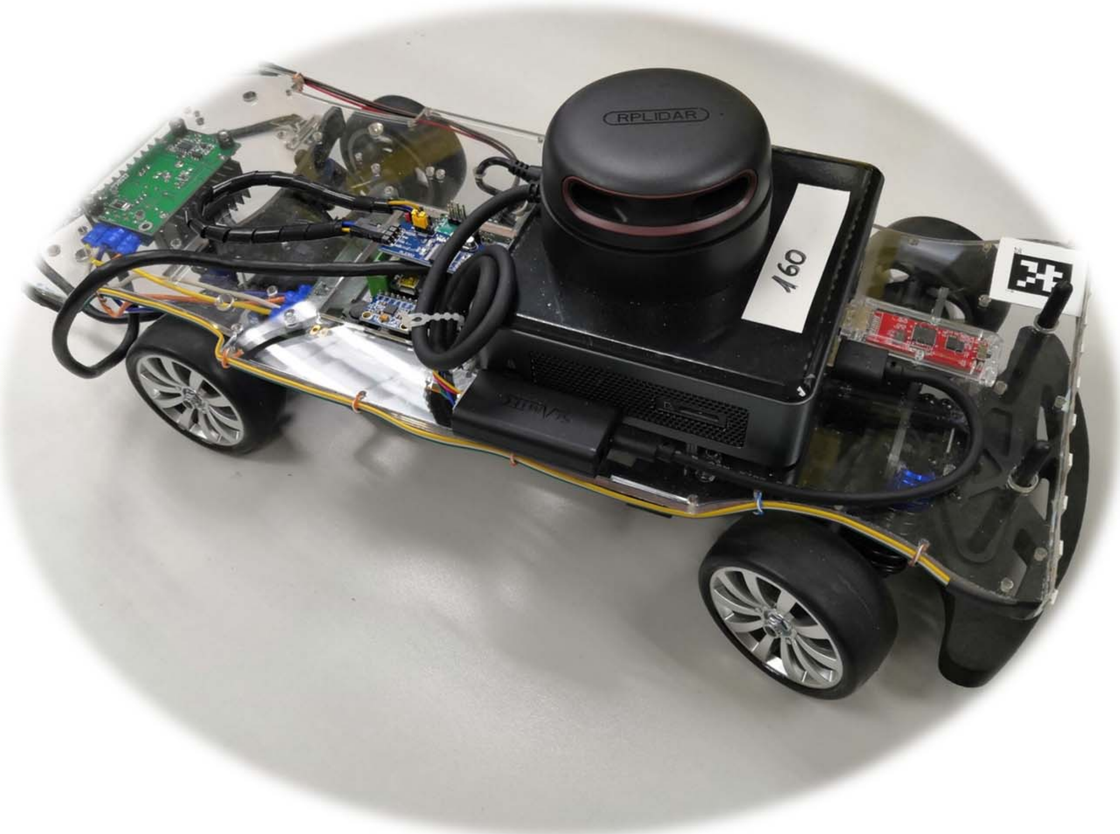
Autonomous Car Autominy

Dr. Roger Miranda Colorado



Introduction

CONACYT
Consejo Nacional de Ciencia y Tecnología

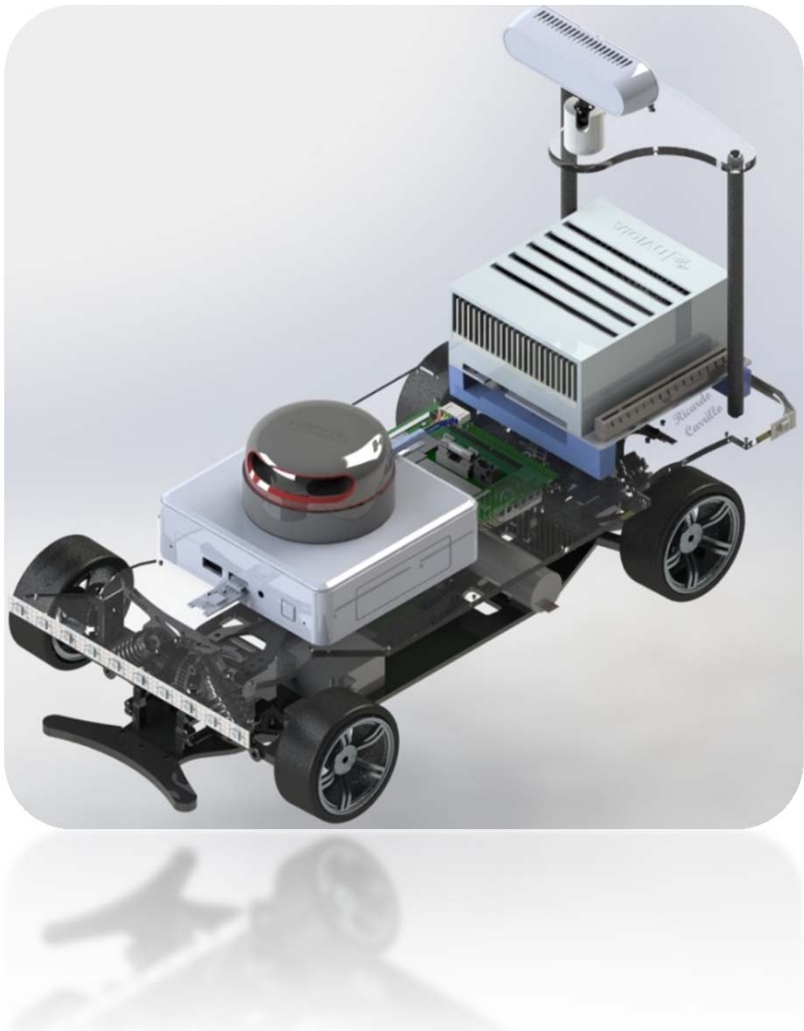
IPN
CITEDI
Centro de Investigación y Desarrollo de Tecnología Digital

