Scaled physical model of electric self-driving bus

Robert Seredenko
Department of Electrical Power Engineering and Mechatronics
Taltech
Tallinn, Estonia
rosere@taltech.ee

Meelis Kobin
Department of Electrical Power Engineering and Mechatronics
Taltech
Tallinn, Estonia
mekobi@taltech.ee

***Abstract*—This paper presents the design and development of a scaled physical model of an electric self-driving bus, focusing on the mechanical, electronics, firmware, and Android app aspects. The project successfully met most of the key performance indicators, but due to delays in obtaining mechanical parts, the physical prototype was not completed. Future possible developments include updating the model body, improving the app, and exploring autonomous driving capabilities.**

Keywords—Scaled model, electronics, self-driving bus

I. Introduction

The project described in this paper is conducted as part of the UTT0110 Mechatronics and Smart Systems Project course, where project topic was proposed by the Mechatronics and Autonomous Systems Research Group. Originally, the project topic proposed was the "Design of a Smart Toy Car for a promotional competition", which was initiated by another team one year prior. However, after careful consideration and discussions with the supervisors, the final topic name was refined to "Scaled physical model of an electric self-driving bus." The decision to shift the project focus was inspired by TalTech University's own self driving bus project ISEAUTO v1.

It is worth noting that originally a team of four members started working on this project, but due to unforeseen circumstances, the team experienced a reduction in team size, with only two members remaining at the end. This resulted in a significant increase in workload per team member and a reduction in the final scope of the project.

Documentation for the project can be accessed [1].

II. Project Scope and Scale model Requirements

To effectively tackle the project, the workload was distributed between the team members by their skills. Robert took on the responsibility of handling electronics, firmware, and the Android application (app), while Meelis focused on mechanical design. The project commenced by reviewing the work previously completed by the initial team, with an emphasis on identifying parts that could be reused.

A. Scale model Requirements and Guidelines

Following extensive discussions with supervisors, a comprehensive list of requirements and guidelines was established:

- The scale model was intended to be assembled by students in grades 9-12, serving as an introduction to mechatronics.
- Model should be straight forward to assemble and simple to use without prior experience.
- All the parts of the model were expected to be open-source, from CAD files to printed circuit board (PCB) design files.
- All model parts need to possess good repairability.
- If applicable, incorporate multiple protections to protect the user and the model.

B. Project Scope and Key Performance Indicators

Several key performance indicators (KPIs) were set to gauge the success of the project:

- A working PCB with requirements fulfilled.
- Verified functionality of the microcontroller firmware and Android app.
- Assembled physical prototype of the scaled bus model.
- Model capable of remote-controlled movement.

III. Scale Model Design

This chapter is divided into three main parts:

- Mechanical design
- Electronics
- Firmware and Android app

Each part will be discussed briefly, highlighting how it addresses the project requirements and how the team achieved their goals.

A. Mechanical Design

The mechanical part of this project did not go as planned. The biggest factor why it suffered was that the team was cut short by two members, when they announced that they will not continue with them in this project, because, the mechanical side was supposed to be carried out by two people minimum. This was also mentioned in the beginning.

When they first started exploring the ideas between each other and their supervisors, they did not have a specific plan or a direct example to follow on the mechanical side. Project similar to this was done by a group of students a few years ago, but supervisors wanted the new group to learn from the previous groups mistakes and inefficiencies and create something better, more unique and make the design look more futuristic – which were the words they needed to follow when they thought about possible designs for the outer part of this project.

Once the guidelines were given, they started thinking about the smaller details and by that they thought about how to make the car remotely controllable and which parts they needed for it. The car base itself was the key factor to get the vehicle running, so they started with that. The supervisors mentioned that using already existing work to make the process easier was more than welcome. Because of that, they went on to search for already modelled 3D designs, which would fit their criteria for details and size. The main aspect for them was to find something, which had an existing transmission system to have torque to the wheels from a brushed DC motor, a steering system, which they would use to control the steering with a servo motor and one the requests from their supervisors – to have a suspension system. It took some time, but they were able to find a suitable design from an internet website which checked all of their criteria for the design.

