Building the f1tenth platform modified for CriSp

Björn Andersson

January 25, 2022

Summary

List of Figures

1 The f1tenth platform

1.1 System overview

The f1tenth platform is an open source platform meant for easy protyping in scientific projects. It is a modified RC-car from traxxas with added sensors. They also developed a software integration for the system using ROS. The project builds upon the MIT project Racecar. [1] Several university courses are held on the topic of autonomous driving using this platform. More info can be found on the f1tenth webpage.[2]

[1] https://mit-racecar.github.io/
[2] https://f1tenth.org/

2 Building the track

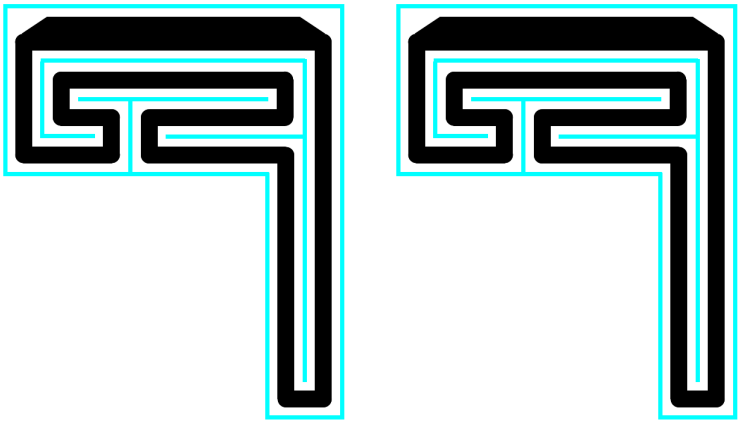
Several ideas for building the track were considered.

1. Precolored playwood with vinyl lines to mark the road. This had been used in a previous project with good results. Also using a levered curved section. But this was considered to be:
   (a) Too time consuming to construct.
   (b) Too inflexible and heavy workload incase the project need to move into a new location.
   (c) Not working well with oils and other fluids to decrease friction on the road.
2. Construct the track with black squared yoga mats. And creating the walls around the track with pool noodles. This was also ruled out becasue:
   (a) It would be timeconsuming to clean the yoga mats after performed experiments.
   (b) It was hard to find the neccessary parts in Norway, and importing was very expensive.
3. Using special robotics tape straight on the floor and vinylprinting the curved sections. Also constructing the walls around the track with a harder paper roll. Held together with 3D printed wall mounts.

In the end method nr.3 was choosen. As illustrated in the figure 1. The origianal layout in figure 2 had to be reconstructed into a simple model due to lack of space. The final layout can be seen in figure 3.

Before taping laser were used to get decently straight lines, also a whiteboard marker where used to mark the lines where the track would go and also where the wall mounts would go. The distance from the wall to the track is $30cm$ and the width of the track is $50cm$. The upper part of the track is $100cmwide$ and is a double lane where the middle is marked with green stripes.

Figure 1: Part of the track seen from the car



(a) Original track layout (b) Final track layout

Figure 2: Track layouts

3 Hardware - Building the car

The car is a modified version of the Traxxas Slash 4x4 Ultimate RC car seen in [figure 3a]. It is equipped with a Velineon® 3500 brushless motor [figure 4a], Velineon® VXL-3s ESC[figure4b], high speed metal gear servo [figure 4c], and radio transmission module and ackerman steering. The ESC and the radio transmission module need to be removed in order to build the f1tenth version. The ESC is replaced by a VESC[3] [figure 5] and the radio transmission is handled by the Nvidia TX2 computer[4]. Some more modifications need to be done before starting the building. The side bumbers need to be removed and the an extra 3D-printed platform need to be fasted to have space for all computers and sensors. See [figure yyyxxx]. Most of the build is descriebed in detail at f1tenth webpage for build.[5] As this is an open source project the descriptions can be varying in level of detail, but also in depreceiation. The computer used in the CriSp project is as mentioned the Nvida TX2 computer, which by the f1tenth build site is considered to be depreched. It is instead recommended to use the newer Nvidia Nano which will simplify the process of building the car and installing the neccessary software. Below is a clarification of some particular issues that are slightly unclear form the documentation on the f1tenth build site alone. Difficulties regarding the software setup of the TX2 and VESC is discussed under the section "Software".