Instituto Politécnico Nacional-CITEDI

Introduction

The autominy car is:

- Fully autonomous scale vehicle 1:10.
- Developed for getting started with the study and control of autonomous cars.
- Is part of the Intelligent Systems and Robotics group of Professor Dr. Raúl Rojas at Freie Universität Berlin.
- A complete guide for understanding the Autominy car can be found in: https://autominy.github.io/AutoMiny/docs/quick-start-guide/



Introduction

The prototype cars are:

- **MadeInGermany[1]**

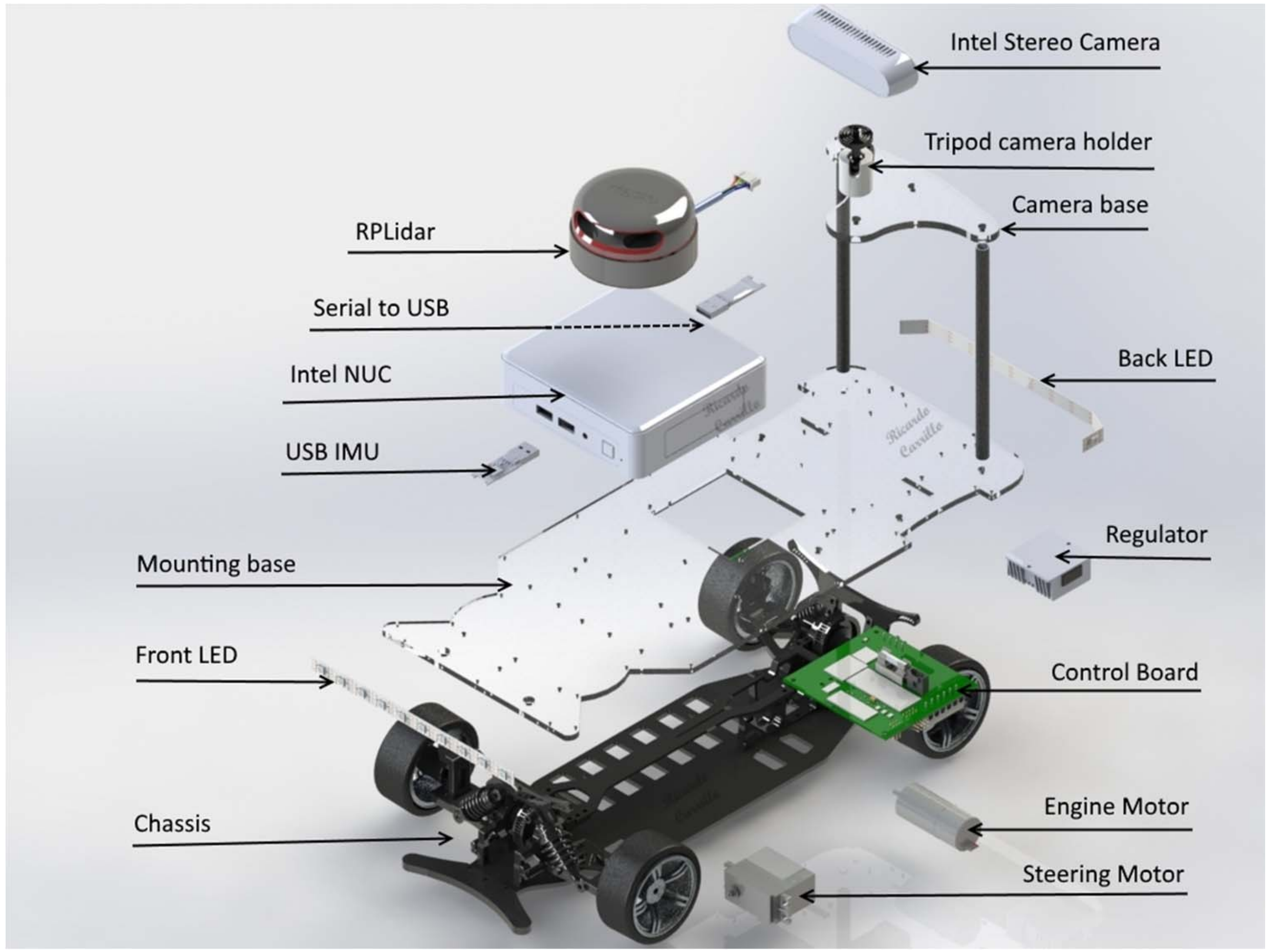
- **Electric car E-Instein[2]**

1. https://autonomos.inf.fu-berlin.de/vehicles/made-in-germany/
