Proceedings of the Canadian Society for Mechanical Engineering International Congress 2022
CSME Congress 2022
June 5-8, 2022, Edmonton, AB, Canada

CONTROL ARCHITECTURE DEVELOPMENT OF AN 8X8 SCALED ELECTRIC COMBAT VEHICLE

Jonathan Tse[1], Michael Peiris[1*], Moustafa El-Gindy[1], Zeinab El-Sayegh
[1]Faculty of Engineering and Applied Science, Ontario Tech University, Oshawa, Canada
*michael.peiris@ontariotechu.net

Abstract—This paper details recent advancements made to the control architecture of a 1:8 scaled multi-wheeled combat vehicle capable of eight-wheel drive and eight-wheel steer. The vehicle was previously developed to be used as a platform for autonomous navigation research with an autonomous parking algorithm being proposed and applied to the vehicle. Currently, a hybrid remote control system was developed to coexist alongside the existing hardware and autonomous software. The enhancements provide additional functionality during non-autonomous testing of the vehicle. Furthermore, these updates also reflect the networked vehicle architecture of next generation combat vehicles as seen in North Atlantic Treaty Organization Standardization agreements.

Keywords-military vehicles, steering configuration, 8x8, Ackermann

I. INTRODUCTION

At present, the automotive industry is in a widespread transition to designing vehicles with more sophisticated electronics and autonomous capabilities. Advances attributed to this are responsible for enhanced safety by using sensors that automate select tasks involved with driving [1]. Examples of this include blind spot detection [2] and automated lane change assistance [3] resulting in the bridging together of the fields of automotive engineering and mobile robotics. The vast majority of the existing literature in this field involves the application of algorithms and other control methods towards passenger vehicles or similarly constructed mobile robots [4]. As such there is a wide array of experimental research involving robots composed of four wheels and two axles [5,6,7]. To highlight some studies in this field, Blok et al. [8] use a four wheel, differential drive robot for navigation within an apple orchard and Peterson et al. [9] use a similarly constructed robot for exploring environments to find harmful radiation.

However, an area where increased investigation is required concerns mobile robots used for military purposes, commercial transport and other heavy vehicles. These types of vehicles are traditionally used in environments composed of rugged or uneven terrain while simultaneously carrying substantial loads. In order to satiate these requirements, these vehicles are usually constructed with arrangements utilising 6 or more wheels alongside additional axles which causes additional complexity [10]. In order to address this gap in the literature a 1:8 Scaled Electric Combat Vehicle (SECV) utilising eight-wheels has been constructed which will be used for future autonomous testing.

The SECV is a 1:8 scaled model of an 8x8 combat vehicle. The SECV can be seen in Figure 1. The SECV has several unique characteristics including complete independent eight-wheel (8x8) drive due to each wheel being connected to a dedicated motor. In addition, each wheel can be assigned its own steering angle value and be independently steered allowing for eight-wheel steer. As such, several steering configurations using various combinations of the axles and wheels are possible for varying levels of turn accuracy. The steering capabilities of the vehicle conform to the Ackermann Steering Criterion [12] which is a relationship between the steering angles of the inner and outer wheels of a vehicle when navigating a turn

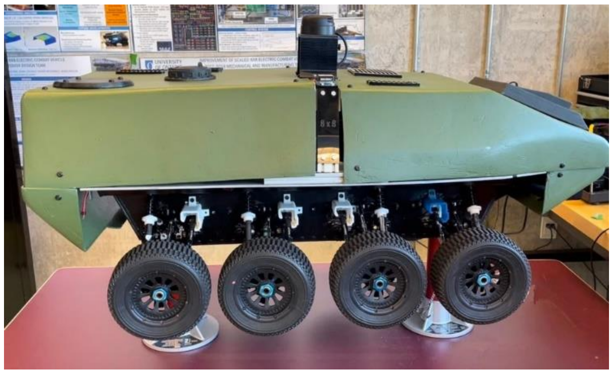
Figure 1. Side View of SECV

In addition, in order to facilitate autonomous motion on the SECV has been outfitted with several sensors, including: a

RPLIDAR Laser Scanning Sensor [13], AMT10 Rotary Encoders [14] and a UM7-LT orientation sensor [15]. Robot Operating System (ROS), an open source software environment is used to develop autonomous software and will be used for future autonomous development of the SECV [16]. ROS operates using node-based communication methods. In this case different robot components including both hardware and software are symbolized as nodes denoting executable files. ROS is heavily used in research and industry both for practical real-world applications and simulation [17]. Using these hardware and software components, a two stage autonomous parking algorithm, allowing the vehicle to locate a predetermined parking and successfully orient itself in the parking location was developed in 2021 and can be seen in Hao et al. [18].

