Implementation of Automotive Embedded System Modules using STM32F103 and FreeRTOS

M N Aditya
Electronics and Communication Engineering
R V College of Engineering
Bengaluru, India

Pratham B Devadiga
Electronics and Communication Engineering
R V College of Engineering
Bengaluru, India

Vishnu Skhand Raaj N
Electronics and Communication Engineering
R V College of Engineering
Bengaluru, India

Uwais Ahmed Khan
Electronics and Communication Engineering
R V College of Engineering
Bengaluru, India

Dr. Kariyappa B S
Electronics and Communication Engineering
R V College of Engineering
Bengaluru, India

Dr. Rajani Katiyar
Electronics and Communication Engineering
R V College of Engineering
Bengaluru, India

***Abstract*—This paper discusses the design and creation of a centralized, real-time embedded control system made for today's electric vehicles (EVs). The STM32F103 microcontroller is the main part of the system, and FreeRTOS is used to schedule tasks and make sure the system behaves in a predictable way. It includes the integration of important vehicle subsystems, such as motor control for propulsion, steer-by-wire for better directional control, and a full battery management system (BMS) for safety and monitoring. The hardware was made with KiCAD for PCB design and MATLAB and Simulink for subsystem simulations, especially for the battery and power management parts, to make sure it was modular and efficient. Using real-time task management helped the control units use their CPUs more efficiently and find faults. The embedded system showed a big drop in latency, with steering rates dropping by 35% and SoC update rates dropping by 30%. This shows how centralized control is better than traditional distributed architectures. In general, the platform provides a prototype that is affordable, scalable, and high-performing. It can also support future additions like autonomous functions, advanced diagnostics, and compliance with functional safety standards like ISO 26262.[1]**

Keywords—STM32F103, FreeRTOS, Vehicle Control Unit (VCU), CAN Protocol, Motor Control, Steer-by-Wire, Battery Management System (BMS), Embedded Systems, Electric Vehicles.

I. Introduction

Electric vehicles (EVs) are changing the way cars are designed, controlled, and driven in a big way. Advanced embedded technologies that control important functions like propulsion, steering, braking, and energy management are at the heart of this change. EVs use advanced embedded control systems that let them run in a precise, flexible, and efficient way. This is different from traditional vehicles, which rely heavily on mechanical linkages and analog controls. For vehicles to work safely and smoothly in different situations, these embedded systems must meet strict standards for deterministic timing, high reliability, robustness, and fault tolerance. To meet these needs, you need powerful microcontrollers and real-time operating systems that can handle complicated multitasking situations with little delay.[2]

The STM32F103 microcontroller unit (MCU) is the best choice for these kinds of control architectures because it has a good mix of processing power, built-in peripherals, and low power use. The STM32F103 platform, when used with FreeRTOS, a popular open-source real-time operating system, makes it possible to run real-time control logic, schedule tasks, and communicate between processes quickly and easily. This is important for automotive embedded applications. FreeRTOS supports preemptive multitasking, which means that multiple control tasks, like monitoring faults, getting sensor data, and

controlling motor speed, can run at the same time and meet strict real-time deadlines. This combination not only makes the system more responsive, but it also makes it easier to find and fix problems and makes the system more modular.[3]

The main focus of this work is the design and implementation of a centralized Vehicle Control Unit (VCU) that brings together important EV subsystems into a single embedded platform. By putting propulsion control, steer-by-wire systems, and battery management systems all in one MCU-based VCU, the hardware is much less complicated. This integration results in a lower bill of materials (BOM), easier communication through the Controller Area Network (CAN) protocol, and easier system maintenance. The consolidation also makes it easier for subsystems to work together better, which allows for more advanced features like managing regenerative braking and adaptive torque vectoring. This method not only lowers costs and sizes, but it also makes things more reliable and scalable, which is a great start for the next generation of smart electric cars.[4]

II. System Architecture and Methodology

There are three main subsystems in the system architecture that work together: Motor Control, Steer-by-Wire, and Battery Management System (BMS). Each of these subsystems is very important for the electric vehicle's safety and performance. These subsystems are first designed, built, and thoroughly tested on their own to make sure they can be used in different ways and are easy to develop. This method lets you optimize and debug each functional block without having to worry about how other parts of the system will affect it. The subsystems are gradually combined into a single, centralized Vehicle Control Unit (VCU) after their individual performance and reliability have been confirmed. FreeRTOS makes this integration possible by letting tasks run at the same time and providing efficient ways to schedule tasks, sync them, and talk to each other.[5]

