Welcome to the Robotics Assignment ("Lab") component of Intelligent Robotic Systems!

Over the semester, we will implement various methods for robot decision-making, both in simulated environments and on physical robotic hardware. In the upcoming "zeroth" lab, we will learn the key concepts of ROS (the Robot Operating System) to control the Mini Trucks that will accompany us throughout the class.

This is a Pre-Lab Tutorial to help you get ready for our labs.

Contents

1 Intro to the *Mini Truck* Robotic Platform

Autonomous driving has sparked a lot of public interest in the last few years. In this lab, we will work with a 1/14-scale autonomous *mini truck* as our mobile robot platform (Figure 1).

Figure 1: The 1/14-scale autonomous *mini truck* used in Intelligent Robotic Systems.

Figure 2 shows an overview of the robot's key physical components. The robot's **body** carries an NVIDIA Jetson Xavier NX onboard computer, a ZED 2 stereo camera, and a battery that powers them. These components form the core of the robot's perception and decision-making hardware.

The robot's **chassis** consists of drivetrain components that steer and drive the truck. We use a TAMIYA TT-01 Type-E RC chassis (Figure 3), which is a shaft-driven 4WD platform powered by a single 25 turn 540 brushed DC motor. The Jetson is connected with a Mastro Servo Controller, which translates the command from the Jetson and sends a Pulse Width Modulation (PWM) signal to the steering servo and motor ESC. In order to allow the robot to be driven through a remote controller, we added an additional servo multiplexer to switch signals.

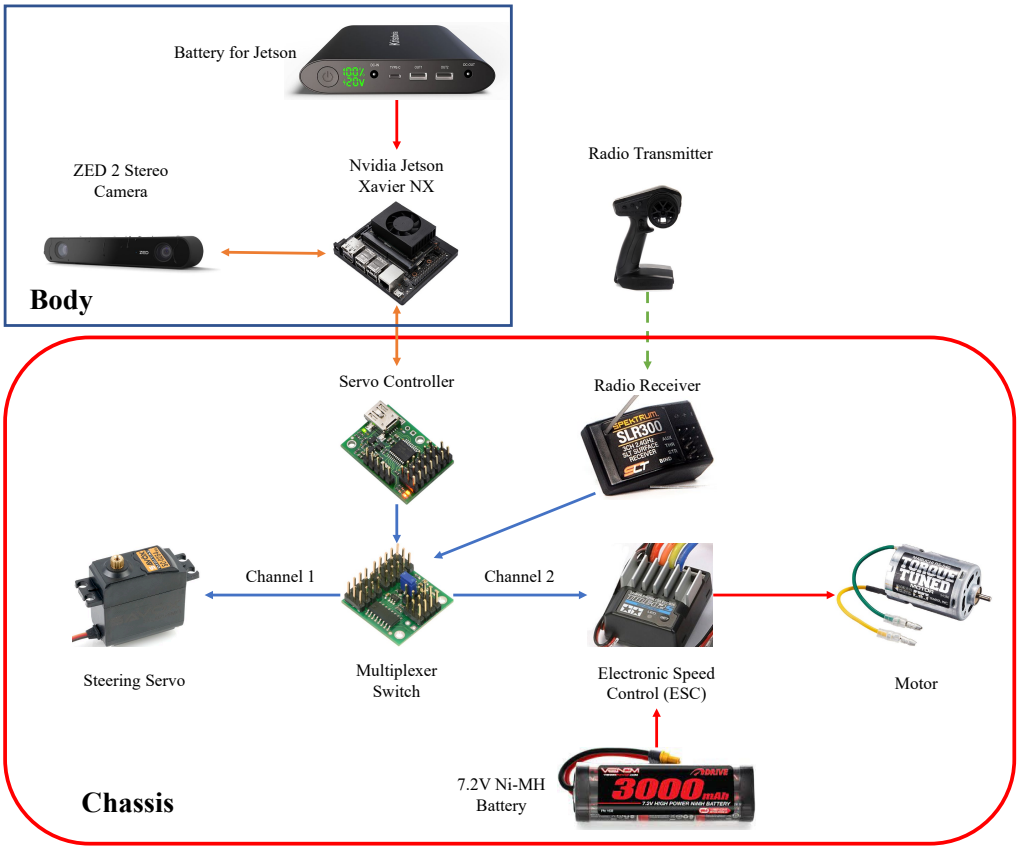


Figure 2: A hardware schematic of the robotic platform.