Simultaneously occurring with the shift towards autonomous navigation technologies, sophisticated vehicle electronics are becoming more prevalent in full sized military and heavy vehicles. These electronics can include devices such as digital controllers, observation devices or other sensors. The continued improvement of digital electronics has also allowed a shift away from analog systems to digital systems capable of communicating with other electronics. These two factors have resulted in the implementation of standardization in the design of next generation combat vehicles regarding communication of vehicle electronics systems. One such organization producing the latest standards in this field include: the North American Treaty Organization (NATO) who have released the standard, NATO Generical Vehicle Architecture (NGVA) for land systems[19].

As such the current enhancements to the 8x8 SECV will focus on redesigning the remote-control architecture, as well as ensuring the architecture will mimic communication architecture standards that will appear on the next generation of armored vehicles. Thus, the main objectives of this work include ensuring communication between all devices (controllers, and sensors), addition of a remote-control system for non-autonomous use, testing capability of the steering configurations of the SECV and finally this work will also allow the vehicle to receive steering commands for autonomous operation and future autonomous work

II. BACKGROUND

Prior to the undertaking of current research on the SECV, the vehicle had limited autonomous navigation capability through the use of ROS. The SECV was able to perform a parking maneuver at low speed in a known environment [18]. All low-level control of the vehicle was handled through one set of controllers assigned for the steering actuators and one set of controllers assigned for the motors. The steering system involved using 2 Arduino Mega micro-controllers with each micro-controller responsible for 4 steering actuators. The drive system involved 4 Roboteq motor controllers with each controller responsible for driving two wheels.

The steering micro-controllers had identical programming that received commands from ROS via Universal Serial Bus (USB). ROS functions as a network of software "nodes" that process and publish data from locations known as "topics". Each node may be thought of as a black box integrated controller with its own specific functionality that may be modified. Topics may be thought of as a public bulletin board that nodes can subscribe to for information, and publish data to. Each topic can only contain one type of information. An example of how the motor controller operates in ROS can be seen in Figure 2.

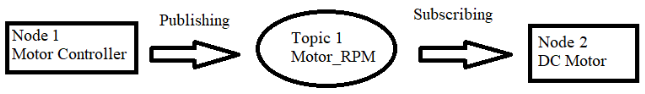

Figure 2. ROS Communication Example

The calculation for required steering angles, involves finding the correct steering angles according to Ackerman All Wheel Steering equations seen in the following sections of this paper. Each Arduino controller had a motor shield attached that was used to control the lengths of four actuators after the calculations were completed. To achieve direction and speed control, the four Roboteq motor controllers were connected through serial communication with one acting as a master controller. The master controller receives velocity commands via a USB connection to the laptop, and transmits these commands to other controllers. Each controller is capable of controlling two motors under closed loop speed control.
Finally, a major drawback of the SECV overall is that it could not be operated without initializing ROS on the laptop. In addition, the vehicle was only able to steer using all four axles and no other steering configuration and unable to switch between configurations during operation.

III. IMPLEMENTATION

To achieve the objectives of: remote control addition, device communication and addition of the ability to test all steering configurations several major changes were applied to the vehicle. The work was completed in two main phases detailed in the following sections.

