MAD76 Academy: H. MATLAB/Simulink

Frank Tränkle*
Hochschule Heilbronn, Germany

February 25, 2026

*frank.traenkle@hs-heilbronn.de

Contents

1 Agenda

- MATLAB/Simulink must have been installed and configured on your Linux PC (see Installation Guide)
- Model-in-the-Loop Simulation for Controller Design (see Section 1.1)
- Interfaces of vehicle dynamics and motion control (see Section 1.2)
- Software-in-the-Loop Simulation for Controller Design (see Section 1.3)
- Real MAD76 Control (see Section 1.4)

Teaching Objectives

- Understand the operation and the interfaces of the Simulink models
- Learn how to control MAD76 with Simulink
- Learn how to use Simulink for adjusting the RC controls of MAD76 (see B. Adjust MAD76)

1.1 Model-in-the-Loop Simulation

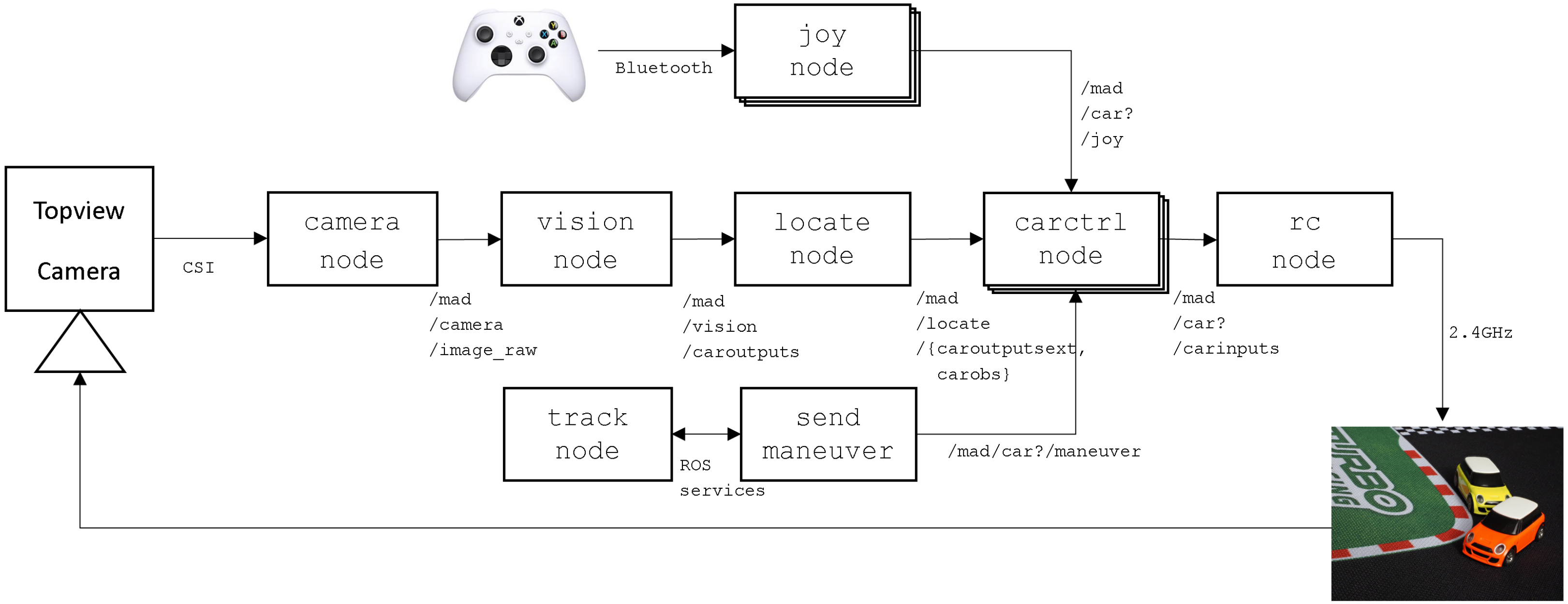

Figure 1: ROS2 nodes of MAD76 Driving Stack

- The Simulink model `s06_sig_template.slx` is the template model for vehicle dynamics modeling and controller design.

