**Lab 1. Get familiar with F1/10 Race Car**

Project Lead: submit codes and peer evaluation form.

Subscriber: submit lab report and peer evaluation form

Due date: Wednesday Feb. 25, 2021 on Course Site

**Major Tasks:**

1. Check/reassemble the race car; learn basics of motor control via the VESC Tool;
2. Install/update basic software on host computer and Jetson device;
3. Learn basics of Ubuntu 18.04 Linux operating system;
4. Work through the first few parts of ROS Tutorial at the Beginner Level.

**Preparation before the lab:**

0.1 Get to know your teammate. Review safety rules for lab operation.

0.2 Gather the race car, other hardware components, and the hand tool set from the Instructor or TA. Figure out how to store your tools, parts, and race car in your work space;

0.3 Have a laptop ready to install Ubuntu.

**Part 1. Hardware assembling:**

Follow the instructions provided on Course Site on how to check or reassemble the F1/10 car. If you decide to reassemble the car, take photos along the way and document difficult encounters and interesting findings. The major tasks in this part include:

* 1. Check battery and battery charger; charge the batteries;
  2. Check the car for any installation defects, loose screws, loose wires or connections;
  3. Use the VESC Tool to tune the motor and VESC.

Your report shall summarize the main steps you take in the. Make sure to answer the following questions and present your results.

Q1.1. What is the type of battery used in the car? What voltage is good to use and what voltage indicates that the battery needs recharge?

Q1.2. What are the definitions of pitch, roll, and yaw? What is a differential drive system? What is an Ackerman steering system?

Q1.3. What are the differences between a brushed and a brushless motor?

Q1.4. How does the PID control work? How do you achieve PID control with the VESC 6 for the F1/10 race car? What do you learn from the VESC Tool? Include the motor testing results from the VESC Tool. If you tried the VESC-Tool Mobile App, report your experience to get a maximum of 10 bonus points.

* PID stands for Proportional, Integral, Derivative. When we try to set the speed of the motor, a control signal is sent to the VESC that defines our target speed. The VESC then outputs a pulse width modulated signal that drives the motor and reports back how far away from the target the current motor speed is. The control input is then adjusted in relationship to the current error using the P, I and D values. Using the PID transfer function we are able to predict the growth of the error and adjust smoothly to a new target speed.
* We achieved PID control with the VESC by first starting with the upenn configuration XML file. This was supposed to set reasonable starting values for the VESC, however we found that all of the values were far too large and were causing instability. We reduced Ki and Kd to very close to 0 and adjusted Kp until we saw corrections of < 500 rpm for our targeted 500 rpm steps. Then we increased Ki until we achieved exactly 500 rpm steps. Finally to combat overshoot and provide better damping, we increased Kd until we saw the overshoot start to grow again.

Q1.5. What problems did you encounter working through assembling the hardware of the car? What did you learn or enjoy most in Part 1 of Lab 1? What suggestions do you have to improve the design or assembling procedures of the car?

* We did not disassemble and reassemble our car as this car seemed well built to start with. However, one note to make car use easier in the future, the wires for the USB hub and the Lidar should be zip tied on the underside of the custom top plate to make accessing the screws for the top plate easier to allow for faster removal of the battery.

Your code submission for this part shall include

File 1: the config xml file you create from the VESC Tool for your vehicle;

File 2: any video or film you took that may help others with the assembling procedures.

**Part 2. Software Installation:**

Follow the instructions on Course Site to install and update basic software packages. If you installed software on your own computer or set up wireless router for your own test space, report your experience. The major tasks in this part include:

2.1. Install/update Ubuntu 18.04, ROS and Nvidia Jetpack on a host computer

2.2. Update ROS, Nvidia Jetpack, and Obitty BSP on the Jetson device by using a monitor, a keyboard and a mouse;

2.3 Setup and test WiFi access with static IP on the wireless routers

Your report shall include answers to the following questions:

Q2.1 How long did it take to install ROS, flash Orbitty, and set up wireless network, respectively? Give a rough estimate of time spent on those tasks. Show the results of the installations in your report.

* Installing ROS: 30 mins to download, 30 mins to enter all of the commands to install.
* Flashing Orbitty: this was a multi-day process due to many errors. Installing Jetpack 4.4.1 and extracting the CTI-L4T tool was completed without issue, however, running the ./flash.sh never seemed to complete without error. The errors we always got were: BootRom not running and CPU Bootloader is not running. Googling these errors did not yield any immediate fixes unfortunately.
* <https://forums.developer.nvidia.com/t/flash-sh-failing-with-cpu-bootloader-is-not-running-on-device-message-with-jetpack-4-2/72762/10>
* This forum post pointed to a bad install of Jetpack as a potential source of error, so we uninstalled and reinstalled Jetpack to attempt again. Flashing just Jetpack using the SDK manager was unsuccessful as well, it seemed to not detect the Jetson via usb.
* We have uninstalled and reinstalled SDK manager and Jetpack 3 times with no success. Our new plan is to flash the TX2 with another computer.
* As a result, setting up a wireless network has not been attempted yet.