A. *Phase A Modifications*

The primary objective of the first phase was to remove the need for an additional outside laptop and re-design the system to accept both ROS autonomous input and remote-control input from a human user. The major modifications include implementation of a Futaba T8J remote control system and changes to the steering code to allow for multiple steering configurations as seen in the Results section. The modification used serial communication to receive the Futaba remote control systems commands. Code for the Arduino controllers allowed them to receive commands from the Futaba system's wireless receiver via serial communication, and steer the axles accordingly. In terms of changes to the hardware, the steering controller responsible for the first two axles also received a connection to a new RS232 shield. The one steering controller was also designated as the front side controller in charge of the two front axles and the Roboteq motor controllers. While, the other steering controller was designated as the rear controller in charge of the two rear axles. The RS232 connection also allowed the remote control to connect to the motor controllers, allowing for speed control of the motors through the remote control. This allowed us to implement the independent remote-control capability requirement.

B. *Phase B Modifications*

After the conclusion of the phase A modifications, testing was carried out on the vehicle using the new remote-control system. It was found that the Futaba T8J system was not capable of sending consistent signals to the vehicle's controllers causing the steering and motor controllers to behave erratically without input from the operator. This can be seen in Figure 3 where a graph of the output signals of the remote control was taken while no operator input was provided. All curves for each channel should have no slope or steps indicating no change in controller input, but instead they are fluctuating rapidly. This was due to a compatibility issue with the frequency at which the remote control was transmitting and its associated receiver. It was also found that the front Arduino steering controller did not have the processing capability to perform all required tasks (steering angle calculations) at the required speed due to the frequency of its connection to the Roboteq motor controllers. These findings necessitated further changes to the remote-control system.

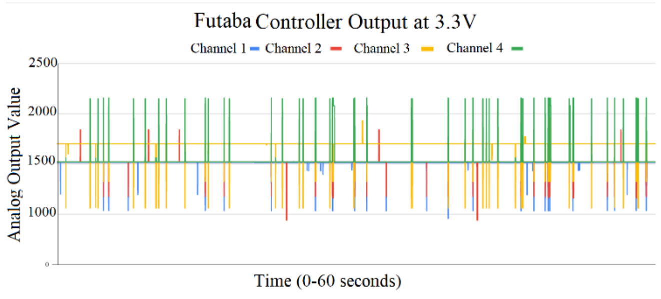

Figure 3. Sample output from Futaba controller

The current system is a complete redesign of the control architecture reusing the four Roboteq SDC 2130 controllers for motor control, and three Arduino Mega 2560 controllers linked together with serial communication for steering control. One steering controller acts as the master controller, and the two remaining steering controllers are slaved to it. The slaved steering controllers control the steering actuators according to serial commands received from the master through Transistor-Transistor Logic (TTL) communication. It should be noted that the Arduino controllers are only responsible for steering control under autonomous control, and the Roboteq motor controllers are directly plugged into the Nvidia TX2 computer to receive commands from ROS. This system has a built-in remote-control system that operates the vehicle independently from ROS, and also has the capability to communicate with a computer running ROS. This architecture is capable of achieving all objectives, and the effect of the modifications can be seen in detail in the following sections.

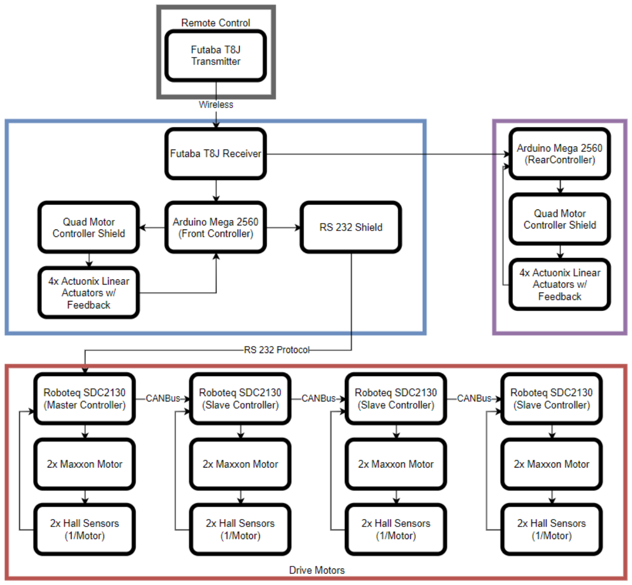
Remote Control
Futaba T8J Transmitter
Wireless
Futaba T8J Receiver
Quad Motor Controller Shield
Arduino Mega 2560 (Front Controller)
RS 232 Shield
4x Actuonix Linear Actuators w/ Feedback
Arduino Mega 2560 (RearController)
Quad Motor Controller Shield
4x Actuonix Linear Actuators w/ Feedback
RS 232 Protocol
Roboteq SDC2130 (Master Controller)
CANBus
Roboteq SDC2130 (Slave Controller)
CANBus
Roboteq SDC2130 (Slave Controller)
CANBus
Roboteq SDC2130 (Slave Controller)
2x Maxxon Motor
2x Maxxon Motor
2x Maxxon Motor
2x Maxxon Motor
2x Hall Sensors (1/Motor)
2x Hall Sensors (1/Motor)
2x Hall Sensors (1/Motor)
2x Hall Sensors (1/Motor)
Drive Motors