- Its purpose is to simulate the complete MAD76 including cars, camera, and control functions (software of driving stack).
- It is a template model with empty subsystems for vehicle dynamics and motion control functions modeling.
- It contains the subsystems modeling and simulation the ROS2 nodes of MAD76 in Figure 1 plus the vehicle dynamics:

ROS2 Node	Description	Simulink-Subsystem
`camera_node`	Rasberry Pi camera driver	`Vision`
`visionnode`	computer vision	`Vision`
`locatenode`	multi-object tracking	`Locate`
`carctrlnode`	motion planning and control for each individual car	`Car0/Motion Control`
`rcnode`	remote control signals output to $2.4\mathrm{GHz}$ channel via SPI	not modeled
`tracknode`	track map	`s06_data.m`, `t71_mad76_small.m`
`joy_node`	optional node for manual control via joystick	not modeled
`send_maneuver`	maneuver messages (reference signals) for motion control	`Maneuver0`
real car	vehicle dynamics	`Car0/Vehicle Dynamics`

- Above table is for car with id 0 (orange/red car).
- The subsystems of car id 0 may be duplicated to simulate multiple cars.
- The vehicle dynamics of car id 0 are modeled by the subsystem `Car0/Vehicle Dynamics`.
- The motion control functions (driving stack software) for car id 0 are modeled by the subsystem `Car0/Motion Control`.

- As soon as you have finished testing in model-in-the-loop (MiL), you can reuse `Car0/Motion Control` for software-in-the-loop (SiL) testing and real MAD76 control in Sections 1.3 and 1.4.
- `rcnode` for remote control is not modeled. The actuation `Car0/Vehicle Dynamics` by `Car0/Motion Control` is assumed to be ideal. Input delays will be modeled as part of texttttCar0/Vehicle Dynamics.
- Test your MATLAB/Simulink installation by simulating this model.
  1. MATLAB on the Linux PC can only be started in the terminal. Enter
     ```
     cd ~/src/mad76/matlab/vertical
     matlab2025a
     ```
  2. Enter on the MATLAB prompt:
     ```
     s06_sig_template
     ```
  3. The model should run without errors and display initial positions of the vehicles in the MATLAB figure. But the car in the MATLAB figure is not moving, yet, because subsystems `Car0/Vehicle Dynamics` and `Car0/Motion Control` are empty.
- Make sure that `/src/mad76/matlab/vertical` always is the current directory in MATLAB.

- Never add this directory to the MATLAB search path. Otherwise, MATLAB slows down and there will be conflicts in opening Simulink models and MATLAB scripts.

1.2 Interfaces

- ROS2 messages in Figure 1 are modeled as Simulink bus signals in `s06_sig_template.slx`
- The main ROS2 messages / bus signals for motion control and vehicle dynamics are:
  - `mbmadmsgs::msg::CarInputs` for control signals

ROS2 Topic	`/mad/car?/carinputs`
ROS2 Message Type	`mbmadmsgs::msg::CarInputs`
Simulink Bus Signal Type	`CARINPUTS`
`Car0/Vehicle Dynamics`	`carinputs` inport
`Car0/Motion Control`	`carinputs` outport

The main elements are:

Element	Symbol	Datatype	Unit	Description
`pedals`	u_n	`single`	$\in [-0.2, 0.2]$	motor signal for thrust and braking
`steering`	δ_n	`single`	$\in [-0.93, 0.93]$	steering signal

– `mbmadmsgs::msg::CarOutputs` for measurement signals

ROS2 Topic	`/mad/locate/caroutputs`
ROS2 Message Type	`mbmadmsgs::msg::CarOutputsList`
Simulink Bus Signal Type	`CAROUTPUTS`
`Car0/Vehicle Dynamics`	`caroutputs` outport
`Car0/Motion Control`	-

The main elements are:

Element	Symbol	Datatype	Unit	Description
`s`	$[s_1, s_2]^T$	`single`	m	rear axle center position in fixed frame
`psi`	ψ	`single`	rad	yaw angle

– `mbmadmsgs::msg::CarObsList` for car observations / process variables (position, speed, yaw angle, etc.)

