Automated Driving Using External Perception

Individual Lab Report

MRSD Project Course I

Spring 2023

Shreyas Jha

Team E: OuterSense

Teammates: Atharv Pulapaka, Dhanesh Pamnani, Jash Shah, Ronit Hire

ILR02

Feb. 15, 2023

Contents

Individual Progress

I have primarily worked on the on-board electronic hardware and embedded system for the RC car. I enabled teleoperation of the RC car using a ESP32 microcontroller board and the Blynk IoT platform. The setup is shown in Figure 1 and the code is available at this link. The combination of all the electronic hardware required on-board: 3S LiPo or 6-cell NiMh battery, a microcontroller (ESP32), a servo motor powered by the microcontroller for controlling the Ackermann steering mechanism, and a 30A ESC (Electronic Speed Controller) connected to the BLDC (Brushless DC) motor and it's 5V BEC (Battery Eliminator Circuit) output to power the microcontroller. After connections on the vehicle are made, the RC car can be teleoperated by a different mobile device by following the steps below:

1. Install the Blynk IoT mobile app and find the authorization token for the device.
2. In the code (linked above) update the new authorization token and flash it onto the ESP32. Edit the Wi-Fi SSID and password in the code as required prior to flashing.
3. Configure two slider buttons in the Blynk app, with virtual pin "V4" configured for steering inputs and virtual pin "V1" configured for throttle inputs.

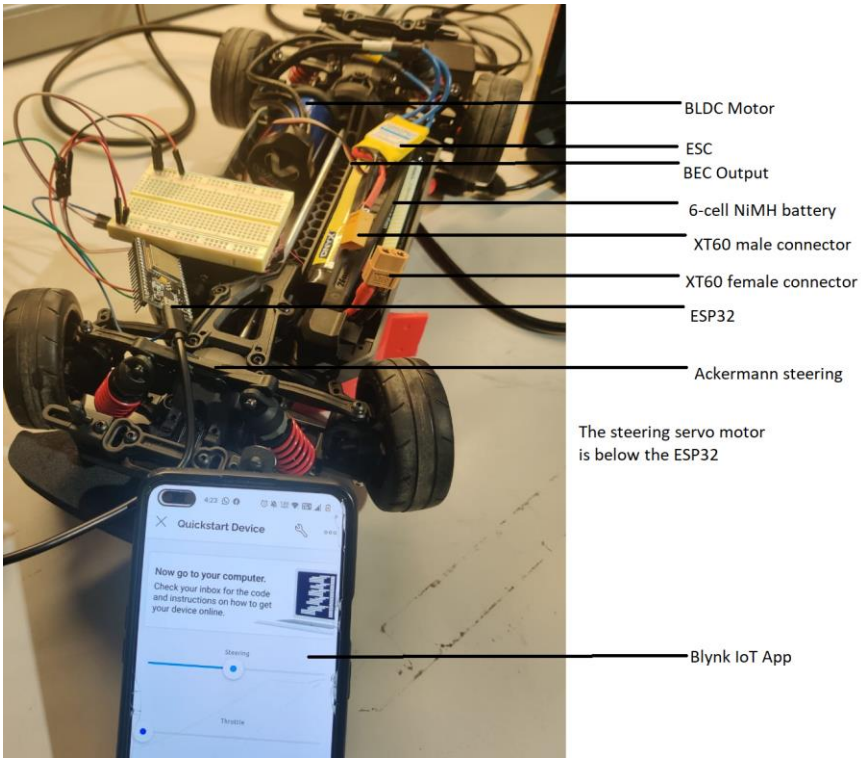

Figure 1: Teleoperation of the RC Car

I also tried different libraries [1][2] for the UM7 orientation sensor and the VESC [3][4] to collect accurate IMU data. I also identified a suitable Raspberry Pi model, which we intend to use in the long-term for controlling our RC cars as this is a Linux based single-board computer - has integrated Wi-Fi, a built-in clock and ROS can be easily used with it.