2. https://autonomos.inf.fu-berlin.de/vehicles/e-instein/

Introduction

For the **Autominy** car:

- The base hardware in the car processes all the algorithms on a ***Intel NUC CPU***.
- Also, we have the basic configuation to run the car autonomously.
- The camera allows the car to seek the track ahead, process images, and execute tasks such as ***localization***, **lane detection**, **obstacle detection**, etc.

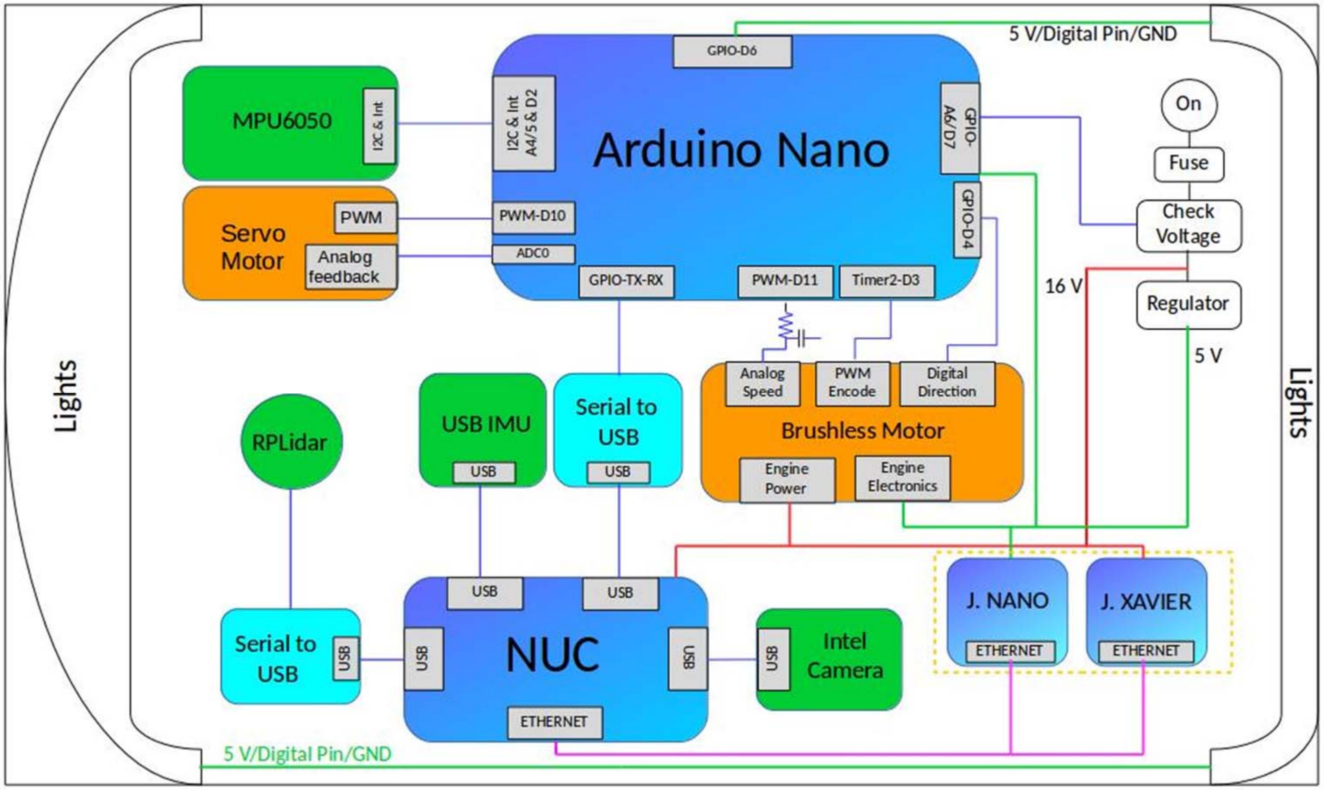


1. https://autominy.github.io/AutoMiny/docs/quick-start-guide/

Introduction

The Autominy has two main processing modules:

1. **One controller board with microprocessor**. It has the arduino controller and an additional IMU MPU6050. The main tasks are:
   - ❑ Battery voltage checker
   - ❑ Chassis sensor data acquisition
   - ❑ Voltage distribution
   - ❑ Control of chassis

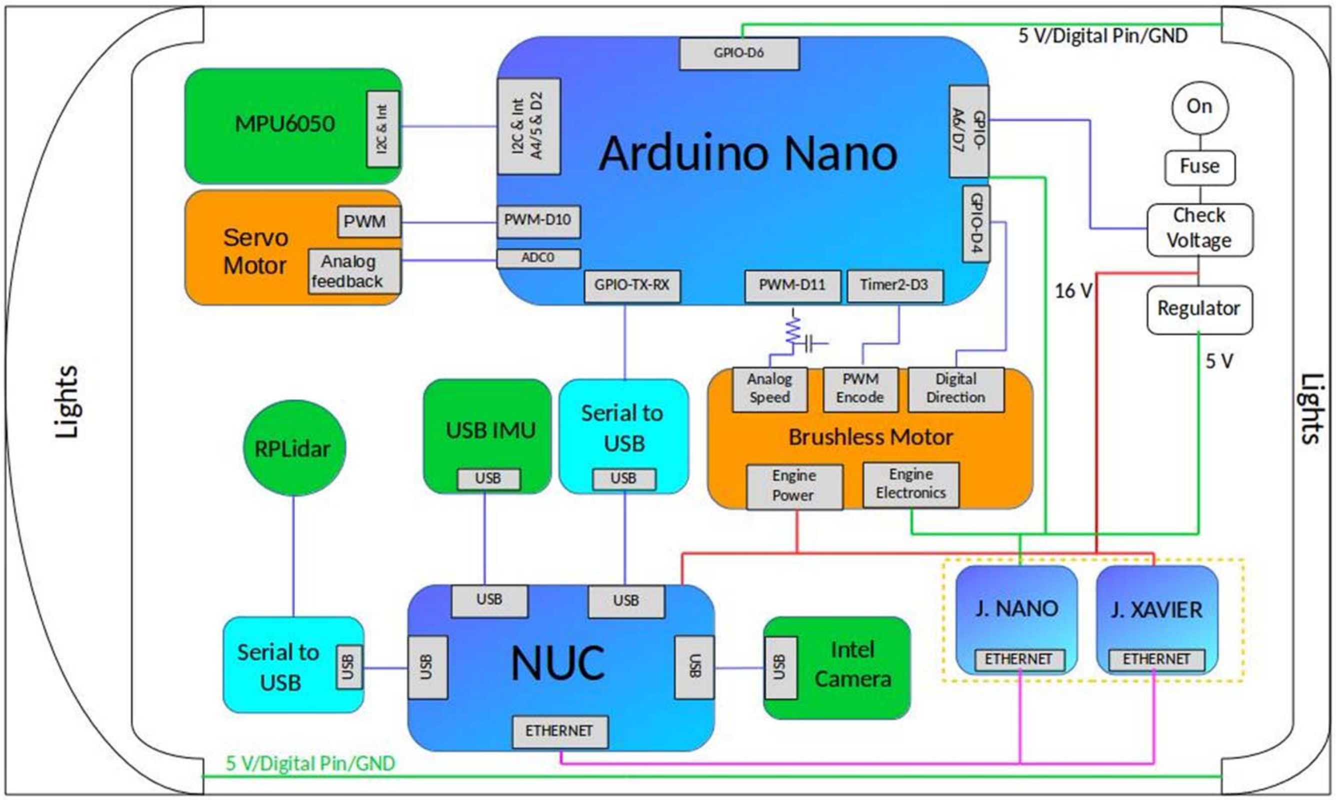


https://autominy.github.io/AutoMiny/docs/autominy-core/

Introduction

The Autominy has two main processing modules:

2. **Intel NUC Computer**. Is the Autominy's main processor and:
   - Handles the data coming from the controller board, LIDAR, Bosch USB IMU, and the stereo camera.

