ID is 100. The DLC for this message is 8.

Name	CAN-ID	Node-ID	DLC	Data						Event
				Charge Control	SoC	Reserved	Master Warning Status Register	Master Error Status Register	Battery Status	
batt_to_charging_msg	0x264	100	8	uint8	uint8	uint8	uint16	uint16	uint8	100 ms

Figure 7.6: BMS Charging Message Structure

Repetition rate: 100 ms

Arbitration field:

- Bit 0-10 = 01001100100
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Charge Control
- State of Charge (SoC)
- Master Warning Status Register
- Master Error Status Register
- Battery Status

7.4. CAN Messages from the ATR Main Controller (AMC)

7.4.1. AMC State Control Message

The General description:

AMC State Control message is sent from AMC on the CAN Bus. It is received by TUC. It contains information regarding the AMC current state and the state TUC should go in. This message's CAN-ID is 0x264 and Node-ID is 40. This message has a priority of 1 and the DLC is also 1.

Name	CAN-ID	Node-ID	DLC	Data	Event
				ATR Current State	
amc_state_control_msg	0x148	40	1	uint8	100 ms

Figure 7.7: AMC State Control Message Structure

Repetition rate: 100 ms

Arbitration field:

- Bit 0-2 = 001 (Priority is 1)
- Bit 0-10 = 00101001000
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 0001 (1) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains: ATR Current State

7.4.2. Common Motor Drive Parameter Message

The General description:

Common Motor Drive Parameter message is sent from AMC on the CAN Bus. It is received by left and right motors. It contains information about the common parameters required for driving the motors. This message's CAN-ID is 0x265 and Node-ID is 40. This message has a priority of 2 and the DLC is 8.

Name	CAN-ID	Node-ID	DLC	Data				Event
				Max. Torque	Max. Ramp Speed	Max. Ramp Acceleration	Max. Ramp Deceleration	
common_motor_drive_param_msg	0x265	40	8	int16	int16	int16	int16	100 ms

Figure 7.8: Common Motor Drive Parameter Message Structure

Repetition rate: 100 ms

Arbitration field:

- Bit 0-2 = 010 (Priority is 2)
- Bit 0-10 = 01001100101
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Maximum Torque
- Maximum Ramp Speed
- Maximum Ramp Acceleration
- Maximum Ramp Deceleration

7.4.3. State Specific Motor Mode Message

The General description:

State Specific Motor Mode message is sent from AMC on the CAN Bus. It is received by left and right motors. It contains information about the mode in which motors have to go. This message's CAN-ID is 0x166 and Node-ID is 40. This message has a priority of 1 and the DLC is 8.

Name	CAN-ID	Node-ID	DLC	Data		Event
				Motor Mode	Reserved	
state_specific_motor_mode_msg	0x166	40	8	uint16	-	100 ms

Figure 7.9: State Specific Motor Mode Message Structure

Repetition rate: 100 ms

Arbitration field:

- Bit 0-2 = 001 (Priority is 1)
- Bit 0-10 = 00101100110
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains: Motor Mode

7.4.4. Target Speed/Torque Command Message

The General description:

Target Speed/Torque Command message is sent from AMC on the CAN Bus. It is received by left and right motors. It contains information about the target Speed / Torque commands for driving the motors. This message's CAN-ID is 0x167 and Node-ID is 40. This message has a priority of 1 and the DLC is 8.

Name	CAN-ID	Node-ID	DLC	Data		Event
				Right Motor Target Speed / Torque register value	Left Motor Target Speed / Torque register value	
target_speed_torque_cmd_msg	0x167	40	8	int32	int32	100 ms

Figure 7.10: Target Speed/Torque Command Message Structure

Repetition rate: 100 ms

Arbitration field:

- Bit 0-2 = 001 (Priority is 1)
- Bit 0-10 = 01001100111
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Right Motor Target Speed / Torque Register Value

- Left Motor Target Speed / Torque Register Value

7.5. CAN Message from Top-Unit Controller (TUC)

7.5.1. Top-Unit Box Status Message

The General description:

Top-Unit Box Status Message is sent from TUC on the CAN Bus. It is received by AMC. It contains information about the button values, joystick values and the TUC state. This message's CAN-ID is 0x174 and Node-ID is 50. This message has a priority of 1 and the DLC is 8.

Name	CAN-ID	Node-ID	DLC	Data								Event
				Status TUCB Local Error	TUCB Button Status 1	TUCB Button Status 2	Forward Right Joystick Value	Forward Left Joystick Value	Reverse Right Joystick Value	Reverse Left Joystick Value	TUC State	
tuc_HB_msg	0x174	50	8	uint8	uint8	uint8	uint8	uint8	uint8	uint8	uint8	50 ms

Figure 7.11: Top Unit Box Status Message Structure

Repetition rate: 50 ms

Arbitration field:

- Bit 0-2 = 001 (Priority is 1)
- Bit 0-10 = 00101110100
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- TUC Local Error Status
- TUC Button Status
- Joystick Values
- TUC Status

7.6. CAN Messages from the Motors

7.6.1. Right Motor Status Message

The General description:

Right Motor Status Message is sent from Right Motor on the CAN Bus. It is received by AMC. It contains information about the current motor speed and torque. This message's CAN-ID is 0x141 and Node-ID is 61. This message has a priority of 1 and the DLC is 8.

Name	CAN-ID	Node-ID	DLC	Data			Event
				Current Speed Value Register	Current Torque Value Register	Reserved	
right_motor_drive_status_msg	0x141	61	8	int16	int16	-	50 ms

Figure 7.12: Right Motor Status Message Structure

Repetition rate: 50 ms

Arbitration field:

- Bit 0-2 = 001 (Priority is 1)
- Bit 0-10 = 00101000001
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Current Speed Register Value
- Current Torque Register Value

7.6.2. Left Motor Status Message

The General description:

Left Motor Status Message is sent from Left Motor on the CAN Bus. It is received by AMC. It contains information about the current motor speed and torque. This message's CAN-ID is 0x143 and Node-ID is 61. This message has a priority of 1 and the DLC is 8.

Name	CAN-ID	Node-ID	DLC	Data			Event
				Current Speed Value Register	Current Torque Value Register	Reserved	
left_motor_drive_status_msg	0x143	62	8	int16	int16	-	50 ms

Figure 7.13: Left Motor Status Message Structure

Repetition rate: 50 ms

Arbitration field:

- Bit 0-2 = 001 (Priority is 1)
- Bit 0-10 = 00101000011
- Bit 11 = 0 (RTR always 0)
- Bit 12-15 = 1000 (8) Data length Code, DLC. Indicates the number of payload bytes.

Data field contains:

- Current Speed Register Value
- Current Torque Register Value

7.7. CAN Internal Message on the Bus

There are also some internal CAN messages which are always on the CAN bus though they are used internally by the nodes and are not used for any applicational use.