Figure 4. Final Control Architecture

IV. Results

The objective of the work was to develop a new control architecture for the 8x8 SECV that included an accurate remote control system for testing vehicle capabilities, and allowed for communication between onboard electronics for remote control and autonomous operation.

A. *Remote Control*

The initial use of a Futaba T8J controller resulted in an inconsistent signal received that caused random unmitigable stutters on the steering actuators and motor controllers. As a result, the Futaba T8J controller was replaced with a Logitech USB gaming controller connected to the master Arduino controller using a USB host shield, and USBHID Arduino library. This change has removed the problems exhibited by the Futaba T8J, and allows for accurate control of the vehicle. The current control scheme allows the user to control the vehicle's movement at multiple speeds, and select between crab or Ackerman multi-axle steering modes. The secondary benefit of using the Logitech USB controller is that its output may be communicated to other devices compatible with USB devices such as the computer running ROS.

B. Control of Steering Actuators

The steering of the vehicle is accomplished through the use of three interconnected Arduino controllers. One of the Arduino controllers acts as a master controller that receives input from either the remote control or a device running ROS, and sends commands to two Arduino controllers slaved to it.
The master Arduino controller receives high level commands on the desired angular changes in direction, and calculates the desired angle of all eight wheels based on Ackerman steering equations or crab steering angles. The required actuator extension for each wheel is calculated, and then sent as packet data to the two slaved Arduino controllers through the TTL serial connection. Each slaved Arduino board acts as a Proportional-Integral-Derivative (PID) controller for four actuators through the use of a custom shield that interfaces with the actuators, and one motor shield that drives the actuators. Each actuator contains a wiper that provides feedback to the Arduino controller by generating an analog signal that increases with extension length. The analog signals are used by the PID control code on the Arduino controller to set the power output for their respective actuator.
One example command is <25,25,25,25> which is a sample packet of data sent from master controller to slaved controller indicating that all actuators should be extended to 25mm. Each packet of data is contained within a “<...>” and contains four numbers. The abstracted form of each packet would be <L1,R1,L2,R2> where L1 and R1 are the left and right actuator extensions for the front axle respectively, and L2 and R2 are the left and right actuator extensions for the rear axle respectively controlled by the Arduino controller.

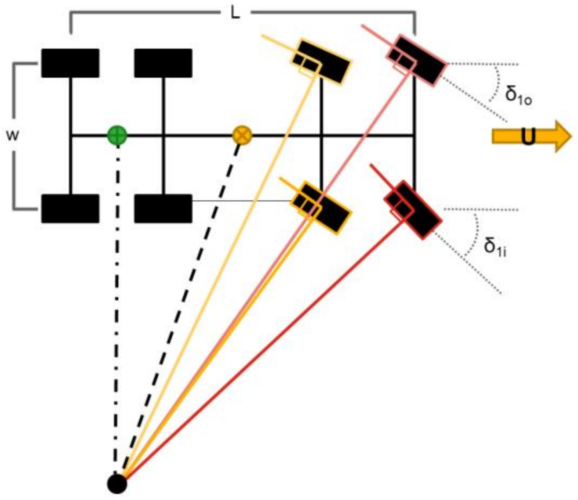

Figure 5. Diagram of Ackerman steering configurations for two axle steering, also known as front wheel steer [20]

The front wheel steering layout illustrated in Figure 5 is currently in use with the LAV III series of vehicles as well as most other four axled military vehicles in the world. The steering controllers use the values shown in Table 1 to calculate the appropriate steering actuator extension for each wheel based on a given input. This steering layout is able to achieve a minimum turn radius of 1.35m .