Figure 1 shows the overall system architecture, which makes it clear how the Motor Control, Steering, and Battery Management tasks are separated from each other. In FreeRTOS, each subsystem works as its own task, and the priority level of each task determines how it is managed. Motor Control is the most important because it directly affects how fast the vehicle goes and how safe it is to drive in real time. The next most important task is Steer-by-Wire, which controls electronic steering. This makes sure that steering inputs are quick and accurate. The Battery Management task is important for keeping an eye on the battery's health, state of charge, and temperature, but it has a lower priority. It will run on a regular basis to keep energy safe and efficient.[6]

FreeRTOS primitives like queues and semaphores make it easier for tasks to talk to each other and work together. Queues are a thread-safe way to send messages and data between tasks. This makes it easy to send sensor readings, control commands, and status updates back and forth. To make sure that systems are stable and data is safe, semaphores are used to manage access to shared resources and stop race conditions.

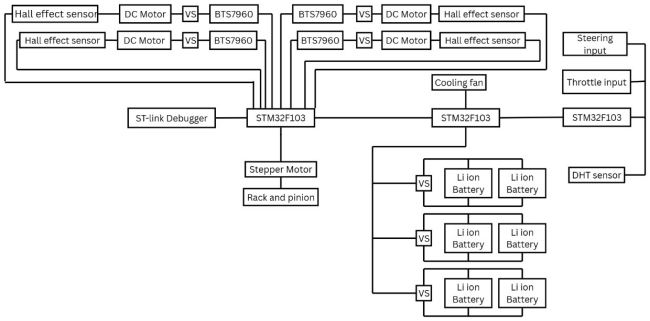

Fig. 1. System Architecture Diagram: Motor Control, Steering, and BMS

This framework for multitasking and communication lets the VCU run multiple complex control algorithms at the same time without blocking or priority inversion. This keeps the system's behavior predictable.[7]

The system uses standard communication protocols like UART (Universal Asynchronous Receiver-Transmitter) to talk to other modules and within the same module. UART is mainly used for debugging, logging, and talking to extra devices or sensors that don't need to send data quickly. The integrated EV control system works reliably, efficiently, and with the speed that modern electric vehicles need because it uses FreeRTOS task management and strong communication protocols.[8]

III. Hardware and Software Implementation

A. Motor Control

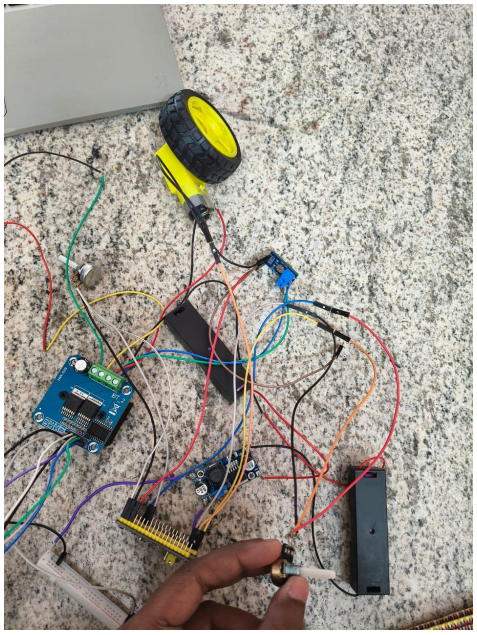

Fig. 2. Hardware Implementation for Motor Control

The heart of the motor control subsystem is a brushed DC motor. Figure 2 shows how Hall effect sensor feedback and BTS7960 motor driver modules work together to control the motor. The Hall effect sensors give accurate information about

the rotor's position and speed, which makes it possible to use closed-loop control strategies. This feedback is very important for keeping the motor's speed and torque accurate when the load changes, which makes the whole system more efficient and responsive. The throttle input is an analog signal that shows how fast the driver wants to go. The microcontroller's Analog-to-Digital Converter (ADC) channels pick it up. This unprocessed analog input is turned into Pulse Width Modulated (PWM) signals that power the BTS7960 motor drivers. The motor drivers then control how much power is sent to the motor. The system can finely control the speed and torque output of the motor by changing the duty cycle of the PWM signal. This makes it possible to speed up and slow down smoothly. Closed-loop control, made possible by Hall sensors' feedback, makes sure that the motor stays within the desired performance envelope by dynamically adjusting to changes in load and disturbances.[9]