Table 7.2: CAN Internal Messages on the bus

Name	Sender Node	Receiver Node	CAN-ID	Node-ID	DLC	Event
bms_HB_msg	BMS	AMC	0x701	1	1	500 ms
charger_HB_msg	Charger	AMC	0x764	100	1	1 sec
int_batt_Vl_msg_1	Battery_1	Battery_2	0x181	1	8	1 sec
int_batt_Vl_msg_2	Battery_2	Battery_1	0x182	2	8	1 sec
int_temp_Vl_req_msg_1	Battery_1	Battery_2	0x281	1	8	1 sec
int_temp_Vl_req_msg_2	Battery_2	Battery_1	0x282	2	8	1 sec
int_batt_cap_msg_1	Battery_1	Battery_2	0x381	1	6	1 sec
int_batt_cap_msg_2	Battery_2	Battery_1	0x382	2	6	1 sec
int_batt_status_msg_1	Battery_1	Battery_2	0x481	1	8	100 ms
int_batt_status_msg_2	Battery_2	Battery_1	0x482	2	8	100 ms

These CAN messages include the heartbeat messages from each node and internal CAN messages which are used by individual batteries which are stacked in parallel. Here, we have two batteries in parallel, so both batteries internally communicate to each other over the CAN bus. These CAN messages help the multiple battery modules to be stacked together and still have the same voltage and charging current. These internal CAN messages have a fixed priority, DLC and the repetition rate predefined by the battery manufacturer. These CAN messages are always present on the CAN bus.

7.8. MQTT Protocol via Wi-Fi Communication

The MQTT (Message Queuing Telemetry Transport) is a lightweight, publish-subscribe based message protocol that is designed for use in IoT and M2M contexts. MQTT design principles are to minimize bandwidth and device resource requirements while having high reliability and assurance of delivery to some degree. MQTT was developed originally by IBM in the late 1990's for monitoring oil pipelines. In 2014,

MQTT became standardized with the release of version 3.1.1. an open standard maintained by OASIS. This open standard is now widely used for wireless communication protocol across various industries [15].

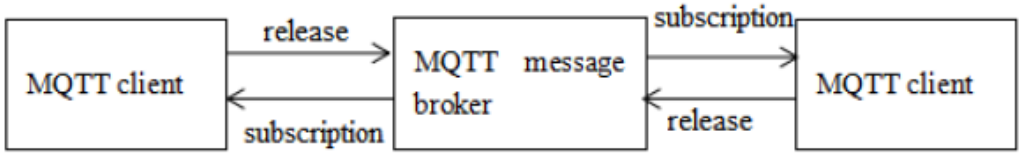

Figure 7.14: Structure diagram of MQTT

In *Figure 7.14*, MQTT has two parts which are MQTT message broker and MQTT clients. The MQTT clients are connected through a common MQTT message broker. The MQTT client needs a central MQTT message broker to publish data on a topic, which can be subscribed by another client on the same topic. Multiple clients can publish a topic to the same broker and subscribe to a topic from the same broker. MQTT broker acts like a central unit which collects all data and sends the data to the devices which request for it based on a predefined set of rules. MQTT works over the TCP / IP protocol. It uses port 1883 which is the default port assigned by IANA, but technically MQTT can use any port. Mosquitto and HiveMQ are some of the local MQTT brokers [16]. MOTT supports three levels of QoS for message delivery, QoS 0 which provides at most once delivery of the message, QoS 1 which provides at least once delivery of the message and QoS 2 which provides exactly once delivery of the message.

bit	7	6	5	4	3	2	1	0
byte1	Message Type				DUP flag	QoS level	RETAIN	
Byte2	Remaining Length							

Figure 7.15: MQTT fixed header message

The MQTT message structure consists of:

- **Fixed header**. It contains information about the type of message and the level of QoS, and other flags. Fixed header is always present in the message and takes up two bytes.
- **Variable header.** This is an optional part of the message and length of the variable header depends on the type of the message and the level of QoS. This part contains information like the ID of the message, the name of the topic, and other message specific data.
- **Payload.** This is the data which is actually being transmitted in the message.

In GPSS, the MQTT protocol is used to send applicable ATR data which is used to drive the ATR autonomously with the help of the cameras in the ceiling and computing at edge.

7.9. ATR MQTT Message

The MQTT messages are sent to the edge controller and received from the edge controller over the Wi-Fi. The MQTT protocol runs on top of a TCP/IP using a publish-subscribe topology. MQTT is an event driven protocol. There is no periodic or ongoing data transmission. It allows the transmission to be minimum. The information is only sent when a client publishes the data, and a broker only sends out information to subscribers when new data arrives.

The ATR AMC controller sends MQTT messages to the edge controller with information regarding the ATR. The AMC sends the odometry information from the motors to the Edge. The current state of the ATR is sent along with it with the same message on the same topic. This message is sent every 100 ms to the edge.

The Edge also sends MQTT messages based on the event driven basis to the AMC with information about state change information and / or the wheel speed commands from the edge in the autonomous transport states.

8. RESPONSE TIME ANALYSIS OF THE ATR

The Response time of the ATR includes the time response for both AMC and TUC. The response time of the whole ATR will have loop time of the AMC and loop time of the TUC. The loop time of the AMC was evaluated. The AMC has a loop time of approximately 5 ms when the AMC is in Idle state, Pre-Maneuvering state, Charging state, Shutdown Preparation state, Emergency stop activated state and Error stop. These states have a shorter loop time as most of the tasks in this particular state are very less time consuming and do not include the task of running the motor.

Table 8.1: Loop time in different AMC States

AMC States	Loop Time
Idle State	~ 5 ms
Manual Move State	~ 55 ms
Pre-Maneuvering State	~ 5 ms
Maneuvering State	~ 25 ms
Transport State	~ 25 ms
Charging State	~ 5 ms
Shutdown Prep. State	~ 5 ms
E-stop State	~ 5 ms
Error State	~ 5 ms

The loop time of the AMC is more when it is in mobile states like Manual Move state, Maneuvering state and Transport state. The Manual Move state takes almost 55 ms to run though the loop while being in that state. The Manual Move States includes running the motor which consists of more CAN messages for the motor which makes the loop time a bit more. The Autonomous states like Maneuvering state and Transport states also take like 25 ms to go through the loop. The loop time of the AMC varies in different states. This is because the tasks in different states vary as in some states the tasks AMC has to do are simpler and executed faster while in other states the tasks are complex and take more time in execution.

While, in the TUC, almost all the states have a loop time of approximately 40 ms. This is because the TUC has almost similar tasks in all the states. The most common task in all the states is to poll the controller pins for the incoming signals from buttons, joysticks, emergency stops and, to

send them to AMC by framing it in a CAN message and, to light and unlight the LEDs on the top unit box based on the ATR state.