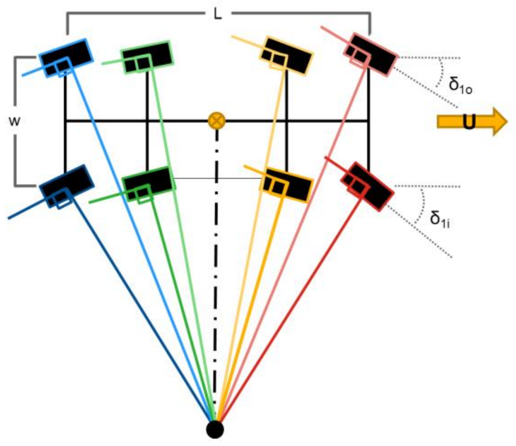

Figure 6. Diagram of Ackerman steering configurations for all wheel or four axle steering [20]

The all wheel steering layout illustrated in Figure 6 uses all wheels to steer the vehicle and has the smallest minimum turn radius. The values used by the controller for this steering layout are shown in Table II. It has a minimum turn radius of 0.87m that is the smallest of all the possible steering configurations.

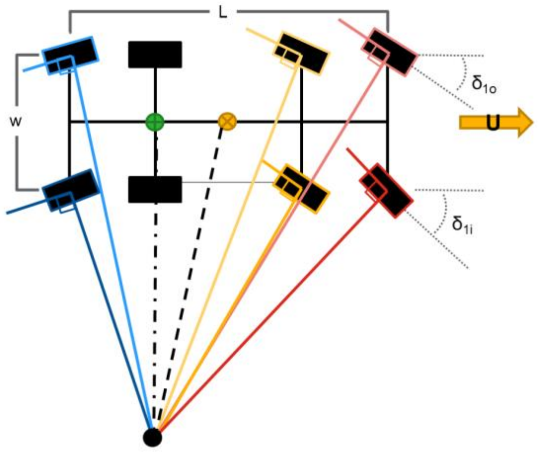

Figure 7. Diagram of Ackerman steering configurations for fixed third axle steering, (3rd axle is unable to steer) [20]

The fixed third axle steering layout illustrated in Figure 7 only steers the axles on the first two axles and rear most axle using calculations based on the values in Table III. It is able to achieve a minimum turn radius of 1.10m. While the minimum turn radius achieved with this steering layout is larger than that of all wheel steering, it is also less complex from a design perspective as fewer wheels need to be steered.

$$cot\phi_o - cot\phi_i = \frac{w}{l} \tag{1}$$

Equation 1 shows the process for calculating Ackerman Steering equation for calculating steering angles of inner and outer wheel on an axle. The values are calculated based on ϕ_o, the angle of the outer wheel, ϕ_i the angle of the inner wheel, w the track width, and l the horizontal distance of the wheel from the center of rotation. The steering angles calculated in the code are proportional to values calculated for the minimum turn radius based off of the diagram in Figure 5, and Equation 1. As the angle of the inside wheel of the first axle is always the highest in every steering configuration, it is always set to 24.79 degrees of angle and 20.00 mm of extension due to limitations of the mechanical system.

C. *Control of Steering Actuators*

TABLE I. FRONT WHEEL STEERING CONFIGURATION

		Angles (degrees)			**Extension (mm)**	
Axle[1]	***Length (m)***	***Inner***	***Outer***	***Average***	***Inner***	***Outer***
1	0.50	24.79	17.98	21.39	20.00	14.06
2	0.30	15.34	10.91	13.13	11.88	8.35

1-The Track width is constant for all axles at 0.46m

TABLE II. ALL WHEEL STEERING CONFIGURATION

		Angles (degrees)			**Extension (mm)**	
Axle[1]	***Length (m)***	***Inner***	***Outer***	***Average***	***Inner***	***Outer***
1	0.30	24.79	15.17	19.98	20.00	11.74
2	0.10	8.54	5.04	6.79	6.50	3.82
3	0.10	8.54	5.04	6.79	6.50	3.82
4	0.30	24.79	15.17	19.98		11.74

1-The Track width is constant for all axles at 0.46m.