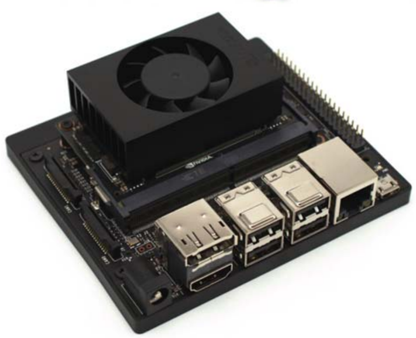


https://autominy.github.io/AutoMiny/docs/autominy-core/

Introduction

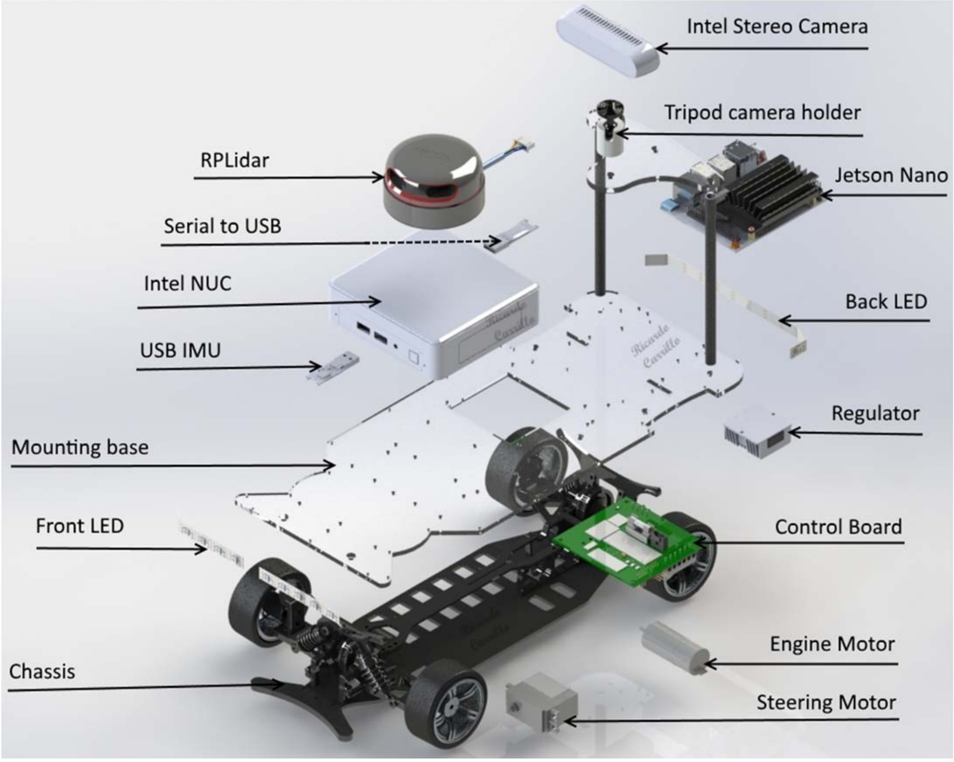
The Autominy can be upgraded with:

- NVIDIA Jetson Nano[1]
- NVIDIA Jetson Xavier[1]



These boards allow for:

- Using ANN and reinforcement learning.
- The board sends the data to the NUC through an ethernet connection.

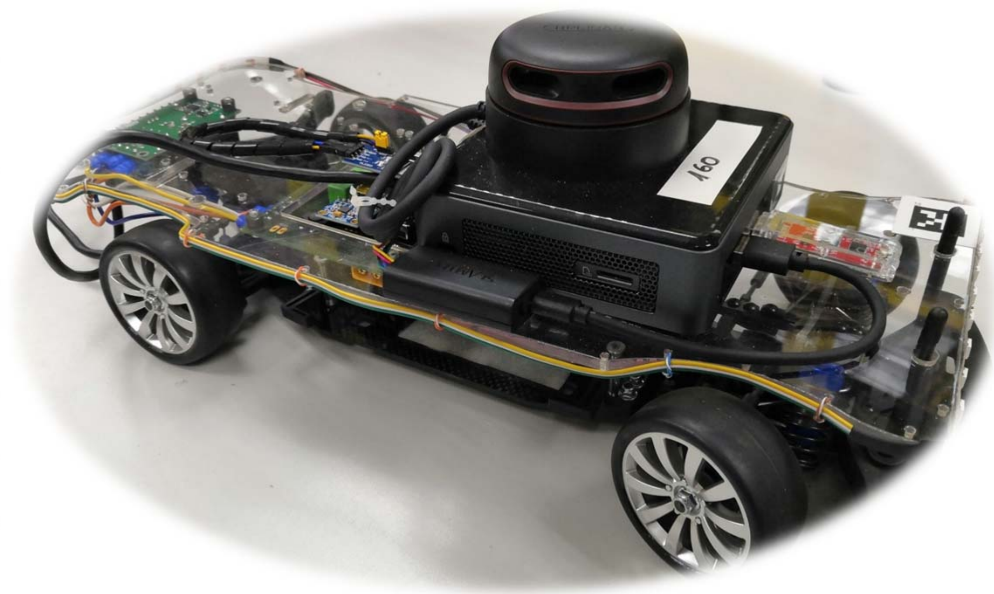


1. https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano/

Introduction

General specifications of the Autominy car with a fully charged LiPo battery are:

Name	Value
Dimensions	387.5mm x 160 mm x 270 mm
Power consumption (core, Nano, Xavier)	80 W, 85 W, 90 W
Max. power consumption (core, Nano, Xavier)	112 W, 122 W, 142 W
Max. forward velocity	2.5 m/s
Max. backwards velocity	-2.5 m/s



Introduction

The Autominy also has:

- ✓ **Sensor modules**: publish raw data from sensors (steering angle, encoder ticks, stereo camera, IMU).
- ✓ **Virtual sensors**: process raw sensor data (lidarpose estimation, camera pose estimation, obstacle detection, odometry, road marking localization, kalman filter hardware calibration, pointcloud, fake GPS).
- ✓ **Autonomics modules**: run in the background and can take over control if the car is in danger.

Introduction

To power the car:

- ❖Use a 14.8 V (4 cells with 3.8 V each) LiPo battery.
- ❖Voltage on each cell must be always above 3.2 V.
- ❖The car turns off if the battery voltage drops below 12.8 V.

https://www.indiamart.com/proddetail/lipo-battery-for-flower-dropping-drone-camera-23338091130.html

Introduction

The car has a **emergency stop** module that monitors obstacles present in the driving direction. Then, it can stop the car if crash is imminent. This module uses the LiDAR and intercepts the actuator's communication. Its configuration can be done through:

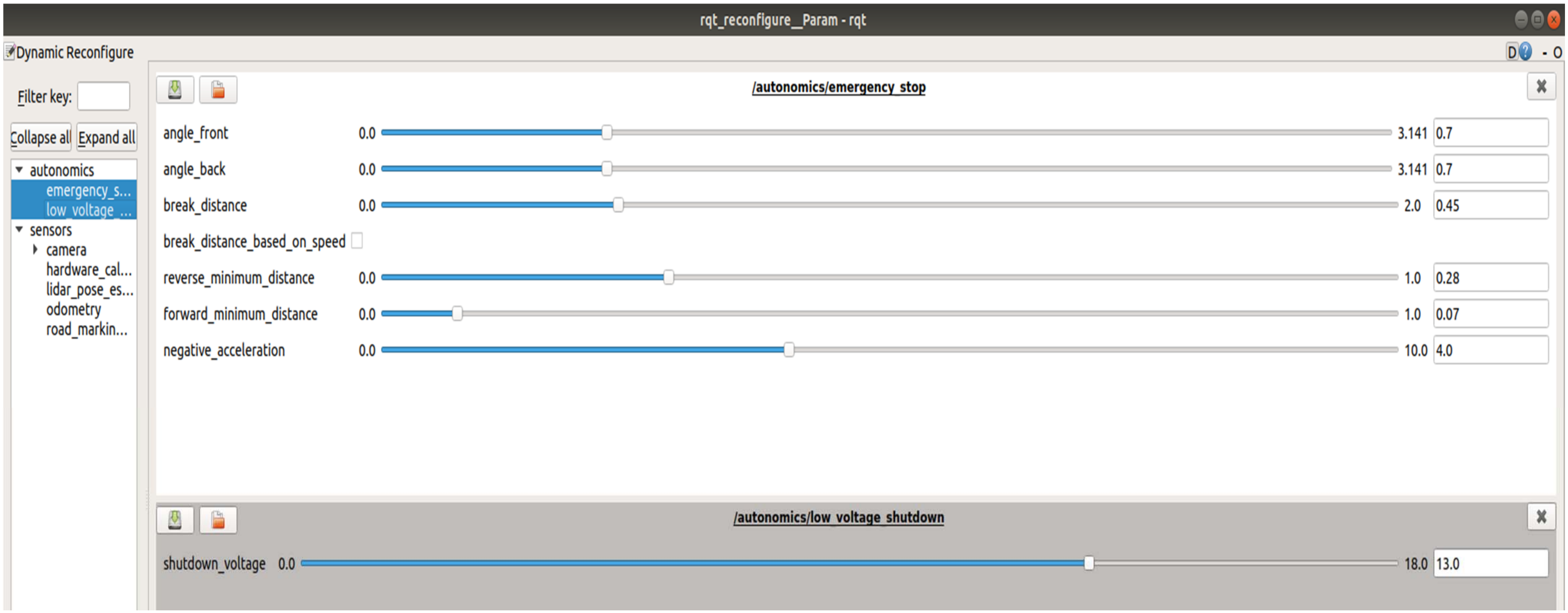
❖ **$ rqt dynamic reconfigure**

Table 1. Emergency stop configuration.

