Embedded

Getting Started

This page: *structure of this subsystem*

`components/driving_board` folder contains 4 libraries:

- firmware
- motor
- parser
- simulink

Firmware contains the generated CubeMX code. Don't edit this after generating, it will be rewritten after you generate again.

> NOTE: do make sure that after generating your code in CubeMX, you run the post generation script. You can set a post generation script in CubeMX itself. However, if you use Windows run: scripts/post_code_generation.bash, if you use Mac run: scripts/post_code_generation_mac.bash

The **Simulink** folder contains code generated by the control team, and it shouldn't be modified by embedded. This code is not used on the embedded side, but it provides a clear reference for the type of input we give to control algorithm and the output we receive from it.

In `control.h`, struct `ExtY` gives the external outputs and `ExtU` is the struct for external inputs.

`src/driving_board` contains `main.c`

This is the code that runs when we build and update to the board by pio. Main includes multiple threads for some tasks that need to run concurrently.

`ERC-Protobufs/components/driving_board` contains protobuffers for this subsystem

Protobuffers are used to send information to software and debugging board and receiving information from software.

Motor Folder

`components/driving_board/motor`

All of the motor functions in this folder are called in Main

BLDC Motors:

They are used to control the speed of the rover.

`bldc.c` contains:

The set_bldc_pwm() method. This methods scales the control signal value calculated from control code and adjust the pwm **duty cycle** outputted by the STM32 accordingly.

Stepper Motors:

`stepper.c` contains:

The set_stepper_pwm() method. This methods scales the control signal value calculated from control code and adjust the pwm **frequency** outputted by the STM32 accordingly.

Encoders:

`read_encoder.c` contains:

The read encoders() method. This method reads encoder values of **encoder angle, voltage generated** and **rpm.**

Parser Folder

`components/driving_board/parser`

This page: structure of message encoding in parser.c

The encoding logic includes:

copy_motor_to_pb():

Copies each motor from internal MotorDiagnostic struct into protobuf MotorInformation.

DBMDiagnosticsEncode():

Encodes full diagnostics data into a protobuf message.

This function: Initializes the protobuf message Copies board state Copies all 10 motors into the protobuf structure Calls pb_message_encode to serialize data

The diagnostics message contains 10 motors:

front_left, middle_left, back_left, front_right, middle_right, back_right, steering_front_left, steering_back_left, steering_front_right, steering_back_right,

Each motor is copied using a loop into the protobuf message.

message_add_envelope():

This method wraps a protobuf message into a PBEnvelope using the same pb_message_encode function.

It: Takes a populated protobuf message (e.g. DrivingBoardDiagnostics) Initializes a PBEnvelope Sets the correct oneof field (e.g. drive_diag) Calls pb_message_encode on the envelope struct Returns encoded envelope buffer and length

Main

src/driving_board/main.c

contains the main firmware entry point and runtime logic for the driving board. The system is built on FreeRTOS (CMSIS-RTOS v2) and runs multiple tasks concurrently.

Initialization

init_board()

This function initializes the full system before starting the scheduler.

It:

- Configures MPU and cache
- Initializes HAL and system clock
- Initializes GPIO, timers, and UART logging
- Initializes control algorithm (control_initialize)
- Initializes Ethernet and MAC filtering Creates FreeRTOS tasks Starts PWM and encoder peripherals
- Starts the RTOS kernel. After this, the scheduler takes over.

Tasks

The system runs 3 main threads:

MainTask

This task handles: Ethernet communication Diagnostics message creation and sending Periodic data updates

Example Flow For Sending A Message:

- Initialize UDP Fill DiagnosticsData using FillDiagnostics
- Encode using DBMDiagnostics
- Encode Send encoded data over Ethernet
- Free allocated memory after sending Also sends test UDP and raw Ethernet messages.

PwmTask

This task runs the control loop.

It:

- Calls control_step() from control module(Simulink folder)
- Reads outputs from rtY
- Updates motor PWM using set_bldc_pwm Runs periodically (1 ms)

DrivingEncoderTask

This task processes encoder data.

It:

- Reads timer counters
- Calculates: revolutions radians (position) rpm (speed)
- Populates motor data (rpm, voltage, angle).
- Updates global variables used in diagnostics
- Runs every 100 ms