MAD76 Academy: B. Adjust MAD76

Frank Tränkle*
Hochschule Heilbronn, Germany

February 25, 2026

*frank.traenkle@hs-heilbronn.de

Contents

1 Agenda

- Adjust the Raspberry Pi camera (see Section 2)
- Adjust the radio controllers (RC) (see Section 3)

2 Camera

This section explains

- how to mount the camera (see Section 2.1),
- how adjust focus and aperture (see Section 2.2),

2.1 Mounting Camera

- The camera must be mounted above the MAD76 track for bird's eye view.
- The camera lens should be at a height of approx. 106cm above the track.
- The ideal position is above the center of the track.
- If the camera is mounted on a tripod next to the track, a tilt in one direction is necessary, which should be as small as possible.
- In all other directions, the camera should be aligned in parallel to the track and not be tilted.

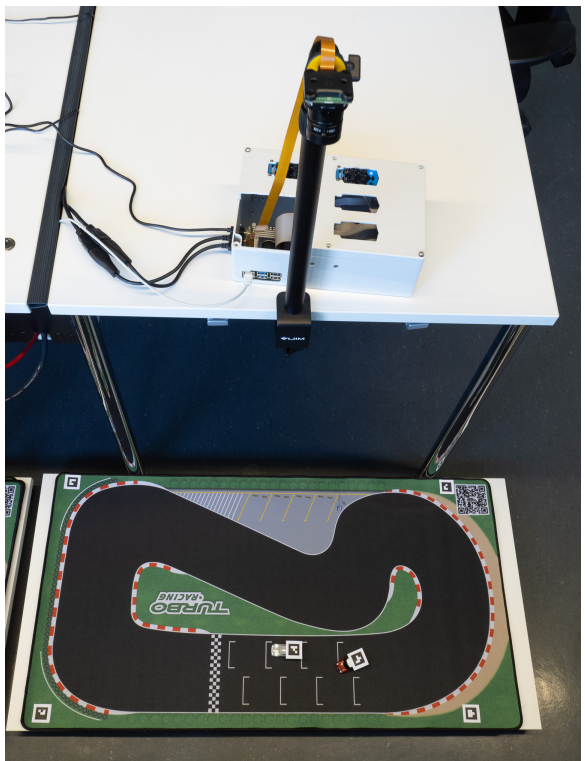

Figure 1: Mounting camera above the MAD76 track.

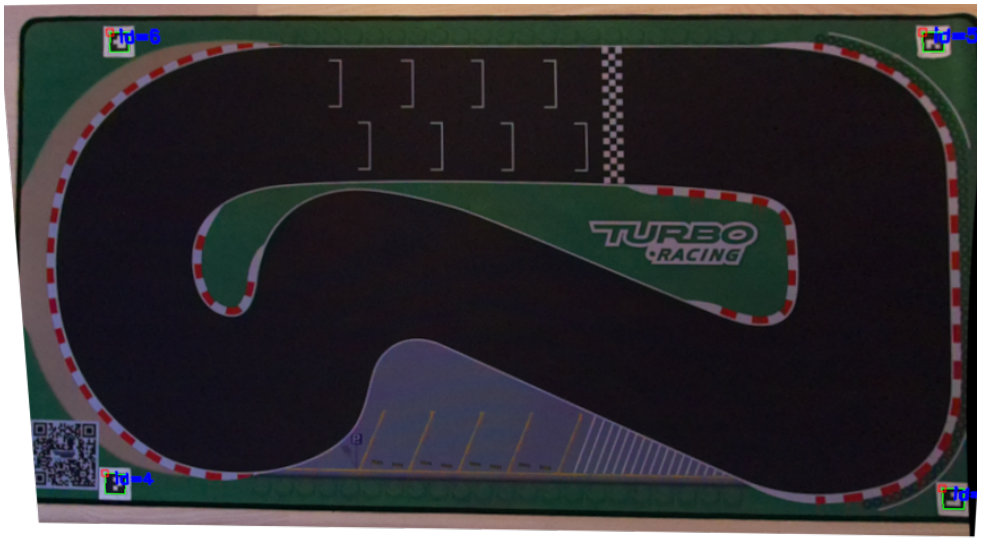

Figure 2: Camera image.

- Place the track as close as possible to the tripod, such that
  - the camera can capture the entire track without any obstruction,
  - the whole track area is as close as possible to the camera for higher pixel resolution,
  - and is in the focal plane of the camera

- For adjusting the position and tilt of the camera, run the following ROS command

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

  This opens a window with the camera image as depicted in Figure 2

- All 4 frame markers and track borders shall be visible.
- The lower and upper track borders shall be in parallel to the image borders.
- The lower track border shall be as close as possible to the lower image border if the camera is titled.