Name	Default value	Description
angle_front	0.7	Car's front angle to monitor
angle_back	0.7	Car's back angle to monitor
break_distance	0.45	Constant brake distance
break_distance_based_on_speed	False	Calculate brake distance based on speed
reverse_minimum_distance	0.28	Minimum distance of obstacles to be considered while reverse driving.
forward_minimum_distance	0.07	Minimum distance of obstacles to be considered while forward driving.
negative_acceleration	4.0	Acceleration used in speed-based braking

Introduction

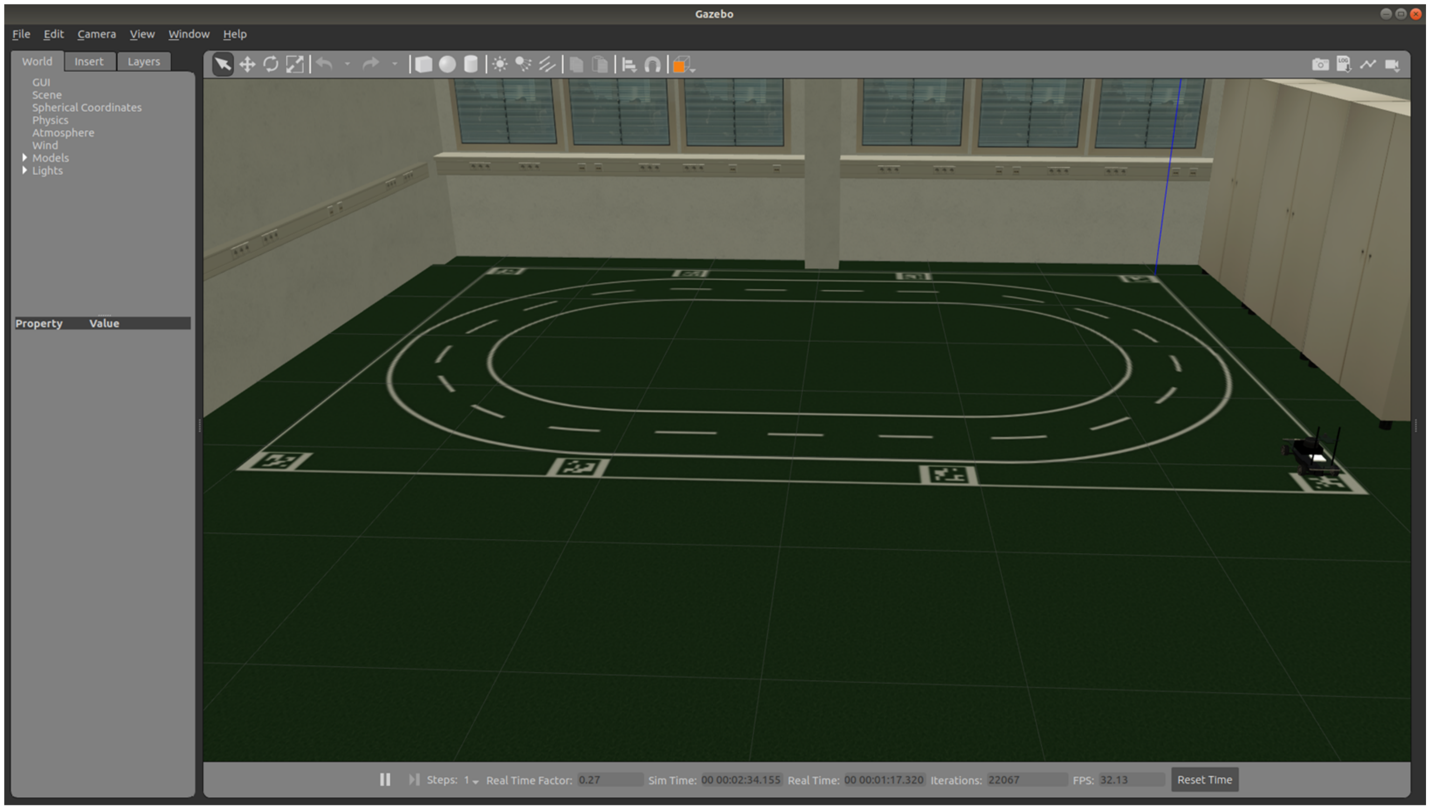
The **rqt dynamic reconfigure** tool can be visualized as depicted in the following figure.



Introduction

For **simulation purposes**, the Autominy package uses the **Gazebo** tool. It creates a physical model of the car. To use this simulator, we employ the command:

- **$ roslaunch autominy Simulated.launch**



Introduction

It is essential to emphasize that:

- **ROS** is a software framework/middleware for robot applications.
- It can be programmed by using:
  - C++, Python, LISP
- Package management (over 3000 packages available)
- Publisher/subscriber approach, services, and actions.
- Big community developed and documented by thousands of contributors.
- There are many libraries and tools free to be used (motion planning, object recognition, hardware interfaces, plotting data, 3D visualization, among others).

