# AUTOMATIC CONTROL COURSE

**Basic Information - Car Platform**

November 1, 2016

Most of the details about the platform can be found at http://f1tenth.org/.

---

## HARDWARE DESCRIPTION

The car is built based on these instructions,
http://f1tenth.org/misc-docs/BuildInstructions

1. **Traxxas 1/10 Scale Rally VXL**

   This is the basic car platform. It's a 4WD and can reach speeds over 40 mph. The complete specs can be found here, https://traxxas.com/products/models/electric/74076rally.

   Some useful links:
   https://traxxas.com/support/Programming-Your-Traxxas-Electronic-Speed-Control
   https://traxxas.com/support/How-do-I-bind-transmitter-receiver

   Issues:
   This is a Race car and not designed to drive slow. Hence, the minimum speed is quite high.

2. **Nvidia Jetson**

   This is the GPU. It has Ubuntu 14.04 and ROS Indigo installed.
   More details about the specs,

http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

The ROS IP of your computer is exported to Jetson. Hence, your computer and Jetson can share a common ROS master. This way, the commands generated from your computer is available to the car.
http://wiki.ros.org/ROS/Tutorials/MultipleMachines

3. **Teensy 3.1 Microcontroller**

   The commands from Jetson are received by Teensy. An arduino code converts these commands to PWM signals. You can control the steering and throttle.
   https://www.pjrc.com/teensy/teensy31.html

   The ROS commands are sent to Teensy via ROS Serial. The default serial port is /dev/ttyACM0.
   http://wiki.ros.org/rosserial_python

4. **Picostation**

   This is used to create an access point to communicate with the car. The SSID is 'picostation1' or 'picostation2' based on your car. The password is 'f1@tenth'.
   https://www.ubnt.com/airmax/picostationm/

   Create a static IP for the picostation network on your system,

   IP- 192.168.2.110, Mask- 255.255.255.0, GW- 192.168.2.110 (Car A)
   or
   IP- 192.168.2.x, Mask- 255.255.255.0, GW- 192.168.2.x

   IP- 192.168.3.110, Mask- 255.255.255.0, GW- 192.168.3.110 (Car B)
   or
   IP- 192.168.3.x, Mask- 255.255.255.0, GW- 192.168.3.x

   x - any desired IP

5. **LIDAR**

   The HOKUYO-UST-10LX-LASER is available for scanning the environment.
   https://www.hokuyo-aut.jp/02sensor/07scanner/ust_10lx_20lx.html

The LIDAR data is available over the topic /scan. This can be visualized in RViz. By combining the laser scan data with Hector Slam package in ROS, you can generate the map of the environment.
http://wiki.ros.org/hector_slam
http://f1tenth.org/lab_instructions/W3_T1_Using%20the%20Hector%20SLAM.pdf

6. **Inertial Measurement Unit**

   The Razor IMU gives you 9DOF data.
   https://www.sparkfun.com/products/10736

   A ROS package is available for this.

   $ sudo apt-get install ros-indigo-razor-imu-9dof

   Launch the IMU and subscribe to the topic "/imu".

   $ roslaunch razor_imu_9dof razor-pub.launch

   More details on LIDAR and IMU,
   http://f1tenth.org/lab_instructions/t5.pdf

7. **Qualysis Motion Capture System**

   The SML lab provides a localization system by using Qualysis. This can give the position and orientation (6DOF) of the car with great accuracy. It updates the position at the rate of 120 Hz.

   To receive Qualysis data, you have to either have a LAN connection or connect to Wi-Fi network 'SML'.

---

## GETTING STARTED WITH THE CAR

You can check the basic functionalities of the car with the following steps.

Some steps that can make life easier,

- Install 'Terminator' for handling multiple terminals.
  http://gnometerminator.blogspot.se/p/introduction.html

- Include these lines at the end of file ".bashrc".

  source /opt/ros/indigo/setup.bash
  source ~/el2425_ws/devel/setup.bash

  This loads "el2425_ws" workspace for ROS. Saves you from doing it from the terminal every time.
  (Note: If you are working with multiple workspaces, you'd have to source them each time from the terminal.)

## 1. Teleoperation

Connect to Picostation network.

Access Jetson via SSH,

$ ssh ubuntu@192.168.2.2 (Car A)
or
$ ssh ubuntu@192.168.3.2 (Car B)

Start the ROS Master,

$ roscore

Then, initialize the publisher which accepts your key press and generates drive commands.

$ rosrun f1tenth_task1 keyboard.py

We then need a listener that can subscribe to the topic and scale the commands according to PWM.

$ rosrun f1tenth_task1 key_receiver.py

Finally, the PWM commands need to be sent to Teensy.

```
$ rosrun rosserial_python serial_node.py /dev/ttyACM0
```

2. **PID Control**

   In this case, we'll have the entire control algorithm on the computer and only send the final commands to Jetson.

   Before you start, ensure that Qualisys is running and the car is visible to it. For Qualisys, use the project "labhybrid2".

   Connect to Picostation network and SML network.

   On your computer:

   Source "el2425_ws" workspace in every new terminal,

```
$ source /opt/ros/indigo/setup.bash
$ source ~/el2425_ws/devel/setup.bash
```

   Set the ROS IP according to your connection with picostation. This should be the computer's IP over picostation.

```
$ export ROS_IP=192.168.2.110
```

   Launch the code,

```
$ roslaunch f1tenth_task2 starter.launch
```

   Launch files help you to start all your nodes in one go. You can also assign any special initializations and arguments in your launch file.

   Have the emergency stop node running,

```
$ rosrun f1tenth_task2 kill_switch.py
```

   Pressing the 'Delete' button sets the flag to 'STOP' and 'Home' button sets it back to 'NORMAL'.

   On Jetson:

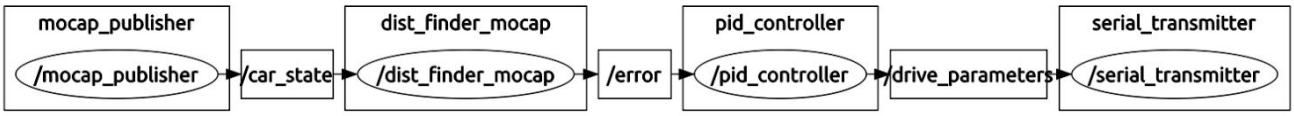
Set it accept your computer as Master,

```
$ export ROS_MASTER_URI=http://192.168.2.110:11311
```

Run the serial node to communicate with Teensy,

```
$ rosrun rosserial_python serial_node.py /dev/ttyACM0
```

The structure of the code is straightforward,




NOTE: It is possible to have the car to be completely autonomous and run the entire controller on Jetson. Only the START-STOP commands and Qualisys data could be provided from your computer.

---

## CODE - GITHUB

https://gits-15.sys.kth.se/jonas1/f1tenth