Table 8.2: Loop time in different TUC States

TUC States	Loop Time
Idle State	~ 40 ms
Manual Move State	~ 40 ms
Pre-Maneuvering State	~ 40 ms
Maneuvering State	~ 40 ms
Transport State	~ 40 ms
Charging State	~ 40 ms
Shutdown Prep. State	~ 40 ms
E-stop State	~ 40 ms
Error State	~ 40 ms

The loop time is almost constant in all the states in TUC but the loop time in different states differs in the AMC. So, to have a constant loop time in AMC as well, delays are added to make the loop time of AMC to be 100 ms. By doing this, we secure that the loop time in all the states in AMC also is constant. Having a constant loop time helps in synchronizing the time taken to run the AMC in any state. A delay of 10 ms is also added to make the TUC to have a loop time of 50 ms in all the states. This ensures that the TUC loop runs twice until the AMC is running once, securing that AMC does not miss any messages sent from TUC over the CAN bus.

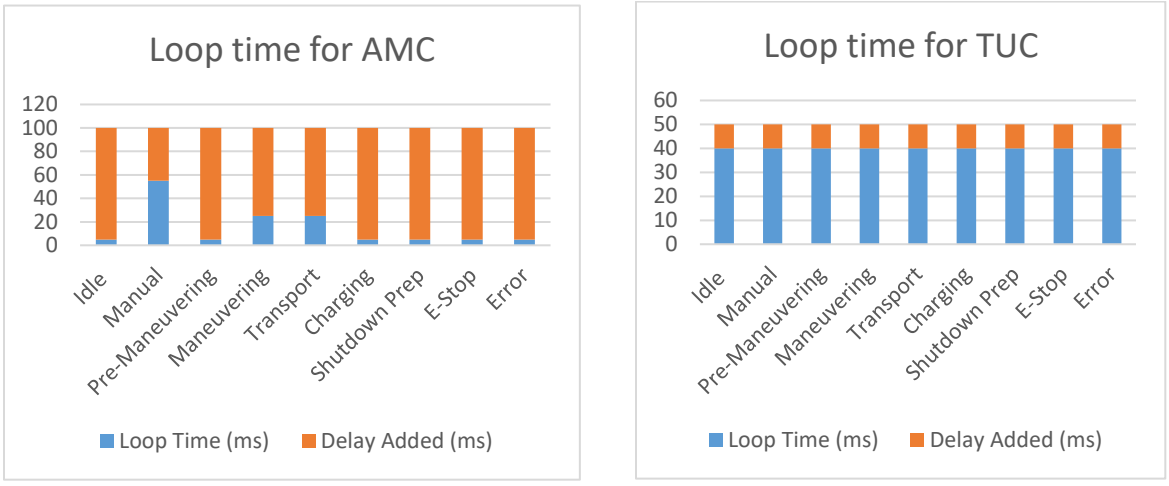

Figure 8.1: Bar Graph of Loop time for AMC and TUC

The response time for a particular state is depending on the data flow from TUC to AMC and back to TUC. Generally, all the input from the top unit box is taken by TUC which takes 50 ms to poll the TUC I/O pins and bind the data received from pins in a CAN message and send it to AMC. Then AMC takes 100 ms to run the loop and perform all the tasks in that state and send the AMC state as a CAN message to the TUC. The TUC again takes 50 ms to go into commanded state from AMC and light up the relevant LEDs for that state.

So, the maximum response time from switching the states is 200 ms. Although staying in the same state without changing the state could have even shorter response time than 200 ms. By doing this we secure that the response time is not more than 200 ms in any state.

9. ANALYSIS OF LOADING OF CAN BUS OF THE ATR

The bus load on a CAN bus is affected by following factors:

- **Number of Messages on CAN bus:** The total number of CAN messages on the CAN bus affects the bus load. Higher number of CAN messages transmitted on CAN bus will require higher bandwidth which in turn increases the bus load.
- **Frequency of Messages on CAN bus:** The repetition rate of a CAN message or the frequency at which CAN message affects the bus load. Higher frequency of CAN messages requires more frequent bus utilization which in turn increases the bus load.
- **Length of CAN Message (DLC):** The length of CAN messages which is defined by Data length code of the message affects the bus load. Longer the message, it occupies more time on the bus thus requires more bandwidth which in turn increases the bus load on the CAN bus.
- **Bit Rate of the CAN Bus:** The operating bit rate of the CAN bus affects the bus load. Higher bit rate gives faster transmission of data on CAN bus, but faster transmission also increases the amount of data to be transmitted in a defined time, which results in higher bus load.
- **Arbitration of the CAN bus:** CAN bus works on priority-based arbitration mechanism, where CAN messages with higher priority have higher preference on the bus. The bus load can be affected by the number of CAN messages contending for bus access and their priorities.
- **Bus utilization efficiency:** The efficiency of CAN message transmission on the bus affects the bus load. Inefficient utilization of CAN messages, such as a large number of message collisions or re-transmission of the messages, increases the bus load.

To have a smooth operation on the CAN bus, the bus load should be only 50 % of available bandwidth. The bus load on the CAN bus can be more than 50 % for a short period of time but the average bus load should not exceed over 50 %. However, overloading the CAN bus can lower the performance, increase the latency, and can lead to data corruption. So, a suitable designing and monitoring of the CAN bus system are significant to maintain reliable and efficient communication. It is important to calculate the necessary transmission speed which is also known as baudrate of the CAN bus to reduce the communication as much as possible. Communication should be designed to

communicate as little as possible but as much as necessary. Higher baudrate on the CAN bus can also generate faster switching, which can cause increased electromagnetic emissions and be vulnerable to external interference. This interference can disrupt communication and reduce the reliability of the CAN bus.

Table 9.1: Percentage bus load at different baudrate

CAN Message Name	Sender Node	Receiver Node	CAN-ID	Node-ID	DLC	Update Rate(sec)	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	BMS	AMC	0x19B	27	8	1	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	BMS	AMC	0x29B	27	8	1	0,0512	0,0256	0,0128	0,0064
bms_charge_cap_msg	BMS	AMC	0x39B	27	8	1	0,0512	0,0256	0,0128	0,0064
bms_status_msg	BMS	AMC	0x49B	27	8	0,2	0,256	0,128	0,064	0,032
battery_HB_msg	Battery	AMC	0x701	1	1	0,5	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	Battery	Charger	0x264	100	8	0,1	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	Battery 1	Battery 2	0x181	1	8	1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_2	Battery 2	Battery 1	0x182	2	8	1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	Battery 1	Battery 2	0x281	1	8	1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_2	Battery 2	Battery 1	0x282	2	8	1	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	Battery 1	Battery 2	0x381	1	6	1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_2	Battery 2	Battery 1	0x382	2	6	1	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	Battery 1	Battery 2	0x481	1	8	0,1	0,512	0,256	0,128	0,064
int_batt_status_msg_2	Battery 2	Battery 1	0x482	2	8	0,1	0,512	0,256	0,128	0,064
charger_HB_msg	Charger	AMC	0x764	100	1	1	0,0064	0,0032	0,0016	0,0008
amc_state_control_msg	AMC	TUC	0x148	40	1	0,1	0,064	0,032	0,016	0,008
tuc_HB_msg	TUC	AMC	0x174	50	8	0,05	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	Right Motor	AMC	0x141	61	8	0,05	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	Left Motor	AMC	0x143	62	8	0,05	1,024	0,512	0,256	0,128
In ATR Driving mode only										
common_motor_drive_param_msg	AMC	Left & Right Motor	0x265	40	8	0,1	0,512	0,256	0,128	0,064
state_specific_motor_mode_msg	AMC	Left & Right Motor	0X166	40	8	0,1	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	AMC	Left & Right Motor	0X167	40	8	0,1	0,512	0,256	0,128	0,064
						Total % Bus load =	6,9184	3,4592	1,7296	0,8648

