Aalto University
School of Electrical
Engineering

Master’s programme in ICT Innovation

Connected Vehicle Platooning: Stability Analysis on an Experimental Vehicle Platform

Niladri Dutta

Master’s Thesis
2023

Aalto University
School of Electrical Engineering

Author Niladri Dutta

Title Connected Vehicle Platooning: Stability Analysis on an Experimental Vehicle Platform

Degree programme ICT Innovation

Major Autonomous Systems

Supervisor Prof. Themistoklis Charalambous

Advisor Thomas Höglund

Date 24 April 2023 **Number of pages** 58+7 **Language** English

Abstract

Rapid development of driving assistance technology and vehicular communication for intelligent transportation systems has proved that it can improve the safety, efficiency and sustainability of vehicles. This thesis endeavours to develop an experimental platform to demonstrate the use of cooperative adaptive cruise control (CACC) systems. The work analyzes existing longitudinal controllers and their string stability in homogeneous platoons.Furthermore, field tests are carried out using the longitudinal controller with multiple autonomous experimental vehicle platforms to verify the effectiveness of the controller. According to the results of simulation and field tests, the proposed CACC platooning approach shows great benefits for longitudinal vehicle platooning.

Keywords cooperative adaptive cruise control, vehicle platoon, multiple-predecessor following topology, communication delays, robot operating system.

Preface

I would like to first express my gratitude to my thesis supervisor, Dr. Themistoklis Charalambous, for his invaluable support and patience throughout the thesis. Dr. Charalambous allowed the development of the research work to be my own, providing me with all the necessary resources to make it possible. I want to thank him for always being available for a discussion when I had a question regarding the research and for steering me in the right direction whenever needed.

I couldn't have done it without the members of the Distributed and Networked Control Systems (DNCS) research group, particularly Elham Abolfazli and Dr. Wei Jiang, who generously shared their expertise and passionately participated in providing feedback at various stages of my thesis. Furthermore, without the generous support of the Department of Electrical Engineering and Automation, which funded the project's research, this endeavour would not have been possible.

I also thank my classmates and cohort members, for their editing help, critical feedback, and moral support.

Finally, I am grateful to my family, particularly my parents and sister. Their confidence in me has kept my spirits and motivation high throughout.

Espoo, 24 April 2023

Niladri Dutta

Contents

Symbols and abbreviations

Symbols

ω	Angular Velocity
τ	Time lag in the powertrain
Δ	Time delay in the wireless communication
d	Desired stand still gap
h	Time headway
u	Throttle control signal input
r	Number of preceding vehicles, directly ahead of a vehicle
p	Position of a vehicle
v	Velocity of a vehicle
a	Acceleration of a vehicle

Operators

$\frac{\mathrm{d}}{\mathrm{d}t}$	derivative with respect to variable t
$\sum_i$	sum over index i

Abbreviations

CACC	Cooperative Adaptive Cruise Control
V2V	Vehicle-to-Vehicle
EKF	Extended Kalman Filter
PLF	Predecessor Leader Following
TPF	Two Predecessor Following
MPF	Multi Predecessor Following
ROS	Robot Operating System
IMU	Inertial Measurement Unit
OBC	On-board Computer
WLAN	Wireless Local Area Network
LAN	Local Area Network
GCS	Ground Control Station
IFT	Information Flow Topology
EPV	Experimental Platform Vehicle

1 Introduction

1.1 Background

Increased dependency on automobiles for transportation poses a high demand for road infrastructure. A possible solution is the use of onboard sensors and inter-vehicular wireless communication to allow cooperation between vehicles.

Vehicle platoons are one of the proposed Intelligent Transportation System (ITS) solutions that are gaining popularity in recent times. Vehicle platoons employ a form of autonomous or semi-autonomous cruise control in a group of vehicles that move together maintaining a specified inter-vehicular distance. Such systems improve road network throughput by reducing traffic congestion and improving traffic safety reducing total travel time. The reduced air drag on each vehicle also reduces fuel consumption and exhaust emissions. Thus, such systems would help road networks accommodate increasing traffic and minimize traffic accidents [3] [4] [5].

1.2 Problem Definition

1.2.1 Vehicle Control Algorithms

Autonomous and semi-autonomous vehicles use control algorithms to set the velocity and direction, which in turn determine the desired trajectory to follow. The right control algorithm would prevent the vehicle from driving off the lanes and causing a collision or traffic congestion.

This work approaches the longitudinal controller design challenge while assuming the lateral guidance control is managed separately [11][13][14][27][41]. When multiple vehicles use longitudinal controllers simultaneously, they form a vehicle platoon.

1.2.2 Connected Vehicle Platooning

A vehicle platoon is a group of interconnected vehicles travelling in close proximity while maintaining inter-vehicular distance. Each vehicle is a subsystem that keeps track of its vehicle states (position, velocity, acceleration, etc.) using measurement sensors. Vehicle subsystems communicate their vehicle states with other subsystems (vehicle) through an inter-vehicle wireless communication method. The platoon consists of a lead vehicle followed by n followers, with indexes $i = 1, 2, 3, \ldots, n$. The follower vehicles maintain a set inter-vehicle distance (d_i) to their predecessor vehicles, closely matching their speed and manoeuvres. The leader can be human-operated or programmed to autonomously follow a given trajectory, which would also make the whole platoon fully autonomous. [40]

Since the vehicle systems are interconnected to a form platoon, it simultaneously couples all their sub-system dynamics to a singular system. Therefore, along with the understanding of a single vehicle's dynamics, the dynamical properties in the longitudinal and lateral domains are required for the whole platoon. [17]

If all vehicles in a platoon are identical, i.e. they have the same dynamics and physical constraints, then it is classified as a homogeneous platoon. If a platoon consists of non-identical vehicles then it is defined as a heterogeneous platoon.

In this thesis, all the experiments are conducted on homogeneous platoons. Each vehicle in the platoon is equipped with the same controllers, sensors, and communication systems to facilitate inter-vehicular cooperation [13]. Figure 1 shows a diagram with a homogeneous connected vehicle platoon consisting of five vehicles.

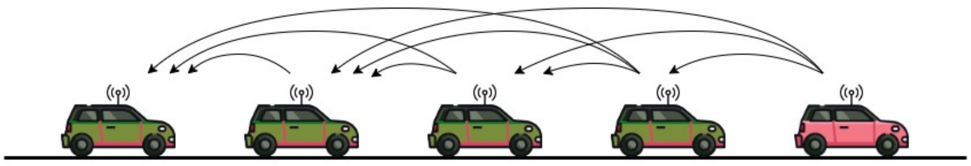


Figure 1: A five-vehicle homogeneous platoon. The arrows indicate the information flow over the wireless communication network.

1.2.3 Experimental Platform Design for Platooning

The reliability of a controller's design can be determined when it can be verified in scenarios in both simulation and real-world cases. Virtual simulators are inexpensive to test the controllers but may lack the ability to accurately consider the influence of the physical world on the system.

Since a full-size vehicle is expensive to set up and can be a safety hazard, a better approach is to first implement a small-scale model of the scenario with a robotic hardware platform.

This thesis describes the development of hardware and software for a vehicle platooning system that has similar behaviour compared to full-size vehicle platoons. The aim has been to provide an environment to test platooning control algorithms rapidly. This experimental platform makes the transition from theory to simulation to real-world hardware on vehicles quicker, allowing for faster improvements before they are tested on a full-scale vehicle.

The controller on this platform consists of two parts - a high-level controller which is being tested for the purpose of the vehicle platooning application, and a low-level controller which controls the vehicle propulsion system based on the commands from the high-level controller. Making the high-level controller isolated from the low-level controller makes it easier to use the same high-level controller in a similar full-size vehicle, where the low-level controller may be different due to the difference in vehicle propulsion hardware. Section 4.2 discusses the details of the software developed to implement various high-level controllers under test.

1.3 Research Motivation

There have been several research projects that demonstrate the use of a well-designed control system for various predecessor leader following (PLF) systems in simulation environments, but the study of these controllers in physical environments with hardware components has been limited. The motivation behind this project is to be able to demonstrate distributed controllers on robot hardware platforms and provide more compelling evidence of the benefits of vehicle platooning which can encourage both private and public sector stakeholders to deploy this functionality in the near future.

This thesis aims to focus on the implementation of distributed controllers on multiple-vehicle systems in a physical environment. By experimenting on vehicle hardware, the proposal is to study the effect of communication limitations and environmental uncertainties (disturbances) on homogeneous platoons. The comparison would be on the basis of parameters such as spacing policy, network topology, time headway, etc., in dynamic physical environments.

Thus, the thesis would also have the possibility to consider scenarios to determine if the controller algorithms are able to maintain overall system stability for a platoon leader with time-varying velocity or in the event of a change in the number of vehicles.

1.4 Objectives

To study the multiple-predecessor following topology for vehicle platoons, this thesis would use various control parameters. The study of the effect of these parameters on hardware systems operating in a physical test environment is the primary objective of the thesis. Researchers have suggested multiple parameters which affect the vehicle platoon, some are mentioned below.

- *Spacing policy:* It is the policy for the separation between two consecutive vehicles.
- *Communication delays:* Delays caused by transmission over a wireless network.
- *System stability (internal and string stability):* A platoon's capacity to minimise the impact of disturbances.

Other factors that influence platoons are the wireless network topologies used by the platoon vehicles and uncertainties due to external disturbances caused by environmental factors like wind gusts or road slopes. These factors are not analysed in this thesis.

Another objective of this thesis is the development of the hardware and software platform for running experiments related to vehicle platoons. This system needs to have a similar behaviour when compared to a full-size vehicle platoon, which would make the transition from theory to simulation to real-world hardware on vehicles quicker, allowing for faster improvements before they are tested on a full-scale vehicle.

1.5 Thesis Outline

This thesis consisted of six chapters and two appendices. The outline of each chapter is summarized as follows:

- *Chapter 1* introduces the thesis by describing the motivation behind this research and its objectives.
- *Chapter 2* presents an overview of vehicle platooning algorithms and reviews previous work on controllers used for cooperative longitudinal control.
- *Chapter 3* discusses the vehicle state estimation methodologies using sensor data and presents the use of Robot Operating System (ROS) based architecture as a framework for inter-vehicle and intra-vehicle information transportation.
- *Chapter 4* introduces the hardware and software systems developed during this thesis.
- *Chapter 5* demonstrates the proposed various approaches discussed in the thesis and presents the results of the experiments for these approaches.
- *Chapter 6* summarizes the work and recommends future improvements for research.
- *Appendix A* presents additional technical specifications of the experimental platform developed in this thesis.
- *Appendix B* briefly describes the Robot Operating System (ROS) software framework.

2 Vehicle Platoon Control

2.1 Overview

In this chapter, the theoretical foundation for vehicle platooning that will be used throughout the thesis report is introduced. Section 2.2 is a review of past work covering the underlying concepts like graph theory followed by an explanation of cooperative systems and platoon topologies. In section 2.5.1, we discuss the longitudinal platooning used in this work, detailing the vehicle dynamic model, controller design, and the methodology for the system stability analysis.

2.2 Cooperative Systems

2.2.1 Cruise Control

A real-world active automation system called cruise control keeps moving at a predetermined speed that the driver chooses. Most highway roads are composed of long, flat, low-curvature paths, which makes it possible for programming an embedded system to assist the driver with the vehicle's speed regulation task. The majority of contemporary commercial vehicles come with cruise control and the use of cruise control systems in long-distance vehicles is extremely common. By controlling vehicle propulsion, this system reduces driver fatigue and improves comfort while also maximising the vehicle's energy efficiency [15]. Figure 2 illustrates a common control structure for cruise control systems.

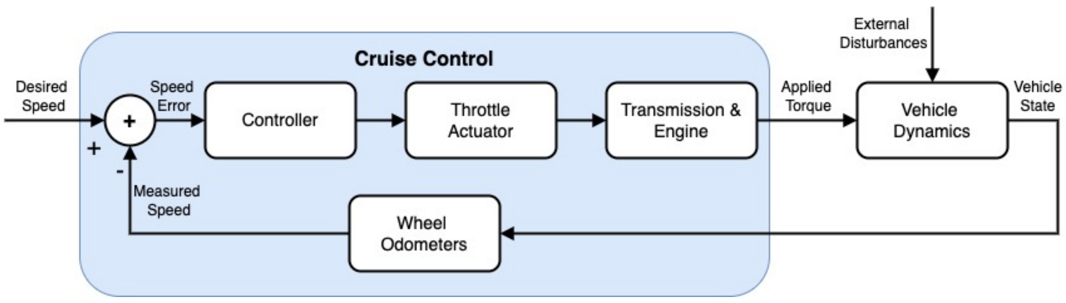


Figure 2: Control structure of a Cruise Control system.

2.2.2 Adaptive Cruise Control (ACC)

Cruise control systems provide satisfactory results but fail to perform optimally in high-traffic scenarios on highways and urban areas. In high-traffic situations, the user must frequently change the set speed during traffic intersections, congestion, etc. The Adaptive Cruise Control (ACC) system was developed to mitigate the shortcomings of the cruise control system. Adaptive Cruise Control uses brake actuators and ranging sensors, to detect preceding vehicles and dynamically set the cruising speed of the vehicle. Figure 3 illustrates a common control structure for ACC systems.

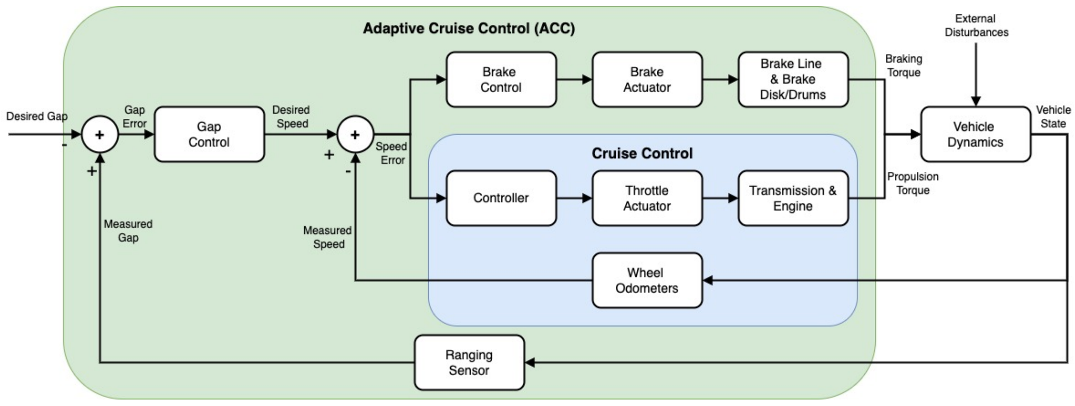


Figure 3: Control structure of an Adaptive Cruise Control (ACC) system.

2.2.3 Cooperative Adaptive Cruise Control (ACC)

Cheaper sensor technology has made it easier to equip cars with ACC and motivated research in autonomous car-following systems. This improved system is called Cooperative Adaptive Cruise Control (CACC).

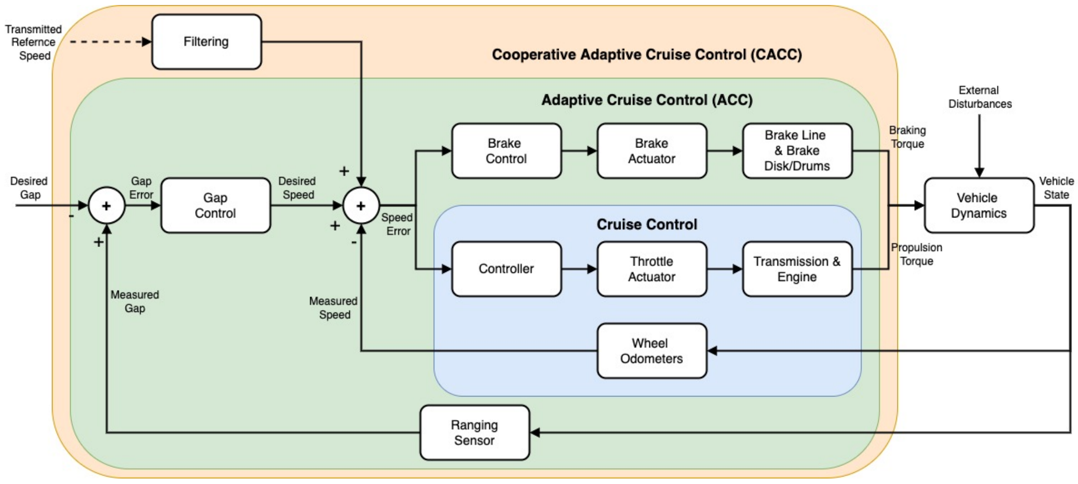


Figure 4: Control structure of a Cooperative Adaptive Cruise Control (CACC) system.

Figure 4 illustrates a common control structure for CACC systems. This system takes advantage of vehicle-to-vehicle (V2V) communication channels between ACC-equipped vehicles to exchange data about the vehicle's state (position, velocity and acceleration) [16]. When there are speed oscillations in the leading vehicles, CACC permits quicker reaction time for the following vehicle's onboard system to control its speed.

The objectives of CACC are listed below [35] :

- Shorter distances between vehicles travelling in the same lane are made possible by increasing traffic throughput without sacrificing stable vehicle following.

- By improving the ability to communicate with moving vehicles and the response time for transmitting their changes, rear-end collision risk can be decreased.
- Enhance driver comfort by dampening propagated string disturbances and acting as an ADAS.
- If distances are close enough, decrease fuel consumption by enhancing traffic flow and aerodynamic drafting.

2.3 Platooning Topologies

A vehicle platoon includes a leader vehicle and N follower vehicles, i.e. a total of $N + 1$ vehicles. Vehicles in a platoon communicate with each other using various information flow topologies. Some common information flow topologies are shown in Figure 5, which include [43]:

(a) Predecessor following topology (PF)

(b) Predecessor-leader following topology (PLF);

(c) Bidirectional topology (BD);

(d) Bidirectional-leader topology (BDL);

(e) Two predecessors following topology (TPF);

(f) Two predecessor-leader following topology (TPLF).

(g) Multiple-predecessor following (MPF).

In the information exchanged between the vehicles can be one or more states of vehicle systems, that is, position, velocity, and acceleration. In this thesis, we have demonstrated results from experiments using Predecessor Following Topology (PF), Two predecessors following topology (TPF) and Multiple-predecessor following (MPF) topologies.