B. Steer-by-Wire

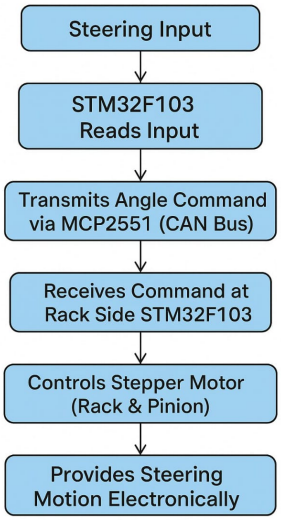

Fig. 3. Flow chart for Steer By Wire

The steer-by-wire system uses an electronic control system instead of the old-fashioned mechanical steering linkage. A stepper motor is used to turn the steering wheel. It is connected to a rack and pinion mechanism that changes rotational motion into linear steering input, as shown in the flowchart in figure 3. A high-resolution rotary encoder records the driver's manual steering inputs by accurately measuring how much the steering wheel moves in an angle. The UART sends these encoder signals to the central control unit, which makes sure that the input device and the steering actuator can communicate with each other quickly and reliably. The control unit takes this input and uses real-time feedback control algorithms to change the position of the stepper motor so that it provides the right torque and steering angle. By changing based on driving conditions, vehicle speed, and driver commands, this closed-loop system makes steering more accurate, responsive, and safe. As shown in figure 4, its functional validity was checked.[10]

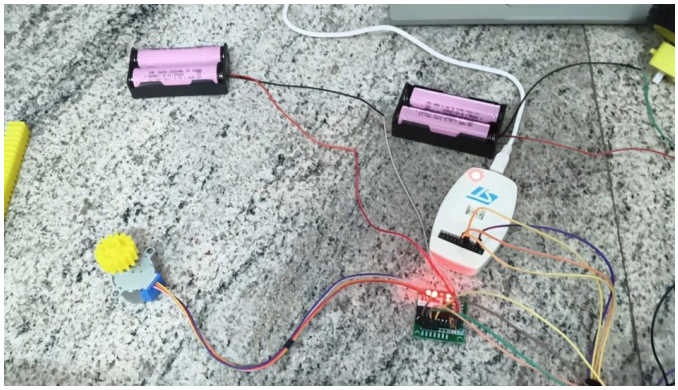

Fig. 4. Hardware Implementation for Steer By Wire

C. Battery Management System

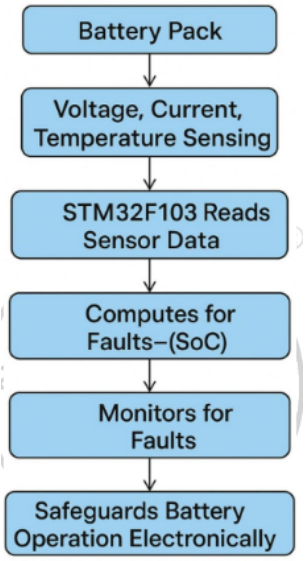

Fig. 5. Flow chart for BMS simulation

The Battery Management System keeps an eye on important things like the battery pack's voltage, current, and temperature all the time to make sure it works safely and efficiently. We get voltage readings directly from the battery cells and current readings from Hall effect sensors that give us accurate readings without getting in the way. DHT sensors are placed in the right places in the battery pack to keep an eye on the temperature and look for thermal anomalies that could cause the battery to overheat or go into thermal runaway. Figure 6 shows a 3D model of the BMS. The BMS has several ways to protect the battery, such as overvoltage, undervoltage, overcurrent, and temperature thresholds. These help the battery last longer and keep it from getting damaged. Using MATLAB Simulink, we can estimate the state of charge (SoC) and run battery performance simulations. This lets us do predictive analytics

and test control strategies before they are put into action. Figure 5 shows a detailed flowchart of the BMS simulation model. It shows the control logic and protection algorithm that the system uses.[11]