Table 9.1 shows all the CAN messages on the CAN bus. Some of the messages are predefined by the manufacturer of the battery and the charger and the rest of them are defined based on the application of the ATR. All the application messages are provided with the priority. The lower the priority value, the higher the priority of the message. The priority-based arbitration mechanism is used on the CAN bus. Lower the CAN-ID of the message higher the priority of the message. The data length code (DLC) is also chosen based on the transmitted data in a CAN message. The DLC can be from 1 to 8 bytes as one CAN message can hold up to 8 bytes or 64 bits of information data. The frequency of CAN messages is also selected based on the requirement of data by the receiving nodes.

9.1. Percentage Bus Loading by Individual CAN Messages

Here, the percentage bus load occupied by individual CAN messages are calculated on different baudrate of CAN bus. The formula used to calculate the bus load by individual message is:

$$\%\ Bus\ Load\ by\ a\ CAN\ Message = \frac{fr * 8 * DLC}{Bd} * 100$$

where, fr = frequency of the CAN message, DLC = Data length Code and Bd = baudrate of the CAN bus.

For each message on the CAN bus, the percentage bus load by individual message is calculated at a baudrate of 125 Kbits/s, 250 Kbits/s, 500 Kbits/s and 1 Mbits/s. The baudrate affects the bus load on the CAN bus which can be seen in the fig. The individual CAN message shows a higher bus load when then the baudrate is lower. All the individual CAN messages show the highest bus load at 125 Kbits/s and the lowest bus load at 1 Mbits/s. By increasing the baudrate of the CAN bus, the bus load by the individual message is decreasing.

The maximum percentage bus load by a message is 1.024% at 125 Kbits/s, 0.512% at 250 Kbit/s, 0.256% at 500 Kbits/s and 0.128% at Mbits/s which is acquired by top-unit box status message (tuc_HB_msg), right motor drive status message (right_motor_drive_status_msg) and left motor drive status message (left_motor_drive_status_msg). As the frequency of these messages is the highest, which is 20 Hz (0.05 ms), the bus load occupied by these messages is maximum.

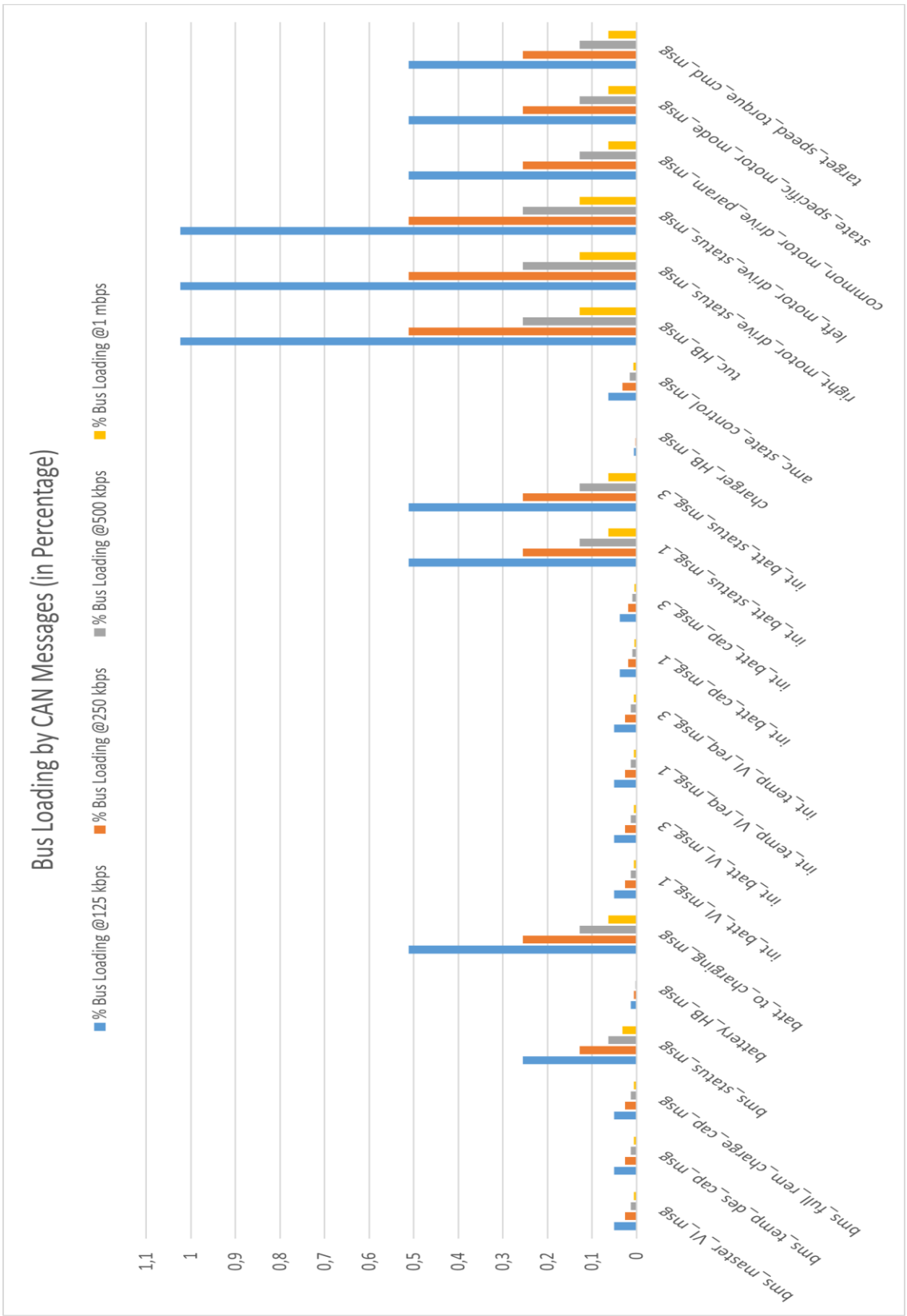

Figure 9.1: Bar Graph of Bus Loading by Individual CAN Messages

The minimum percentage bus load by a message is 0.0064% at 125 Kbits/s, 0.0032% at 250 Kbit/s, 0.0016% at 500 Kbits/s and 0.0008% at Mbits/s which is acquired by charger heartbeat message (charger_HB_msg). As the frequency of this message is lowest which is 1 Hz (1 sec) and the data length code for this message is only 1, which is 1 byte of data to be transmitted on the bus in one CAN message, that's why the bus load occupied by this message is minimum. It shows that the bus load depends on the frequency of the message and the data length code of the CAN message.