TABLE III. FIXED THIRD AXLE STEERING CONFIGURATION

		Angles (degrees)			**Extension (mm)**	
Axle[1]	***Length (m)***	***Inner***	***Outer***	***Average***	***Inner***	***Outer***
1	0.40	24.79	16.79	20.79	20.00	13.07
2	0.20	12.77	8.42	10.59	9.81	6.41
4	0.20	13.25	8.74	11.00	10.19	6.66

1-The Track width is constant for all axles at 0.46m.

D. *Motor Drive Control*

The control of all eight driving motors is achieved through the use of four Roboteq SBL 2130 motor controllers that are linked together through serial communication connections. One of these controllers is designated as a master controller, and has authority over the other three controllers. The master controller receives Roboteq CANBUS language commands through a RS232 protocol serial connection from the master Arduino controller during remote control operation, or through a USB connection to a computer when under ROS control.

Each motor is controlled individually using CANBUS commands. As an example, *@00 !G 100* is a sample command sent to controllers indicating 100% output for all motors where *@00* indicates all motors, *!G* is the command for power output, and *100* is the desired power output value. Use of the Roboteq CANBUS commands allows for motors to be individually controlled, and it is possible to spin individual motors at different outputs or velocities if desired. Each controller is connected to its own power source, and is capable of controlling two motors each through open loop or closed loop control. Closed loop speed control is accomplished by the use of a rotary encoder connected to the output shaft of a pulley gearbox attached to each motor that provides rotational velocity feedback to the controller. These two options allow for the torque output or velocity of each wheel to be controlled if necessary.

E. *ROS Control*

The master Arduino controller may be connected by USB connection to a computer running ROS, and is programmed to subscribe to the cmd_vel topic. and receives twist geometry messages that contain vector values for the desired yaw of the vehicle, and forward movement. These two values are then used to calculate the required steering angle for the vehicle to follow the path generated in ROS when under autonomous operation.

```
rostopic pub /mobile_base_controller/cmd_vel
geometry_msgs/Twist "linear:
x: 0.5
y: 0.0
z: 0.0
Angular:
x: 0.0
y: 0.0
z: 0.0"
```

Sample ROS twist geometry message that is published to the cmd_vel topic to move the vehicle forwards. Only the linear x value and angular z value are required [3]. At present, this code has been tested and it is confirmed that it can receive input from ROS, but there has been no operational testing of the vehicle under ROS control yet.

V. Conclusion

In conclusion, the current architecture allows for the vehicle to meet the objectives required for the vehicle to be used for future autonomous research. This was accomplished by adding hardware such as an additional steering controller, creating a network of controllers linked with serial communication, and writing new code for the three Arduino steering controllers.

In comparison, previous versions of the 8x8 SECV's control required additional peripheral devices such as a laptop as the initial architecture could only function with a laptop running ROS connected. Furthermore, the SECV did not have remote control functionality. The second iteration of the architecture had remote control functionality that performed erratically due to an incompatible remote-control system, and could not receive ROS commands. Under the second phase of modifications the architecture involved each Arduino controller functioned independently, and the front Arduino controller did not have the processing capability required to function properly. Neither control architectures had the capability for serial communication between the Arduino controllers. As a result of the control architecture upgrade, the processing and steering control tasks are distributed between three Arduino controllers resulting in a reduced computing load for each controller. The master Arduino controller is now only responsible for the reception and processing of high-level commands, and transmitting commands to the slaved controllers. The serial communication network allows for data to be communicated between all of the controllers, and will allow for future expansion by allowing the steering controllers to be used as input or output ports for additional equipment if necessary.

Future development of the SECV should entail refining the control system's programming and hardware. While testing to see if the vehicle is able to receive input from ROS has been successful, operational testing has not yet been conducted with the vehicle under ROS control. It would also be desirable for all of the Arduino microcontrollers to be placed onto a custom printed circuit board with all of the shields integrated to reduce the wiring complexity in the vehicle.

