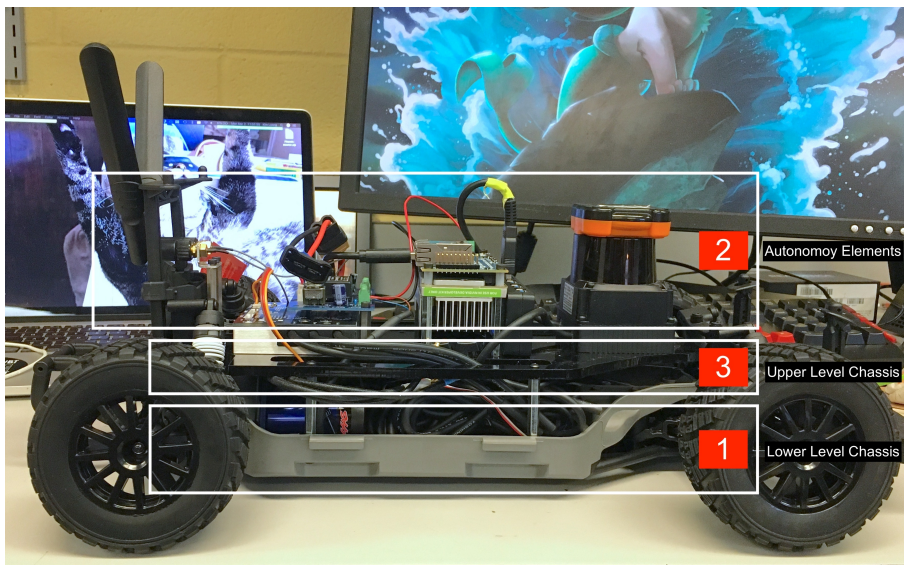


Docs » Building the Car

Building the Car

At the end of this build section, you will have a fully functioning F1TENTH Autonomous Vehicle.

There are three main sections to the car.



[1] will be referred to as the **Lower Level Chassis**

[2] will be referred to as the **Autonomy Elements**

[3] will be referred to as the **Upper Level Chassis**

1. First, we start off by setting up the [Lower Level Chassis](#) , which serves as the foundation of the vehicle.
2. Then, we put together all of the [Autonomy Elements](#).
3. Next, all of the Autonomy Elements will be mounted on the [Upper Level Chassis](#).
4. Finally, the Upper Level Chassis will be [connected](#) with the Lower Level Chassis.

Danger

LIPO (LITHIUM POLYMER) BATTERY SAFETY WARNING

The F1TENTH Autonomous Vehicle uses lithium polymer batteries. LiPO batteries allow your car to run for a long time, but they are not something to play with or joke about. They store a large amount of

energy in a small space and can damage your car and cause a fire if used



- When charging batteries, always monitor them and place them in a fireproof bag on a non-flammable surface clear of any other objects.
- Do not leave a LIPO battery connected to the car when you're not using it. The battery will discharge and its voltage will drop to a level too low to charge it safely again.
- Unplug the battery from the car immediately if you notice any popping sounds, bloating of the battery, burning smell, or smoke.
- Never short the battery leads.
- Do not plug the battery in backwards. This will damage the VESC and power board (and likely the Jetson as well) and could cause a short circuit.
- See this [video](#) for an example of what might happen if you don't take care of your batteries. Be safe and don't let these happen to you!

Difficulty Level: Intermediate

Approximate Time Investment: 2 hours

[Contact Us](#)

Copyright ©2020 FITENTH Foundation
All rights reserved



Content on this site is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#)

[Docs](#) » [F1TENTH Simulator](#) » Linux/ROS Installation

Linux/ROS Installation

ROS only runs natively in Linux so we are only supporting using the simulator in Ubuntu at this time. If you do not have ROS Melodic installed, follow the instructions from <http://wiki.ros.org/melodic/Installation/Ubuntu>.

Dependencies

You will need the following dependencies:

- tf2_geometry_msgs
- ackermann_msgs
- joy
- map_server

Install them using

```
sudo apt-get install ros-melodic-tf2-geometry-msgs ros-  
melodic-ackermann-msgs ros-melodic-joy ros-melodic-map-  
server
```

The full list of dependencies can be found in the `package.xml` file.