To edit the existing 3D designs, the team would use SOLIDWORKS as their main program, since the existing files were also modelled there. They had some issues in the beginning with the program activation, but once that was resolved, they proceeded to start understanding the program, since they did not have much prior experience with it.

Once they agreed on the base of the vehicle, they set out to start printing the parts with existing 3D printers in their school laboratories. Meanwhile they also thought about their outer design of the vehicle, and they came up with an idea – to replicate the first version of TalTech self-driving bus. They proposed the idea to their supervisors, and they gladly accepted it. The supervisors also had a few ideas, how to make it more fun and intriguing if the group had the spare time for it, like making the doors as realistic as possible with the movement similar to what the real one has. One of their supervisors also got them a geometric model of the TalTech self-driving bus so that they could follow the replicating process as precisely as possible on the modelling side.

Printing the base of the vehicle was first suggested by their supervisors, that they will do it themselves, because they just had maintenance going on for the printers and they did not want to jeopardize it. A few weeks went by, when they announced that they will start printing the base of the car once they get time for it. They discovered that they did not have much free time left on their hands and offered the group to do the printing instead. The timing was a bit bad for the group, since we were dealing with other classes during that time, and they had to postpone it. During that time, they also started modelling the outer part of our project (see Fig. 1).

Some time went on and they received the PCB 3D model, which was supposed to be used to control the vehicle. To implement it into our design by fitting it on the base of the car, we needed to modify the design by adding supports to the base, where the PCB would be mounted.

Since the printing got delayed, there is no finalized physical model of the vehicle existing yet.

Fig. 1. 3D model of Scaled electric self driving bus.

B. Electronics

When the previous team was building “Smart Toy Car” prototype, they used breadboard to assemble and connect all electronics parts. Breadboard approach could work, but it will not meet the requirement of simple assembly and connection reliability would be questionable in high vibration environment. Based on previous points PCB approach was selected.

To meet simple assembly requirement, the choice of PCB components was limited to only through-hole components, which are easier to solder than surface mount components. The choice of Raspberry Pi Pico as microcontroller (MCU) was done beforehand by the supervisors. To satisfy repairability requirements, the PCB design incorporated a as many as possible discrete components.

After extensive consultation with supervisors, the team decided to employ a brushed DC motor for controlling the model's movement in both directions, necessitating the use of an H-bridge circuit. In addition to the H-bridge circuit, the PCB had to include circuits for four individually controllable LED, a 5 V linear regulator, battery voltage monitoring using a voltage divider and Analog to Digital Converter, an external Bluetooth module, two servo headers, and a connector for the Pi Pico microcontroller.

Robust input protections needed to be also implemented, as 3S 18650 Li-Po battery will be used as power source. Protections included two different methods for reverse polarity protection (P-Channel MOSFET and Diode method), a physical SPST switch for battery disconnection, and a 3 A slow-blow fuse to safeguard against short circuit events.

Before proceeding with the PCB design, the team conducted thorough simulation of the H-bridge circuit. The design of the H-bridge circuit presented challenges due to the microcontroller's 3.3 V general-purpose input-output (GPIO) pins. To overcome this limitation, three potential solutions were explored:

- Using logic level MOSFETs with a gate threshold below 2V and controlling MOSFETs directly from GPIO pins.
- Employing external gate driver IC,
- Creating a custom gate driver.

The team opted for the third solution as it offered greater flexibility, although it required a more meticulous approach.