2.3.1 Multiple Predecessor Following

Multiple predecessor following (MPF) is a platooning strategy which involves the information flow from multiple predecessor vehicles in the control loop, starting from the immediate predecessor vehicle. This method uses weighted information from the leading part of a vehicle platoon to achieve better performance while braking, emerging or splitting the platoon[15].

It is ideal for scenarios where intra-platoon interruptions may occur, e.g. a pedestrian crossing or a collision of a platoon member vehicle. It also permits the upstream vehicles to react at the moment a perturbation occurs in the platoon downstream, even before perceiving the perturbation from the immediate predecessor.

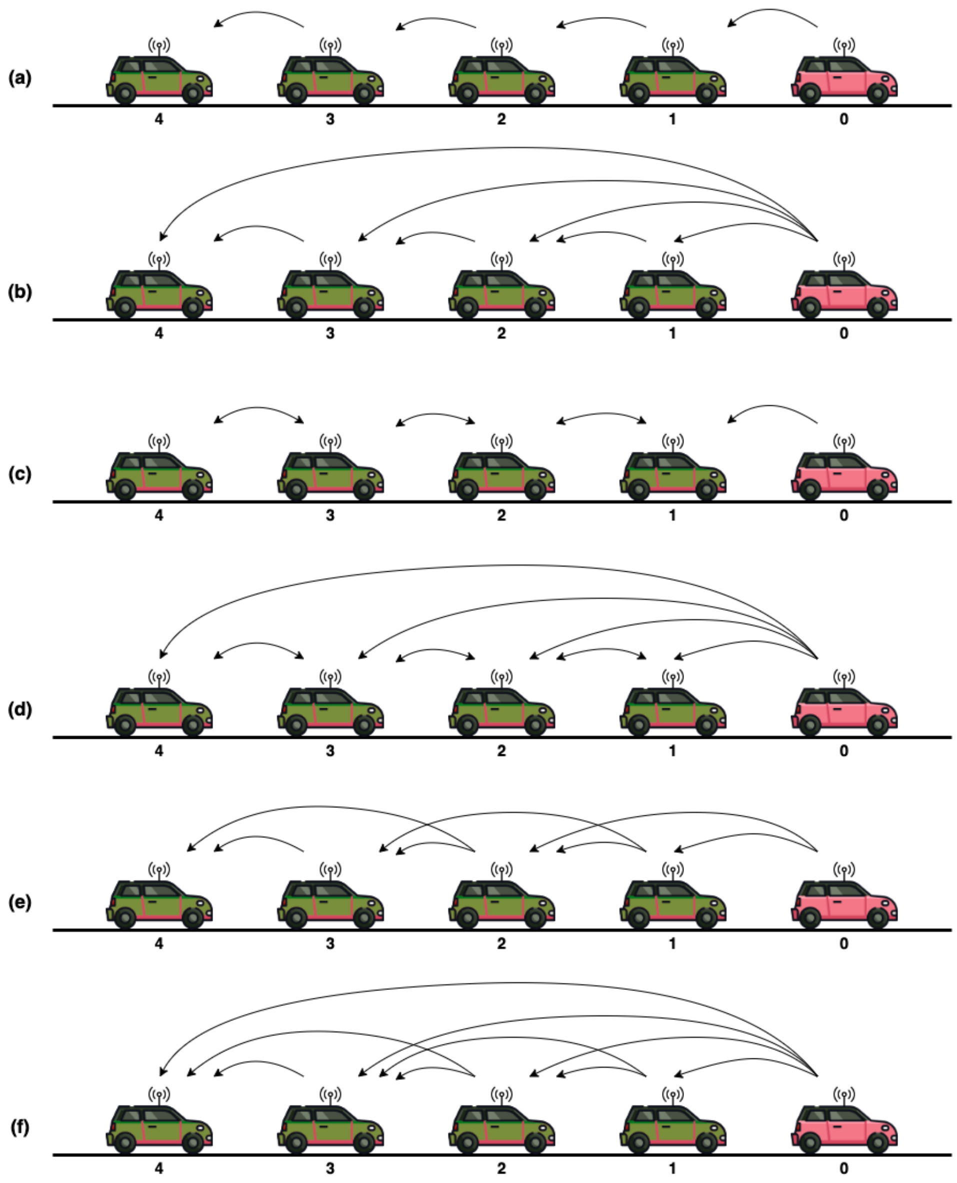


Figure 5: Platooning Topologies

Ploeg et al. [25] have discussed the two-vehicle look-ahead approach, comparing it with the predecessor following strategy. In the two-vehicle look-ahead approach, the controller of the i-th vehicle utilised the state of the $(i-1)$-th and $(i-2)$-th vehicles to compute the motor throttle command. This approach also shows good string stability results.

Thus, this is the approach that is the focus of this thesis, and experiments in later sections demonstrate its advantages.

2.4 Graph Theory

In this thesis, the information flow between networked dynamic vehicle systems is modelled with the directed graph concept from graph theory. A directed graph is

In this thesis, the information flow between dynamic vehicle systems are modelled with the directed graph concept from graph theory [1]. Considering a networked dynamic vehicle system, with one leader vehicle and N following vehicles with M communication links. Then, the following vehicles are represented by a set of nodes $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ which are connected by edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ representing the information flow through a between pairs of following vehicles. The resulting directed graph is represented by $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

The dynamic vehicle system's network topology is represented by the Laplacian matrix $\mathcal{L}$ and the incidence matrix D. The Laplacian matrix $\mathcal{L} = [l_{ij}], i, j \in \mathbb{N}^N$ is associated with $\mathcal{G}$, where

$$l_{ij} = \begin{cases} -a_{ij}, & i \neq j \\ \sum_{k=1}^{N} a_{ik}, & i = j \end{cases} \tag{1}$$

where $a_{ij} = 1$ represents information flow from vehicle j to vehicle i.

$$a_{ij} = \begin{cases} 1, & (v_j, v_i) \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Here a uni-directional communication structure is assumed where vehicles are able to receive information only from their predecessors, and hence $a_{ij} = 0$ if $j > i$.

The connections between the vehicles and the leader can be modelled by [2],

$$\mathcal{P} = \mathbf{diag}\{p_{11}, p_{22}, ..., p_{NN}\} \tag{3}$$

where

$$p_{ii} = \begin{cases} 1, & \text{if vehicle } i \text{ is connected to the leader} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Then, a new information topology matrix can be defined as

$$\mathcal{L}_p = \mathcal{L} + \mathcal{P} \tag{5}$$

where $\mathcal{L}_p$ is a lower triangular matrix.

2.5 Longitudinal Platooning

2.5.1 Longitudinal String Stability

The longitudinal string stability component of vehicular platooning will be the main emphasis of this work. The longitudinal controller is used to control the distance between two successive platoon vehicles.

Any platoon member (vehicle i) braking or accelerating will cause oscillations between vehicles i and vehicle $(i-1)$, which can spread further down the platoon if there is no inter-vehicle communication. The slinky effect [36], another name for this phenomenon, is depicted in figure 6.

If the first vehicle in the platoon, i.e. the leader vehicle ($i = 0$) decelerates by braking, the distance to the next vehicle (vehicle 1), d_1 decreases. As a reaction to this, the internal controller on vehicle 1 also commands its motor controller to decelerate, causing d_2 to decrease. As a result, d_1 increases, and a similar oscillation motion propagates throughout the entire platoon's inter-vehicle distances (d_i).

The "slinky effect" may be amplified further by wireless communication lag, sluggish sensor data processing, and actuator lag. The platoon's spacing policy is maintained with stable margins by using a longitudinal controller that satisfies the string stability criterion and guarantees that longitudinal oscillations are reduced.

It is assumed that the lateral controller present in the system is uncoupled from the longitudinal controller, hence the idea of lateral string stability is not taken into account in this thesis.

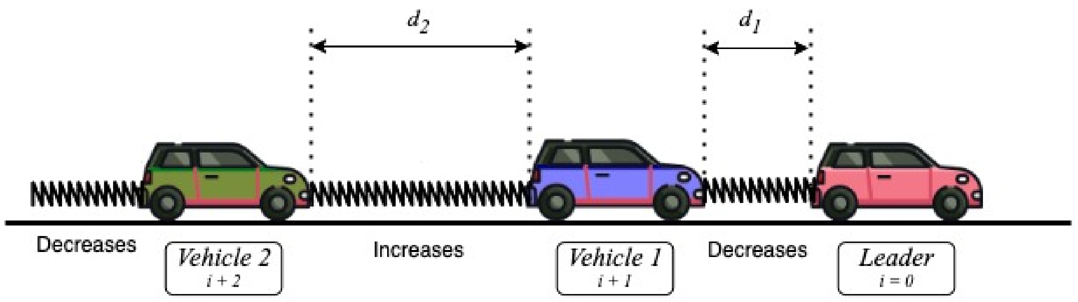


Figure 6: Slinky effect in the longitudinal aspect of platooning.

2.5.2 Vehicle Dynamics Model

In this thesis work, the following vehicle model is considered. The vehicle model used in this work is referred from various previous research publications, like [25]. In a platoon consisting of N vehicles following the vehicle model given by,

$$\begin{cases} \dot{p}_i(t) & = v_i(t), \\ \dot{v}_i(t) & = a_i(t), \\ \tau_i \dot{a}_i(t) + a_i(t) & = u_i(t), \end{cases} \tag{6}$$

where for vehicle i, $p_i(t)$ is the position, $v_i(t)$ is the velocity, $a_i(t)$ is the acceleration, $u_i(t)$ is the control input and the time-lag in the powertrain is represented by $\tau_i > 0$. In multiple predecessors following, multiple vehicles preceding vehicle i transmit their vehicle state information to vehicle i. Figure 7 shows the nomenclature for the predecessor and successor of vehicle i, which are all wirelessly connected to each other. The desired distance between vehicle i and the l-th vehicle ahead of vehicle i [10] is,

$$d_{i,i-l}(t) = \sum_{k=i-l+1}^{i} (h_k v_k(t) + d_k), \tag{7}$$

where $h_k \geq 0$ is the time-headway of the k-th vehicle and the expected standstill distance from the rear of vehicle $k-1$ to the front of vehicle k is given by $d_k > 0$.

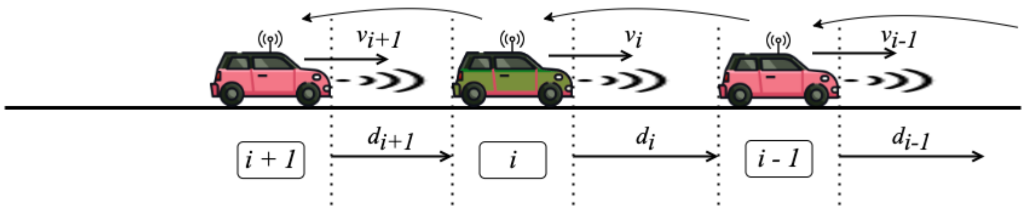


Figure 7: Platooning Notation

2.5.3 Controller Design

The objective of to designing a controller for a vehicular platoon is to achieve individual node stability and the platoon's string stability.

A linear feedback controller is used to determine the control signal input to the vehicle at the i^{th} position. [10]

$$u_i(t) = \sum_{l=1}^{r_i} \Big(k_{pi} \big(p_i - p_{i-l} + \sum_{k=i-l+1}^{i} (h_k v_k + d_k) \big) + k_{vi}(v_i - v_{i-l}) + k_{ai}(a_i - a_{i-l}) \Big), \tag{8}$$

where the number of the vehicles preceding vehicle i is given by $r_i \leq i$. For a heterogeneous platoon following the MPF topology, r_i may be different for each vehicle, since each vehicle's control command can be determined by the states of a different number of predecessors. The feedback on distance, velocity, and acceleration errors between vehicle i and its predecessor vehicles can be adjusted by tuning the k_{pi}, k_{vi}, and k_{ai} control gains.

2.5.4 Problem Formulation

The i-th vehicle's controller has access to the states of its predecessor vehicles due to a local wireless communication network established between the platooning vehicles. As a result, the difference between its own states and all predecessors can be determined. It is assumed that wireless communication has a uniform time delay Δ. The control law can be modified to include the communication time delays in the following equation based on the controller in equation (8) proposed in [10]. Using $u_i'(t) = u_i(t - \Delta)$ as the control input for vehicle i.

$$u_i'(t) = \sum_{l=1}^{r_i} \Big[k_{pi}\big\{p_i(t-\Delta) - p_{i-l}(t-\Delta) + \sum_{k=i-l+1}^{i} (h_k v_k(t-\Delta) + d_k)\big\} \\ + k_{vi}\big\{v_i(t-\Delta) - v_(t-\Delta)\big\} + k_{ai}\big\{a_i(t-\Delta) - a_{i-l}(t-\Delta)\big\}\Big] \tag{9}$$

2.6 Stability Analysis

In this section, we present the stability analysis studied by Abolfazli *et al.* [1]. The errors in the platooning vehicle states are defined by,

$$\begin{cases} \bar{p}_i(t) = p_i(t) - p_0(t) + \Sigma_{k=1}^{i}(h_k v_k(t) + d_k) \\ \bar{v}_i(t) = v_i(t) - v_0(t) \\ \bar{a}_i(t) = a_i(t) - a_0(t) \end{cases} \tag{10}$$

The leading vehicle is assumed to be moving at a constant speed, hence $u_0(t) = 0$ and $a_0(t) = 0$.
Using equation (22), the dynamics equation is expressed as,

$$\begin{cases} \dot{\bar{p}}_i(t) = v_i(t) + \Sigma_{k=1}^{i}(h_k \bar{a}_k(t)) \\ \dot{\bar{v}}_i(t) = \bar{a}_i(t) \\ \dot{\bar{a}}_i(t) = -\frac{1}{\tau}\bar{a}_i(t) + \frac{1}{\tau}u_i(t) \end{cases} \tag{11}$$

By modifying equation (10) the control law from equation (9) can be written as,

$$u_i(t) = -\sum_{l=1}^{r_i} \Big[k_{pi}\big\{\bar{p}_i(t-\Delta) - \bar{p}_{i-l}(t-\Delta)\big\} + k_{vi}\big\{\bar{v}_i(t-\Delta) - \bar{v}_(t-\Delta)\big\} \\ + k_{ai}\big\{\bar{a}_i(t-\Delta) - \bar{a}_{i-l}(t-\Delta)\big\}\Big] \tag{12}$$

We define the augmented errors,

$$\bar{p} = [\bar{p}_1, \bar{p}_2, ..., \bar{p}_N]^\top \\ \bar{v} = [\bar{v}_1, \bar{v}_2, ..., \bar{v}_N]^\top \\ \bar{a} = [\bar{a}_1, \bar{a}_2, ..., \bar{a}_N]^\top$$

Substituting equation (12) into equation (11). The closed-loop network dynamics model can be written as,

$$\dot{\xi} = A\xi(t) + A_\Delta \xi(t-\Delta), \tag{13}$$

$$\xi = \Phi(t), \qquad t \in [-\Delta, 0] \tag{14}$$

where the state vector is

$$\xi = [\bar{p}^\top, \bar{v}^\top, \bar{a}^\top]^\top \tag{15}$$

the system's initial state is given by

$$\Phi(\cdot) \in \mathbb{C}([-\Delta, 0], \mathbb{R}^\nu), \tag{16}$$

and other parameters of this closed loop equation system are,

$$A = \begin{bmatrix} 0 & I_N & H \\ 0 & 0 & I_N \\ 0 & 0 & -T \end{bmatrix} \qquad \in \mathbb{R}^{\nu\times\nu)}, \nu = 3N \tag{17}$$

$$A_\Delta = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -TK_p\mathcal{L}_p & -TK_p\mathcal{L}_p & -TK_p\mathcal{L}_p \end{bmatrix} \qquad \in \mathbb{R}^{\nu\times\nu)}, \nu = 3N \tag{18}$$

$$K_m = \text{diag}\{k_{m1}, \dots k_{mN}\}, \qquad m \in \{p, v, a\}, \tag{19}$$

$$T = \text{diag}\{1/\tau_1, \dots 1/\tau_N\}, \tag{20}$$

$$H = \begin{bmatrix} h_1 & 0 & \cdots & 0 \\ h_1 & h_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ h_1 & h_2 & \cdots & h_N \end{bmatrix} \tag{21}$$

2.6.1 Internal Stability

The objective of designing a controller for a vehicular platoon is to guarantee the individual node stabilities and the platoon's overall string stability. Internal stability indicates that the closed loop system is stable, and thus vehicles in the platoon can track the desired inter-vehicle distance and maintain the desired velocity.

Mathematically, the following conditions are required for internal stability, with $1 \le l \le r$,

$$\begin{cases} \lim_{t\to\infty} \left(p_i(t) - p_{i-l}(t) + \sum_{k=i-l+1}^{i}(h_k v_k(t) + d_k)\right) = 0, \\ \lim_{t\to\infty} \left(v_i(t) - v_(t)\right) = 0, \\ \lim_{t\to\infty} \left(a_i(t) - a_{i-l}(t)\right) = 0 \end{cases} \tag{22}$$

The sufficient condition for internal stability is stated according to the following theorem [1],

Theorem 1 *By selecting the control gains (k_{pi} , k_{vi} , k_{ai}) such that the following conditions hold*

$$k_{pi} > 0, \tag{23a}$$

$$k_{ai} > 0, \tag{23b}$$

$$k_{pi} - \tau_i(k_{vi} + k_{pi}h_i) + \tau_i^2 k_{pi} \neq 0, \tag{23c}$$

$$k_{vi} + k_{pi}h_i) \ge k_{pi}\tau_i, \tag{23d}$$

the closed loop system in equation (13) is asymptotically stable for any time delay Δ that satisfies the following inequality

$$\Delta r_i(k_{vi} + k_{pi}h_i) < 1, i \in \mathbb{N} \tag{24}$$

2.6.2 String Stability

It is essential for a vehicular platoon to ensure that all vehicles in the platoon perform the same manoeuvre when an external disturbance is introduced, for example, while changing lanes or a sudden deceleration for collision avoidance [43]. Thus, along with following the internal stability of the individual sub-systems, the interconnected vehicles must also respect the conditions for the stability of the complete platooning system. This concept is called string stability which is described as; "The attenuation of errors propagating in an upstream direction of interconnected vehicles forming a platoon" [12]. This means that string stability is achieved when a perturbance due to an external disturbance is introduced into the system of platooning vehicles and does not propagate along the platoon.