There are some messages which are always available on the CAN bus like the messages transmitted by the battery which includes battery internal messages between the individual batteries and the battery application messages which are the messages sent by the BMS system which combines all the individual batteries as a whole BMS unit. The AMC state control message (amc_state_contol_msg), top-unit box status message (tuc_HB_msg), right motor drive status message (right_motor_drive_status_msg) and left motor drive status message (left_motor_drive_status_msg) are also always present on the CAN bus.

The remaining messages on the CAN bus are transmitted based on the application and requirements in the different states. The CAN messages like charger heartbeat message (charger_HB_msg) and BMS charging message (batt_to_charging_msg) are only available on the CAN bus when the ATR is connected to the charger. The CAN messages like common motor drive parameter message (common_motor_drive_param_msg), state-specific motor mode message (state_specific_motor_mode_msg) and target speed/torque command message (target_speed_torque_cmd_msg) are only available when the ATR is in driving mode, can be manual driving mode with joysticks or in the autonomous mode.

9.2. Total Maximum Percentage Bus Loading by all the CAN Messages at different Baudrate

The total percentage of bus load differs at different baudrate of CAN bus. In the below fig., on the x-axis, it has maximum bus load at a baudrate of 125 Kbits/s, 250 Kbits/s, 500 Kbits/s and 1 Mbits/s whereas on the y-axis, it has bus load in percentage. The total maximum bus load at 125

Kbits/s is 6.9184 %, at 250 Kbits/s is 3.4592%, at 500 Kbits/s is 1.7296% and at 1 Mbits/s is 0.8648%. The bus load at 125 Kbit/s is the highest while the bus load at 1 Mbits/s is the lowest.

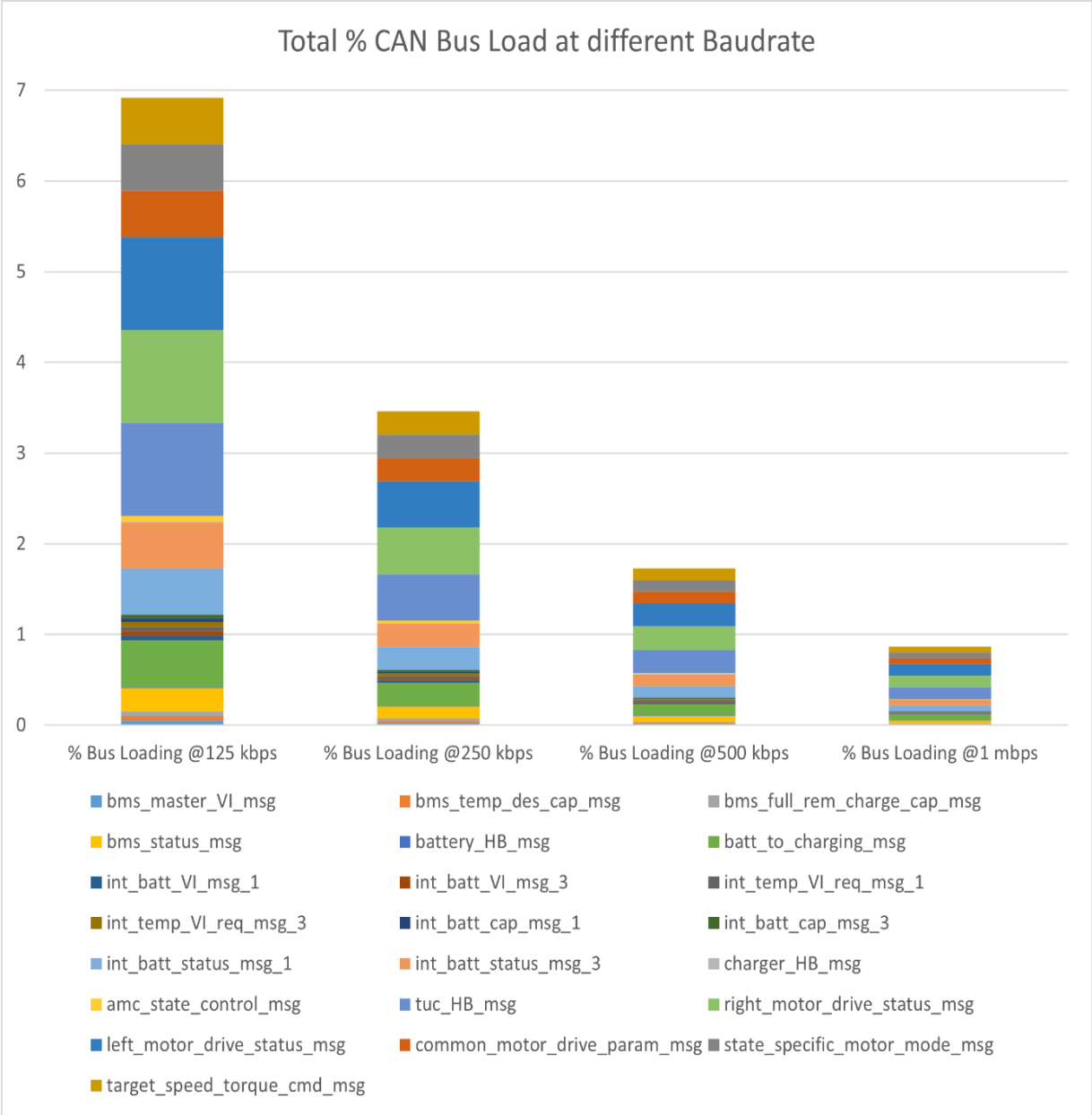

Figure 9.2: Bar Graph of Total percentage CAN Bus Load at different baudrate

This trend shows that when the baudrate is lower, the bus load is higher and while increasing the baudrate, the bus load on the CAN bus is decreasing.

9.3. Total Maximum Percentage Bus Loading by all the CAN Messages at different Baudrate in various State

The below table shows the maximum percentage bus load by the CAN messages at various ATR States. The bus load was calculated at different baudrate like 125 Kbits/s, 250 Kbits/s, 500 Kbits/s and 1 Mbits/s at various ATR states.