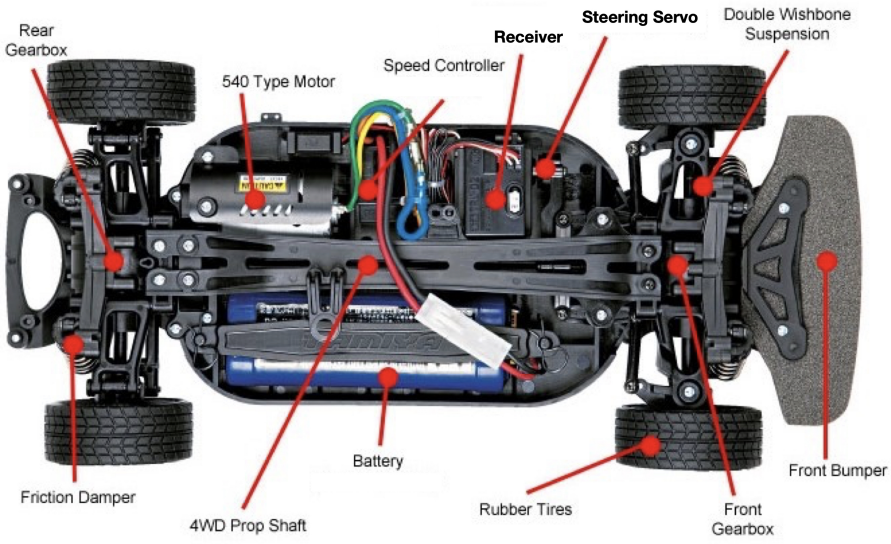


Figure 3: TT-01 Type-E Chasis that is used as the base of our robot platform

1.1 Safety First

Read the following instructions **carefully** before operating the robot, and keep them handy.

Robot Rules & Responsibilities

Please keep the robot and its accessories stored when you are not working with them. You can keep the robot in F111 or take it with you if it is convenient. Either way, please note that **you are responsible for your assigned robot for the duration of the semester**.

- **Use caution and common sense when operating your robot.** Avoid driving it around potential hazards—including stairs—or unsuspecting civilians! Beyond basic "bench tests", we strongly recommend you only drive the robot inside the F111 lab space.
- **Let us know immediately if any component is damaged or goes missing,** so that we can address the issue as quickly as possible. If you are experiencing hardware issues with the robot, please consult with a course AI **before** you try to fix it by yourself.
- **Practice with the remote control and drive slowly!** The robot can achieve a maximum speed of over 15 mph (25 km/h). **Injuries and bone fractures are likely** if a person is hit at that level of speed.
- Be **extremely careful** when handling and connecting batteries. **Shorted batteries will cause a fire**.
- Motors can become very **hot** while driving the robot **and** shortly after!
- The motor ESC will automatically cut off power if the Ni-MH battery voltage is too low. However, if the robot has been running for over 30 minutes or you notice a significant power drop, you should **immediately** place the battery in the **Uncharged Battery Bin** and replace it with a charged battery.
- Please be aware that our ability to repair the robots is limited, so take care of your robot accordingly. **If your robot suffers severe damage, you may not be able to complete the Lab and Final Project components of the class.**

1.2 Connect to the Robot

1.2.1 Turn On the Jetson

The NVIDIA Jetson Xavier NX takes 12 V – 19 V DC power supplied by the power bank mounted right below the Jetson. The power bank can output various DC voltages ranging from 5 V to 20 V. In order to avoid damage to the Jetson, **make sure you unplug the power cord before turning on the battery**. First, hold the power button on the battery until it lights up. Then, cycle through the voltage by double-clicking the power button until it shows **16 V**. Finally, plug in the power cord, and the Jetson will turn on automatically.

You will see the remaining battery life on the screen. Please note that the power will last less than two hours. To program the Jetson, use the external power supply, labeled **NX PWR**, instead.