References

[1] O. Saeed Asadi Bagloee Madjid Tavana Mohsen Asadi Tracey, "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies," Journal of modern transportation, vol. 24, no. 4, pp. 284-303, 2016.

[2] N. De Raeve, M. de Schepper, J. Verhaevert, P. Van Torre, and H. Rogier, "A Bluetooth-Low-Energy-Based Detection and Warning System for Vulnerable Road Users in the Blind Spot of Vehicles," Sensors (Basel, Switzerland), vol. 20, no. 9, p. 2727, 2020.

[3] M. Park, S. Lee, M. Kim, J. Lee, and K. Yi, "Integrated differential braking and electric power steering control for advanced lane-change assist systems," Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, vol. 229, no. 7, pp. 924-943, 2015.

[4] J. F. Sekaran, H. Kaluvan, and L. Irudhayaraj, "Modeling and Analysis of GPS–GLONASS Navigation for Car Like Mobile Robot," Journal of electrical engineering & technology, vol. 15, no. 2, pp. 927-935, 2020.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," The International Journal of Robotics Research, vol. 30, no. 7, pp. 846-894, 2011.

[6] P. Quillen and K. Subbarao, "Minimum control effort–based path planning and nonlinear guidance for autonomous mobile robots," International journal of advanced robotic systems, vol. 15, no. 6, p. 172988141881263, 2018.

[7] L. Yan and L. Zhao, "AN APPROACH ON ADVANCED UNSCENTED KALMAN FILTER FROM MOBILE ROBOT-SLAM," International archives of the photogrammetry, remote sensing and spatial information sciences., vol. XLIII-B4-2020, pp. 381-389, 2020.

[8] P. M. Blok, K. van Boheemen, F. K. van Evert, J. Ijsselmuiden, and G.-H. Kim, "Robot navigation in orchards with localization based on Particle filter and Kalman filter," Computers and electronics in agriculture, vol. 157, pp. 261-269, 2019.

[9] J. Peterson, W. Li, B. Cesar-Tondreau, J. Bird, K. Kochersberger, W. Czaja, and M. McLean, "Experiments in unmanned aerial vehicle/unmanned ground vehicle radiation search," Journal of field robotics, vol. 36, no. 4, pp. 818-845, 2019.

[10] M. El-Gindy and P. D'Urso, "Development of control strategies of a multi-wheeled combat vehicle," International Journal of Automation and Control, vol. 12, p. 325, 01/01 2018.

[11] G. D. L. Systems. (2020). Light armoured vehicles (LAV) [Online]. Available: https://www.gdlscanada.com/products.html.

[12] R. F. Carpio, C. Potena, J. Maiolini, G. Ulivi, N. B. Rossello, E. Garone, and A. Gasparri, "A Navigation Architecture for Ackermann Vehicles in Precision Farming," IEEE robotics and automation letters, vol. 5, no. 2, pp. 1102-1109, 2020.

[13] RobotShop. (2020). RPLIDAR A2M8 360° Laser Scanner [Online]. Available: https://www.robotshop.com/ca/en/rplidar-a2m8-360-laser-scanner.html.

[14] C. Devices. (2020). AMT10 Rotary Encoders [Online]. Available: https://www.cuidevices.com/product/motion/rotary-encoders/incremental/modular/amt10-series.

[15] RobotShop. (2020). UM7-LT Orientation Sensor [Online]. Available: https://www.robotshop.com/ca/en/um7-lt-orientation-sensor.html.

[16] O. S. R. Foundation. (2020, January). ROS, About ROS.

[17] A. Koubaa, Robot Operating Systems (ROS) - The Complete Reference. Cham: Springer International Publishing AG, 2016.

[18] A. H. Tan, M. Peiris, M. El-Gindy, and H. Lang, "Design and development of a novel autonomous scaled multiwheeled vehicle," Robotica, pp. 1-26, 2021.

[19] NATO GENERIC VEHICLE ARCHITECTURE (NGVA) FOR LAND SYSTEMS, 2016.

[20] A. Odrigo, El-Gindy, M, "“Application of Optimal Control for Skid Steering in Future Electric Combat Vehicle," International Journal of Automation and Control.