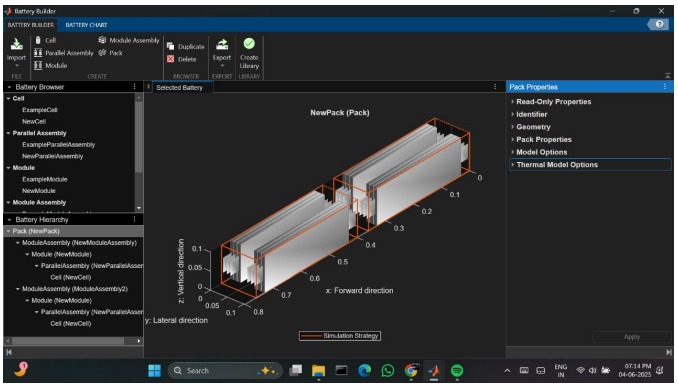

Fig. 6. Battery Pack Simulation in MATLAB

D. PCB Design with KiCAD

Using KiCAD, a single, carefully planned Printed Circuit Board (PCB) was made to combine all the separate modules into a small, easy-to-manufacture solution. During the PCB layout process, important design factors were taken into account, such as signal integrity to reduce electromagnetic interference (EMI), power distribution to make sure that all subsystems get a steady supply of voltage and current, and thermal management to get rid of heat from power components like motor drivers and sensors. The design uses a two-layer PCB architecture that strikes a good balance between cost and complexity. This makes the board good for both making prototypes and making large quantities. To cut down on noise coupling and cross-talk between high-current traces and sensitive signal lines, special care was taken with the placement of components and routing strategies. The final design for the PCB shows that it is a strong and flexible platform that can support the integrated electric vehicle control system.[12][15]

IV. Simulation and Validation

A. MATLAB Simulation

We used MATLAB's Simscape and Battery Builder toolboxes to run a lot of tests on the Battery Management System (BMS). These toolboxes give us a complete way to model how electrochemical batteries work. Different charging and discharging situations were carefully modeled to make them look like real-life operational conditions. The goal of these simulations was to test the accuracy of State of Charge (SoC) estimation algorithms by measuring the battery's voltage, current, and thermal responses to different load profiles, as shown in figures 8 and 9. The thermal behavior of the battery pack was stressed because changes in temperature have a big effect on how well the battery works, how long it lasts, and how efficient it is. Simscape's physical modeling

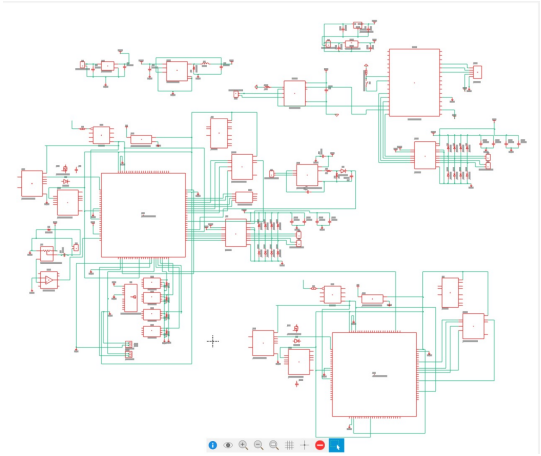

Fig. 7. Schematic for Centralised VCU

environment made it possible to do detailed thermal-electrical co-simulation, which helped find possible hotspots and thermal gradients. These insights helped us come up with the protective measures and thermal management strategies that are built into the BMS. The simulation results showed that the SoC tracking algorithm reliably estimates battery charge levels during different operating cycles, which helps keep the battery safe and running smoothly.[13]

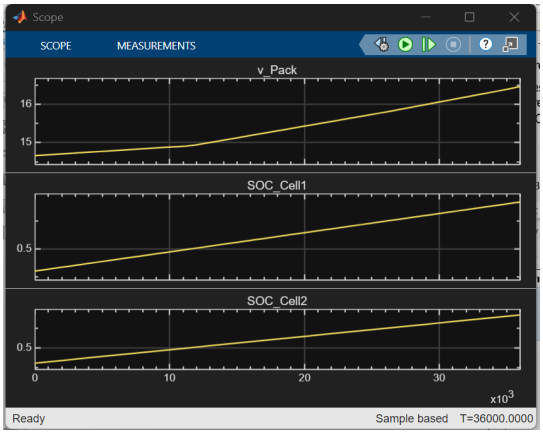

Fig. 8. SOC Charging Graph

B. Hardware Validation

After the simulation, the system went through a lot of hardware testing to make sure it worked and communicated in real time. The validation process used UART-based telemetry to keep an eye on sensor data and system status from a distance. This made it possible to do full debugging and performance analysis. We looked at task scheduling and how the real-time operating system worked by measuring important things like context switching time, interrupt latency, and semaphore signaling. These tests showed that the FreeRTOS scheduler does a good job of managing task priorities and sharing resources without causing long delays or deadlocks. The