1.2.2 Work on the Jetson Directly

The Jetson NX runs Ubuntu 20.04 desktop OS and can be used as a regular PC by connecting with a keyboard, a mouse, and a monitor through the HDMI port. The password for login is **nvidia**. The F110 Lab space provides Windows workstations, and you can borrow a mouse, keyboard and monitor for the Jetson.

1.2.3 Work on the Jetson Through SSH Connection

When the robot cannot be connected to a monitor and keyboard, for example when it is running on the ground, SSH becomes a useful tool to log into your robot and run programs through the command-line interface. SSH stands for **S**ecure **Sh**ell, a protocol that allows you to securely and remotely connect to your robot using a wired/wireless network connection.

All robots are connected to the local Wi-Fi network **ECE346** at startup. Each will have a reserved IP address **192.168.1.X**, where XX is the ID of the robot. For example, if the Jetson on your robot has the label NX-7, the IP address is 192.168.1.7. Similarly, if your robot is NX-11, the IP address is 192.168.1.11. Before running SSH, first, connect your computer to the **ECE346** Wi-Fi with password **ece346sp2023**. Once you have connected to the network, open a terminal/power shell and type

```
ssh nvidia@192.168.1.X
```

It will ask you for the login password for the Jetson: enter **nvidia** and you are all set.

SSH is a feature that can be found in any modern OS (Mac/Linux/Windows) machine. You can either use one of the Windows workstations in the F111 lab or bring your own laptop to connect to the robot. The process to get SSH working may differ across operating systems. If you do not have previous experience with SSH, we provide a detailed set of instructions in the class GitHub repo.

1.3 Drive the Robot with the Remote Controller

The remote controller (Figure 4) allows you to drive the robot manually—as you would a regular RC car—and also serves as a *dead man's switch* for the robot. We list each element's function below.

A. **Throttle Trim**: Adjusts the throttle neutral point

B. **Steering Trim**: Adjusts the steering centerpoint. Normally, the steering trim is adjusted until the vehicle tracks straight.

C. **LED**: Indicates the power is ON

D. **Steering Wheel**: Controls the steering angle of the front wheels.

E. **Throttle/Brake**: Controls the vehicle's acceleration.

F. **Steering Rate**: Adjusts the sensitivity (gain) of the steering wheel.

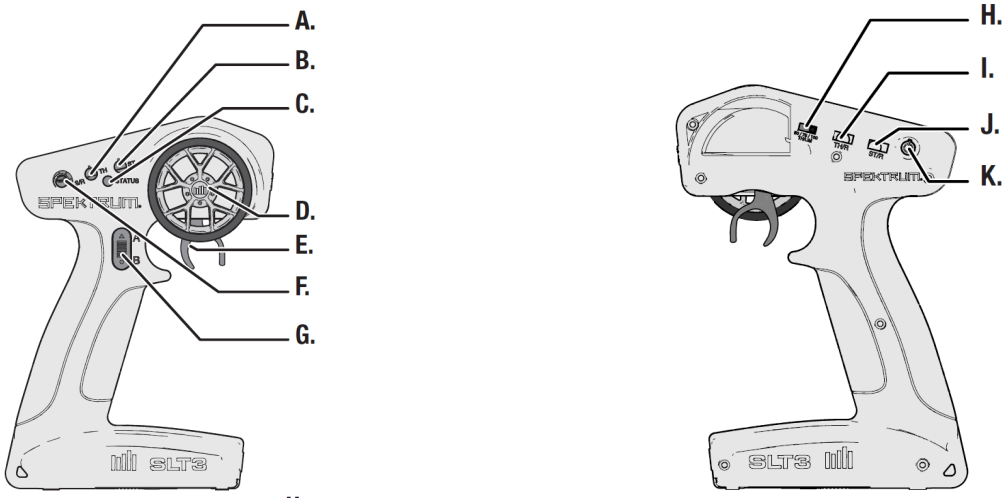
G. **Channel 3**: Three-position momentary switch (not used here).

H. **Throttle Limit**: Limits throttle output to 50/75/100%. Note: you should keep it at 50%.

I. **Throttle Reversing**: Flip the switch to reverse the throttle channel.

J. **Steering Reversing**: Flip the switch to reverse the steering channel.

K. **Power Button**: Turns the controller on and off.



(a) Right diagram of the remote controller.