(a) Traxxas 4x4 Ultimate (b) Under the hood

Figure 3: Traxxas RC-car

3.1 VESC

VESC (Vedder ESC) is originally a project from Benjamin Vedder to imporve speed control of electrical skateboards. The VESC used in this project is VESC MK6. It is connected to the Traxxas brushless engine with "three wires". The connectors of the motor had to be cut off and replaced with new ones in order to fit with the VESC connectors. These had to be soldered on and afterwords a rubber cover had to be melted ontop of them to avoid short circuits. The color

[3] https://vesc-project.com/
[4] https://developer.nvidia.com/embedded/jetson-tx2
[5] https://f1tenth.org/build.html

(a) Brushless DC motor (b) ESC (c) High speed metal gear servo (d) VESC 6 MK V

Figure 4: Components

code is :.... If for some reason the wires are connected in the wrong way than the result will be that the car is driving

3.2 Power board

The PCB board was cretaed by the f1tenth team and was ordered. Although it come without components so some soldering was neccessary. Also it was neccessary to add two wires for power and ground from the powerboard to the TX2 computer. The powerboard and its components can be seen in [figure 5].

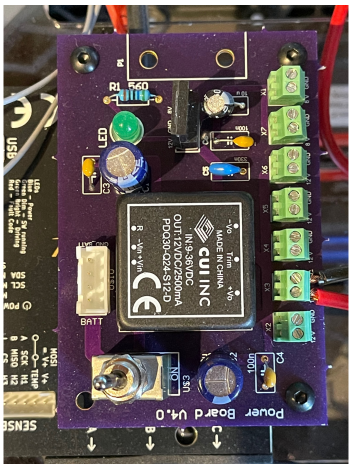


Figure 5: Power board

3.3 Wire between servo and the VESC

The wire to connect the servo to the VESC is not included with the Traxxas car. This cable has to be fabricated on your own. It's quite an easy one to do. It's a three wire connector (power, ground, signal) as shown in [figure xxxxx].

4 Software

Due to the fact that the computer being used is the Nvidia TX2 some "limitations" exist.

- The host computer (meaning the computer not on the car) has to use Ubuntu 18.04 in order to be able to flash the neccessary JetPack v.x. to the TX2 computer on the car.
- An extra orbit carrier board has to be attached to the TX2 computer to make available several normal I/0 like HDMI and USB.
- A USB-hub has to be connected to the TX2 computer (actually the orbit carrier board) to be able to use more than 1 USB port.
- The orbit carrier board must be flashed with the correct firmware

Some difficulties were encountered while flashing the VESC firmware which will be explained under "Flashing the VESC".

The system is using ROS to communicate between the different sensors and motors on the car, but also between camera and image processing libraries like OpenCV. Which is discussed in detail under ROS and Camera.

4.1 Flashing OS and firmware for the carrier board

The horror and the pain. I will update this section with some proper instructions.

4.2 Flashing the VESC

To flash the firmware to the VESC the VESC TOOL is needs to be downloaded and installed. This can be done for free from the VESC webpage. A throughful guide of how to do this is provided by the f1tenth build page xxyyy.. Although at this time of writing it does not exist the neccessary "$servo_out$" firmware for this model of the VESC. Many different approces and firmware were tried with little success. In the end the firmwire were built from scratch using the official VESC source code created by Benjamin Vedder.

To create a binary file from Vedders source code:

1. Download the source code from his github
2. Change the "ENABLE_SERVO_OUT" from 0 to 1 in "conf_general.h".
3. Be sure to comment out the version of the hardware you are using in the "conf_general.h"

4. Type "make" in the terminal. Make sure to follow the instructions on Vedder's github page regarding which compiler should be used.

5. Flash onto the VESC using the VESC_TOOL provided from the official VESC home page

After this please run FOC calibration and set up the correct input communication for the servo. This can be done in the vesc_tool in the motor configuration wizard and the input wizard.