ROS2 Topic	`/mad/locate/carobs`
ROS2 Message Type	`mbmadmsgs::msg::CarObsList`
Simulink Bus Signal Type	`CAROBS` array for list of all cars
`Car0/Vehicle Dynamics`	-
`Car0/Motion Control`	`carobs` inport

The main elements are:

Element	Symbol	Datatype	Unit	Description
`v`	$y = v$	`single`	$\mathrm{m/s}$	rear axle center speed
`s`	$[s_1, s_2]^T$	`single`	m	rear axle center position in fixed frame
`psi`	ψ	`single`	rad	yaw angle

– `mbmadmsgs::msg::DriveManeuver` for maneuvers and reference signals

ROS2 Topic	`/mad/car?/maneuver`
ROS2 Message Type	`mbmadmsgs::msg::DriveManeuver`
Simulink Bus Signal Type	`DRIVEMANEUVER`
`Car0/Vehicle Dynamics`	-
`Car0/Motion Control`	`maneuver` inport

The main elements are:

Element	Symbol	Datatype	Unit	Description
`vmax`	$w = v^*$	`single`	$\mathrm{m/s}$	setpoint (reference) for speed control

1.3 Software-in-the-Loop Simulation

- Simulink model `c71_car0_template.slx` is the template model for code generation
- Simulink simulates `c71_car0_template.slx` as a ROS2 node replacing the ROS2 node `carctrlnode` in Figure 2
- All other ROS2 nodes of Figure 2 run as usual
- Start this ROS2 environment in manual simulation mode (without MAD76 driving stack in `carctrlnode`)

```
ros2 launch mbmad madpisimman.launch
```

- Start the simulation of `c71_car0_template.slx`. Enter on the MATLAB prompt:

```
cd ~/src/mad76/matlab/vertical
c71_car0_template
```

- Send a maneuver message to the Simulink model, so that the main subsystem is enabled

```
ros2 run mbmadcar send_maneuver.py 0 0.2 0.5
```

- You can now manually control the orange car with id 0 by manipulating the sliders in subsystem `Motion Control` and `carinputs msg/Motion Control/Sliders` or `Joystick/Sliders`

