F1Tenth Documentation:

**Replicating the Linux Environment on Laptop:**

The F1Tenth Car uses the robot operating system (ROS). ROS has multiple versions, ROS, ROS1 and ROS2. This F1Tenth Car uses ROS which is only compatible with Ubuntu version 16. Note Ubuntu 22 and later releases will not work with the ROS that is currently on the car. If you which to install Ubuntu on an operating system so that you can set up the environment that is on the F1Tenth car currently, then follow this guide:

**Step 1**: Download Ubuntu 16.04.7 LTS (Xenial Xersus)

Link to past releases of Ubuntu: <https://releases.ubuntu.com/?_ga=2.168765858.2046390350.1681326261-1221278850.1680913303>

**Step 2:** Flash the Ubuntu version to a Flash Drive so that the Flash Drive can be used when booting your computer.

**Step 3:** On a computer you are okay with either making dual boot or with losing all your previous data and making a single boot, insert the Flash Drive into the computer, power on the computer, press F12 (probably) to get into boot settings, and select boot from usb drive then it should proceed to ubuntu installation.

Link to installing Ubuntu: <https://ubuntu.com/tutorials/install-ubuntu-desktop-1604#1-overview>

NOTE: when installing ubuntu it is easiest in later steps if you have your name as: aimslab

**Step 4:** With ubuntu installed follow the guide below to install ROS: Note the car uses ros-kinetic so with every noetic replaced it with kinetic. So for example when it is time to install the full desktop do ‘sudo apt install ros-kinetic-desktop-full’

Link to installing ros: <http://wiki.ros.org/noetic/Installation/Ubuntu>

**Step 5:** Copy the ‘f1tenth’ folder from the car and paste the folder onto your ubuntu computer.

**Step 6:**  Delete the files generated by the catkin\_make that are in the f1\_tenth/f1\_tenth\_ws/build folder as highlighted by the image: Table

Description automatically generated

**Step 7:** using the cmd terminal install: rospy, python3-numpy, yaml, rospkg, pyqt5-dev-tools, libspnav-dev, ros-kinetic-driver-base, ros-kinetic-ackermann-msgs, ros-kinetic-joy, ros-kinetic-map-server. NOTE to install the command will probably be something like ‘sudo apt install installname’

**Step 8:**  In the terminal, in the f1tenth, f1tenth\_ws run the following: catkin\_make

The cakin\_make might have errors, it means you either need to remove old makefile data or install more stuff as the errors will say

**Step 9:**  if the catkin\_make works, then here are the commands for starting the simulator:

In one terminal run:

catkin\_make

source\_devel/setup.bash

roslaunch f1tenth\_simulator simulator.launch

(once the simulator is running, press n to get the program you are using to run), For manual controls press k, then use w a s d.

In another terminal run: python3 yourcode.py simulator

For Running the car in real life use:

In one terminal run:

catkin\_make

source\_devel/setup.bash

roslaunch racecar teleop.launch

In another terminal run: python3 yourcode.py car

**Step 10:** For your python script to work with out error you need the simulator.launch or teleop.launch. For example python program that a previous group made look at Scipio.py in the Desktop/f1tenth/f1tenth\_ws/src/AvoidObject directory

**Helpful Hint:** In the python program you are trying to run in the rospy.Publisher, if it is set to drive\_topic\_car it will only work in real life and not in the simulator. To get it to work in the simulator it needs to be drive\_topic\_simulator

self.drive\_pub = rospy.Publisher(drive\_topic\_car, AckermannDriveStamped, queue\_size = 10) #drive\_topic\_car

**Helpful Hint2:**

On F1tenth car

click wifi logo

click hokoyuo to connect **Need to do this to get lidar data**

**Running the Simulator**

In one terminal run:

cd Desktop/f1tenth/f1tenth\_ws

catkin\_make

source\_devel/setup.bash

roslaunch f1tenth\_simulator simulator.launch

(once the simulator is running, press n to get the program you are using to run), For manual controls press k, then use w a s d.