2.2 Focus and Aperture

- In MAD76, the camera acquires images with a frame rate of 40 frames per second (fps) (or a sampling time of $25\mathrm{ms}$)
- The MAD76 SW sets the exposure time to a fixed value of $2\mathrm{ms} = 1/500\mathrm{s}$ to avoid motion blur.
- The focus of the lens must be adjusted to ensure that the entire track area is in sharp focus.
  1. Fasten the screw for aperture adjustment (`OPEN<>CLOSE`).
  2. Loosen the screw for focus adjustment (`NEAR<>FAR`).
  3. Adjust the focus by turning the focus ring until the entiretrack area is in sharp focus.
     - You may zoom in by hitting the menu button + of the camera image window.
  4. Fasten both screws.

- The aperture must be adjusted to the small value (large f-stop number) to achieve a large depth of field.
  1. Fasten the screw for focus adjustment.
  2. Loosen the screw for aperture adjustment.
  3. Adjust the aperture by turning the aperture ring, such that all 4 frame markers are reliably detected.
     - The detection is successful if all 4 frame markers are highlighted in the camera image by green bounding boxes.
     - A darker image is better than a brighter image for reliable detection and greater depth of field.
  4. Fasten both screws.

3 RC Calibration

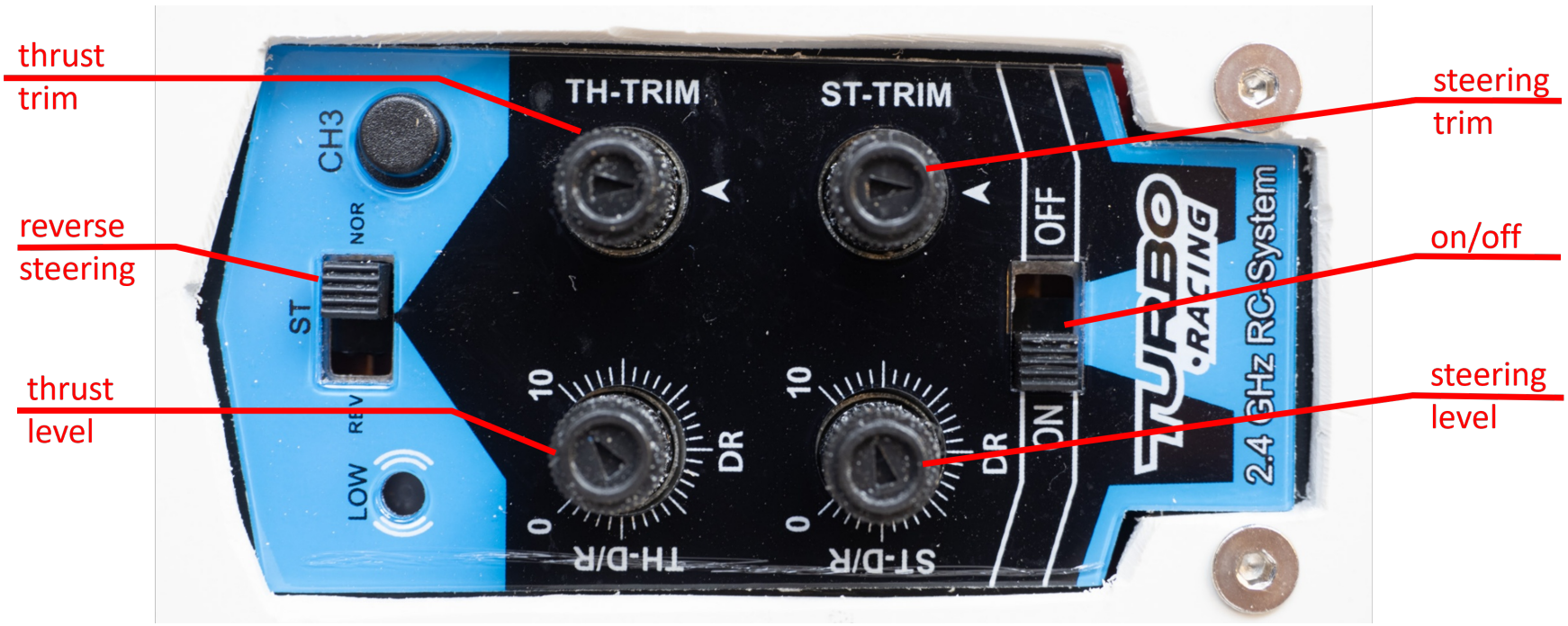

Figure 3: RC Buttons

3.1 Rationale

- The cars have tolerances in steering, motor propulsion and friction
- The RC microcontrollers (µCs) have tolerances in the 3.3V power supply for the potentiometers
- Due to these tolerances, each RC-to-car-coupling must be calibrated individually
- For calibration, **the car must be switched on before switching on the RC**
- By calibration, all cars will show similar dynamic behavior, such that all cars can be controlled by the same MAD76 driving stack with good performance