Table 9.2: Percentage Bus load at different baudrate in various ATR states

States	% Bus Loading @125 Kbps	% Bus Loading @250 Kbps	% Bus Loading @500 Kbps	% Bus Loading @1 Mbps
Start-up State	5,312	2,656	1,328	0,664
Idle State	6,4	3,2	1,6	0,8
Manual State	6,912	3,456	1,728	0,864
Pre-Maneuvering State	6,4	3,2	1,6	0,8
Maneuvering State	6,912	3,456	1,728	0,864
Transport State	6,912	3,456	1,728	0,864
Charging State	6,4064	3,2032	1,6016	0,8008
E-Stop State	6,4	3,2	1,6	0,8
Pre-Shutdown State	6,4	3,2	1,6	0,8

In *Figure 9.3*, on the x-axis, it has various ATR states, and, on the y-axis, it has bus load in percentage. These bus loads are calculated at various baudrates which can be seen as with different color lines.

The bus load at 125 Kbits/s is the highest in all the states while the bus load at 1 Mbits/s is the lowest in all the states. The bus load is the highest in all the ATR driving states like Manual Move state, Maneuvering state and Transport states because the number of CAN messages transmitted on CAN bus in these states are also more as all the driving parameter CAN messages are transmitted in these states. The bus load fluctuation on the lower baudrate is also high as on 125 Kbits/s, the bus load fluctuates approximately between 5% to 7% whereas on the higher baudrate, the bus fluctuation is very low as on 1 Mbits/s, the bus load fluctuates approximately between 0.6% to 0.8%.

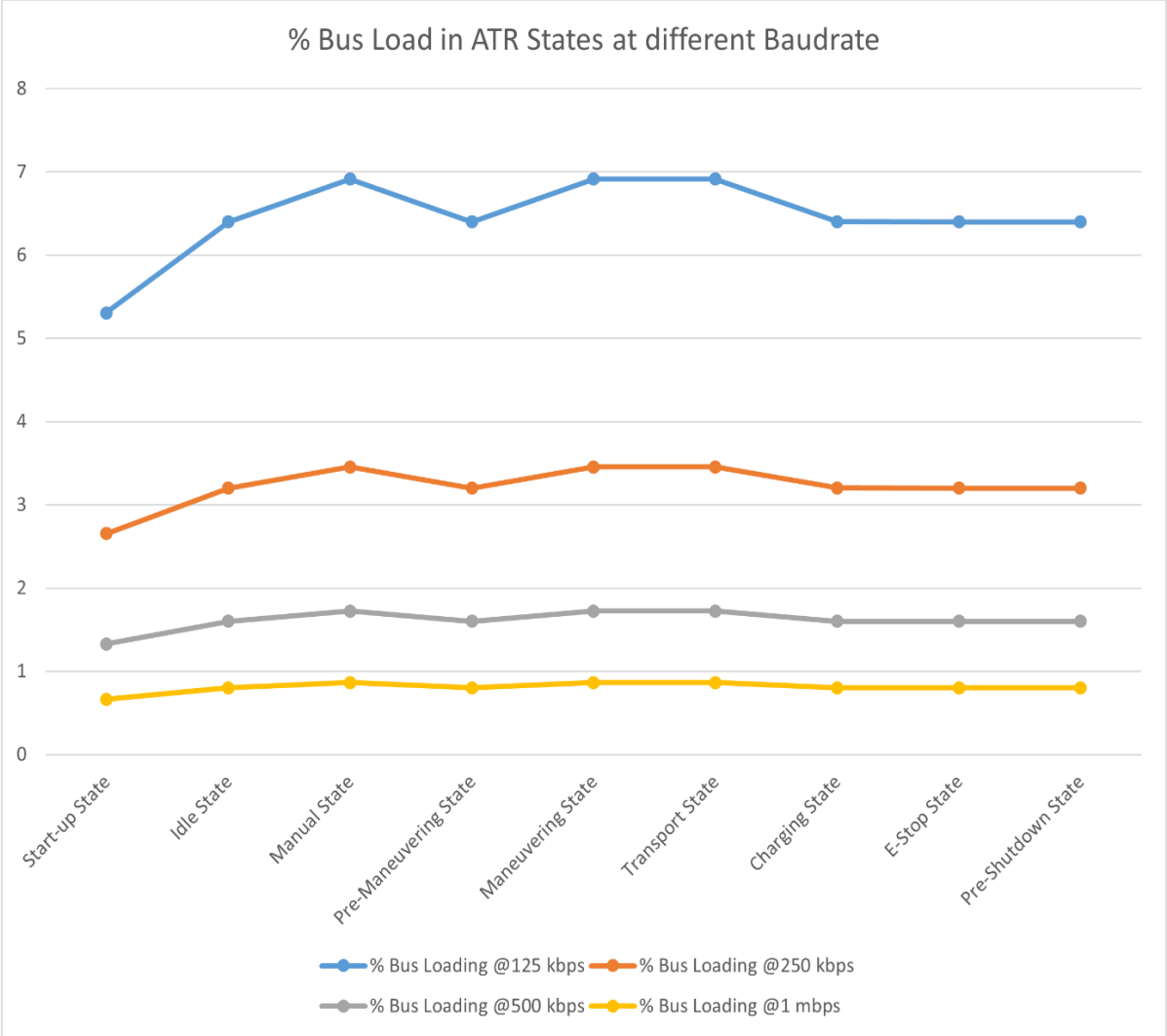

Figure 9.3: Line Graph of Percentage Bus load in ATR States at different baudrate

9.4. Selection of Baudrate for the CAN Bus of the ATR

The baudrate for the CAN bus is selected to be 250 Kbits/s. This particular baudrate was selected for the CAN bus keeping in mind that the maximum bus load at 250 Kbits/s is just 3.4592% which is quite low and good as it leaves a lot of room for addition of more CAN messages in the future on the ATR. At this baudrate, the speed of message transmission on the bus is fast enough for smooth operation of the motors, when the motor speed commands are sent from the AMC node over the CAN bus. All the nodes on the CAN bus are also compatible with the 250 Kbits/s baudrate. At this baudrate, the interrupt service routine for the incoming CAN messages to AMC and TUC is not overcrowded. If the baudrate is increased more like 500 Kbits/s or 1 Mbits/s,

there could be abnormal behavior because of hardware constraints of the AMC and TUC controllers as higher baudrate can generate more processing burden on the controllers and can have signal integrity issues because it could have higher noise content or reflections in the signal. If the baudrate is decreased to 125 Kbits/s, there is a higher bus load fluctuation across different states. Hence, a baudrate of 250 Kbits/s seems to have less fluctuation in bus load and still have decent bus transmission speed to run the motors without any issue. So, choosing the baudrate as 250 Kbit/s is the best choice.