Package

To install the simulator package, clone the simulator repository into your catkin workspace:

```
cd ~/catkin_ws/src  
git clone https://github.com/f1tenth/f1tenth_simulator.git
```

Then run catkin_make to build it:

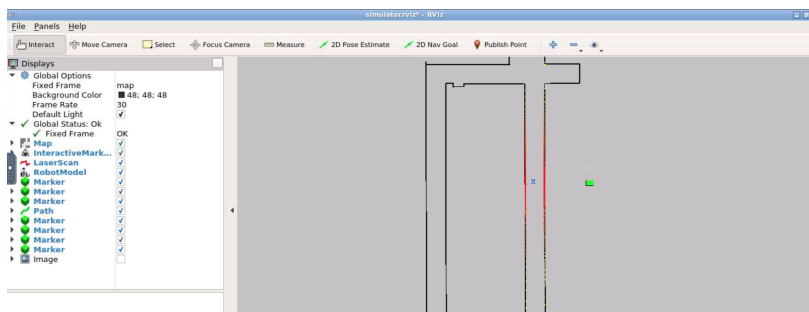
```
cd ~/catkin_ws  
catkin_make  
source devel/setup.bash
```

Quick Start

To run the simulator on its own, run:

```
roslaunch f1tenth_simulator simulator.launch
```

This will launch everything you need for a full simulation: roscore, the simulator, a preselected map, a model of the racecar, and the joystick server.



[Contact Us](#)

Copyright ©2020 FITENTH Foundation
All rights reserved



Content on this site is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#)

Docs » F1TENTH Simulator » Using the Simulator

Using the Simulator

Driving

Try controlling the racecar manually using your keyboard. Press **K** to give driving control to the keyboard, you should see the following in your terminal:

```
k[ INFO] [1566770434.254597400]: Keyboard turned on
MUX:
0
1
0
0
```

Then, drive using the standard WASD keys:

- **W** drives the car forward
- **A** steers to the left
- **S** drives it backwards
- **D** steers to the right.

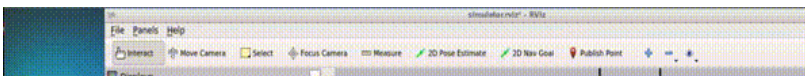
Press **spacebar** to bring the car to a stop.

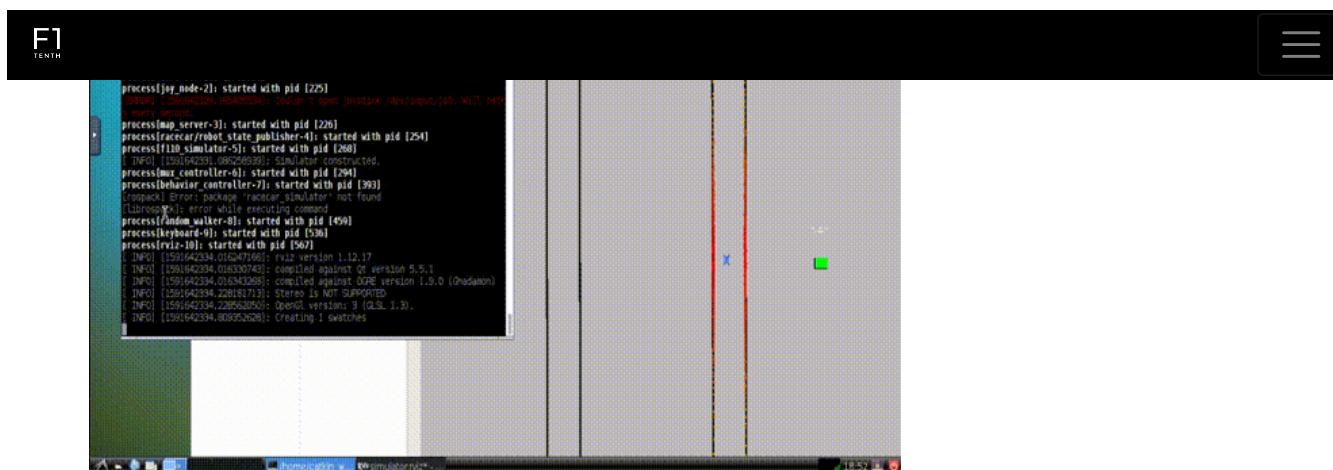
! Note

Be aware that if you crash, the keyboard will be turned off, and you'll have to press **K** again to turn it back on. Also, it can only handle one key press at a time, so holding down multiple keys at once does not work. Lastly, pressing **A** or **D** will steer to a fixed angle, and the only way to straighten out is with **spacebar**.

