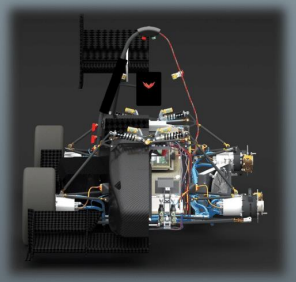
McMaster University

McMaster Formula Electric

Vehicle Controls System



Team 28: Controls Freaks

2023/03/25

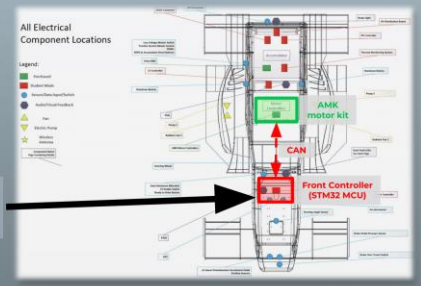
MECHTRON 4TB6: Mechatronics Capstone

User and Context

McMaster Formula Electric

→ Vehicle's mechanical & electrical systems designed, fabrication in process

★ *Car needs **embedded software** to control the motor kit*



Objective

→ Within FSAE rules, determine motor commands based on driver input sensors, BMS CAN feedback, and motor kit CAN feedback - send commands to the motor over CAN

Goals

- ❑ Modularity
- ❑ Maintainability
- ❑ Support for SW simulation
- ❑ Hardware compatibility

Team


Dharak Verma
Comp Eng IV


Abhishek Magdum
Mechatronics IV

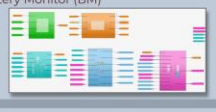

Jason Surendran
Mechatronics Mgmt. V


Derek Paylor
Mechatronics Mgmt. V

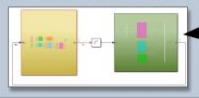

Laura Yang
Mechatronics Mgmt. V

Model-based Development (MATLAB/Simulink)

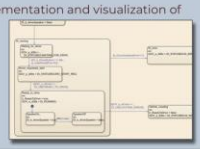
Modularity
Each subsystem responsible for handling a specific vehicle function:
- → Driver Interface (DI)
- → Vehicle Dynamics (VD)
- → Motor Interface (MI)
- → Governor (Gov)
- → Battery Monitor (BM)



Hardware hiding
Eg. signal scaling, data type conversion, CAN interfacing occurs independently of the control system

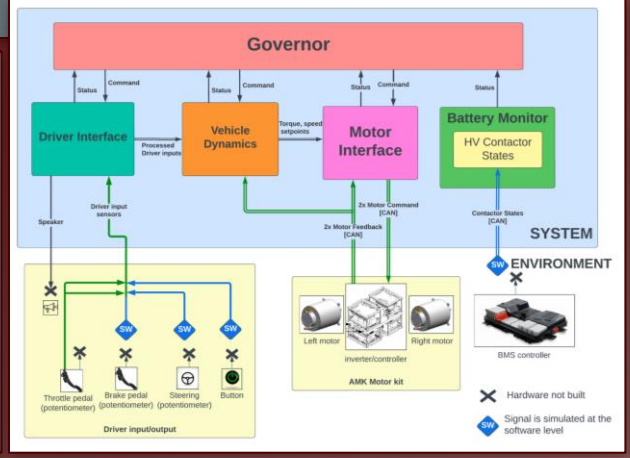

Handled separately by the Plant model

Stateflow
Unmatched implementation and visualization of state machines
- → Used in MI, DI, BM and Gov



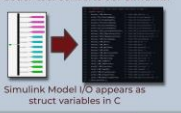
Built-in tools, add-ins
Aid in testing and code deployment to hardware

Embedded Coder | Data Inspector | Simulink Coder Support Package for STMicroelectronics Nucleo Boards version 22.2)



Deploying to Hardware

1. Code Generation - *Embedded Coder* tool converts our Simulink model to a state machine in C


Simulink Model I/O appears as struct variables in C

2. Code Integration

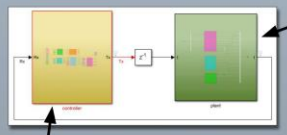
a) Map STM I/O in CubeIDE (CAN, ADC)

b) Assign STM I/O to Simulink I/O

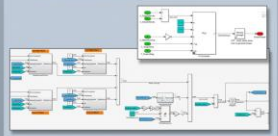
3. Code Runtime - step the state machine periodically via timer interrupt

Model Testing Environment

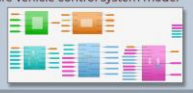
System-level testing



Plant model
Simulates vehicle's dynamic response to controller commands
- → Driver input signals are custom defined
- → Motor and vehicle response (torque, acceleration, speed) simulated using a physics-based model



Controller
Entire vehicle control system model



Model Testing Environment

Workflow

1. Build dynamic signal inputs using *SignalEditor* tool
- → Trace test scenarios to SRS
- → Store test scenario data to allow quick & consistent testing across new SW versions

2. Import data, run simulation
- → Load test case data from .mat file (system test); use *root Inport mapper* (unit test)

3. Analyze logged output data using *Simulink Data Inspector*
- → Oscilloscope-like viewing of timeseries signals

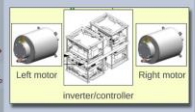
Hardware Demonstration


Throttle pedal (potentiometer signal)

ADC via STM board + UART


Control System running on Simulink in real time

CAN bus via Kvaser dongle



Motors commanded to spin!