(b) Left diagram of the remote controller.

Figure 4: Diagrams of the remote controller

The robot's drivetrain uses a separate power source that is connected directly to the ESC. To power up the robot, *first* turn on the remote controller by pressing the Power Button (K), *then* turn the switch on the bottom of the chassis to the ON position.

By default, pull the throttle towards you to go forward, and push the throttle away to brake and reverse. In order to steer the truck, you need to rotate the steering knob by the desired amount. You can invert the throttle and steering using the corresponding (I, J) switches.

1.4 Drive the Robot with the onboard Jetson computer

We use a Maestro 6-Channel USB Servo Controller to control the motor ESC and steering servo. We have provided you with a ROS wrapper of Mastero Servo Controller API. It subscribes to a ROS topic and sends inputs to the controller. Due to safety concerns, the multiplexer switch automatically disables the control signal from the Mastero Servo Controller. **In order to drive the robot with Jetson, you need to press the Down button of Channel 3 (G) all the time**. The system will immediately switch to the remote controller mode if you release this button.

2 Move Mini Truck Autonomously

Before we run any decision-making algorithms, we need to ensure our mini-truck knows where it is and can execute our control command. These functionalities have been built on your robot as ROS nodes.

In addition, we want to easily visualize the state and future plan of our robot on your own computers. Luckily, ROS has made it easy for us since it is naturally a distributed computing environment. A running ROS system can comprise dozens, even hundreds of nodes, spread across multiple machines with network setups.

To pass ROS messages between your laptop and the robot, your computer must connect to the **ECE346** WiFi. We provide two useful scripts for network setups.

```
# If the computer hosts ROS Master
source network_ros_host.sh <HOST_IP>

# If the computer is a client of the ROS Master
source network_ros_client.sh <HOST_IP> <CLIENT_IP>
```

In ECE346, `<HOST_IP>` is the IP address of your Robot, and `<PC_IP>` is the IP address of your computer under ECE346 WIFI. You can look it up from the network settings.

Step 1: Launch Perception and Control Nodes On Robot

To launch perception and control nodes, **open a new terminal** and **ssh into your robot**. Then,

```
cd ~/StartUp
./start_ros.sh <HOST_IP>
```

Please make sure your robot is static on the track because the localization algorithm requires accurate gravitational direction for initialization.

Step 2: Launch Visualization On Your PC

Next, we open a new terminal on your PC and navigate to the `ROS_Core` under your Git repository. Then, we activate the conda ROS environment, and (optionally) rebuild the workspace by:

```
cd <Path of your repo>/ROS_Core
conda activate ros_base
# Optional: if you have new packages
catkin_make
```

Moreover, let's configure the network setting using the script

```
source network_ros_client.sh <HOST_IP> <PC_IP>
```

Finally, we can launch visualization nodes

```
source devel/setup.bash # .zsh for Mac user
roslaunch racecar_interface visualization.launch
```

After around 30s, Rviz (Figure 5) and RQT (Figure 6) windows will show up.