The final configuration of the H-bridge circuit utilized two P-Channel and two N-Channel power MOSFETs, with four NPN bipolar junction transistors controlled by four MCU GPIO pins (See Fig. 2). Each NPN transistor controlled an individual power MOSFET by either pulling up to supply or pulling down to ground voltage on the MOSFET gate. As the MOSFETs already have reverse-biased body diodes, discrete protection diodes were deemed unnecessary. Simulation results were favourable, and the same circuit configuration was implemented on a breadboard for final verification. During 30 minutes of continuous stress testing with different control and pulse-width modulation (PWM) patterns at average current of 1A, no issues were observed.

Once the design verification was completed, the team used KiCad 7 software to design the PCB. The size of the PCB was limited to 100mmx100mm to manage costs. Special attention was given to the high-current H-bridge circuit, necessitating traces as wide as possible. Furthermore, efforts were made to segregate circuits based on their functionality and incorporate text and shapes on the silkscreen to enhance assembly clarity. Male and Female Headers pins were strategically employed wherever applicable to facilitate the replacement of broken components such as the MCU, Bluetooth module, LEDs. Extensive silkscreen markings were incorporated to simplify assembly and identify various parts of the PCB (See Fig. 3).

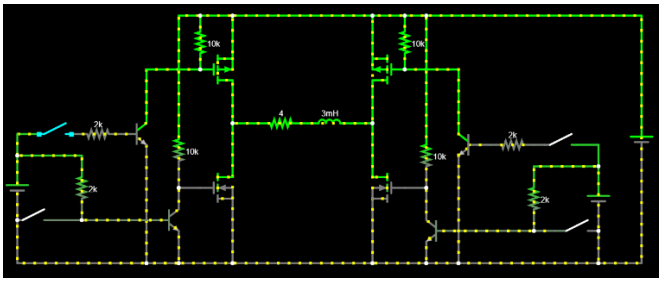

Fig. 2. H-bridge circuit simulation in Falstad simulator [2].

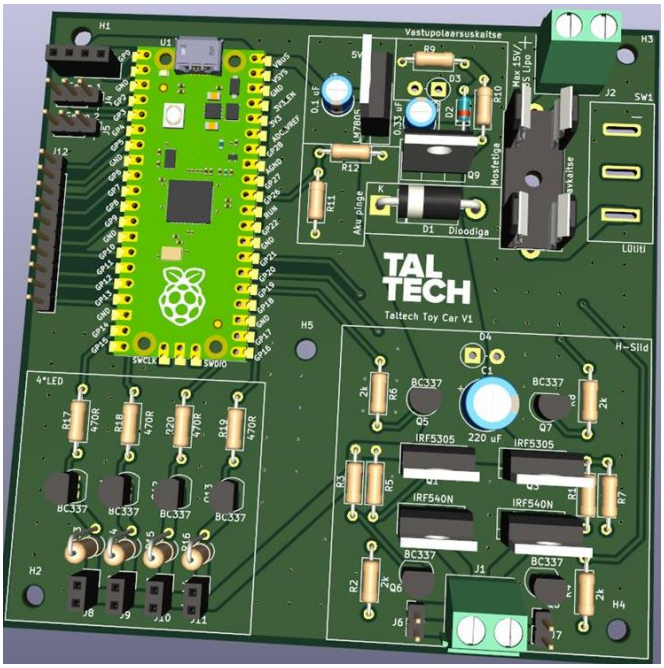

Fig. 3. 3D model of PCB in KiCad 7.

C. *Firmware and Android application*

Originally, the plan involved developing both an Android and iOS app. However, due to the departure of a team member responsible for this aspect, the plan was revised, and the focus shifted solely to the Android app. Fortunately, Robert had prior experience with a similar project and possessed a basic Bluetooth remote control app. During the initial phase of development, the team utilized the ESP32C3 microcontroller since Raspberry Pi Pico MCUs were not available. The firmware code was initially written in C using PlatformIO extension in Visual Studio Code. When the Pi Pico became available, the existing code was ported to it, as using C with PlatformIO offered better performance compared to interpreted languages like MicroPython.

