Red Bird Racing EVRT Vehicle Control Unit (VCU)

Project Documentation

For 2025 (Gen 5), last updated: 2025-05-20

Contents

1 Introduction

This document provides an overview of the Red Bird Racing EVRT Vehicle Control Unit (VCU) for the year 2025 (Gen 5) The VCU firmware is designed to manage pedal input, CAN communication, and vehicle state transitions for our Formula Student electric race car.

2 Setup and Tuning

1. Adjust pedal input constants in Pedal.h.
2. Flash the VCU firmware. Ensure the car is jacked up and powered off during this process.
3. Clear the area around the car, especially the rear.
4. Test the minimum and maximum pedal input voltages and adjust the constants accordingly.

3 Debugging

Debugging is performed using the serial monitor. Enable specific debug messages by setting flags in Debug.h. Note that enabling debugging may introduce delays due to the slow serial communication. Refer to section *Future development* for DBC debugging.

4 System Design

TBD

Include diagrams where necessary

Data structure

Program procedure

5 Folder/File descriptions

5.1 src

The src folder contains the main firmware for the VCU.

5.1.1 main.cpp

The main file initializes the pedal, CAN communication, and state machine for the car. It handles transitions between states when starting the car. Refer to the rulebook to learn about the different stated needed.

5.2 lib

The lib folder contains all libraries (except headers, see *Include*).

In general, each library is in its own folder. In certain libraries, a library.json file in included to link the library with other files in include/lib.

5.2.1 Pedal

The Pedal library implements the logic for reading pedal input, filtering signals, handling reverse mode, and constructing CAN frames for throttle control. It ensures safe operation by checking for pedal faults and managing transitions between forward and reverse modes.

5.2.2 Queue

The Queue library provides static FIFO queue and ring buffer implementations. These are used for managing pedal input data and other time-sequenced information within the firmware.

5.2.3 Signal_Processing

The Signal_Processing library provides simple digital signal processing (DSP) functions for filtering and processing pedal input signals, helping to ensure accurate and stable readings.

5.3 Include

The include folder contains all header files for the project. These headers define interfaces, constants, macros, and pin mappings used throughout the firmware and libraries.

5.3.1 Debug.h

This header defines macros for enabling or disabling debug messages throughout the firmware. By setting specific flags, developers can control the verbosity of serial output for troubleshooting and development purposes.

5.3.2 pinMap.h

This header maps the microcontroller's physical pins to meaningful names used in the codebase. It centralizes pin assignments for easier maintenance and modification, reducing the risk of pin conflicts or errors.

5.4 platformio.ini

The platformio.ini file configures the PlatformIO environment for the project. It specifies the target board, framework, build flags, and library dependencies required to build and upload the firmware.

6 Future development

- Add more CAN channels for BMS, data logger, and other components.
- Improve the torque curve for better performance.
- DBC bugging
- Implement reverse mode for testing only

7 References

References used are located in the same folder as this document.

Rulebook: 2025 中国大学生方程式大赛规则_最终版.pdf

Outline (for those who can’t read Chinese, find the relevant section and ask a member who can read Chinese to help you): FSEC2025_Rules_Outline_Planeson.docx