10. CONCLUSION

In this thesis, a distributed control system for the ATR is developed. One of the benefits of using a distributed control system is that it reduces the response time of the system. The load on each controller in the distributed control system is also reduced as the total load of the machine is shared between different controllers. In the case of GPSS, the ATR computational load is distributed between the AMC and TUC controllers as in where AMC does most of the decisive computation like deciding the state of the ATR while TUC does more of the sensory computation like reading and writing the peripherals and I/O pins.

The ATR is working as a state machine in which several operating states are defined. It is important for the ATR to be in only one state at a time otherwise a conflict can occur if ATR is in two different states at the same time. To eliminate this problem of having conflict among the different states claiming the ATR state at the same time, a priority-based state transition is developed where all the states have been assigned with a priority value as which state has higher significance over the other states like the emergency states (E-Stop Activated state or Error State) are prioritized over the driving states (Maneuvering state or Transport State) keeping in mind the safety regulations.

An interrupt service routine is developed for AMC and TUC controllers. This interrupt service routine helps the controller to ensure that all the necessary CAN messages arriving on the CAN bus are received by the controller and no important messages are missed or lost in transmission. A IRQ list and a working list is introduced to ensure receiving all the CAN messages. The IRQ list regularly gets filled by the data from incoming CAN messages and is copied to the working list. The working list is used for reading with received data from the CAN bus while the IRQ list is receiving all the incoming data during that time. Once the working list is read completely, the IRQ list which was receiving all the incoming data during that time is copied, halted and copied to the working list. The IRQ list is then cleared and is ready to receive new CAN message data. This interrupt service routine helps to ensure that all the CAN messages on the bus are received even while reading the CAN message parallelly.

The CAN bus is selected to run at a baudrate of 250 Kbits/s. The maximum bus load of the ATR is 3.4592% which leaves a lot more room for additional CAN messages to be added in future if

needed. Priority-based arbitration is used for all the CAN messages to avoid message collision on the CAN bus. The frequency or the repetition rate of the CAN messages are defined based on the needs of the individual nodes like motors requiring the Target speed/torque command message (target_speed_torque_cmd_msg) more frequently at 20 Hz than the BMS charge capacity message (bms_charge_cap_msg) which is required at 1 Hz to update the SoC of the batteries.

The maximum response time of the ATR is secured to be 200 ms. The maximum loop time in all the states of the AMC is stabilized to be 100 ms by adding appropriate delays in various states which is done to have a constant AMC loop time irrespective of its state. The TUC also has a maximum loop time of 50 ms in all the states. The loop time of TUC is kept almost half of the AMC to ensure that the TUC loop runs twice while the AMC is running once ensuring that even if AMC misses one top-unit box status message, it can get the next one. The general flow of the control is that the input data is received by TUC I/O and sent as a CAN message to AMC which takes 50 ms. The AMC receives and reads the message from TUC, then it evaluates the state change conditions and based on that changes the state and performs state specific tasks which takes about 100 ms. Then it orders TUC to change its states and light up the state relevant LEDs on the top unit box, this also takes 50 ms. All these three steps occur when most of the state transition occurs. By defining a fixed loop time of AMC and TUC, the maximum response time of the ATR is assured to not exceed more than 200 ms.

REFERENCES

[1] https://www2.deloitte.com/us/en/insights/focus/future-of-mobility/electric-vehicle-trends-2030.html (accessed on 24th March 2023).

[2] Carolina Villarreal Lozano, Kavin Kathiresh Vijayan, Literature review on Cyber Physical Systems Design, Procedia Manufacturing (2020), 295-300.

[3] Parthasarathy Guturu, Bharat Bhargava, Cyber-Physical Systems: A Confluence of Cutting-Edge Technological Streams.

[4] Ioan Stefan Sacala, Mihnea Alexandru Moisescu, Ioan Dumitrache Calin Aurel Munteanu, Simona Iuliana Caramihai, Cyber Physical Systems Oriented Robot Development Platform, Procedia Computer Science (2015), 203-209.

[5] https://store.arduino.cc/products/arduino-mkr-wifi-1010 (accessed on 24th April 2023).

[6] https://store.arduino.cc/products/arduino-mkr-can-shield (accessed on 25th April 2023).

[7] https://simplexmotion.com (accessed on 26th April 2023).

[8] https://www.varta-ag.com/en/industry/product-solutions/lithium-ion-battery-packs/asb/easyblade (accessed on 26th April 2023).

[9] https://delta-q.com/our-products/icl-series/ (accessed on 26th April 2023).

[10] https://www.winstar.com.tw/products/smart-display/can-display/wl0f00050000fgaaasa01.html (accessed on 27th April 2023).

[11] Foukarakis, M., Leonidis, A., Antona, M., Stephanidis, C. (2014). Combining Finite State Machine and Decision-Making Tools for Adaptable Robot Behavior. In: Stephanidis, C., Antona, M. (eds) Universal Access in Human-Computer Interaction. Aging and Assistive Environments. UAHCI 2014. Lecture Notes in Computer Science, vol 8515. Springer, Cham.

[12] Schneider, F.B. (1990). The state machine approach: A tutorial. In: Simons, B., Spector, A. (eds) Fault-Tolerant Distributed Computing. Lecture Notes in Computer Science, vol 448. Springer, New York, NY.

[13] Jigang Wang, Research on CAN bus consistency test method. IOP Conference Series: Earth and Environmental Science, Volume 560 (2020).

[14] Zhenwu Shi, Fumin Zhang, An analytical model of CAN bus for online schedulability test, Georgia Institute of Technology (2012).

[15] B. Mishra and A. Kertesz, "The Use of MQTT in M2M and IoT Systems: A Survey," in IEEE Access, vol. 8, pp. 201071-201086 (2020).

[16] Niu, Zuoling. (2021). Research and Implementation of Internet of Things Communication System Based on MQTT Protocol. Journal of Physics: Conference Series. vol. 2023.

[17] Rocha, C. et al. (2020). Development of an Autonomous Mobile Towing Vehicle for Logistic Tasks. In: Silva, M., Luís Lima, J., Reis, L., Sanfeliu, A., Tardioli, D. (eds) Robot 2019: Fourth Iberian Robotics Conference. ROBOT 2019. Advances in Intelligent Systems and Computing, vol 1092.

[18] Giuseppe Fragapane, René de Koster, Fabio Sgarbossa, Jan Ola Strandhagen, Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda, European Journal of Operational Research, Volume 294, Issue 2, 2021, Pages 405-426.

[19] Koseoglu, Murat & Celik, Orkan & Pektas, Omer. (2017). Design of an autonomous mobile robot based on ROS. 1-5. 10.1109/IDAP.2017.8090199.

ABSTRACT

This thesis presents the development of a control system for Autonomous Transport Robots (ATR) which will be used in the Volvo Factories. The research also evaluates the timing and frequency of the control messages sent over the Controller Area Network (CAN) bus of the ATR. The ATR is a part of a cyber-physical system having control parameters over Wi-Fi and CAN communication.