The firmware structure was designed to be straightforward, encompassing tasks such as reading and processing UART communication from the external Bluetooth module, generating PWM signals for servos, and generating the appropriate PWM signals for the H-bridge. To facilitate hardware PWM generation and servo control, the team utilized the RP2040_PWM [3] and RP2040_ISR_Servo [4] libraries. Initially planned GPIO LED control and battery State of Charge calculations were excluded from the scope of the project due to insufficient time to complete the tasks.

A notable improvement compared to the work of the previous team was the development of a custom communication protocol. Instead of relying on 1-byte packets, a custom protocol utilizing 3-byte packets was introduced. Each packet consists of a command byte, motor PWM value byte, and servo angle byte, providing greater control and flexibility.

The Android app was developed using MIT App Inventor. It featured a button for Bluetooth connection, a button for disconnecting Bluetooth, and three additional buttons to control the speed (Slow, Average, Fast). The central element of the app was a smoothly moving joystick that returned to the centre position when not pressed (See Fig. 4).

Adapting Robert's existing app to the current project required only minor changes. The communication protocol was changed to support the new protocol, and three speed control buttons were added. The rest of the app remained unchanged.

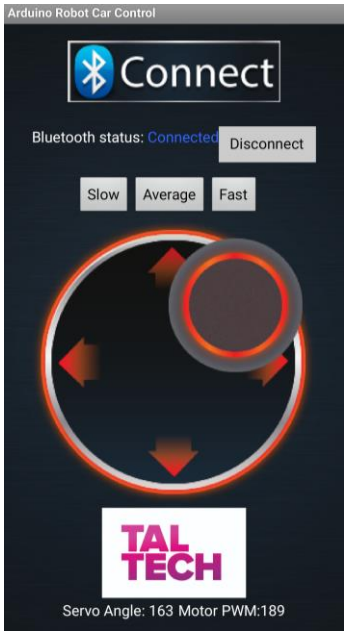

Fig. 4. Android app user interface with Bluetooth in connected state and joystick in top right corner.

IV. Key performance indicator evaluation and potential future development

A. KPI Evaluation

The evaluation of the KPIs indicated the following outcomes:

- PCB successfully fulfilled all the established requirements.
- Microcontroller firmware and Android app demonstrated effective communication with no significant issues.
- The physical prototype was not completed due to delays in printing the mechanical parts, and as a result, the model could not move under its own power.
- However, if the mechanical part of the physical model had been completed, successful achievement of all KPIs would be possible as the PCB, firmware, and Android app are functioning properly.

B. Potential future development.

Several potential future developments could further enhance the project. These include updating the model body to ISEAUTO v2, improving the App to incorporate support for LED control, battery state of charge measurement and offering IOS version of the app. Firmware support needs to be added for LED control and battery state of charge measurement.

Another potential development direction could be possibility of adding autonomous driving functionality to the scaled model. As execution of autonomous driving requires significantly higher computational power and multitude of different sensors, the feasibility of this development direction needs to be studied further.

Acknowledgment

Team wants to thank supervisors Martin Sarap and Viktor Rjabtšikov for great support from start until end of the project course.

References

[1] R. Seredenko and M. Kobin, “Github repository for Scaled physical model of electric self-driving bus,” 17 05 2023. [Online]. Available: https://github.com/Sp1kys/Scaled_Electric_Bus/tree/main. [Accessed 17 05 2023].

[2] P. Falstad, “Circuit Simulator Applet,” [Online]. Available: https://www.falstad.com/circuit/. [Accessed 17 05 2023].

[3] K. Hoang, “Repository for RP2040_PWM library,” [Online]. Available: https://github.com/khoih-prog/RP2040_PWM. [Accessed 17 05 2023].

[4] K. Hoang, “Repository for RP2040_ISR_Servo library,” [Online]. Available: https://github.com/khoih-prog/RP2040_ISR_Servo. [Accessed 17 05 2023].