Challenges

Speed control of Velineon 3500 BLDC motor: The requirement for OuterSense is to drive the RC car at a minimum average speed of 12.5 cm/s. As we build the first iterations of our cyber-physical system, we want to operate the RC car at the slowest possible speed. We saw that it was possible to drive the car off the shelf at much slower speeds than with the current setup. Currently the minimum motor RPM is higher than desired, and we need to find a way or a different combination of electronic components to drive the RC car at lower speeds.

Drift in IMU yaw readings: There is continuous drift in the yaw readings from both the UM7 and VESC IMU. Potential solutions are to use sensor fusion and filtering, process raw sensor data in better ways, dive deep into calibration techniques, to purchase a more accurate IMU with very low drift rates or use a different approach to estimate vehicle pose.

Teamwork

Atharv Pulapaka: Atharv has been primarily working on studying the MPC architecture Model Predictive Control. He has been testing this using Simulink and has come up with estimates of the lane deviation and the communication latency which we can smoothly recover from.

Dhanesh Pamnani: Dhanesh has been owning the mechanical design, procurement, machining, and assembly of the hardware for the track and infrastructure units. Dhanesh has almost built one complete infrastructure sensing unit by assembling all the parts to make a robust and reliable platform to mount our sensors at a height such that they can span the entire track with a birds eye view. He was aided by Ronit and Jash to expedite setting up the mechanical hardware.

Jash Shah: Jash has been working on a pose estimation method. He has used OpenCV and developed an algorithm to detect a square-shaped marker (which we can paste on the hood of the RC car) and estimate its yaw angle from a Birds Eye View. He will be working to make this more robust – invariant to changes in illumination or height. Jash has also actively been involved in the manufacturing of the infrastructure sensing units. Jash, Dhanesh, and Ronit have also been brainstorming about how to mount the Intel RealSense D435i so we can attach it to the infrastructure sensor and have minimum vibrations resulting in better raw sensor readings.

Ronit Hire: Ronit helped me in enabling teleoperation of the RC car. He has been primarily supporting in the manufacturing of the infrastructure sensing unit. He also worked on procurement of parts and components, partnered on the controller simulation in Simulink with Atharv, helped Jash in debugging the pose estimation algorithm. He is also communicating with the supplier for support with the VESC we purchased to mitigate the hardware issue we identified (flagged in ILR01).

Plans

One of our goals for the next lab demonstration is to extend the teleoperation of the RC car & using ROS to do so. We want to send steering and throttle commands via ROS messages from a different device (laptop) over WLAN and collect timestamped IMU data from the car. I intend to work on the following:

1. Set up a RPi (Raspberry Pi) 4B module [5].
2. Control the steering servo and BLDC motor via RPi.
3. Dive deep into ESC control to drive the BLDC motor at even lower RPMs.
4. Select an appropriate version of ROS/ROS2 for our project and install it.
5. Set up the Archer AX10000 Wi-Fi router [6].
6. Develop a ROS node subscribe to a topic receiving commands and control the RC car servo and BLDC motors.
7. Develop a ROS node to read UM7 sensor data and publish it to a topic.
8. Collaborate with Atharv for the conceptual design of PDBs (Power Distribution Board) for RC cars and infrastructure sensing units.

Appendix

Links

[1] https://github.com/PorridgePi/UM7

[2] https://github.com/mikehoyer/UM7-Arduino

[3] https://github.com/SolidGeek/VescUart

[4] https://hackaday.io/project/167706-vescuino-shield-ver-01/log/169168-arduino-vesc-spi-high-speed-communication-library

[5] Recommended Linux distribution for RPi 4B - Index of /releases/20.04 ubuntu-20.04-preinstalled-server-armhf+raspi.img.xz

[6] https://www.tp-link.com/us/support/download/archer-ax11000/#Emulators