In a homogeneous platoon, the vehicles are identical and the parameters can be simplified to $\tau_i = \tau = 0, r_i = r, h_i = h, k_{pi} = k_p, k_{vi} = k_v, k_{ai} = k_a, \forall i \in \mathbb{Z}_0^n$.

The spacing error is defined as,

$$e_i = p_i - p_{i-1} + hv_i + d_i \tag{25}$$

and the laplace tranform of $e_i(t)$ is

$$E_i(s) = \sum_{l=1}^{r} H_l(s)E_{i-l}(s) \tag{26}$$

where

$$H_l(s) = \frac{k_a s^2 e^{-\Delta s} + \{k_v - k_p h(r-l)\}se^{-\Delta s} + k_p e^{-\Delta s}}{\tau s^3 + s^2 + rk_a s^2 e^{-\Delta s} + r(k_v + k_p h)se^{-\Delta s} + rk_p e^{-\Delta s}} \tag{27}$$

Since the platoon is homogeneous, $H_l(s)$ in equation 27 is identical for all member vehicles. In the work by Bian *et al.*, the authors mention that string stability is guaranteed if, (substituting $s = j\omega$ in 27),

$$||H_l(j\omega)||_\infty \leq \frac{1}{r}, \forall 1 \leq l \leq r \tag{28}$$

Theorem 2 *Consider the system in equation (6) with the input given by equation (9) that is internally stable. Then, the string stability specification in equation (28) holds if all the following conditions are satisfied:*

$$k_v + k_p(h - \tau) \geq 0, \tag{29a}$$

$$2\tau\Delta - \Delta h - \tau h \leq 0, \tag{29b}$$

$$k_a - \tau(k_v + k_p h) \leq 0, \tag{29c}$$

$$\tau - 2rk_a\Delta \geq 0, \tag{29d}$$

$$1 + 2r\big(k_a - \tau(k_v + k_p h)\big) + 2r\Delta\big(k_p(\tau - h) - k_v) - k_v\big)0, \tag{29e}$$

$$r^2k_p^2h2\big(1 - (r - l)^2\big) + 2r^2k_pk_vh(1 + r - l) - 2rk_p \geq 0, 1 \leq l \leq r \tag{29f}$$

$$k_a > 0, \tag{29g}$$

$$k_p > 0, \tag{29h}$$

$$\tag{29i}$$

for the region defined by equation (29)*, there exists a set of feedback gains k_p, k_v and k_a, such that string stability specification in equation 28 holds if,*

$$h \geq h_{min} = \frac{2(\tau + \Delta)}{2rk_a + 1} \tag{30}$$

3 Vehicle State Estimation

3.1 Overview

This chapter introduces the method required to achieve an accurate state estimation of each vehicle in the platoon using the Cooperative Adaptive Cruise Control (CACC) system. The chapter discusses the sensors used on the vehicles and the sensor fusion algorithms.

3.2 Sensors for State Estimation

A vehicle operating with the Cooperative Adaptive Cruise Control (CACC) can be considered an autonomous mobile robot vehicle, which needs to act intelligently to develop a map of its surroundings and maintain localisation to remain in a stable vehicle platoon formation with the other member vehicles in the platoon.

Sensory organs in animals, allow them to sense and interact with their surroundings. Similarly, autonomous mobile vehicles are equipped with popular sensor systems that provide them with the ability to sense their environment and navigate through it safely. The following is a list of some of the most popular sensors installed in autonomous vehicles [7].

- **Ultrasonic Rangefinder** emits sound pulses that bounce off objects and are received by the receiver. The distance between the objects is determined by the time of flight of the reflected pulse. The ultrasonic rangefinder system is simple to use but only provides accurate distance measurements for large, solid objects at close ranges.

- **IMU (Inertial Measurement Unit)** consists of sensors which are used to detect the physical motion of the vehicle. Typically, an IMU had 6 degrees of freedom (6 DoF), i.e. three accelerometers and three gyroscopes, measuring the corresponding values in the x, y and z axis. Accelerometers measure the acceleration values and the gyroscope measures the angular velocities. Thus, with these measurements, it is possible to calculate the three-dimensional position and orientation of the IMU.

- **Odometry (Wheel encoders)**: are mounted on the wheels of vehicles, to count the number of turns of the rotating wheels while the robot is in motion. By accumulating the information about wheel rotation, it is possible to use a method called 'dead reckoning' to trace the path taken by the vehicle with respect to the initial position of the vehicle. The dead reckoning algorithm has a high chance of accumulating large errors due to reasons like - wheel slipping, road conditions, or the resolution of the encoder.

- **LIDAR (Light Detection and Ranging)** is a sensor that emits pulsed laser light beams and monitors the reflected beams to detect objects in its surroundings. LIDAR sensors can be focused on smaller areas at greater distances with much higher accuracy than radar. LIDAR sensors are expensive, but the cost is decreasing as the demand for autonomous vehicles is increasing steadily.

- **RADAR (Radio Detection and Ranging)** systems on autonomous vehicles emit radio waves with radar transmitters that reflect off environmental objects and return to the receiver of the RADAR system. The information can be used to model environmental objects. RADAR operates well in long ranges and in most weather types, but isn't useful in object identification.
- **GNSS (Global Navigation Satellite System)** is a system on an autonomous vehicle consisting of a GNSS receiver which obtains time and location information in numerical coordinates (e.g. latitude, longitude) from a global satellite network.
- **Stereo Cameras** is capable of providing image and video data which could be used to recognise objects. Using stereo cameras, it is possible to get depth information about the surroundings and model real-time 3D scenes.

Each type of sensor has its strengths and weaknesses, making it imperative for autonomous vehicle systems to combine data from multiple sensors to generate the best model of its environment and make the best decision to navigate around obstacles.

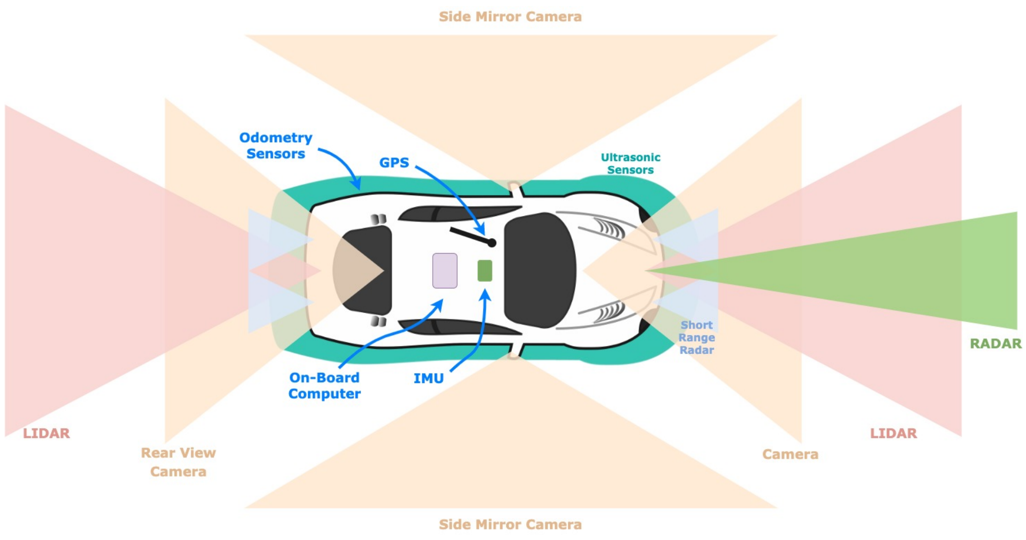


Figure 8: Sensors for vehicle state estimation on an autonomous vehicle.

3.3 Sensor Fusion

An autonomous vehicle in a connected platoon moves on the road and its environment is not static, thus it uses sensor fusion to merge data from its various sensors to improve its own localization. Based on the localisation information, it calculates the best trajectory to follow the desired path. Sensor fusion provides numerous advantages to autonomous robots [18]:

- increases the accuracy, reliability, and fault tolerance of sensor data;
- extends spatial and temporal coverage of sensor systems;
- by improving the resolution, it allows better recognition of the robot's surroundings, which is critical in dynamic environments;
- by using algorithms for data preprocessing, sensor fusion reduces the cost and complexity of a robot;
- allows multiple sensors to be used without modifying the robot's software or hardware architecture.

3.3.1 Bayesian Filters

The most popular algorithm for the state estimation of a robot is using Bayesian state estimation. These filters are named after their application of Bayes' law, [23]:

$$P(A|B) = \frac{P(B|A)}{P(B)}P(A) \tag{31}$$

Bayes' law states that "$P(A|B)$ [the probability of event A occurring given that event B has occurred], can be calculated by $P(B|A)/P(B)$ [the normalised probability of event B occurring given that event A has occurred], multiplied by $P(A)$ [the probability of event A occurring]".

Bayes' law allows the estimation of an unobservable event based on the information in the observable data. For example, to estimate the likelihood of having a disease given a positive result, Bayes' law is applied with the observable quantities, i.e. false positives and false negatives, as input for the calculations.

The equation for a Bayesian Filter can be derived using equation (31), which states Baye's law, and considering the Markov assumption, where the current state depends purely on the previous state and not on any other state before it ([23]).

$$P(s_t|z_t, u_t) = \eta P(z_t|s_{t-1})P(s_{t-1}|u_t) \tag{32}$$

In this case, η is a normalization factor, z_t is a system observation, u_t is a transition that modifies the state, and s_t is the current state vector. According to Montella, equation (32) is the basis of all Bayesian state estimators [23]. There are many popular Bayesian Filter algorithms like the Kalman Filter, Extended Kalman Filter (EKF) and Particle Filter, which are used based on the scenario.

The Kalman Filter algorithm propagates a system's state, characterised by a Gaussian distribution, using linear transition functions over time[23]. The Kalman Filter is one of the few algorithms that have been proven to be an optimal solution for its domain.

The Kalman Filter only applies to a strict set of problems consisting of linear state transition and linear measurements with added Gaussian noise. This condition applies to many real-world problems, but some problems consist of non-linear transitions. To address these problems, the Extended Kalman Filter (EKF) was invented. The EKF includes the step to linearise non-linear transition functions and observation functions using Taylor Series expansion. [19] [23]

In this work, the Extended Kalman Filter has been used to fuse sensor data and estimate the robot's states (position, velocity and acceleration).

4 Experimental Platform: Design and Development

While designing a longitudinal controller for a Cooperative Adaptive Cruise Control system it is necessary to conduct a series of experiments, which should demonstrate results that are close to practical scenarios. Therefore, a reliable vehicle dynamics model must be used. This chapter presents the CACC algorithm used for this thesis. Then the physical constraints and limitations are considered to make realistic hardware in a loop experimental environment. Finally, the experimental vehicle model is explained.

4.1 Hardware Platform

The goal of this project is to analyse and improve platooning algorithms used in autonomous vehicles. As a result, the experimental platform vehicle must be designed to fulfil the following requirements in order to simulate an experiment analogous to vehicle platooning on the real world highways.[33]

1. The dynamics of the hardware experimental platform must be similar to that of a full-scale vehicle. (For this experiment, the dynamics must be similar at least in the longitudinal direction).
2. The experimental platform must be able to achieve the speeds necessary to demonstrate platooning.
3. In the event of a controller failure, the robot must be strong enough to withstand collisions with other robots travelling.

Taking these requirements into account, the Magni robot platform of Ubiquity Robotics was selected. The Magni robot has a metal frame, which can withstand collisions. Even though the Magni robot model is a differential drive robot, rather than an Ackermann model steering system, the dynamics are similar to a full-scale vehicle in the longitudinal direction, which is the scope of the experiments in this project. The robot can achieve speeds of up to 1m/s (or higher with software updates), which is enough to simulate experimental results in a closed lab environment. A detailed description of the experimental platform vehicle is available in Appendix A.

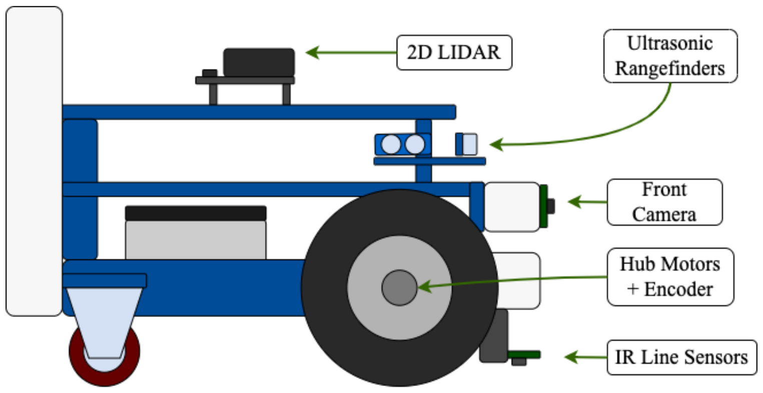


Figure 9: A side view of the Magni robot with sensors and actuators.

4.2 Software System

The software system would complement the hardware components of the experimental platform, which would together make the system complete. This section discusses the software architecture and the choices for the components used to design the system's software.

4.2.1 Software Architecture

The software system for the experimental platform must be designed to accommodate the various requirements of the experimental setup. Some of the most important requirements are listed below:

- The experimental platform shall have the ability to run various CACC controllers with minimal effort in configuring new controllers.
- The software system must allow the functionality to record sensor data and control commands.
- The performance of the controller can be affected by large delays, so the software system shall be designed to minimise the overhead due to data processing and other software applications.
- The software shall be modular, making it easier to accommodate prototyping and allowing the development of sub-components for future updates.

Taking into account these requirements, the Robot Operating System (ROS) was chosen as a framework for the implementation of the software architecture. ROS is an open-source middleware system which supports modules developed using C++ and Python, making it extremely useful for prototyping experimental platforms. This framework facilitates message passing and the scheduling of task execution. It also provides a simple data-logging interface and the flexibility to choose computer hardware. Due to these advantages, ROS suits the requirements of the experimental platform, and it was selected as the software framework for this system. A description of the ROS framework is presented in Appendix B.

In the work by Rajamani [28], the usage of a split-controller configuration is discussed for automotive applications. In this configuration, a high-level controller commands the linear and angular velocity/acceleration, while a low-level controller achieves the commanded velocities and accelerations to the necessary motors. To capture the commands from the high-level controller and relay them to the motors effectively, the low-level controller must run at a significantly higher rate (~50 Hz), compared to the high-level controller (~10 Hz).

The system is categorized into two parts: the experimental platform vehicle and the ground control station (GCS). The details of the software architecture and its various modules are described in the following subsections. Figure 10 illustrates the system architecture of the software running on the onboard computer in each experimental platform vehicle. It also shows the network interface modules that run in the experimental vehicle to facilitate data exchange with other vehicles in the platoon and at the GCS.

4.3 Experimental Platform Vehicle Software

The experimental vehicle's software system consists of four main categories, which are

- Physical Level (Sensor-Actuator Level)
- Low Level (Hardware Interfaces/Drivers)
- High Level (Perception, Localisation and Navigation)
- Application Level (Control and Network)

4.3.1 Sensor-Actuator Level

The sensors and motor controller are supplied with pre-installed firmware software provided by the Magni robot platform manufacturer. The experimental platform is installed with the following sensors:

- 2D Lidar
- Depth Camera
- Inertial Measurement Unit (IMU)
- Sonar Rangefinders
- IR Sensor Array

The motor controller is the only actuator in the experimental platform vehicle.