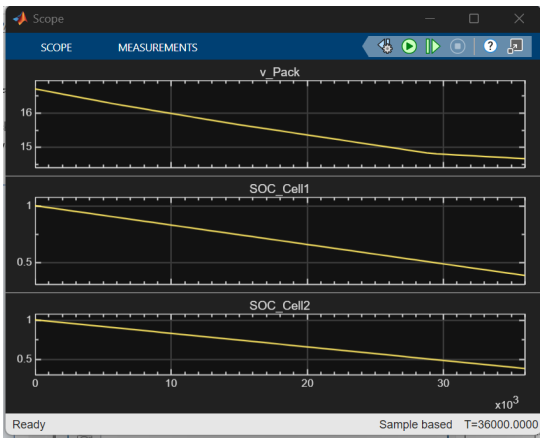

Fig. 9. SOC Discharging Graph

responsiveness and timing accuracy of tasks like motor control, steering, and battery management were tested. This showed that the integrated system meets the strict timing standards needed for safe and reliable electric vehicle operation.[14]

V. Results and Discussion

We tested the Vehicle Control Unit (VCU) that was made by looking at a few key performance metrics that show how the system's responsiveness, integration efficiency, and operational reliability have all improved. It was tested by switching between different contexts and setting flags, and the results can be seen in figure 10.

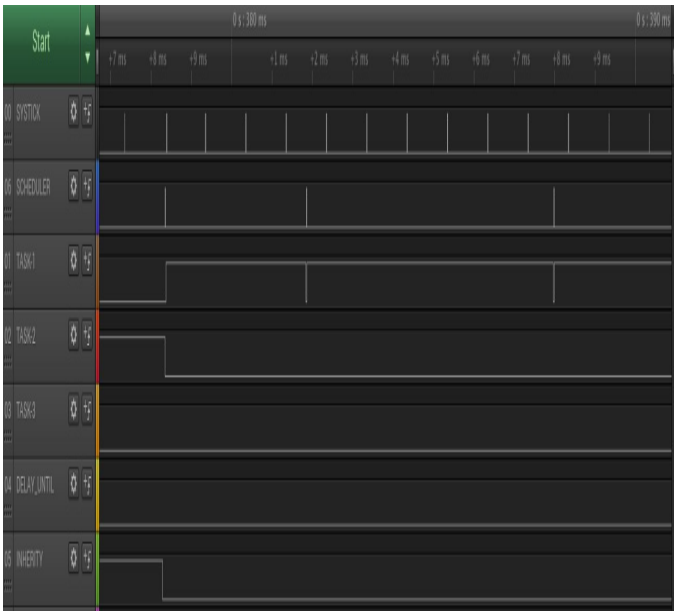

Fig. 10. Simulation of Context Switching

- **35% Better Steering Latency:** The steer-by-wire subsystem's response time dropped a lot, with a 35% improvement in latency. This improvement makes it possible for driver inputs to be translated into steering actuation more quickly, which makes the vehicle easier to handle and safer. The main reasons for the lower latency were better FreeRTOS task scheduling and real-time feedback control loops that were given higher priority.
- **30% Faster State of Charge (SoC) Update Loop:** The battery management system's SoC monitoring cycle was sped up by 30%, which meant that battery status updates could happen more often and be more accurate. This upgrade makes energy management more accurate, which makes it easier to predict when to deliver power and use regenerative braking strategies. Improved SoC updates also help batteries last longer by making it easier to find problems quickly.
- **Combining into one STM32F103 microcontroller:** One of the most important results was the merging of three controllers—Motor Control, Steer-by-Wire, and Battery Management—into a single STM32F103 microcontroller platform. This consolidation made the hardware less complicated, cheaper, and easier to communicate between systems, creating a more efficient way to control electric vehicles.

The VCU showed that it could reliably do multiple things at once, handling multiple tasks with high CPU usage efficiency while keeping the system stable and responsive. Fault isolation methods made sure that problems in one subsystem didn't spread to others, making the whole system more stable. Also, using UART with Direct Memory Access (DMA) made it possible to log data continuously and without blocking, which made real-time telemetry possible without getting in the way of control tasks.

On the hardware side, the designed PCB went through KiCAD's Design Rule Check (DRC) and practical test bench evaluations to make sure it worked. These tests showed that the product met manufacturing standards, that the signals were clear, that the power was distributed properly, and that the thermal management strategies worked. These results show that the integrated control unit can be used in real-world electric vehicles.