- f1tenth firmware for VESC. Was not up to date with the current VESC used in this project at the time of writing.
  `https://github.com/f1tenth/vesc_firmware`
- The official VESC source code, this is the one we used to build from scratch
  `https://github.com/vedderb/bldc`

With all of this set up the motor could drive forward and backward properly, but there was still no response on the servo. After long investigation it was found that the reason for this was that the ROS bridge implementation for the VESC was configured for a different axxis on the servo. Changing this one to the correct one made the servo work as expected.

4.3 ROS

A lot of thought was put to what programming language should be used within this project. The f1tenth team already has created a functional implementation of the car system in ROS (Robot Operating System). ROS is meant to be a middleware which can easily blend hardware and software from different manufacturers. Although this leads to a bit of a overhead in needed software this is a well tested enviroment for integrating robotic systems. A self made platform could be done but this would require a lot of expertise and time which is beyond what this project has asked for. Thus it was decided not to reinvent the wheel and make use of an already existing system.

It proved to be a bit more difficult then expected as ROS is quite an extensive system with many concepts to get the hang of. At the time of this writing all components of the system has been successfully integrated and can be controlled. That is, the motor, the servo, the camera and openCV are all operating on the same ROS core.

4.4 Intel Real Sense Depth Camera

In order to make use of the Intel Real Sense D435i camera on the TX2 computer it waas neccessary to make some changes in the linux kernel. Jetson Hacks have luckily found a way to do this, and shared their method. Will write more about how to do this ...

4.5 Code implementation

Not so much valuable code and algorithms have been developed at this stage of the project. But some fundamentally important bridges between components have been constructed. More about this can be read on this projects github page.

CriSp github:
`https://github.com/crispTesting/crisp`

5 Future work

5.1 More sensors

Althoug both the VESC and the Intel Real Sense Depth Camera has integrated IMU's, there are available external 9DOF IMU's to be attached to either the car or the trailer.

There are also available Range Finders which could be used to detect the walls around the car for further sensor fusion and giving a better estimate of the position.

There has been proposed that a camera system shall be mounted over the track and should be able to provide a better estimate of the ground truth of the cars position.

5.2 Image processing

Some success has already been achieved with the use of OpenCV functionallity to detect edges using the Hough Transform. Further work would include determing only the important two edges representating the lane, finding the center point of the lane and project this center point a certain distance in front and drive towards that one.

5.3 Servo control

Although the servo can be controlled autonomously, the tuning of turn ratio has to be found.

5.4 Emergency Stop

A formal emergency stop button should be implenented. In the way the code is constructed right now everything will shut down when the script is terminated. So in some sense this mechanism is already in place, but it could be improved upon.

5.5 Controller for navigation

Has to be considered what should be implemented here. PID? LQR? Something else?

5.6 Braking

There will be a need to implement an algorithm to slow down during more curved section of the road, or when the sideslip is large and the car is about to drive off track.

5.7 Data extraction

The algorithm for data extraction has to be created. Goals for this is to be able to record data during the curved section of the road and later be able to analyse and display this data for different quantiles of the section. Here things like, sidesslip torque, speed, acceleration and time should be messaured.

5.8 GUI for data analysis

A GUI for data analysis of the extraced data has to be created. The goal is to create a GUI where you can set how many rounds the car should drive and simply press a button "Drive" to start the experiment. Ideally the car would then drive the set amount of laps and then return to its "resting place". There should be a button for emergency stop. Data should be able to be analysed depending on both race nr. and lap nr.

5.9 Permanent network solution

In order to control the car one must use SSH to log into the car. Currently neither the host computer or the car has a static IP-address resulting in that they often are located on different subnets providing difficulties connecting to each other. A solution to this is to log into the IOT network, which have been suggested.

5.10 Camera mount

A camera mount has to be either 3D-printed or bought and be mounted on the car. Some suggestions for 3D-printed models have been made using Autodesk Fusion 360. Theese files can be found on the projects github page. See under code implementation [subsection 4.5].

6 Food for thoughts

6.1 System upgrade

If there would be a need to buy a new or more cars. I suggest that the newer Nvidia Nano will be bought. This would simplify much of the build process and integration of sensors.

6.2 Code

There is a possibility that machine learning algorithms could be used to train the car to see how it should act. When should it slow down, when should it speed up etc. The car could then be driven manually with a joystick to gather as many pictures as neccessary for the algorithm to work.