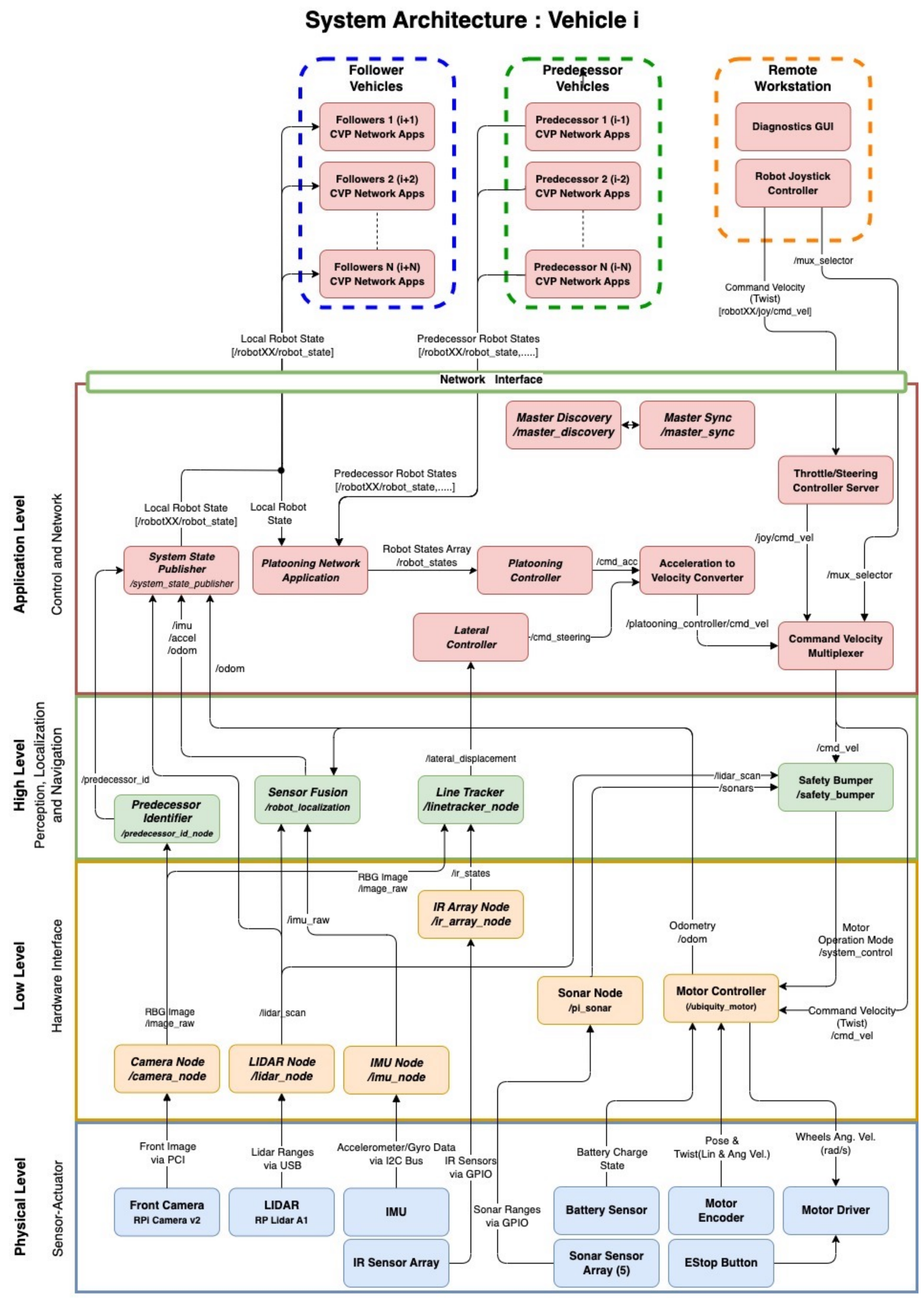

System Architecture : Vehicle i
Follower Vehicles
Followers 1 (i+1) CVP Network Apps
Followers 2 (i+2) CVP Network Apps
Followers N (i+N) CVP Network Apps
Predecessor Vehicles
Predecessor 1 (i-1) CVP Network Apps
Predecessor 2 (i-2) CVP Network Apps
Predecessor N (i-N) CVP Network Apps
Remote Workstation
Diagnostics GUI
Robot Joystick Controller
/mux_selector
Command Velocity (Twist) [robotXX/joy/cmd_vel]
Local Robot State [/robotXX/robot_state]
Predecessor Robot States [/robotXX/robot_state,.....]
Network Interface
Application Level
Control and Network
Master Discovery /master_discovery
Master Sync /master_sync
Predecessor Robot States [/robotXX/robot_state,.....]
Throttle/Steering Controller Server
Local Robot State [/robotXX/robot_state]
Local Robot State
System State Publisher /system_state_publisher
Platooning Network Application
Robot States Array /robot_states
Platooning Controller
/cmd_acc
Acceleration to Velocity Converter
/joy/cmd_vel
/mux_selector
/imu /accel /odom
/odom
Lateral Controller
/cmd_steering
/platooning_controller/cmd_vel
Command Velocity Multiplexer
High Level
Perception, Localization and Navigation
/predecessor_id
Predecessor Identifier /predecessor_id_node
Sensor Fusion /robot_localization
/lateral_displacement
Line Tracker /linetracker_node
/cmd_vel
/lidar_scan
/sonars
Safety Bumper /safety_bumper
Low Level
Hardware Interface
RBG Image /image_raw
/ir_states
IR Array Node /ir_array_node
/imu_raw
Odometry /odom
Motor Operation Mode /system_control
Sonar Node /pi_sonar
Motor Controller (/ubiquity_motor)
Command Velocity (Twist) /cmd_vel
RBG Image /image_raw
/lidar_scan
Camera Node /camera_node
LIDAR Node /lidar_node
IMU Node /imu_node
Physical Level
Sensor-Actuator
Front Image via PCI
Lidar Ranges via USB
Accelerometer/Gyro Data via I2C Bus
IR Sensors via GPIO
Battery Charge State
Pose & Twist(Lin & Ang Vel.)
Wheels Ang. Vel. (rad/s)
Front Camera RPi Camera v2
LIDAR RP Lidar A1
IMU
Sonar Ranges via GPIO
Battery Sensor
Motor Encoder
Motor Driver
IR Sensor Array
Sonar Sensor Array (5)
EStop Button

Figure 10: System Architecture Diagram

4.3.2 Hardware Interfaces

These modules consist of the hardware interfaces to communicate with the sensor and actuator firmware software. This includes the drivers for reading sensor data and sending command signals to the wheel motor controllers.

1. **Sensors**

   - **2D LIDAR Node:** The rplidar open-source ROS package runs the rplidarNode which provides the basic interface with the 2D Laser Scanner (RP-LIDAR A1) which is used on the experimental platform vehicle. This node converts the raw RP-LIDAR sensor measurements to the `sensor_msgs/LaserScan` ROS message format and publishes it on the `/scan` topic data stream [32].
   - **Camera Node:** The camera node (`camera_node`) uses the open source ROS package `raspicam_node` which publishes the raw RGB image data as a ROS topic called `/image_raw` of the ROS message type `sensor_msgs/Image` [38].
   - **Inertial Measurement Unit (IMU):** The IMU ROS package comprises a wrapper ROS node developed in this project for interfacing with the Altimu10v5 IMU module using the open source python package. Through this interface, it is possible to read the accelerometer, gyroscope, and magnetometer sensor measurements and publish them as the ROS message of type `sensor_msgs/Imu` on the `/imu` topic [20].
   - **Sonar Rangefinders:** The measurements from the onboard sonar rangefinders on the vehicle are published as ROS topics using the open-source Pi Sonar ROS Node by Ubiquity Robotics [37].
   - **IR Array Node:** The IR sensors were calibrated to detect the black line marking the test path. The node reads the IR sensor array measurements through GPIO pins and publishes the data as a topic (`/ir_states`) of ROS message type `std_msgs/Int32MultiArray`.

2. **Actuators**

   - **Low-Level Controller:** The low-level controller is a velocity controller which uses the PID control algorithm. It receives the control command from the high-level controller. The PID controller's equation is [6]

     $$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \tag{33}$$

     where time is represented by t and e(t) is the error between the desired state and the current state of the system at the time instance t. The proportional, integral and derivative gains of the system are shown as K_p, K_i and K_d.

4.3.3 Perception, Localisation and Safety

This category of software modules consists of software algorithms that use sensor data to perceive the vehicle environment, improve its localisation and ensure the safety of the system. The following is a description of the modules in this section:

1. **Predecessor Identifier:** Each vehicle has an Aruco code printed tag pasted on the back of the robot frame. Using the open-source Aruco detection ROS package, based on the shape and size of the tag, it is possible to identify an Aruco code tag and its relative position with respect to the camera. Thus, this information is used to identify the predecessor vehicle robot and publish the `/predecessor_id` topic, which is used by other ROS node modules to determine the arrangement of the vehicles in the platoon.

2. **Sensor Fusion:** This module's function is to subscribe to various sensor data sources and perform sensor fusion to improve the localization of the vehicle. Here, the open-source package `robot_localization` has been used to utilize its state estimation node consisting of an Extended Kalman Filter(EKF) `ekf_localization_node` to fuse the IMU readings, LIDAR scans and odometry data provided by the experimental robot vehicle's sensors. The resulting estimated state of the vehicle is published as filtered odometry data (`/odometry/filtered`), i.e. position, velocity (linear and angular), and filtered acceleration data (`/accel/filtered`) [24] [29].

3. **Line Tracker:** This module's main functionality is to determine the lateral displacement of the vehicle with respect to the marked path. The IR sensor array has been calibrated to detect the black line that marks the test path. Based on the measurements of the `/ir_states` given by the IR Array Node, this module determines the lateral displacement. This module also subscribes to the RGB image data from the Camera Node, processes the image to identify the line marking and calculates the lateral displacement with respect to the marked trajectory. The module then publishes the lateral displacement as a topic called `/lateral_displacement` of the ROS message type `std_msgs/Int8`

4. **Safety Bumper:** To prevent any collisions, based on rangefinder readings of nearby environmental objects from the LIDARs and the sonar sensor array, this node performs an emergency stop by disabling the motors if the obstacles are too close to the vehicle.

4.3.4 High-Level Applications: Vehicle Software

The high-level applications consist of most of the vehicle platooning-related software. This also includes applications used to communicate with other vehicles in the network.

1. **Platooning Applications**

(a) **Platooning System State Publisher:** The system state publisher node processes the sensor data to publish a custom system state ROS message consisting of the position, velocity, and acceleration of the robot vehicle system. These measurements are required for vehicle platooning controllers.

(b) **Platooning Network Application:** Since each vehicle on the platoon's network publishes its individual system state, each robot system must compile the necessary robot states of its predecessor and leader according to the required platoon topology. This node performs this function and republishes the required system states for a given robot vehicle.

(c) **Platooning High-Level Controller:** The high-level controller node is the primary controller of the platooning vehicle, and it consists of the various controllers under test, discussed in section 2.5.3. By varying the controller parameters on follower robot vehicles, the platoon stability study is carried out. The output is published as an `acceleration/throttle` command value in the range 0 to 1, which is an input to the low-level controller for the robot's motor control. The high-level controller for the leader robot just follows a predefined path. Therefore, it does not use the controller under test to determine the `acceleration/throttle` command.

(d) **Lateral Controller - Line Follower:** To evaluate the longitudinal platooning controller, all the experimental robot vehicles travel in a straight path marked with a black coloured line on the floor. Thus, each vehicle system must be calibrated to reduce the influence of the lateral motion controller. The implementation of a single vehicle path tracking was achieved by tuning the PID controller gains of each vehicle, such that the deviation from the marked line is minimised. The lateral controller gives the necessary turn command to the motors such that the vehicle returns back to the marked path when a lateral displacement is detected. Figure 11 illustrates the turn commands (marked in green) given to the motor controllers when (i) the vehicle is on the right path, (ii) the vehicle deviates from the path in the left direction and (iii) the vehicle deviates from the path in the right direction. The best performance in the test path was observed with the maximum steady state speed of 0.7 m/s, thus in the experiment the parameters were tuned assuming the robot runs with the speed of 0.7 m/s.

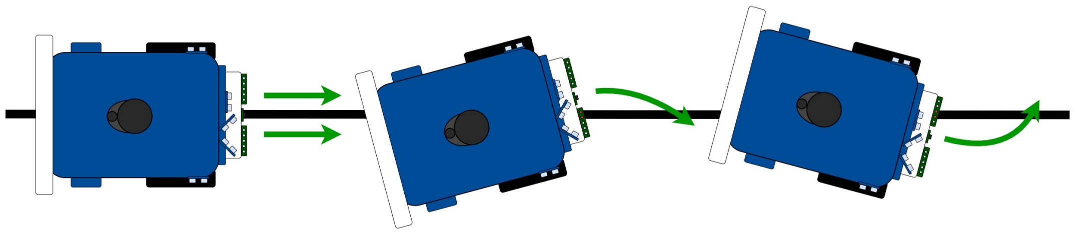


Figure 11: Line follower - Lateral controller

(e) **Acceleration to Velocity Controller:** The high-level controllers publish the command as an acceleration value in the range of 0 to 1, but the low-level controller for the vehicle is a velocity set point controller. This node integrates the acceleration commands over a given time interval to calculate and publish the set point velocity required as input to the low-level motor controller.

2. **Controller Utility Application**

   (a) **ROS Multi-Master Discovery and Master Synchronisation:** The FKIE Multimaster package offers the Master Discovery and Master Synchronisation nodes to initiate and manage a ROS network with multiple ROS masters on various computers. The topics and services on each ROS Master's computer are automatically detected and synchronised with the other ROS masters on the network. In this case, the computer of each robot vehicle runs a ROS Master and ROS network independent of the computers of the other vehicles. It allows the addition and removal of vehicles to the ROS network very easily, with minimal configuration.

   (b) **Command Velocity Multiplexer:** The `mux` node in the `topic_tools` ROS package subscribes to N incoming topics and publishes a single output topic. The input and output messages have the same ROS message format. The node republishes the message data from a selected topic out of the many subscribed topics to the single output topics, i.e., it switches the output among 1 of N inputs. The node handles ROS service requests to switch between input topics and to add or delete input topics. The `cmd_vel_mux` ROS node subscribes to the `/robotXX/joy/cmd_vel`, `/robotXX/platooning/cmd_vel`, etc. topics and publishes its output as the `/robotXX/cmd_vel` topic, where `robotXX` refers to the vehicle ID in the platoon.

4.4 Ground Control Station Software

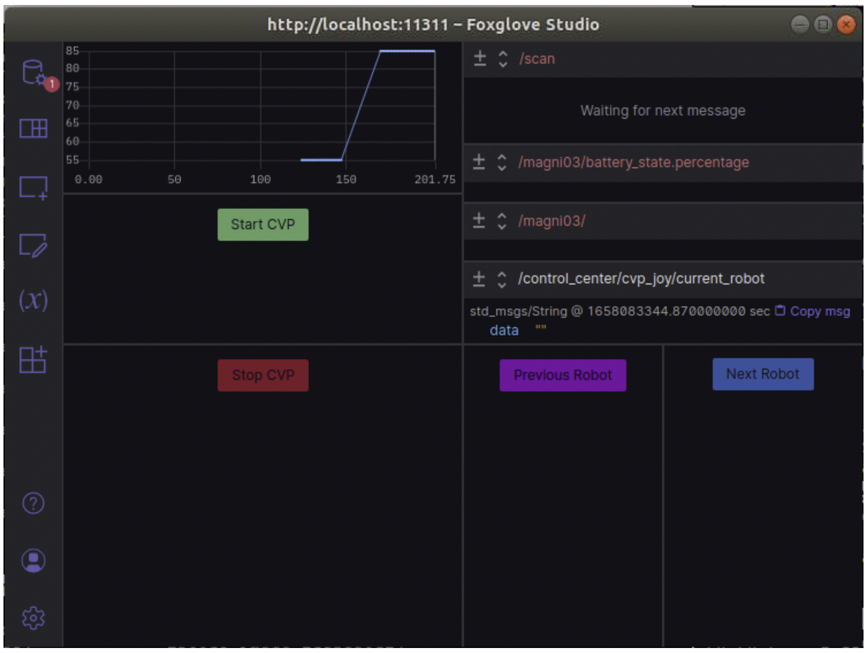
The ground control station (GCS) coordinates vehicle operations and visualizes system state data collected from all vehicles in the platoon. The GCS is powered by Ubuntu and a locally hosted ROS master node. The following modules are running on the GCS computer:

1. **ROS Multi-Master Discovery and Master Synchronization:** Similar to each vehicle computer, the GCS runs the FKIE Multimaster package's Master Discovery and Master Synchronization nodes to manage a ROS network with multiple ROS masters on various computers.

2. **Joystick Apps**: A joystick controller is wirelessly connected to the GCS as a user interface to receive commands for robot control. To map the joystick controller buttons to commands, an open-source ROS node called `joy_node` is used. These joystick commands are then translated to `cmd_vel` topic to be

communicated to the motor controller. A detailed description of the joystick applications is explained in the appendix A

3. **Diagnostics UI**: The GCS uses open-source user interface packages for ROS - Foxglove and Plot Juggler, to display various plots and sensor measurements for each robot in real time. The user interface makes it easier to study the platooning system during an experiment and record experimental results. A screenshot of the user interface view is shown in Figure 12.


http://localhost:11311 – Foxglove Studio
85
80
75
70
65
60
55
0.00
50
100
150
201.75
Start CVP
Stop CVP
Previous Robot
Next Robot
/scan
/magni03/battery_state.percentage
/magni03/
/control_center/cvp_joy/current_robot
Copy msg

```
Waiting for next message
std_msgs/String @ 1658083344.870000000 sec
  data  ""
```

Figure 12: Groundstation UI

4.4.1 Network Architecture

For this experiment, all experimental platform vehicles and the GCS computer are on a WLAN network as shown in Figure 13. A network router handles network traffic consisting of the connected vehicle system computers and the GSCC computer. These computer systems have been configured with a virtual network service, which is described in more detail in appendix A.

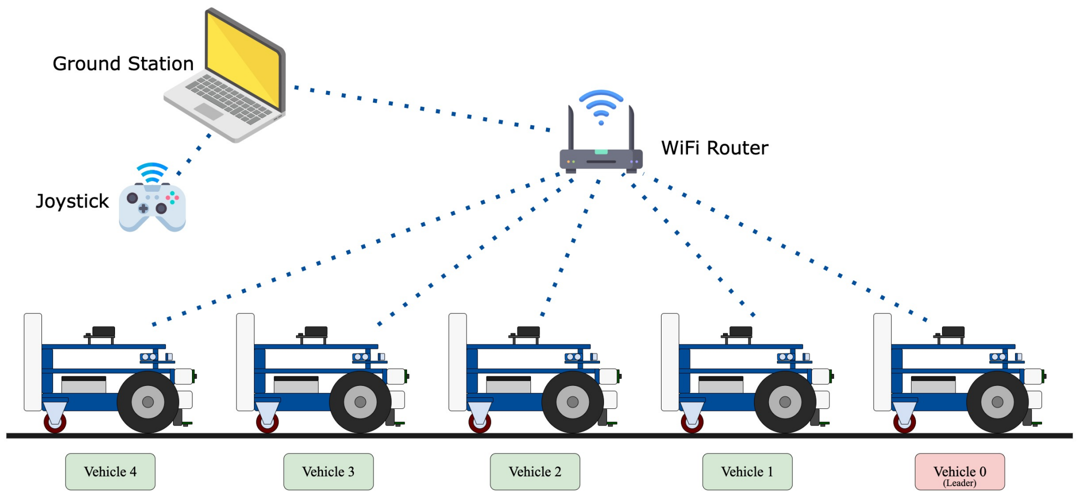


Figure 13: Platooning Network Topology

5 Experimental Evaluation

5.1 Overview

The primary goal of this thesis is to conduct experimental evaluations of the CACC platooning controllers on the experimental platform vehicle. It is necessary for the experimental platform's hardware, controller software and inter-vehicular network communication to function reliably for a successful platooning experiment. Prior to the implementation on the hardware platform, simulation experiments were performed to verify the stability of the parameters used in the control algorithms.