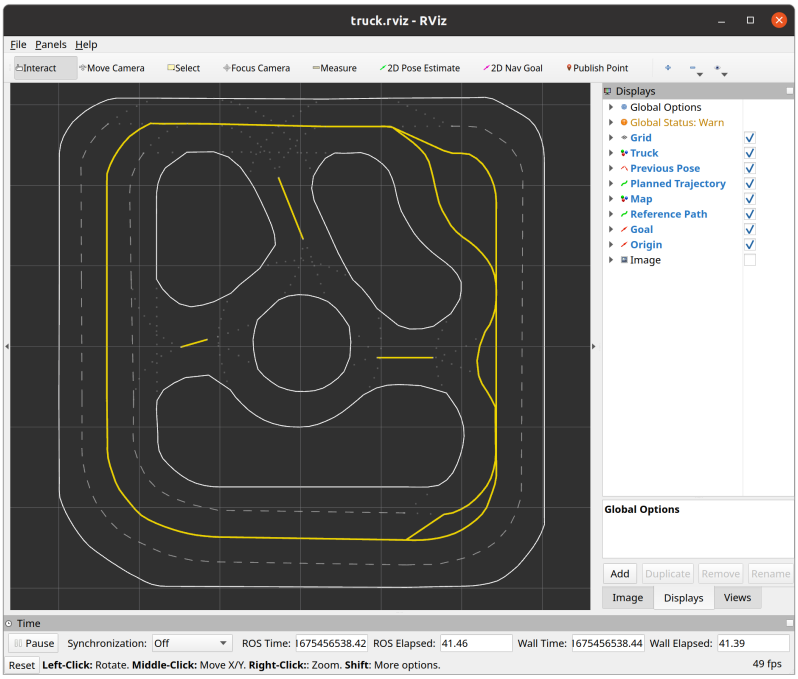


Figure 5: Rviz visualization tool. The orange box indicates the current pose of the robot and the yellow arrows indicate the past poses.

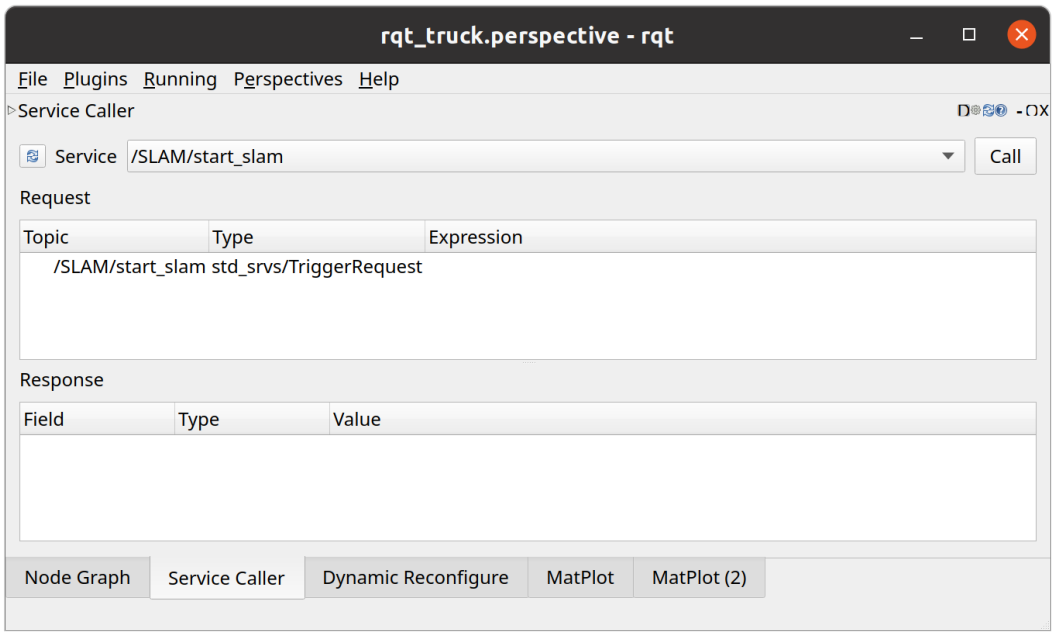


Figure 6: RQT GUI.

Step 3: Start Localization

On your RQT console (Figure 6), first, go to the **Service Caller** page. Then, click the **refresh button** 🔄 and choose **/SLAM/start_slam** from the drop-down menu. Finally, click the **call** button to start localization.

An orange box will appear on your Rviz, which indicates the pose of your robot. Drive around the track and try to verify if the state estimation is accurate.

Important: Localization results will be significantly compromised if fiducial markers are occluded. Please do not stay or place your items inside the room.

Step 4: Launch Your Program On Truck

To launch your own decision-making algorithms, **open a new terminal** and **ssh into your robot**. Then, let's navigate to your workspace, activate the conda ROS environment and configure network setups by:

```
cd <Your Workspace>
conda activate ros_base
source network_ros_client.sh <HOST_IP> <PC_IP>
source devel/setup.bash # .zsh for Mac user
```

Finally, launch your node and remember to press the Down button as described in Section 1.4 to test your algorithm.

```
roslaunch <ROS Package> <Launch File>
```