The ATR has a distributed control system with two controllers, which work simultaneously and interdependently. One controller is the center of the decision-control system, and the other controller handles peripherals, buttons, and sensors. The ATR is developed as a state machine having different operating states. Each operating state has its own tasks to perform and a certain set of state transitions. The ATR uses CAN communication to communicate with different nodes present on the ATR. To provide reliability of the CAN message, an interrupt service routine is developed for both the controllers. This interrupt service routine helps the controller to ensure that all the necessary CAN messages arriving on the CAN bus are received by the controller and no important messages are missed or lost in transmission. For Wi-Fi communication, MQTT Protocol is used to communicate with the edge controller.

The response time of both the controllers as well as the response time of the ATR is calculated in all the ATR states and a constant response time of the ATR for every state is proposed. The frequency of the CAN messages is evaluated and a bus load on the CAN bus is calculated at different baud rates. A suitable baud rate for the CAN bus on the ATR is proposed.

Overall, this thesis contributes to the development of a robust control system for ATR, ensuring reliable communication over both CAN and Wi-Fi interfaces. The analysis of response times and bus loads aids in optimizing the performance and efficiency of the ATR in various operational scenarios.

Keywords: Autonomous Transport Robot, Control System, Frequency Analysis, Timing Analysis, CAN bus, Baud rate, Bus load, State Machine, MQTT Protocol

BIOGRAPHY

ADAMYA SHUKLA

Adamya Shukla was born on 30th December 1996 in Sultanpur, Uttar Pradesh, India. He completed Bachelors of Technology in Electrical and Electronics Engineering in 2019 from Karunya Institute of Technology and Sciences, Coimbatore, India. He did his Bachelors thesis at Sao Paulo State University (UNESP), Botucatu, Brazil. He was awarded with Chancellors awards for achieving highest grade among all departments of engineering. In 2020, he enrolled in the graduate study program, Automotive Computing and Communications, in the Faculty of Electrical Engineering, Computer Science, and Information Technology (FERIT) at Josip Juraj Strossmayor University of Osijek.

During his graduate studies, he did couple of internships. He did his first internship in the field of Embedded Systems in the department of Hardware/Software Co-Design at Friedrich-Alexander University Erlangen-Nuremburg, Germany in 2021. He did his second internship in the field of Embedded Software Development in the department of Volvo Global Trucks Operations at Volvo Trucks, Gothenburg, Sweden in 2022. His field of interests are Automotive Software and Electronics, Embedded Systems, Robotics and AI.

ANNEXES

ANNEX A: Percentage Bus load in Start-Up State at various Baudrate

Frame Name	Start-up State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0	0	0	0
amc_state_control_msg	0	0	0	0
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0	0	0	0
state_specific_motor_mode_msg	0	0	0	0
target_speed_torque_cmd_msg	0	0	0	0

ANNEX B: Percentage Bus load in Idle State at various Baudrate

Frame Name	Idle State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0	0	0	0
amc_state_control_msg	0,064	0,032	0,016	0,008
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0	0	0	0
state_specific_motor_mode_msg	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	0,512	0,256	0,128	0,064

ANNEX C: Percentage Bus load in Manual State at various Baudrate

Frame Name	Manual State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0	0	0	0
amc_state_control_msg	0,064	0,032	0,016	0,008
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0,512	0,256	0,128	0,064
state_specific_motor_mode_msg	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	0,512	0,256	0,128	0,064

ANNEX D: Percentage Bus load in Pre-Maneuvering State at various Baudrate

Frame Name	Pre-Maneuvering State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0	0	0	0
amc_state_control_msg	0,064	0,032	0,016	0,008
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0	0	0	0
state_specific_motor_mode_msg	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	0,512	0,256	0,128	0,064

ANNEX E: Percentage Bus load in Maneuvering State at various Baudrate

Frame Name	Maneuvering State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0	0	0	0
amc_state_control_msg	0,064	0,032	0,016	0,008
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0,512	0,256	0,128	0,064
state_specific_motor_mode_msg	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	0,512	0,256	0,128	0,064

ANNEX F: Percentage Bus load in Transport State at various Baudrate

Frame Name	Transport State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0	0	0	0
amc_state_control_msg	0,064	0,032	0,016	0,008
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0,512	0,256	0,128	0,064
state_specific_motor_mode_msg	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	0,512	0,256	0,128	0,064

ANNEX G: Percentage Bus load in Charging State at various Baudrate

Frame Name	Charging State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0,0064	0,0032	0,0016	0,0008
amc_state_control_msg	0,064	0,032	0,016	0,008
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0	0	0	0
state_specific_motor_mode_msg	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	0,512	0,256	0,128	0,064

ANNEX H: Percentage Bus load in E-Stop Activated State at various Baudrate

Frame Name	E-Stop State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0	0	0	0
amc_state_control_msg	0,064	0,032	0,016	0,008
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0	0	0	0
state_specific_motor_mode_msg	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	0,512	0,256	0,128	0,064

ANNEX I: Percentage Bus load in Shutdown Preparation State at various Baudrate

Frame Name	Pre-Shutdown State			
	% Bus Loading @125 kbps	% Bus Loading @250 kbps	% Bus Loading @500 kbps	% Bus Loading @1 mbps
bms_master_VI_msg	0,0512	0,0256	0,0128	0,0064
bms_temp_des_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_full_rem_charge_cap_msg	0,0512	0,0256	0,0128	0,0064
bms_status_msg	0,256	0,128	0,064	0,032
battery_HB_msg	0,0128	0,0064	0,0032	0,0016
batt_to_charging_msg	0,512	0,256	0,128	0,064
int_batt_VI_msg_1	0,0512	0,0256	0,0128	0,0064
int_batt_VI_msg_3	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_1	0,0512	0,0256	0,0128	0,0064
int_temp_VI_req_msg_3	0,0512	0,0256	0,0128	0,0064
int_batt_cap_msg_1	0,0384	0,0192	0,0096	0,0048
int_batt_cap_msg_3	0,0384	0,0192	0,0096	0,0048
int_batt_status_msg_1	0,512	0,256	0,128	0,064
int_batt_status_msg_3	0,512	0,256	0,128	0,064
charger_HB_msg	0	0	0	0
amc_state_control_msg	0,064	0,032	0,016	0,008
tuc_HB_msg	1,024	0,512	0,256	0,128
right_motor_drive_status_msg	1,024	0,512	0,256	0,128
left_motor_drive_status_msg	1,024	0,512	0,256	0,128
common_motor_drive_param_msg	0	0	0	0
state_specific_motor_mode_msg	0,512	0,256	0,128	0,064
target_speed_torque_cmd_msg	0,512	0,256	0,128	0,064