5.2 Experiment Setup

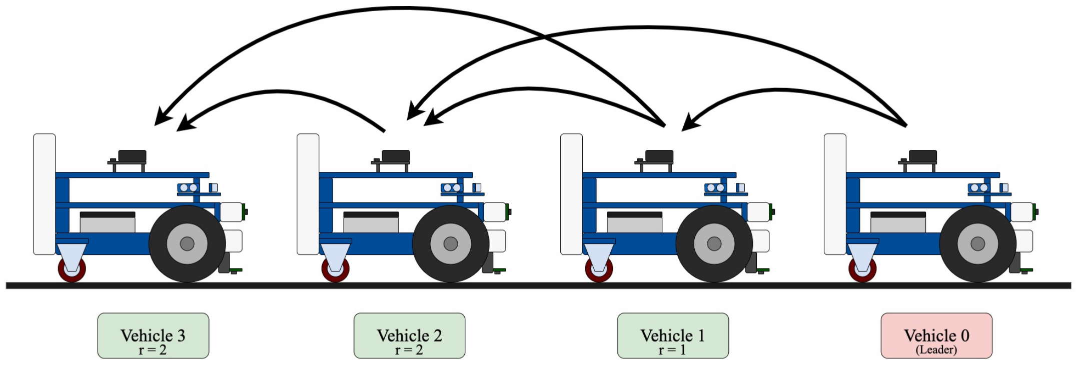


Figure 14: Two-Predecessor Following - Experiment Setup

In the platooning experiment, the aim is to evaluate the multiple predecessors following (MPF) controller. The experiment is set up using the experimental platform vehicles designed in this work and described in Chapter 4. In this experiment, the lead vehicle is followed by identical vehicles, in the order: Lead Vehicle, Vehicle 1, Vehicle 2, Vehicle 3, etc. Figure 14 illustrates the arrangement of these experimental vehicles and their direction of data flow according to the MPF methodology. The longitudinal controller on each vehicle is described below:

- **Leader Vehicle**: The controller on the leader vehicle executes a predefined acceleration control input command, i.e. of $0.1m/s^2$ acceleration at $t = 5s$. A short disturbance deceleration is introduced at $t = 15s$ acting on the leader vehicle's throttle input with $u_0 = A_d sin(\omega_0 t + \pi)$ for the duration of half a cycle, i.e. ($\frac{\pi}{\omega_0}$ seconds). To bring the platoon to a stop, a constant braking deceleration is applied at $t = 40s$. The plot in figure 15 shows the leader controller's throttle (acceleration) input command.

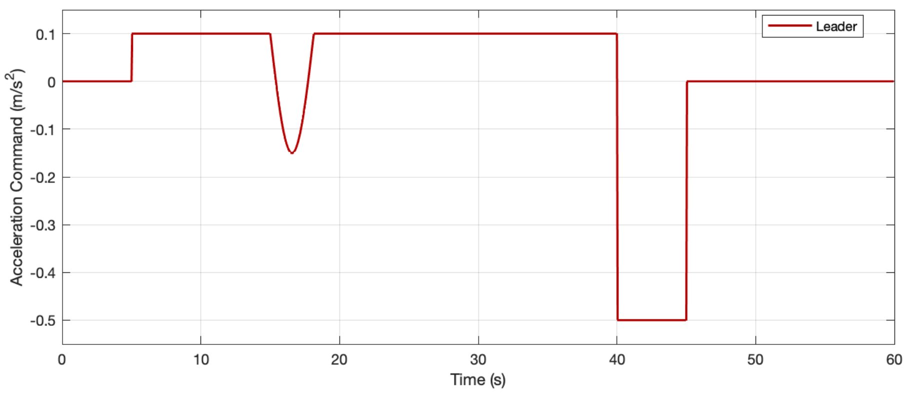


Figure 15: Leader Vehicle : Acceleration Input Command

- **Vehicle 1**: The controller on this vehicle is running the MPF controller described in equation (8)

  with $r = 1$, where $u_1(t) = u_1$, $h = h_1$, $d = d_1$, $k_p = k_{p1}$, $k_v = k_{v1}$ and $k_a = k_{a1}$. The controller input (u_1) calculation is simplified to the following equation:

$$u_1 = -k_p(p_1 - p_0 + hv_1 + d) - k_v(v_1 - v_0) - k_a(a_1 - a_0) \tag{34}$$

- **Vehicle 2**: The controller on this vehicle is running the MPF controller described in equation (8) with $r = 2$, where $u_2(t) = u_2$, $h = h_1 = h_2$, $d = d_1 = d_2$, $k_p = k_{p2}$, $k_v = k_{v2}$ and $k_a = k_{a2}$. The controller input (u_2) calculation is simplified to the following equation:

$$\begin{aligned} u_2 = -k_p[(p_2 - p_1 + hv_2 + d) + (p_2 - p_0 + hv_2 + d + hv_1 + d)] \\ - k_v[(v_2 - v_1) + (v_2 - v_0)] - k_a[(a_2 - a_1) + (a_2 - a_0)] \end{aligned} \tag{35}$$

- **Vehicle 3, Vehicle 4, etc.**: The controller on these vehicles can run the MPF controller described in equation (8) where $u_3(t) = u_3$, $h = h_1 = h_2 = h_3$, $d = d_1 = d_2 = d_3$, $k_p = k_{p3}$, $k_v = k_{v3}$ and $k_a = k_{a3}$. For $r = 2$, the controller input (u_3) is given by a similar equation like (35). For $r = 3$, the controller input (u_3) is simplified to the following equation:

$$\begin{aligned} u_3 = -k_p[(p_3 - p_2 + hv_3 + d) + (p_3 - p_1 + hv_3 + d + hv_2 + d) \\ + (p_3 - p_0 + hv_3 + d + hv_2 + d + hv_1 + d)] \\ - k_v[(v_3 - v_2) + (v_3 - v_1) + (v_3 - v_0)] \\ - k_a[(a_3 - a_2) + (a_3 - a_1) + (a_3 - a_0)] \end{aligned} \tag{36}$$

5.2.1 Simulation Setup

Initially, a software simulation of the experiment setup is created to test the expected performance of the platoon controller and tune the parameters, before the experiment can be accomplished on the experimental vehicles' hardware. Matlab and Simulink are used to create a subsystem block diagram of each platoon vehicle's high-level controllers. The model is designed to handle up to five vehicles in the platoon. It follows the previously discussed vehicle arrangement for the platoon: Leader Vehicle, Vehicle 1, Vehicle 2, Vehicle 3 and Vehicle 4.

This is a homogeneous platoon, where all vehicles in the platoon are identical to each other. The vehicle plant model, common to all the vehicles are designed as a Simulink subsystem block diagram. The model is designed based on the work by Abolfazli *et al.* in [1].

Figure 16 shows the complete Simulink model block diagram with the vehicle subsystem and the controllers under test.

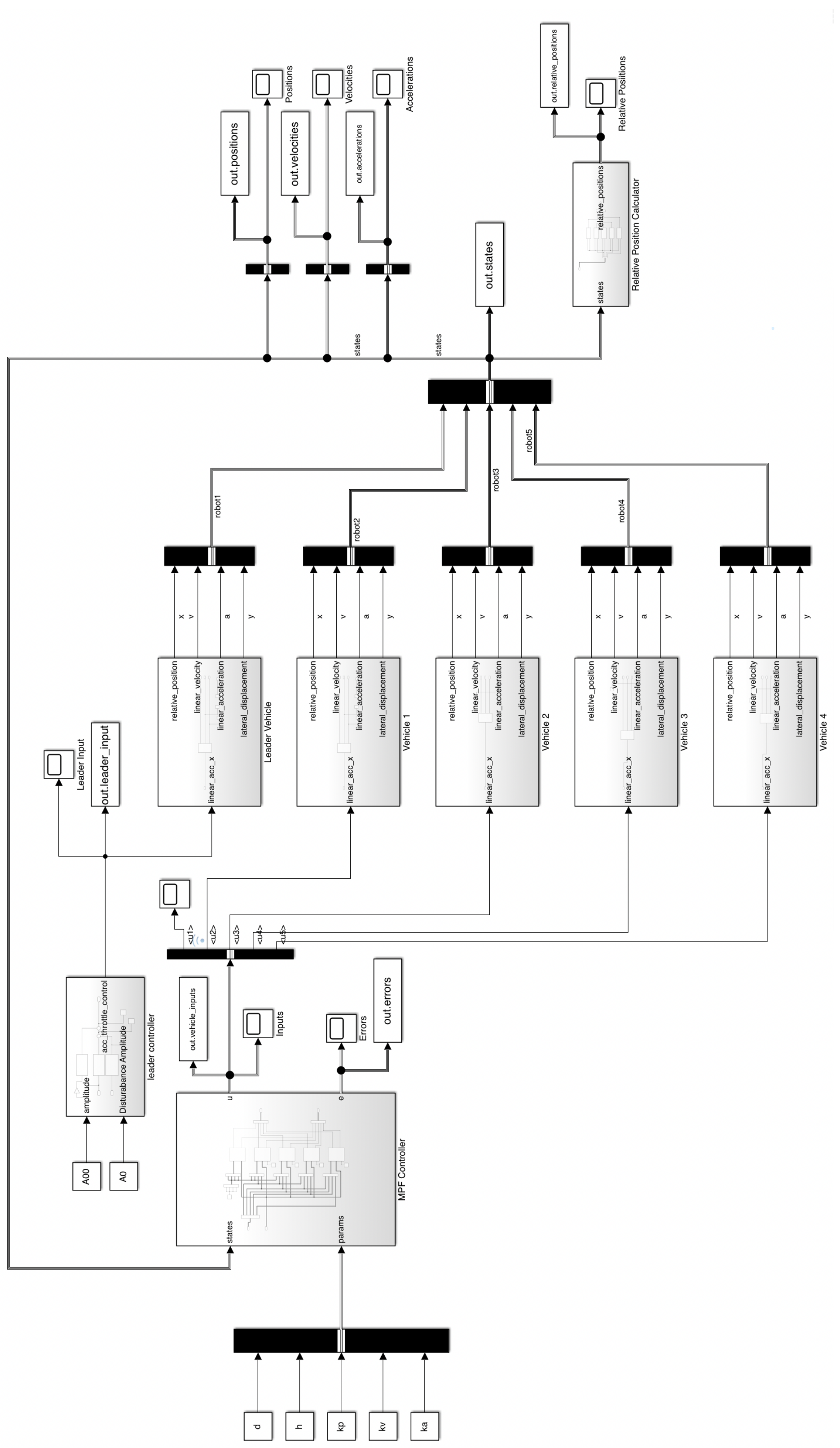


Figure 16: Simulink Model - Experiment Block Diagram

Figure 17 shows the expanded view of the plant model with the vehicle dynamics block diagram for each of the five vehicles blocks, according to the vehicle model equations discussed in section 2.5.2.

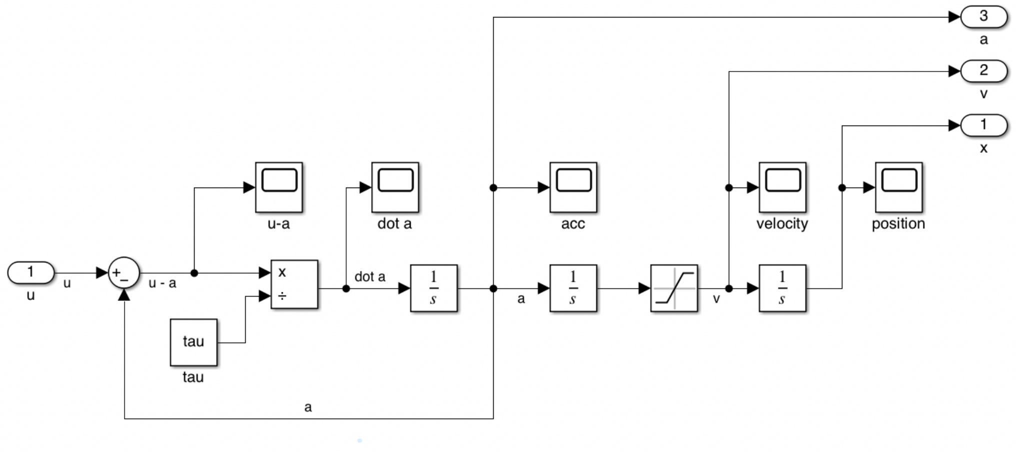


Figure 17: Simulink Model - Vehicle Plant Model Block Diagram

MPF controllers on each follower vehicle are designed as a subsystem model according to the equations (34), (35) and (36).

The MPF controller block in Figure 16 consists of the MPF controllers for all five vehicles in the simulated platoon. An expanded view of this block is shown in the Simulink block diagram in Figure 18. Each vehicle's controller block will follow the equations (34), (35) or (36) based on the value of the block's input parameter r=1, 2, or 3, etc.

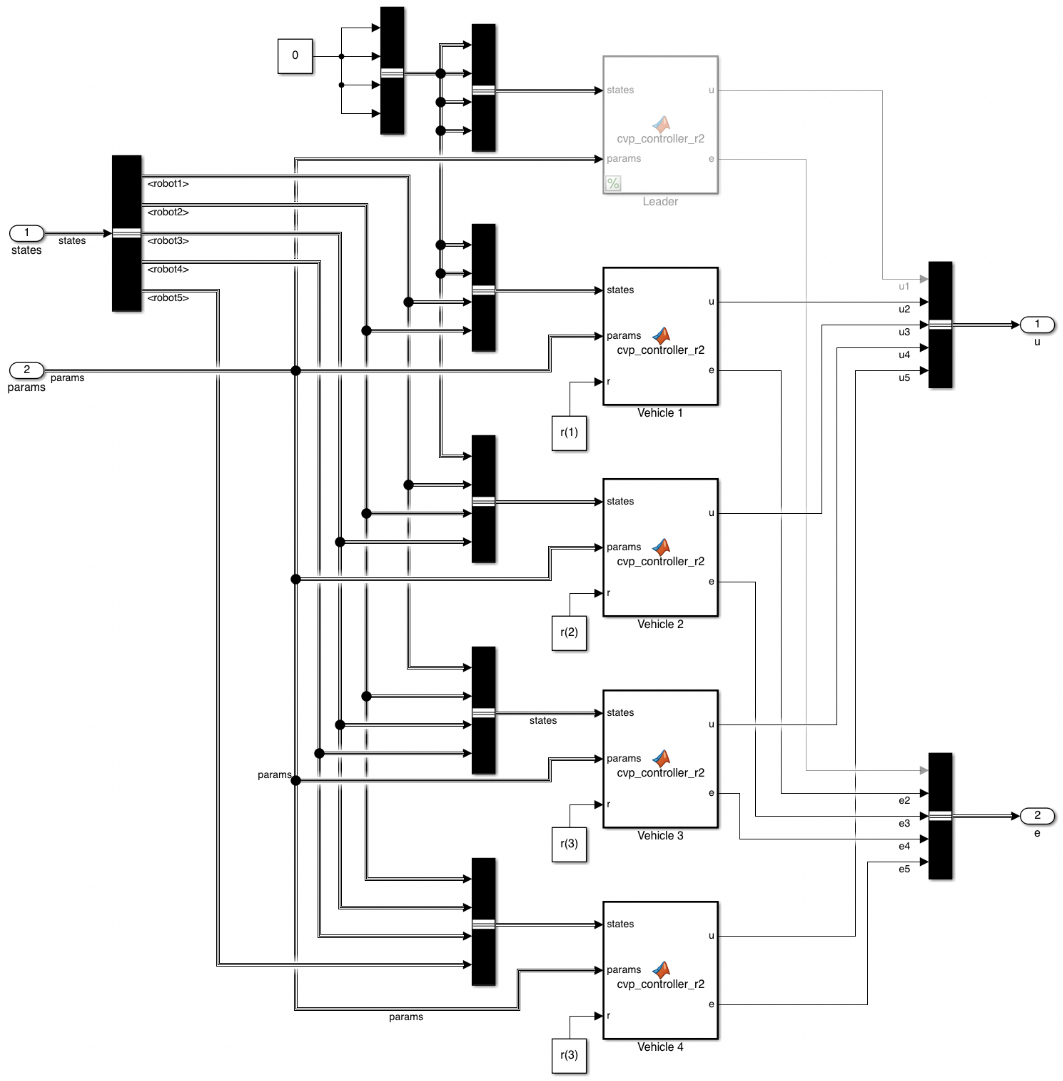


Figure 18: Simulink Model - Controller Subsystem Block Diagram.

5.3 Results: Two-Predecessor Following (TPF)

The platooning in this experiment involves a four-vehicle platoon and the experiment setup shown in figure 14 is according to the description in Section 5.2. This experiment evaluates the TPF controller (with $r \leq 2$ in each vehicle's controller). For this experiment, the values chosen for each system parameter are given in Table 1.

The (k_p, k_v, k_a) parameters are chosen to be $(0.1, 0.61, 0.4)1$ such that the condition in (23) is fulfilled. The values for the parameters $\tau = 0.9$ and $\Delta = 0.05$ are determined through measurements on the experimental platform vehicle. Thus, with these parameters for $r = 2$, the internal stability specification (24) and the string stability criteria (29) are fulfilled. The minimum time headway for these parameters is $h_{min} = 0.719$, and the chosen value $h = 0.78$ is greater than h_{min} which satisfies (30).

Table 1: Two Predecessor Following - Experiment Parameters

Parameter	Value	Description
N	4	Total Number of Vehicles in the Platoon
r	2	Number of predecessor vehicles
τ	0.9	Timelag in the power train, $\tau > 0$
h	0.78	Time headway, $h > 0$
d	0.6	Desired standstill gap, $d > 0$
k_p	0.1	Tunable gain for feedback distance
k_v	0.61	Tunable gain for feedback velocity
k_a	0.41	Tunable gain for feedback acceleration
Δ	0.05	Time delay
A_0	0.05	Amplitude of throttle to leader vehicle's controller
A_d	0.25	Amplitude of disturbance

5.3.1 Simulation Results

The simulation results after running the Simulink model in 16 are presented in this section. Firstly, to perform stability analysis, figure 19 shows the magnitude-frequency diagram of the function $|H_i(j\omega)|$. In each case $|H_1(j\omega)|$ and $|H_2(j\omega)|$ (for $l = 1$ and $l = 2$, respectively) do not exceed the maximum tolerable value for string stability, i.e.,$\frac{1}{r}$ according to (28). This satisfies the (28) and the platoon is string stable.

Next, the plots in figure 20 show the absolute distances travelled by each vehicle in the platoon after running the vehicle platooning simulation for 60 seconds. In this case, the platoon vehicles are initially stationary, and the platoon leader vehicle starts accelerating at $t = 5$ seconds. A disturbance is introduced as a deceleration to the leader vehicle's throttle input at $t = 15$ seconds, which introduces perturbations to the platoon in the longitudinal direction. The spacing error of the vehicles with respect to their predecessor vehicles is plotted in figure 22(a).

The plots of the states (relative position with respect to leader vehicle, velocity and acceleration) of the simulated platoon vehicles are shown in figure 21(a). Similarly, figure 22(a) shows the spacing errors of the follower vehicles according to data from the simulation.