In another terminal run: python3 yourcode.py simulator

Example:

cd Desktop/f1tenth/f1tenth\_ws/src/AvoidObject

python3 Scipio.py simulator

**Running the Car**

For Running the car in real life use:

In one terminal run:

cd Desktop/f1tenth/f1tenth\_ws

catkin\_make

source\_devel/setup.bash

roslaunch racecar teleop.launch

In another terminal run: python3 yourcode.py car

Example program:

cd Desktop/f1tenth/f1tenth\_ws/src/AvoidObject

python3 Scipio.py car

**SSH into the vesc**

Open Putty

# note ip address is not guranteed to be the same and can change

#for windows devices only

Be on same network (u of l secure )

get the ip address on nvidea jetson via clicking on wifi symbol on desktop and the connection tab then selecting ulsecure(default)

or use terminal with ifconfig and look for wlan0

into putty select ssh and type the computer name and ip like as follows into host name

aimslab@10.204.120.54

press open

accept if pop up

**Part 2** transfering files from laptop to jetson (you can do this portion with out putty)

open up powershell

in terminal type 'sftp aimslab@10.204.120.54'

type yes to connect and the the jeston password 123456

command 'pwd' tells you whilch directory on the remote device (print working directory)

command 'lpwd' tells you local working directory

example getting to directory on jetson:

cd Desktop/f1tenth/f1tenth\_ws/src/

cd

on jetson to make new directory

mkdir TestImport

to import file, being the directory you want it to go to

and do 'put filename.extension'

getting to directory locally is command 'lcd directory path'

//you can use putty to open a terminal and run ros launch

then another instance of putty to run the program

to go the other way use the cmd 'get filename.extenstion' to get a whole folder add -r

rm -r folder name

**Useful Programming information:**

Initializing everything: in the rospy.Subscriber the self.mapping\_callback is the main function that kicks everything off in terms of reading data from the lidar via msgs with the msgs.ranges containing an array of distances in meters being found from the lighter.

def \_\_init\_\_(self):

        global prev\_time

        #Topics & Subs, Pubs

        lidarscan\_topic = '/scan'

        drive\_topic\_simulator = '/nav'

        drive\_topic\_car = '/vesc/high\_level/ackermann\_cmd\_mux/input/nav\_0'

        prev\_time = rospy.get\_time()

        self.PreviousTime = rospy.get\_time()

        #consider passing all the surrondings as an object

        self.lidar\_sub = rospy.Subscriber(lidarscan\_topic, LaserScan, self.mapping\_callback)

        #note drive\_topic\_simulator publishes to the simulator and nothing will happen in real life

        #drive\_topic\_simulator for simulator

        #drive\_topic\_car for real life

        self.drive\_pub = rospy.Publisher(drive\_topic\_car, AckermannDriveStamped, queue\_size = 10) #drive\_topic\_car

The Lidar sends data via a ranges array with filled with meters. This is in a msg.ranges

Example: for mapping the area relative to the car:

## set the surrounding object data

    def MapArea(self, surrounding, msg):

        length\_of\_ranges = len(msg.ranges) #Length was 1081

        #array length of each object

        alo = int(length\_of\_ranges / 7)

        surrounding.ObjectBackRightDistance = (min(msg.ranges[0: alo]))

        surrounding.ObjectRightDistance = (min(msg.ranges[alo: (alo\*2)]))

        surrounding.ObjectFrontRightDistance = (min(msg.ranges[(alo\*2): (alo\*3)]))

        surrounding.ObjectFrontDistance = (min(msg.ranges[(alo\*3): (alo\*4)]))

        surrounding.ObjectFrontLeftDistance = (min(msg.ranges[(alo\*4): (alo\*5)]))

        surrounding.ObjectLeftDistance = (min(msg.ranges[(alo\*5): (alo\*6)]))

        surrounding.ObjectBackLeftDistance = (min(msg.ranges[(alo\*6): ((alo\*7)-1)]))

        print("Mapped Area")

        return surrounding

Making The car drive, in your callback function, you need to send drive msgs: example

                print("Driving straight Maintain Speed.  Closest Object in front:")

                print(surrounding.ObjectFrontDistance )

                drive\_msg = AckermannDriveStamped()

                drive\_msg.header.stamp = rospy.Time.now()

                drive\_msg.header.frame\_id = "laser"

                drive\_msg.drive.steering\_angle = 0

                self.CurSpeed = MaxSpeed

                print("Current Speed: ", self.CurSpeed)

                drive\_msg.drive.speed = self.CurSpeed

                self.drive\_pub.publish(drive\_msg)