Introduction

ROS can be used for:

- Academic research: https://robots.ros.org/
- Industrial applications: https://rosindustrial.org/
- Autonomous cars: https://www.ros.org/news/robots/autonomous-cars/
- NASA: https://www.ros.org/news/2014/09/ros-running-on-iss.html

Introduction

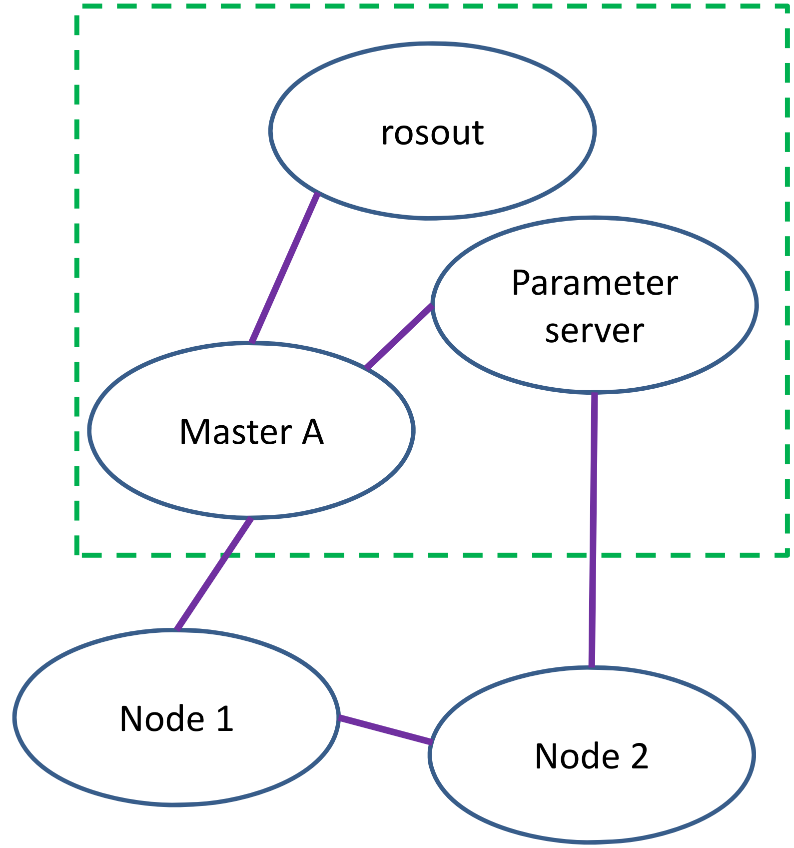
ROS allows for:

- Easier hardware abstraction and code reuse.
- Tasks can be divided in different parts that communicate with each other through **messages**.
- Each part is named "**node**" and is typically run as a separate process.
- Nodes perform a specific computation and share data with the network.
- Nodes can be added or removed while ROS is running.
- ROS can run on different machines (distributed system).

Introduction

The **ROS architectureROS** is as follows:

- There is a **master node**, which can be run by utilizing:
  - **$ roscore**
- The **master node**:
  - Tracks publishers/subscribers
  - Enables peer-to-peer connections between nodes.
- A robot control system usually comprises many nodes.
- Topics are named buses over which nodes exchange messages.
- A message is a simple data structure comprising typed fields.



Introduction

We assume that we are running:

- **Ubuntu 18.04**
- **ROS Melodic**

We can follow the installation guide of **Ubuntu** through the next address:

- https://ubuntu.com/tutorials/install-ubuntu-desktop#1-overview

Also, we can follow the installation guide of **ROS** through the next address:

- https://wiki.ros.org/melodic/Installation/Ubuntu

After finishing the previous tutorial, we are able to install the **Autominy packages**.

Introduction

To install the Autominy packages, we follow the next steps:

1. **$ git clone https://github.com/autominy/autominy**
2. **$ cd autominy/catkin ws**
3. **$ apt install python-catkin-tools**
4. **$ rosdep install --from-paths . --ignore-src --rosdistro=melodic –y**
5. **$ catkin build**
6. **$ source devel/setup.bash**

Introduction

After finishing the previous steps, we have installed all the Autominy packages. Watch the following video.

Dr. Roger Miranda Colorado

Researchgate:
https://www.researchgate.net/profile/Roger-Miranda-Colorado-2
Google Scholar:
https://scholar.google.com/citations?hl=es&user=NmzkrSwAAAAJ&view_op=list_works&sortby=pubdate
Pure-IPN:
https://ipn.elsevierpure.com/es/persons/roger-miranda-colorado-3
Youtube:
https://www.youtube.com/channel/UCeGT1lfNnJt695XGzEI4IxA