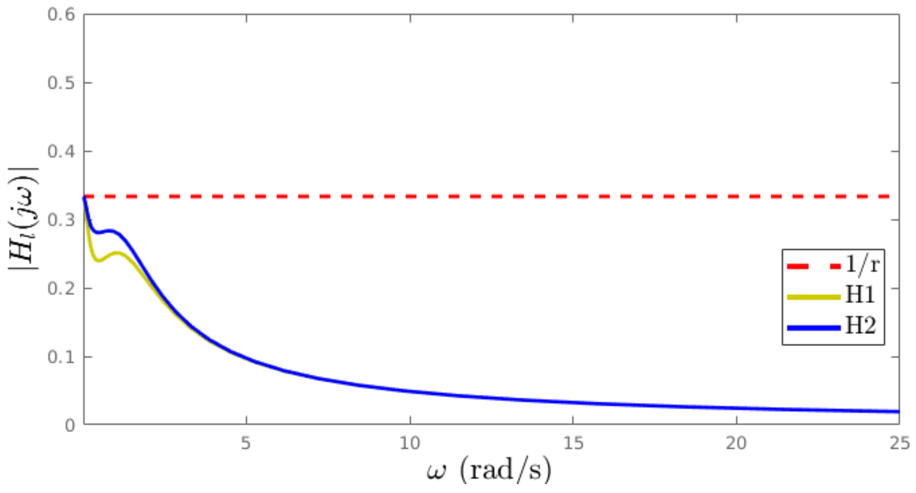


Figure 19: Two-Predecessor Following Simulation - String Stability

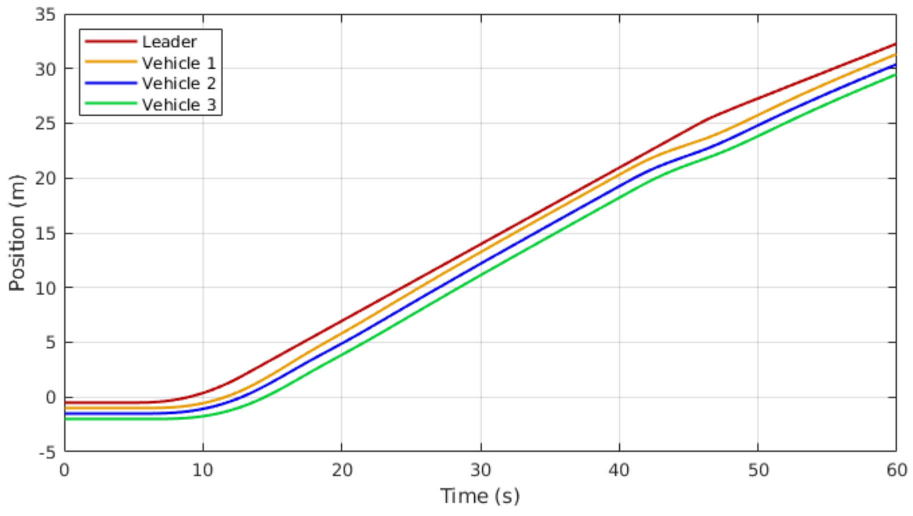


Figure 20: Two-Predecessor Following Simulation - Distance travelled by each vehicle

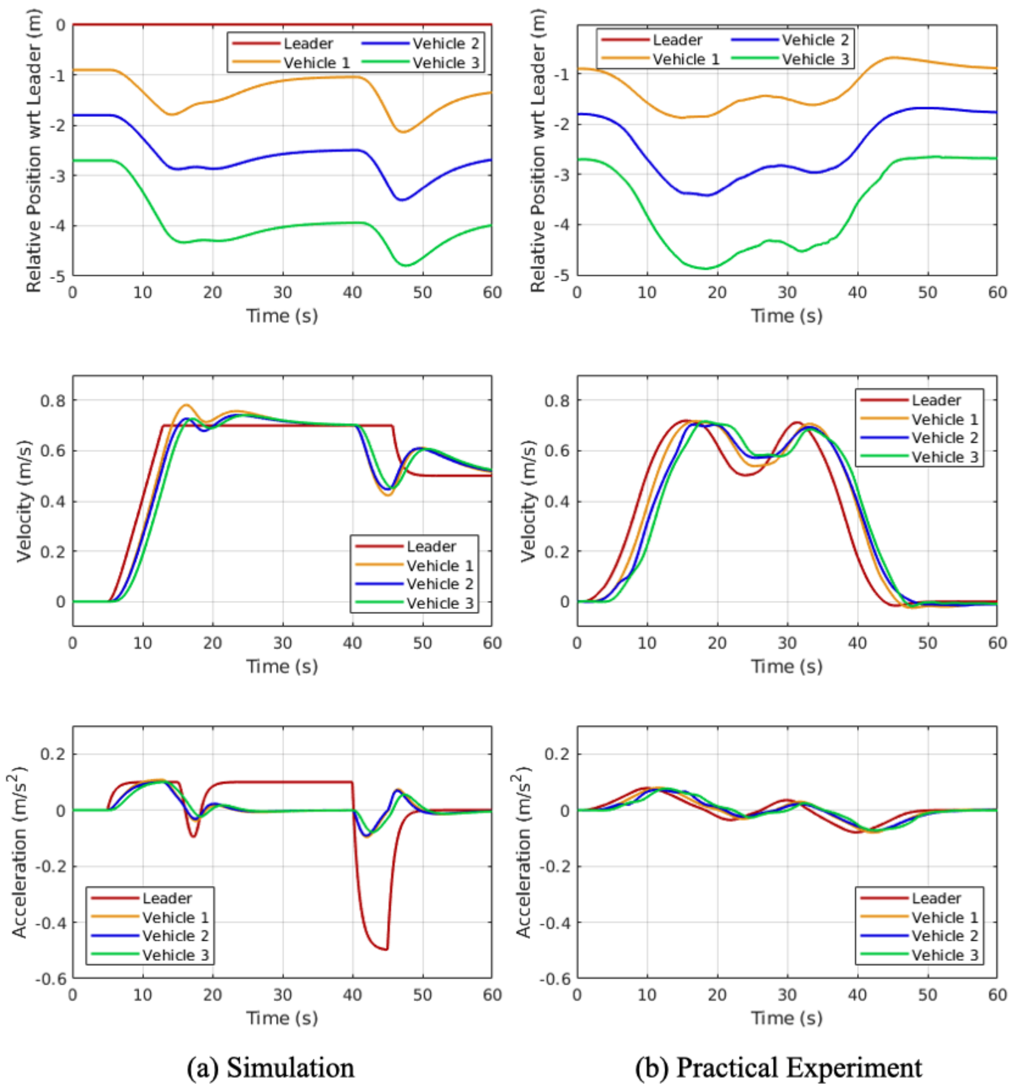
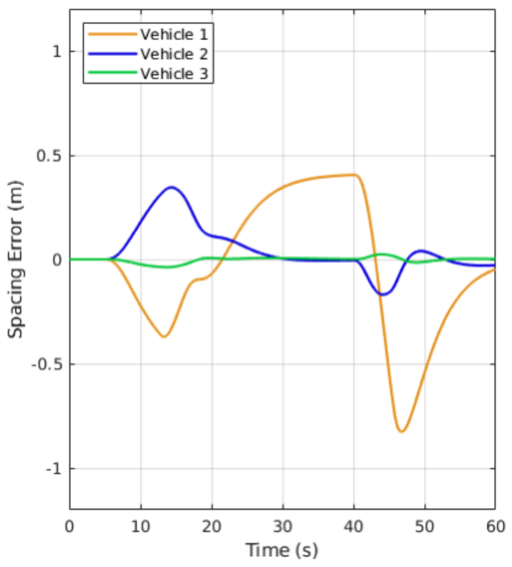
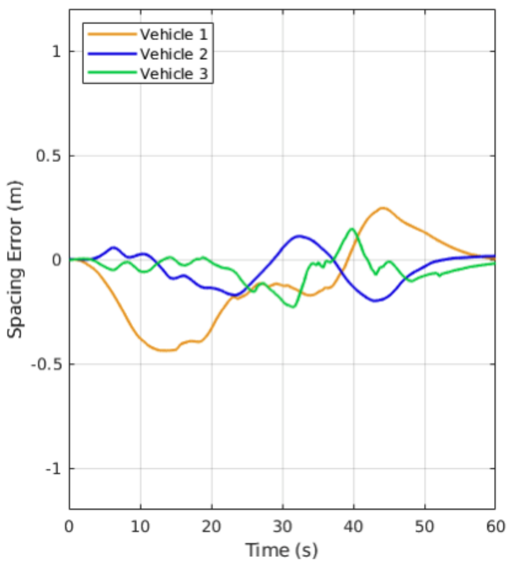


Figure 21: Two-Predecessor Following - System States



(a) Simulation



(b) Practical Experiment

Figure 22: Two-Predecessor Following - Spacing Errors

5.3.2 Practical Experiment Results

The practical experiment is performed on the experimental platform vehicle and the results are discussed in this section.

The results for this experiment are in figure 21(b) which show the vehicles' relative position with respect to the platoon leader, velocities and acceleration of each vehicle in the platoon.

The LIDAR sensor is placed at the center of the experimental vehicle's roof, i.e. 0.2 m with respect to its front. Thus, the plot of the vehicles' relative position with respect to the platoon leader is the distance from the rear of the leader vehicle to the center of the *i*-th vehicle.

Similar to the simulation, in this experiment, the platoon vehicles are initially stationary and the platoon's leader vehicle starts accelerating at $t = 5$ seconds. Similar to the simulation, a disturbance is introduced as a deceleration to the leader vehicle's throttle input at $t = 15$ seconds, which introduces perturbations to the platoon in the longitudinal direction. The spacing error of the vehicles with respect to their predecessor vehicles is plotted in figure 22(b).

5.4 Results: Three-Predecessor Following

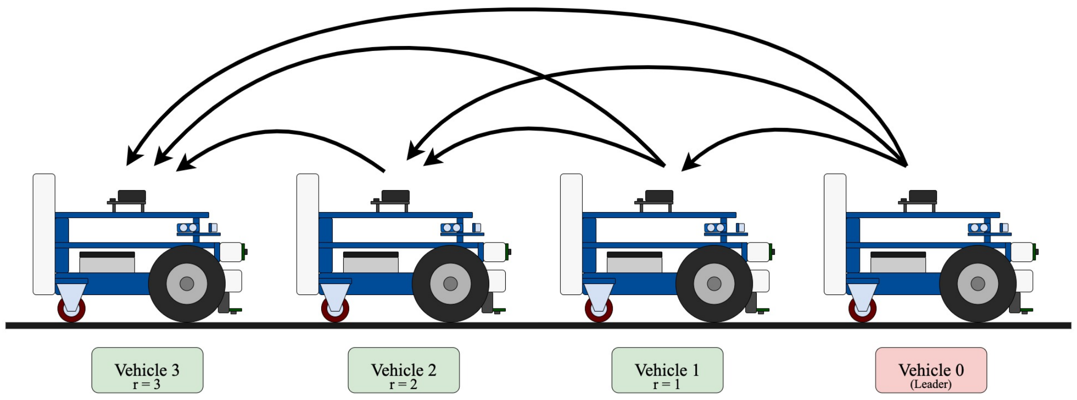


Figure 23: Three-Predecessor Following - Experiment Setup

As described in the experiment setup in section 5.2, this platooning experiment involves a four-vehicle platoon according to the arrangement in figure 23. This experiment evaluates the multiple predecessor following controller (with the member vehicles having $r \leq 3$). For this experiment, the values chosen for each system parameter are given in Table 2.

The (k_p, k_v, k_a) parameters are chosen to be $(0.1, 0.39, 0.41)$ such that the condition in (23) is fulfilled. The values for the parameters $\tau = 0.9$ and $\Delta = 0.05$ are determined through measurements on the experimental platform vehicle. Thus, with these parameters for $r = 3$, the internal stability specification (24) and the string stability criteria (29) are fulfilled. The minimum time headway for these parameters is $h_{min} = 0.549$, and the chosen value $h = 0.78$ is greater than h_{min} which satisfies (30).

Table 2: Three Predecessor Following - Experiment Parameters

Parameter	**Value**	**Description**
N	4	Total Number of Vehicles in the Platoon
r	3	Number of predecessor vehicles
τ	0.9	Timelag in the power train, $\tau > 0$
h	0.78	Time headway, $h > 0$
d	0.6	Desired standstill gap, $d > 0$
k_p	0.1	Tunable gain for feedback distance
k_v	0.39	Tunable gain for feedback velocity
k_a	0.41	Tunable gain for feedback acceleration
Δ	0.05	Time delay
A_0	0.05	Amplitude of throttle to leader vehicle's controller
A_d	0.25	Amplitude of disturbance

5.4.1 Simulation Results

The simulation results after running the Simulink model in 16 are presented in this section. Firstly, to perform stability analysis, figure 24 shows the magnitude-frequency diagram of the function $|H_i(j\omega)|$. In each case $|H_1(j\omega)|$ and $|H_2(j\omega)|$ (for $l = 1$ and $l = 2$, respectively) do not exceed the maximum tolerable value for string stability, i.e.,$\frac{1}{r}$ according to (28). This satisfies the (28) and the platoon is string stable.

Next, the plots in figure 25 show the absolute distances travelled by each vehicle in the platoon after running the vehicle platooning simulation for 60 seconds. In this case, the platoon vehicles are initially stationary, and the platoon leader vehicle starts accelerating at $t = 5$ seconds. A disturbance is introduced as a deceleration to the leader vehicle's throttle input at $t = 15$ seconds, which introduces perturbations to the platoon in the longitudinal direction. The spacing error of the vehicles with respect to their predecessor vehicles is plotted in figure 27(a).

The plots of the states (relative position with respect to leader vehicle, velocity and acceleration) of the simulated platoon vehicles are shown in figure 26(a). Similarly, figure 27(a) shows the spacing errors of the follower vehicles according to data from the simulation.