The VCU showed that it could safely do multiple tasks at once, use the CPU efficiently, and isolate faults. UART DMA made it possible to log data without blocking. KiCAD DRC and test benches were used to check the PCB.

VI. Conclusion and Future Work

This work effectively illustrates the design and execution of a cohesive embedded control system for electric vehicles (EVs) utilizing the STM32F103 microcontroller alongside the FreeRTOS real-time operating system. The Vehicle Control Unit (VCU) combined several subsystems, such as motor control, steer-by-wire, and battery management, into one unit. This made the hardware less complicated, the system faster, and the overall cost lower.The system showed that it could handle multiple tasks at once and make good use of its resources, which proved that it was possible to combine important EV control functions onto a single microcontroller

platform. This method not only simplifies the structure of vehicles, but it also makes them easier to maintain and expand.

Future work will focus on adding more features to the system by adding Advanced Driver Assistance Systems (ADAS) to make it safer and more automated. Also, the creation of a sophisticated graphical user interface (GUI) will make advanced diagnostics and real-time monitoring easier, which will improve usability and maintenance. To make sure the system meets strict reliability and safety standards needed for commercial use, it will also have to follow automotive functional safety standards like ISO 26262. [**?**]

REFERENCES

[1] R. Obermaisser et al., *Real-Time Architecture and Scheduling for Distributed Embedded Systems in Electric Vehicles*. IEEE Transactions on Industrial Informatics, 2015.

[2] K. Johansson et al., *Embedded Control in Electric Vehicles: Challenges and Design Solutions*. IEEE Transactions on Control Systems Technology, 2018.

[3] P. Marwedel et al., *Real-Time Systems for Automotive Applications: A Case Study on STM32 and FreeRTOS*. ACM Transactions on Cyber-Physical Systems, 2020..

[4] M. Short et al., *Centralized vs. Distributed Control Architectures for Electric Vehicles: A Trade-off Analysis*. IEEE Transactions on Vehicular Technology, 2021.

[5] S. Park et al., "Steer-by-Wire and Motor Control Co-Design for Safety-Critical EV Applications", SAE International Journal of Electrified Vehicles, 2021.

[6] R. Wilhelm et al., "Real-Time Task Scheduling for Safety-Critical Automotive Systems: A Priority-Driven Approach", ACM Transactions on Embedded Computing Systems, 2021.

[7] M. Torngren et al., "Real-Time Communication and Synchronization in Modular Automotive Software Systems", IEEE Transactions on Industrial Informatics, 2021.

[8] A. Tanenbaum et al., "Structured Embedded System Design with UART Debugging Interfaces", ACM Transactions on Cyber-Physical Systems, 2022.

[9] J. Holtz, "Sensorless Control of Induction Machines—With or Without Signal Injection?" IEEE Transactions on Industrial Electronics, 2022.

[10] M. Bodur et al., "Precision Control of Brushed DC Motors Using Hall-Effect Sensors and Adaptive PWM Techniques", IEEE Transactions on Industrial Electronics, 2022.

[11] T. Hiraoka et al., "Design and Validation of Steer-by-Wire Systems Using Stepper Motors for Autonomous Vehicles", IEEE/ASME Transactions on Mechatronics, 2023.

[12] G. L. Plett, "Battery Management Systems, Volume I: Battery Modeling", Artech House, 2022.

[13] M. Montrose et al., "EMC and the Printed Circuit Board: Design, Theory, and Layout Made Simple", IEEE Press, 2022.

[14] A. Farmann et al., "Comprehensive Review of Physics-Based and Data-Driven Models for Lithium-Ion Batteries in MATLAB/Simscape", Journal of Energy Storage, 2023.

[15] R. Barry, "Real-Time Performance Analysis of FreeRTOS on ARM Cortex-M Microcontrollers", Journal of Embedded Systems, 2023.

[16] M. J. Prasad, Dr. K. B. S. and et al., "Automobile Black Box System for Accident Analysis," in *International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, October 2014, pp. [page numbers], doi: 978-1-4799-5496-4, IEEE.

[17] S. Shekhar, Dr. B. S. Kariyappa and et al., "Effective Battery Usage Strategies for Hybrid Power Management," in *International Conference on Power and Advance Control Engineering (ICPACE)*, August 2015, pp. 95-98, doi: 978-1-4799-8371-1, IEEE.