The controls are a bit tricky, but hopefully you won't have to do too much manual driving!

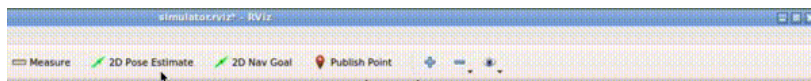
If you are using a joystick, make sure the correct axis is set in params.yaml for steering and acceleration- this changes between different joysticks.





Instant Pose Setting

A useful function of the simulator is that you can instantly move the car without driving it to its new location. To do this, click the **2D Pose Estimate** pose button at the top of the rViz window, and then click the desired location on the track to move the car there.



» [Contact Us](#)

Copyright ©2020 FITENTH Foundation
All rights reserved



Content on this site is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#)

[Docs](#) » [F1TENTH Simulator](#) » [Advanced Information](#)

Advanced Information

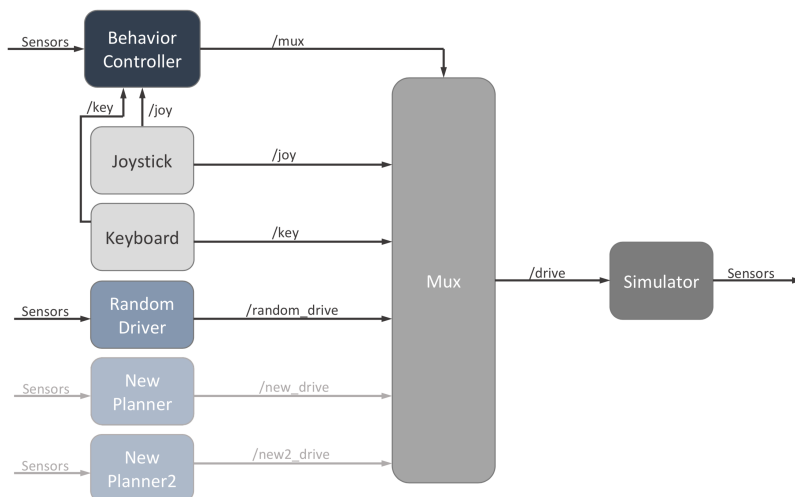
RVIZ Visualization

With the simulator running, open rviz. In the left panel at the bottom click the **Add** button, then in the **By topic** tab add the `/map` topic and the `/scan` topic. Then in the **By display type** tab add the RobotModel type. In the left panel under the newly added LaserScan section, change the size to 0.1 meters for a clearer visualization of the lidar (shown in rainbow).

You can use a keyboard or USB joystick to drive the car around, or you can place the car manually by clicking the **2D Pose Estimate button** on the top of the screen and dragging your mouse on the desired pose.

ROS API

The simulator was set up with two main objectives in mind- similitude to the real car and fast prototyping of racing algorithms. The **simulator** node was written such that it can be swapped out with the F1/10 car itself, and if all topic names remain the same, the same exact code can be run to drive the car. The rest of the ROS nodes are organized so that new planning algorithms can be added quickly and toggled between during driving.



Simplified graph of ROS nodes

for what a planning node should look like. Each planner can listen to the sensor data published by the **simulator** and then publish **AckermannDrive** messages to their own specific topic (e.g., `/random_drive`). The **mux** node listens to all of these topics, then takes the message from whichever planner is turned on and publishes it to the main `/drive` topic, which the **simulator** listens to. Note that only the velocity and steering angle specified in the message are used. The **mux** node also listens to joystick and keyboard messages too, for manual driving. The **behavior controller** node tells the **mux** node which planner is on through the `/mux` topic. By default, each planner (including keyboard and joystick) is mapped to a joystick button and keyboard key, and they are simply toggled on and off manually. Additionally, upon collision, the car will halt and all mux channels will be clear- nothing will be in control until manual intervention.

To instantly move the car to a new state publish **Pose** messages to the `/pose` topic. This can be useful for scripting the car through a series of automated tests.

The simulated lidar is published to the `/scan` topic as **LaserScan** messages.

[Contact Us](#)

Copyright ©2020 FITENTH Foundation
All rights reserved



Content on this site is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](#)