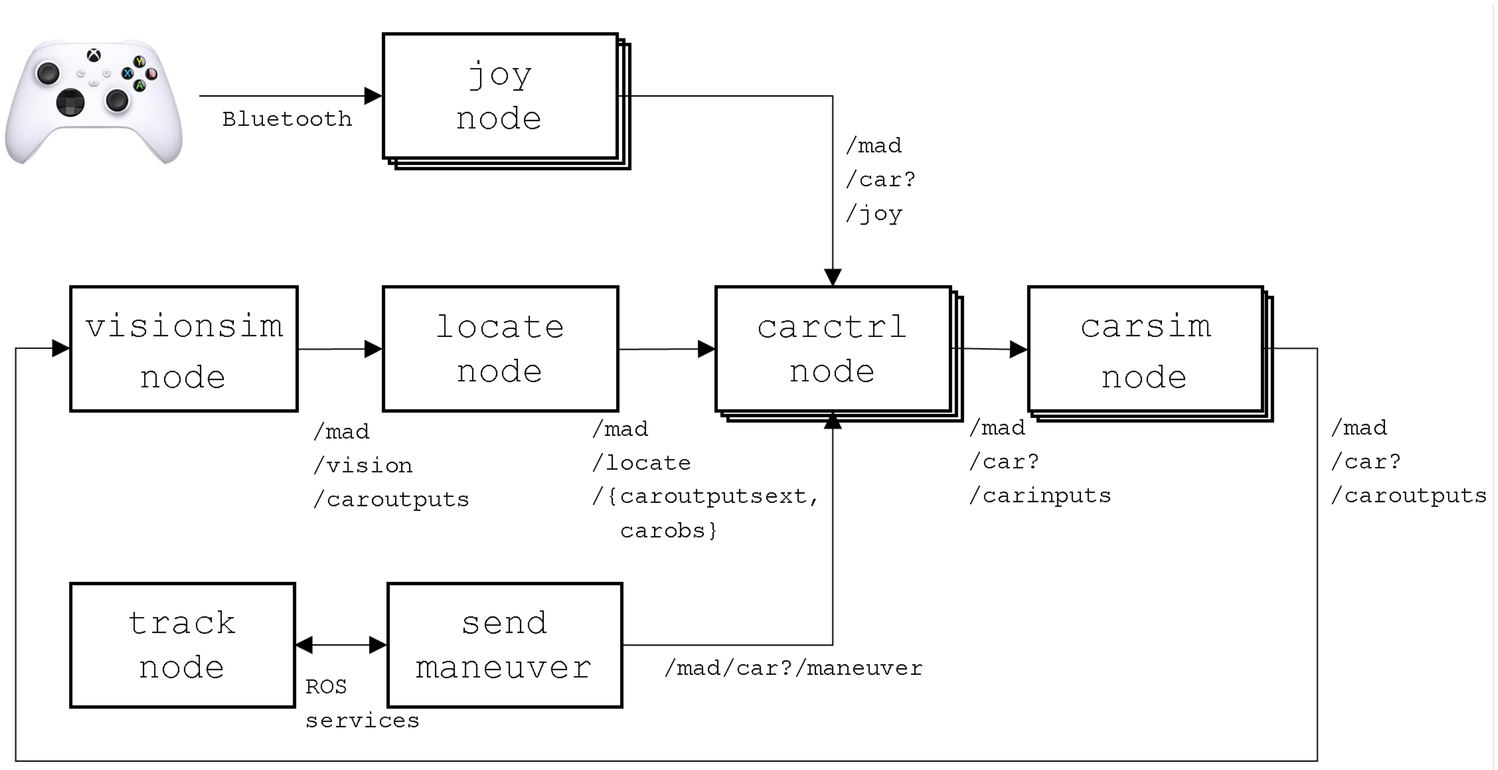

Figure 2: ROS2 nodes in SiL simulation mode

1.4 Real MAD76 Control

- The Simulink model `c71_car0_template.slx` is now simulated on the MAD76 Linux PC to control the real MAD76 cars
- All ROS2 nodes in Figure 1 run on the Raspberry Pi except for `carctrlnode`
- Simulink simulates `c71_car0_template.slx` as a ROS2 node replacing `carctrlnode` on the Linux PC
- ROS2 runs in a distributed environment consisting of Raspberry Pi and Linux PC
- ROS2 establishes network communication on Ethernet between Raspberry Pi and Linux PC
- Start the ROS2 environment on the Raspberry Pi without `carctrlnode`. Enter on the Raspberry Pi terminal:

```
ros2 launch mbmad madpiman.launch
```

- Start the simulation of `c71_car0_template.slx` on the MAD76 Linux PC. Enter on the MATLAB prompt:

```
cd ~/src/mad76/matlab/vertical
c71_car0_template
```

  and start the simulation by hitting the `Run` button in toolbar `Simulation`
- Send a maneuver message to the Simulink model, so that the main subsystem is enabled. Enter on the Raspberry Pi:

```
ros2 run mbmadcar send_maneuver.py 0 0.2 0.5
```

- First argument is the car identifier

0	orange/red car
1	yellow/white car
2	blue car
3	green car

- Second argument is the car reference speed $w = v^*$ in $\frac{\mathrm{m}}{\mathrm{s}}$

- Third argument is the lateral reference position

0	right curb
0.25	right lane
0.5	center line
0.75	left lane
1	left curb
-1	ideal line for low laptimes

- You can now manually control the orange/red car with id 0 by manipulating the sliders in subsystem `Motion Control and carinputs msg/Motion Control/Sliders or Joystick/Sliders`
- You can measure the vehicle states (elements of bussignal `carobs/carobs(1)`) including position, speed, and yaw angle in the Simulink Data Inspector
- Stopping the simulation triggers an emergency stop of the car
- You may now model control functions as part of subsystem `Motion Control and carinputs msg/Motion Control/Sliders or Joystick/Sliders` by replacing the sliders with your control algorithms
- You can use the sliders to adjust the RC controls according to B. Adjust MAD76
- Once finished with MiL testing, you can copy the motion control subsystem

  `Car0/Motion Control` from `s06_sig_template.slx` to the subsystem `Motion Control and carinputs msg/Motion Control` in `c71_car0_template.slx` for controlling the cars on the real MAD76 system