3.2 Before you start

1. If MAD76 is running, switch off the RC by stopping ROS by hitting `Ctrl-C` in the terminal
2. Move the RC button `on/off` to the `on` position, since MAD76 IO will do the powering automatically by the L293B
3. Move the button `reverse steering` to the `NOR` position, which typically is for right-handed drivers
4. Adjust the knobs `thrust level` and `steering level` all the way to the limit `10` in counter-clockwise direction, so that the car will show maximal performance
5. Switch on the car
6. Switch on the RC by starting MAD76 in manual mode with the following ROS command in a terminal:

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

   This command starts the I/O of MAD76 but not the driving stack. Omit this step if you want to use the Python program `rctest.py` from MAD76 Academy MAD76 I/O Programming
7. If the car has not been coupled to the RC, yet (the car's headlights are flashing), the coupling must be done now
   (a) Switch off all other RCs and cars
   (b) Initiate the coupling process by placing a needle into the couple hole on the bottom of the car
   (c) Wait for the car's headlights to stop flashing, indicating a successful coupling

3.3 Calibrate the motor

1. Calibrate standing still

   (a) Adjust RC knobs `thrust trim` and `steering trim` to the neutral center positions, as depicted in Figure 3

   (b) Set the normalized motor signal to zero: $u_n = 0$ by running the ROS command, which sends `CarInputs` messages to the topic `/mad/car0/carinputs` with a rate of 10Hz:

   ```
   ros2 topic pub -r 10 /mad/car0/carinputs mbmadmsgs/msg/CarInputs "{carid: 0, pedals: 0.0, steering: 0.0}"
   ```

   or by running the Python program `rctest.py` from MAD76 Academy MAD76 I/O Programming

   ```
   python rctest.py 0 0.0 0.0
   ```

   The command line arguments are

argument	description
`carid`	ID of the RC and car (0, 1, 2 or 3)
`pedals`	normalized motor signal $u_n \in [-1, 1]$
`steering`	normalized steering signal $\delta_n \in [-1, 1]$

   Or you may run MATLAB/Simulink model `c71_template_car0.slx` from MAD76 Academy H. MATLAB/Simulink and use the sliders in this model.

   (c) Adjust RC knob `thrust trim` until the car stops and is in standing still

2. Calibrate break-off-torques to overcome friction to identical values for forward and reverse motion

   (a) Increase u_n in small steps from 0 to 0.06

```
ros2 topic pub -r 10 /mad/car0/carinputs mbmadmsgs/msg/CarInputs "{carid: 0, pedals: 0.01, steering: 0.0}"
ros2 topic pub -r 10 /mad/car0/carinputs mbmadmsgs/msg/CarInputs "{carid: 0, pedals: 0.02, steering: 0.0}"

...
```

   or by running the Python program `rctest.py` or by running MATLAB/Simulink model `c71_template_car0.slx` from H. MATLAB/Simulink. The car should start moving forward at $u_n \approx 0.04$. If not, adjust the RC knob `thrust trim`.

   (b) Decrease u_n in small steps from 0 to -0.06

```
ros2 topic pub -r 10 /mad/car0/carinputs mbmadmsgs/msg/CarInputs "{carid: 0, pedals: -0.01, steering: 0.0}"
ros2 topic pub -r 10 /mad/car0/carinputs mbmadmsgs/msg/CarInputs "{carid: 0, pedals: -0.02, steering: 0.0}"
...
```

   The car should start moving backward at $u_n \approx -0.04$. If not, adjust the RC knob `thrust trim`.

   (c) Repeat this calibration procedure until the car starts moving forward and backward at the same absolute level, e.g., $|u_n| \approx 0.04$

3.4 Calibrate the steering

1. Set
   - the normalized steering signal to zero for straight driving: $\delta_n = 0$
   - the normalized motor signal for little thrust: $u_n \approx 0.06$

   by running the ROS command

```
ros2 topic pub -r 10 /mad/car0/carinputs mbmadmsgs/msg/CarInputs "{carid: 0, pedals: 0.06, steering: 0.0}"
```

   or by running the Python program `rctest.py` from MAD76 I/O Programming

```
python rctest.py 0 0.06 0.0
```

   or by running MATLAB/Simulink model `c71_template_car0.slx` from H. MATLAB/Simulink.
2. The car should move on a straight line. If not, adjust RC button `steering trim`
3. Check if the steering is operating with no faults in the total range $\delta_n = [-1, 1]$ by running `ros2 topic pub` or `rctest.py` with varying steering inputs