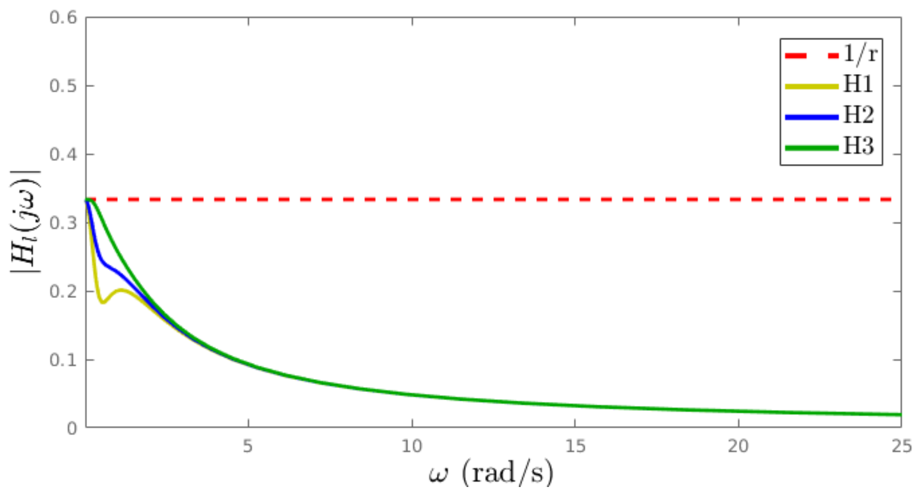


Figure 24: Three-Predecessor Following Simulation - String Stability

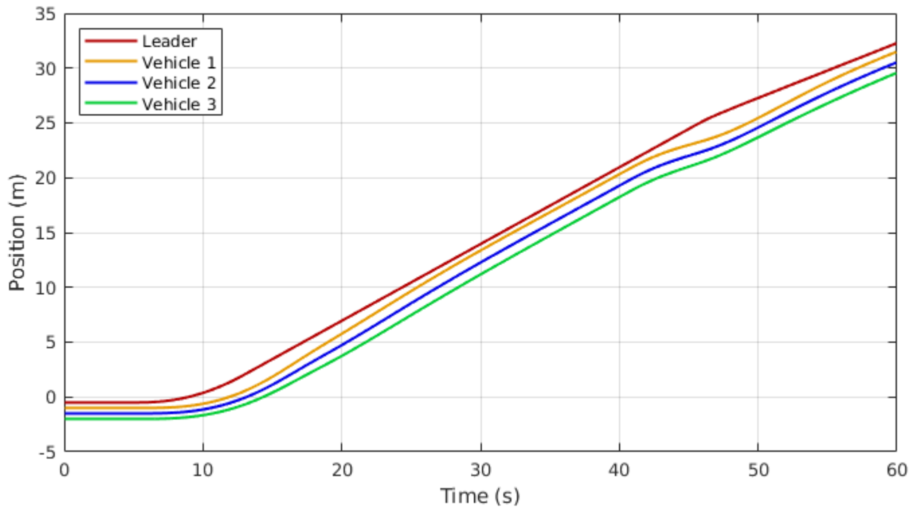
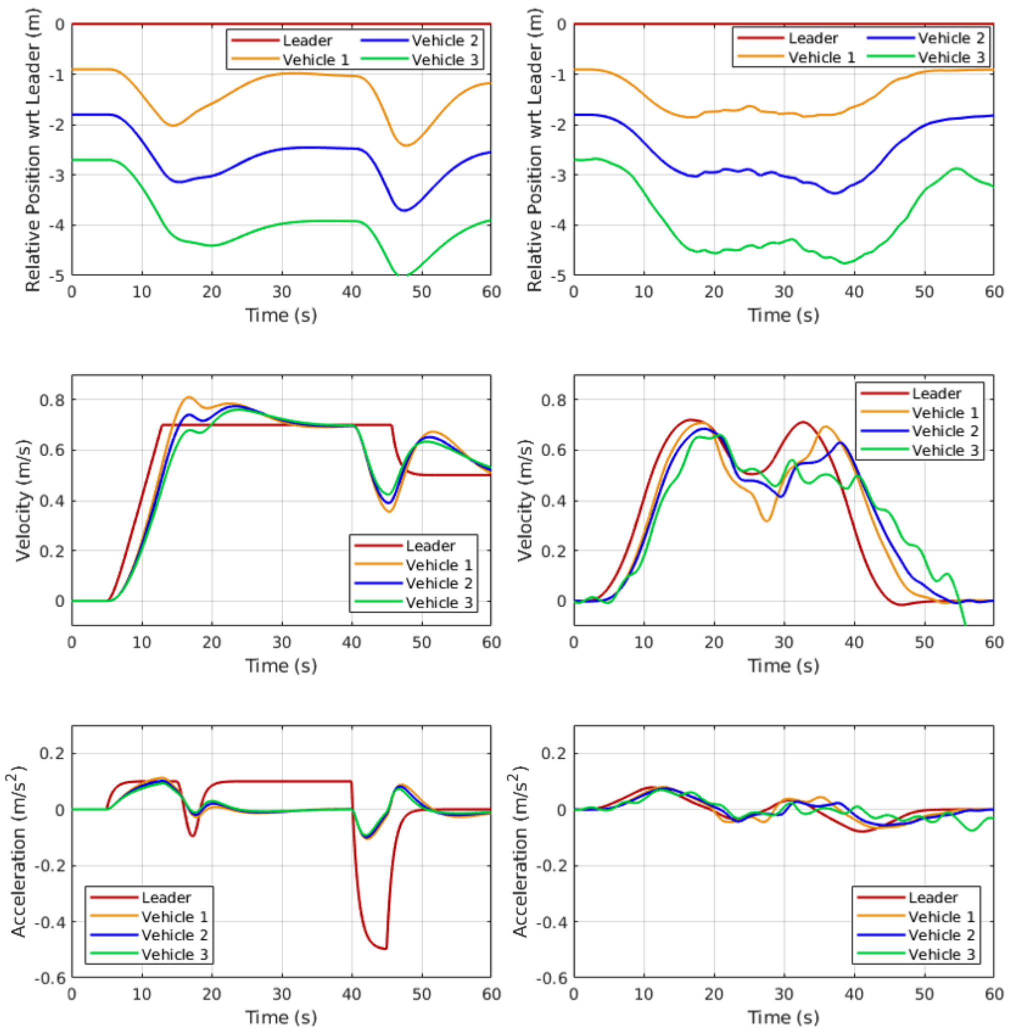
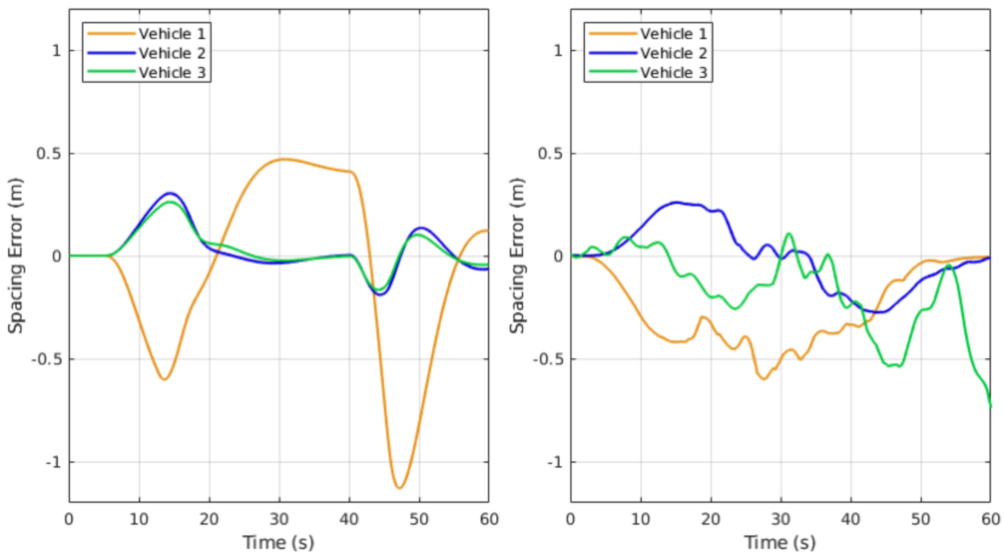


Figure 25: Three-Predecessor Following Simulation - Distance travelled by each vehicle



(a) Simulation (b) Practical Experiment

Figure 26: Three-Predecessor Following - System States



(a) Simulation (b) Practical Experiment

Figure 27: Three-Predecessor Following - Spacing Errors

5.4.2 Practical Experiment Results

The practical experiment is performed on the experimental platform vehicle and the results are discussed in this section.

The results for this experiment are in figure 26(b) which show the vehicles' relative position with respect to the platoon leader, velocities and acceleration of each vehicle in the platoon.

The LIDAR sensor is placed at the center of the experimental vehicle's roof, i.e. 0.2 m with respect to its front. Thus, the plot of the vehicles' relative position with respect to the platoon leader is the distance from the rear of the leader vehicle to the center of the i-th vehicle.

Similar to the simulation, in this experiment, the platoon vehicles are initially stationary and the platoon's leader vehicle starts accelerating at $t = 5$ seconds. Similar to the simulation, a disturbance is introduced as a deceleration to the leader vehicle's throttle input at $t = 15$ seconds, which introduces perturbations to the platoon in the longitudinal direction. The spacing error of the vehicles with respect to their predecessor vehicles is plotted in figure 27(b).

Figure 28: Three-Predecessor Following - Robot States

5.5 Discussion

Control algorithms for the platoon were introduced and discussed in chapter 2. In chapter 5, the chosen control algorithms were implemented in simulation and on the experimental robot platform. Finally, this section compares and analyses the results.

Figure 19 and figure 24 show that the platoon satisfies the conditions of string stability for the chosen parameter configurations for both the two-predecessor following and three-predecessor following platoon experiments. These plots represent the expected results according to the work by Abolfazli *et al.* [1].

In the experiments on the experimental platform vehicle, figure 21 and figure 26 show that the vehicle controllers produce the necessary acceleration commands to follow the predecessor vehicle, similar to the simulation results. On comparing the spacing errors in the practical experiment with the simulation results (as shown in figure 22 and figure 27) it is evident that the expected spacing error does not translate perfectly from the simulation to the practical scenario. Thus the system has internal stability and is string-stable, but the effectiveness of the MPF controller depends on various other factors which might not be considered in its control input equation.

The error in the spacing might be due to physical interactions with the environment, some of the factors can be:

- **Sensor Fusion and Measurement Errors**: All sensors have error margins, including the lidars, IMUs and odometers used in the vehicle designed for this experiment. These errors are considered in the sensor fusion model, yet they can lead to some inaccuracies over time.

- **Friction**: Interaction with the floor of the experiment area, may cause the vehicle to experience uneven friction on each wheel. This can lead to the slipping of each wheel differently, and even though the same command is given to both the wheels of the vehicle, the slipping can lead to the vehicle turning rather than moving forward, due to the differential dynamics of the experiment vehicle platform used.

- **Communication**: The vehicles in this experiment communicate over the wireless LAN network using the TCP/IP communication protocol. The transmission over the wireless medium is subject to lots of noise and delays, which in turn leads to further delays in the processing of the controller's throttle command in each follower vehicle.

These experiment results give a good opportunity to study the MPF controller in the physical environment and improve the controller based on feedback from the external environment.

6 Conclusion

CACC is regarded as a promising solution for smart mobility. By improving on the ACC algorithm, this system allows vehicles to exchange their system state information with other vehicles in the platoon permitting vehicles to cruise with short inter-vehicular distances. This comes with the benefit of increasing traffic flow without increasing any road infrastructure. Reducing the net aerodynamic drag in platooning vehicles and optimizing their individual speeds, CACC systems also reduce the fuel consumption of those vehicles, lowering carbon emissions [21].

The CACC system still needs to overcome some technical challenges before its application becomes widespread in the automotive industry. Improvements in the distributed longitudinal controller for platoons is one such ongoing topic for research. The analysis of interconnected dynamics of platoon vehicles is used to determine the stability of the system.

6.1 Recommendations for further research

CACC was created with the goal of reducing traffic congestion and increasing fuel efficiency. Though, CACC might be a very useful system to improve safety but is primarily not designed to be a safety system. Regardless, safety should be guaranteed since the CACC systems will be used on public roads with heavy vehicles [34].

In platooning vehicles, the inter-vehicular distance is small, which increases the possibility of collisions within the platoon and with surrounding vehicles, in case of disturbances or failures. Hazards associated with platooning systems can be due to external disturbances; internal component failures in wireless communication; and human errors [8].

External disturbances include cut-in manoeuvres from vehicles in the surrounding traffic, bad roads and poor weather conditions. These scenarios are not favourable for the platoon to be operating. The forward-facing range sensor (typically a radar) is the most crucial sensor for CACC controllers because it measures the distance between the vehicles and their relative speeds. Research on implementing redundant systems for taking these measurements can improve the measurements and serve as a backup in case of internal sub-system failures [9].

Wireless communication is an important feature of CACC controllers. These controllers rely on receiving the most recent information from other vehicles in the platoon in order to operate in real time. In the results of this thesis, it can be observed that the packet losses and delays increase in the local network of the experimental setup, once the number of platooning vehicles is more than three. It would be useful to research the effect of packet losses and delays in the communication network [8].

The current software architecture uses the robot operating system (ROS) as a communication framework. The information exchange using ROS messages over the network is not the most efficient option. The current networking subsystem developed in ROS can be upgraded to use more optimal protocols. Research on efficient information flow topologies(IFTs) combined with standardized Vehicle-to-vehicle (V2V) communication protocols can be an effective method to improve the information transfer reliability of this experimental setup [39].

CACC is categorized as “level-1” in terms of driving automation level. In such semi-autonomous vehicle systems, where human drivers are still responsible for monitoring the vehicle’s environment and taking control of the vehicle when needed. Thus, during the transition between manual and automated driving modes, there are possibilities of human error. Neglecting human factors can lead to accidents if the driver is not aware of when the transition occurs. Human error analysis may become an important research topic to improve the human-machine interface of CACC systems in the future. [22].

A basic version of the automatic emergency braking system has been implemented in the experimental platform developed in this thesis. Further research on the sensor data processing for the automatic triggering of emergency brakes in a platooning vehicle can be an improvement to the vehicle’s safety.

This work uses an experimental vehicle platform using the differential drive system, which is simple but also comes with its drawbacks which have been discussed in section 5.5. The Ackerman steering model is used in most commercial road vehicles. This system ensures proper handling of the vehicles by preventing the tyre from slipping outward during turns and also increasing controllability. Using an Ackerman model-based experimental vehicle platform can be very beneficial to observe the behaviour of the CACC controllers in the scenario where lateral motion is involved [42].

Lastly, the experimental platform developed in this thesis is a useful resource for testing various CACC controllers but it must be validated and developed further. It should be tested in real traffic scenarios and its performance can be compared with a benchmark before it can be implemented on full-scale vehicles.

References

[1] E ; Abolfazli, B ; Besselink, and T Charalambous. “On Time Headway Selection in Platoons Under the MPF Topology in the Presence of Communication Delays”. In: *IEEE Transactions on Intelligent Transportation Systems* (2021). DOI: 10.1109/TITS.2021.3087484. URL: https://doi.org/10.1109/TITS.2021.3087484.

[2] Elham Abolfazli and Bart Besselink. “Reducing time headway in platoons under the MPF topology in the presence of communication delays”. In: (2019), pp. 1–14.

[3] Assad Al Alam, Ather Gattami, and Karl Henrik Johansson. “An experimental study on the fuel reduction potential of heavy duty vehicle platooning”. In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* (2010), pp. 306–311. DOI: 10.1109/ITSC.2010.5625054.

[4] Assad Alam. “Fuel-efficient heavy-duty vehicle platooning”. PhD thesis. 2014. ISBN: 978-91-7595-194-2.

[5] Assad Alam et al. “Heavy-duty vehicle platooning for sustainable freight transportation: A cooperative method to enhance safety and efficiency”. In: *IEEE Control Systems* 35.6 (Dec. 2015), pp. 34–56. ISSN: 1066033X. DOI: 10.1109/MCS.2015.2471046.

[6] M. Ya. Alwardat and P. V. Balabanov. “SPEED CONTROL OF DC MOTOR USING PID CONTROLLER BASED ON MATLAB”. In: *Vestnik Tambovskogo gosudarstvennogo tehnicheskogo universiteta* 27.2 (July 2021), pp. 195–202. ISSN: 01365835. DOI: 10.17277/VESTNIK.2021.02.PP.195-202.

[7] M H Ang, O Khatib, and B Siciliano. *Sensors for Mobile Robots*. Tech. rep.

[8] Maytheewat Aramrattana. “A simulation-based safety analysis of CACC-enabled highway platooning”. PhD thesis. 2018. ISBN: 9789188749079.

[9] Jakob Axelsson. “Safety in vehicle platooning: A systematic literature review”. In: *IEEE Transactions on Intelligent Transportation Systems* 18.5 (May 2017), pp. 1033–1045. ISSN: 15249050. DOI: 10.1109/TITS.2016.2598873.

[10] Yougang Bian et al. “Reducing time headway for platooning of connected vehicles via V2V communication”. In: *Transportation Research Part C: Emerging Technologies* 102 (May 2019), pp. 87–105. ISSN: 0968-090X. DOI: 10.1016/J.TRC.2019.03.002.

[11] C. C. Chien and P. Ioannou. “Automatic vehicle-following”. In: *Proceedings of the American Control Conference* 2 (1992), pp. 1748–1752. ISSN: 07431619. DOI: 10.23919/ACC.1992.4792410.

[12] P. A. Cook. “Conditions for string stability”. In: *Systems and Control Letters* 54.10 (Oct. 2005), pp. 991–998. ISSN: 01676911. DOI: 10.1016/J.SYSCONLE.2005.02.011. URL: https://www.researchgate.net/publication/220519176_Conditions_for_string_stability.

[13] J. Eyre, D. Yanakiev, and I. Kanellakopoulos. “A simplified framework for string stability analysis of automated vehicles”. In: *Vehicle System Dynamics* 30.5 (1998), pp. 375–405. ISSN: 00423114. DOI: 10.1080/00423119808969457.

[14] Jennifer Eyre, Diana Yanakiev, and Ioannis Kanellakopoulos. “String Stability Properties of AHS Longitudinal Vehicle Controllers”. In: *IFAC Proceedings Volumes* 30.8 (June 1997), pp. 71–76. ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)43803-1.

[15] Carlos Flores. “Control architecture for adaptive and cooperative car-following”. In: (Dec. 2018). URL: https://pastel.archives-ouvertes.fr/tel-02275824%20https://pastel.archives-ouvertes.fr/tel-02275824/document.

[16] Weinan Gao, Zhong-Ping Jiang, and Kaan Ozbay. “Adaptive optimal control of connected vehicles”. In: *2015 10th International Workshop on Robot Motion and Control (RoMoCo)* (July 2015), pp. 288–293. DOI: 10.1109/ROMOCO.2015.7219749. URL: http://ieeexplore.ieee.org/document/7219749/.

[17] Justin de Geus. *Practically string stable, lateral control solution for a homogeneous platoon of vehicles: A centralized vs distributed approach*. 2021. URL: https://repository.tudelft.nl/islandora/object/uuid%3Ab27a13e7-0900-40c3-ae9d-afa16e153f6c.

[18] JEFF SHEPARD. *What is the role of sensor fusion in robotics?* URL: https://www.sensortips.com/frequently-asked-question-faq/what-is-the-role-of-sensor-fusion-in-robotics-faq/.

[19] Simon J. Julier and Jeffrey K. Uhlmann. “New extension of the Kalman filter to nonlinear systems”. In: *Signal Processing, Sensor Fusion, and Target Recognition VI* 3068 (July 1997), p. 182. ISSN: 0277786X. DOI: 10.1117/12.280797. URL: https://www.researchgate.net/publication/2445827_A_New_Extension_of_the_Kalman_Filter_to_Nonlinear_Systems.

[20] Svetoslav Kuzmanov. *GitHub - SvetoslavKuzmanov/altimu10v5: A Python module for accessing the Pololu AltIMU-10 v5 inertial measurement unit on a Raspberry Pi*. 2017. URL: https://github.com/SvetoslavKuzmanov/altimu10v5.

[21] Michael P. Lammert et al. “Effect of Platooning on Fuel Consumption of Class 8 Vehicles Over a Range of Speeds, Following Distances, and Mass”. In: *SAE International Journal of Commercial Vehicles* 7.2 (Oct. 2014), pp. 626–639. ISSN: 19463928. DOI: 10.4271/2014-01-2438. URL: https://www.researchgate.net/publication/277571756_Effect_of_Platooning_on_Fuel_Consumption_of_Class_8_Vehicles_Over_a_Range_of_Speeds_Following_Distances_and_Mass.

[22] M. H. Martens and A. P. Van Den Beukel. "The road to automated driving: Dual mode and human factors considerations". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* (2013), pp. 2262–2267. DOI: 10.1109/ITSC.2013.6728564.

[23] Corey Montella. *The Kalman Filter and Related Algorithms: A Literature Review The Kalman Filter and Related Algorithms A Literature Review*. Tech. rep. URL: https://www.researchgate.net/publication/236897001.

[24] Thomas Moore and Daniel Stouch. "A generalized extended Kalman filter implementation for the robot operating system". In: *Advances in Intelligent Systems and Computing* 302 (2016), pp. 335–348. ISSN: 21945357. DOI: 10.1007/978-3-319-08338-4{_}25/COVER. URL: https://link.springer.com/chapter/10.1007/978-3-319-08338-4_25.

[25] Jeroen Ploeg, Nathan Van De Wouw, and Henk Nijmeijer. "Lp string stability of cascaded systems: Application to vehicle platooning". In: *IEEE Transactions on Control Systems Technology* 22.2 (Mar. 2014), pp. 786–793. ISSN: 10636536. DOI: 10.1109/TCST.2013.2258346.

[26] *Products Magni - Ubiquity Robotics*. URL: https://www.ubiquityrobotics.com/products-magni/.

[27] R. Rajamani and C. Zhu. "Semi-autonomous adaptive cruise control systems". In: *IEEE Transactions on Vehicular Technology* 51.5 (Sept. 2002), pp. 1186–1192. ISSN: 00189545. DOI: 10.1109/TVT.2002.800617.

[28] Rajesh Rajamani. "Vehicle Dynamics and Control". In: Mechanical Engineering Series (2012). DOI: 10.1007/978-1-4614-1433-9. URL: http://link.springer.com/10.1007/978-1-4614-1433-9.

[29] *robot_localization wiki — robot_localization 2.7.4 documentation*. URL: http://docs.ros.org/en/noetic/api/robot_localization/html/index.html.

[30] *ROS/Concepts - ROS Wiki*. URL: http://wiki.ros.org/ROS/Concepts.

[31] Matija Rossi. *UNIVERSITY OF ZAGREB FACULTY OF MECHANICAL ENGINEERING AND NAVAL ARCHITECTURE MASTER'S THESIS*. Tech. rep. 2014.

[32] *rplidar - ROS Wiki*. URL: https://wiki.ros.org/rplidar.

[33] by A Samuel Mitchell et al. *GROUND VEHICLE PLATOONING CONTROL AND SENSING IN AN ADVERSARIAL ENVIRONMENT*. Tech. rep. 2016.

[34] Steven E Shladover et al. *UC Berkeley Research Reports Title Using Cooperative Adaptive Cruise Control (CACC)to Form High-Performance Vehicle Streams. Definitions, Literature Review and Operational Concept Alternatives Permalink https://escholarship.org/uc/item/3w6920wz*. May 2018. URL: https://escholarship.org/uc/item/3w6920wz.

[35] Steven E Shladover et al. “Using Cooperative Adaptive Cruise Control (CACC) to Form High-Performance Vehicle Streams”. In: June (2014), p. 54. URL: https://escholarship.org/uc/item/8pw857gb%0Ahttp://escholarship.org/uc/item/3m89p611.

[36] Srdjan S. Stanković, Milorad J. Stanojević, and Dragoslav D. Šiljak. “Decentralized overlapping control of a platoon of vehicles”. In: *IEEE Transactions on Control Systems Technology* 8.5 (2000), pp. 816–832. ISSN: 10636536. DOI: 10.1109/87.865854. URL: https://www.researchgate.net/publication/3332200_Decentralized_overlapping_control_of_a_platoon_of_vehicles.

[37] Ubiquity Robotics. *GitHub - UbiquityRobotics/pi_sonar: Package that provides a ROS interface for the sonar array in UbiquityRobotics robots*. 2015. URL: https://github.com/UbiquityRobotics/pi_sonar.

[38] *UbiquityRobotics/raspicam_node: ROS node for camera module of Raspberry Pi*. 2013. URL: https://github.com/UbiquityRobotics/raspicam_node.

[39] Chaojie Wang et al. “Cooperative Adaptive Cruise Control for Connected Autonomous Vehicles by Factoring Communication-Related Constraints”. In: *Transportation Research Procedia* 38 (Jan. 2019), pp. 242–262. ISSN: 2352-1465. DOI: 10.1016/J.TRPRO.2019.05.014.

[40] *What is a vehicle platoon? | Driver Knowledge Test (DKT) Resources*. URL: https://www.driverknowledgetests.com/resources/what-is-a-vehicle-platoon/.

[41] Diana Yanakiev and Ioannis Kanellakopoulos. “A Simplified Framework for String Stability Analysis in AHS”. In: *IFAC Proceedings Volumes* 29.1 (June 1996), pp. 7873–7878. ISSN: 1474-6670. DOI: 10.1016/S1474-6670(17)58959-4.

[42] Andrew Youssef et al. “Comparative Analysis of Various Control Techniques for a ROS-based Platooning Architecture”. In: *2020 8th International Conference on Control, Mechatronics and Automation, ICCMA 2020*. Institute of Electrical and Electronics Engineers Inc., Nov. 2020, pp. 95–99. ISBN: 9781728192109. DOI: 10.1109/ICCMA51325.2020.9301599.

[43] Yang Zheng et al. “Influence of information flow topology on closed-loop stability of vehicle platoon with rigid formation”. In: *2014 17th IEEE International Conference on Intelligent Transportation Systems, ITSC 2014* (Nov. 2014), pp. 2094–2100. DOI: 10.1109/ITSC.2014.6958012.

A System Specifications

The Magni Silver mobile robot platform by Ubiquity Robotics is used as the base robot to develop the experimental platform for this work. The Magni Silver robot provides all the basic functionalities of a mobile robot including a chassis, motors and some basic sensors. In addition to this additional components were integrated into the robot to make it function analogous to a full-scale vehicle. This section describes the experimental platform robot's hardware and software technical specifications in detail. Furthermore, the section also contains a description of the network topology used in the experimental setup. The entire software outlined in this thesis is accessible as an open-source project repository on:
https://github.com/niladut/connected_vehicle_platooning_ros

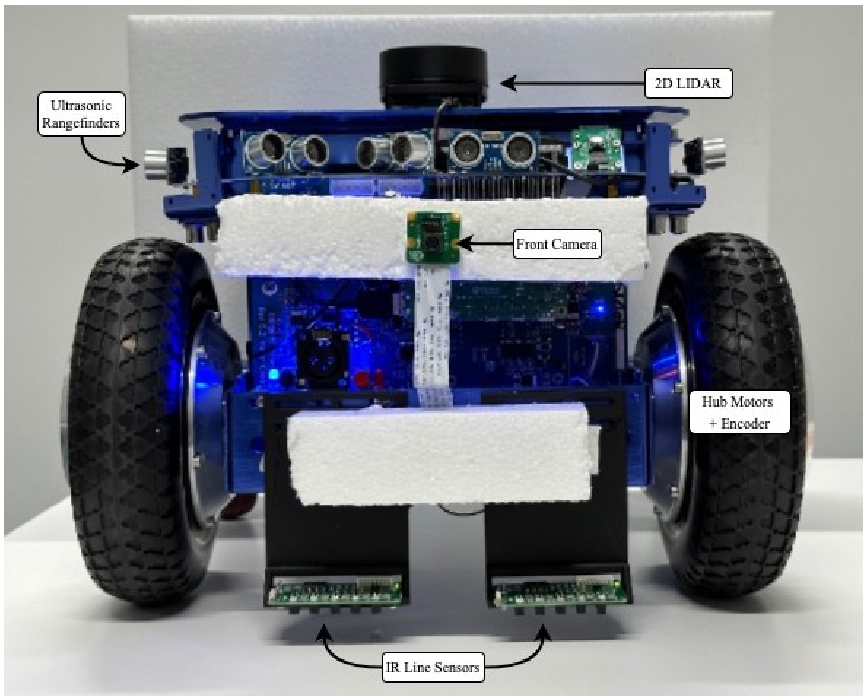


Figure A1: Front view of the Magni Silver robot with sensors and actuators.

Figure A1 displays the front view of the Experimental Platform Vehicle, indicating the position of sensors and actuators. The software bumper comprises ultrasonic sensors and LIDAR, while white styrofoam boards are also mounted at the robot's front and rear acting as a fail-safe, to prevent harm to the structure in case of accidental collisions.

A.1 Experimental Platform Vehicle

Table A1: Experimental Platform Vehicle (Magni Silver) - Hardware Specifications [26]

Category	**Component**	**Description**
Mechanical Specifications	Size (w/l/h) in mm	417.40 x 439.09 x 265
	Weight in kg	13.5
Actuator	Motor Drive System	2 X 200W hub motors – differential drive
	Maximum Linear Speed	1 m/s
	Maximum Angular Speed	0.5 rad/s
Computer System	OBC	Quad Core ARM A9 – Raspberry Pi 4
	Operating System	Ubuntu 16.04
	Middleware Framework	ROS Kinetic
	Connectivity/Ports	WLAN, Ethernet, 4x USB, Raspberry Pi GPIO socket
Sensors	LIDAR	RP Lidar A1 by Slamtec
	Camera (front)	Raspberry Pi Camera Module V2
	Sonars	HC-SR04 5-point sonar array
	IMU	Pololu AltIMU-10 v5 (LSM6DS33 gyro and accelerometer and LIS3MDL magnetometer)
	Odometer	Hall sensor odometer (accurate to 2mm)

A.2 Ground Control Station

Table A2: Ground Control Station - Hardware Specifications

Category	Component	Description
Computer System	OBC	Quad Core ARM A9 – Raspberry Pi 4
	Operating System	Ubuntu 16.04
	Middleware Framework	ROS Kinetic
	Connectivity/Ports	WLAN, Ethernet, 4x USB, Raspberry Pi GPIO socket
User Interface	Joystick Controller	Logitech Wireless Gamepad F710

A.2.1 User Interface: Joystick Controller

A joystick controller is connected to the ground control station with the objective of remotely controlling the experimental vehicle platform while testing. The joystick controller system involves multiple application nodes to convert the joystick controller's button inputs to manual control commands which are transmitted to the vehicle's motion controller. The following is a list of these application modules, describing them in detail.

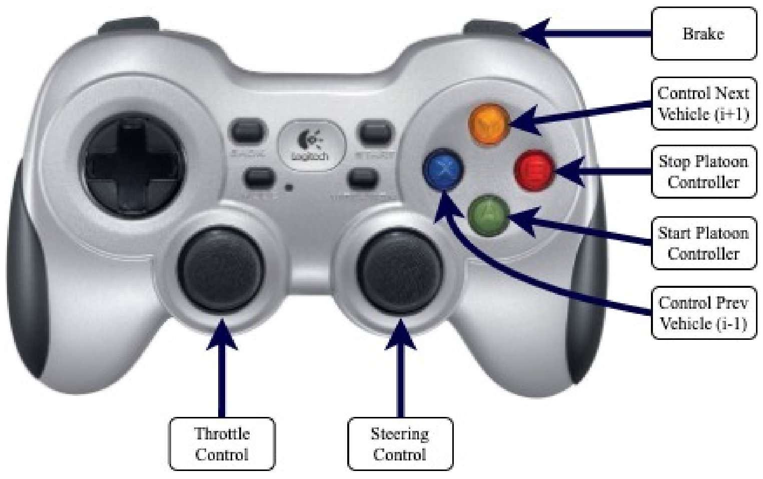


Figure A2: Joystick controller button mapping

1. **Joy Node:** The `joy_node` is a ROS node that runs a driver to interface between a generic Linux joystick and ROS. This node publishes a ROS message of type `sensor_msgs/Joy` containing the current state of each button and axes on the joystick controller. For this experiment the wireless joystick Logitech Wireless Gamepad F710 was used.

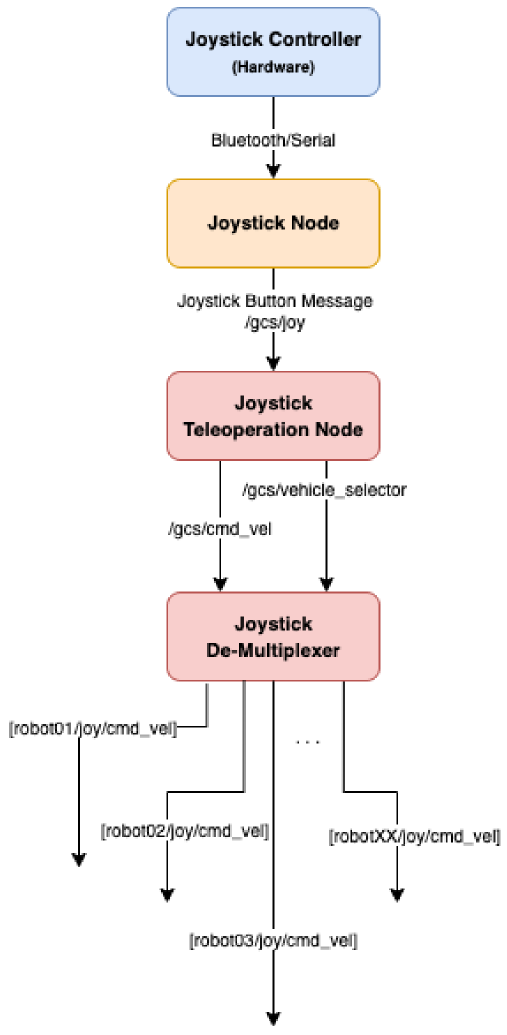


Figure A3: Ground Control Station - Joystick Velocity Controller modules

2. **Joy Teleoperation for Command Velocity:** This package subscribes to the `/joy` topic published by the `joy_node` and publishes `geometry_msgs/Twist` ROS messages (consisting of linear and angular velocity command) on the topic called `/gcs/cmd_vel`. The node facilitates the teleoperation of a velocity controller-based ROS robot using a standard joystick. It is possible to control multiple vehicles in the platoon with one controller. So, this module also maps the buttons of the joystick controller for changing the current vehicle under manual control, i.e. switching manual control from vehicle i to vehicle $(i + 1)$ or vehicle $(i - 1)$. Furthermore, during experimentation, the joystick buttons (A and B, respectively) are mapped to trigger the starting and stopping of the platooning controllers on each vehicle. The R1 and R2 buttons of the controller are also mapped to perform braking and emergency stop (E-Stop). Figure A2 shows the controller button mappings for the joystick controller.

3. **Joystick DeMultiplexer - Cmd Vel:** The `demux` node in the topic_tools

ROS package subscribes to a single incoming topic and publishes a set of N topics of the same message format as the input topic. The node republishes the message data from the subscribed topic to one out of the many output topics, i.e., it switches the input among 1 of N outputs. The node handles ROS service requests to switch between output topics and to add and delete output topics. The `joystick_cmd_vel_demux` ROS node subscribes to the `/control_center/cmd_vel` topic and demultiplexes between the `robotXX/joy/cmd_vel` topics, where *robotXX* refers to the various vehicles in the platoon.

A.3 Network Configurations

The network configuration of the experimental platform is done using the Tailscale VPN service, which allows network devices and applications to be accessible over a private network within a local network (LAN) or even over the internet (WAN). Tailscale provides the MagicDNS service through which the devices on the private VPN can be accessed just based on the hostname, without keeping track of their IP addresses. These features make use of this service ideal for this experimental platform, where the number of vehicles in the platoon under test can vary for each experimental test.

B Robot Operating System (ROS)

B.1 Overview

The Robot Operating System (ROS) is an open-source framework for building robotics applications. ROS provides the necessary middleware for communication between programs in distributed robotic systems. Since the collection of conventions, tools, and libraries provided by the ROS framework is standardized, it simplifies the task of creating complex and robust robotics software for various robotic platforms. Developers or researchers can use ROS standards to develop ROS packages which they can contribute to the global open source community and reuse code developed by other members of the ROS community.

Here we discuss a brief overview of the main concepts which are required to understand the concepts of ROS used in this thesis.

B.2 ROS Structure

ROS uses packages containing executables and libraries to organise the system software. The ROS Computation Graph manages the software at runtime. [30]

Some of the fundamental concepts in ROS are:

- **Nodes:** are the vertices of the ROS graph consisting of the runtime programs (processes) which perform computation for a particular functionality of a robot system.
- **Master:** It is the main program that provides the name registration and lookup for the various nodes in the computation graph. This allows the nodes to locate other nodes and interact with them by exchanging data.
- **Parameter Server:** It is a dictionary storing universal parameters which are publically available across the ROS Computation Graph. It is a part of the Master system, which allows the storage to be in a central location.
- **Messages:** Data is transferred between node packages in the form of data structures called messages. The messages can consist of various types of fields (integer, floating point, boolean, string, etc.) and can include nested structures like arrays or other messages.
- **Topics:** Messages a published from a node via asynchronous channels called topics. The published topics can be subscribed by other nodes in the ROS Computation Graph, after which the transmitted messages can be received by it. For a given topic, there can be multiple subscribers and publishers (though usually, it is only one publisher).
- **Services:** When a request-reply-based interaction between nodes is required, then the synchronous communication approach is done through services. Nodes can offer services under a certain name, other nodes can send requests to these services and wait for a response.

- **Bags:** ROS provides a mechanism to store messages in the form of ROS bags. Bags allow storing of sensor data and other messages transferred over the ROS Graph. Recording messages in this format allow reproducing logged data and facilitates easier diagnosis of issues or testing new features remotely.

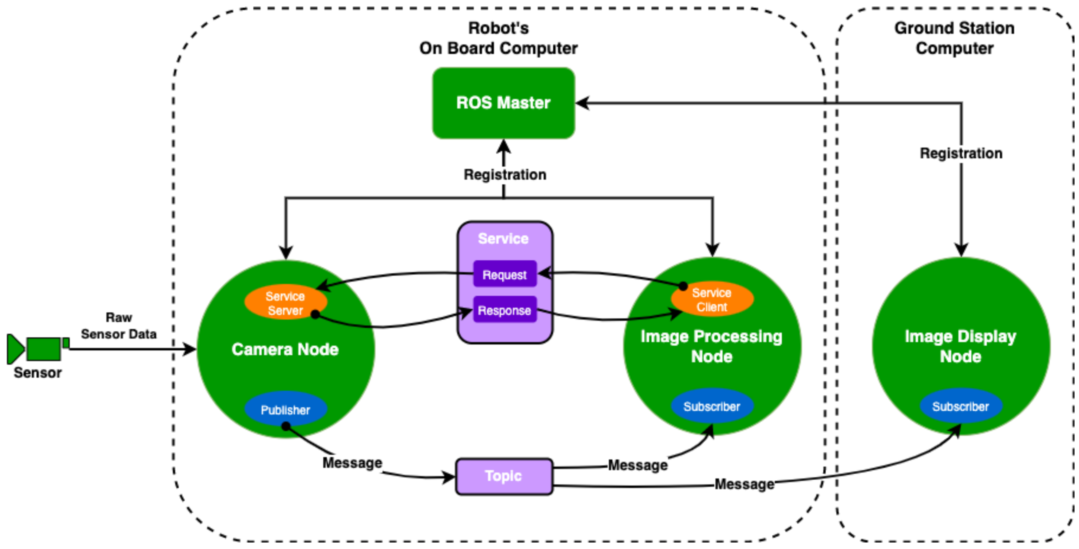


Figure B1: ROS Computation Graph - topics and services

Figure B1 shows an example of the ROS Computation Graph for a system that runs on two computers: one on the robot's onboard computer and one for the ground station computer. The ROS Master is running on the robot. Each node on the ROS graph has to register to the ROS Master to be accessible to other nodes. This system functions to record images from a camera sensor mounted on robot. Three nodes are running in the system [31]:

- the *Camera Node* captures raw image data from the camera,
- the *Image Processing Node* performs image processing(filtering, compression, etc) on the raw image data
- the *Image Display Node* displays the images on the ground station machine.

There are two ways that the *Image Processing Node* can acquire the images from the *Camera Node*:

i Firstly, the *Camera Node* uses a topic publisher to publish the images packaged as messages on a named topic, to which the other nodes are subscribed.

ii Alternatively, the *Camera Node* offers a ROS service server. The ROS service client on the *Image Processing Node* uses a service request call to the server on the camera node and waits for the response message.