Q2.2 If you tried other installation methods, then report your experience even if not successful.

* Since the CTI tool needed Jetpack 4.4.1, we did not attempt any alternative installation methods.

Q2.3 What problems did you encounter working through the software installation or update? What did you learn or enjoy most in Part 2 of Lab 1? What suggestions do you have for improving the instructions or procedures?

* Dual booting was pretty fun

No code or file submission is required for Part 2 of Lab 1. If you took or made any videos or document that may help others with the software installation procedures, then submit them.

**Part 3. Linux Tutorial**

If you are not familiar with Linux command lines, please go through the tutorial here:<https://ubuntu.com/tutorials/command-line-for-beginners>. It takes ~50 minutes for a beginner to work through it. I suggest you also practice effortful retrieval (<http://sites.gsu.edu/scholarlyteaching/effortful-retrieval/> ) with your teammate. Your time is well spent on these tutorials since we will be using the basic commands for the rest of the semester. Also refer to the more in-depth tutorial at [http://linuxcommand.org](http://linuxcommand.org/) and the basic Linux/Unix command list here<https://files.fosswire.com/2007/08/fwunixref.pdf> . At minimum, you shall know how the following commands/key combinations work:

Your report shall answer the following questions:

Q3.1 How do you open a new terminal in Linux?

A new terminal can be opened by pressing Ctrl+Alt+T on the keyboard, or by clicking the app drawer in the lower left corner and selecting “Terminal”.

Q3.2 What text editing tool do you use on Linux? How do you like it?

Different text editing tools are better suited for different applications, but the primary text editor that I use is gedit for general tasks. Additionally, for quick edits I sometimes use nano, and I occasionally use vim when debugging source code due to its tools in seeking to specific code entries

Q3.3 What does “sudo” command do?

Linux contains a permission system to protect certain privileged parts of the operating system that an adversary could potentially use as an attack vector. The “sudo” command allows the command following it to be run at the highest level of privilege. For example, “sudo ls” can list the contents and access properties of any directory on the system, including directories that are owned by other users.

Q3.4 How do you check and change the access property (r,w,x) of a file? Show a result of your exercise;

To check the access properties of a file, the command “ls -l” is used. For example, when “ls -l” is run from the home directory, the following is returned:

total 60

drwxr-xr-x 5 nick nick 4096 Feb 16 16:01 catkin\_ws

drwxr-xr-x 2 nick nick 4096 Feb 11 04:31 Desktop

drwxr-xr-x 2 nick nick 4096 Feb 10 19:45 Documents

drwxr-xr-x 3 nick nick 4096 Feb 10 21:34 Downloads

-rw-r--r-- 1 nick nick 8980 Feb 10 19:30 examples.desktop

drwxr-xr-x 2 nick nick 4096 Feb 10 19:45 Music

drwxr-xr-x 3 nick nick 4096 Feb 10 21:34 nvidia

drwxr-xr-x 2 nick nick 4096 Feb 10 19:45 Pictures

drwxr-xr-x 2 nick nick 4096 Feb 10 19:45 Public

drwxr-xr-x 3 nick nick 4096 Feb 23 15:40 snap

drwxr-xr-x 5 nick nick 4096 Feb 23 15:34 team8\_ws

drwxr-xr-x 2 nick nick 4096 Feb 10 19:45 Templates

drwxr-xr-x 2 nick nick 4096 Feb 10 19:45 Videos

In the second line, there is a “d” in the first character indicating that “catkin\_ws” is a directory. The following nine characters indicate the access properties. The first three characters “rwx” indicate that the owner of the file (“nick” in this case as indicated by the username after the access permissions) has read, write, and execute permissions on the directory. The second group of three letters “r-x” indicates that the group of the file (“nick” in this case as indicated by the second group name listed after the access permissions) has read and execute permissions on the directory, but not write permissions. The third group of three letters “r-x” indicates that any other user or group on the system has read and execute permissions. Note that, for directories, the access properties apply only to the directory itself and not any of its contents. These will have their own access properties.

To change the access properties, we use the command “chmod xxx <somefile>”. The “xxx” can be one of two things. It can either be three numbers indicating the octal representation of the nine access property bits discussed earlier, or it can take the form of:

<o=owner, g=group, a=all><+ or -><r=read, w=write, x=execute>

to add or revoke a specific permission to the owner, group, or everyone. For example, “chmod o+x <somefile>” adds execute permissions to the owner of that file. If you do not have write permission for the file or folder you are attempting to change, “sudo” will need to be used.

Q3.5 What is a shell or shell script?

A shell is the command syntax that is used for the terminal input. For the case of Ubuntu, the “bash” shell is used, but other shells also exist and can be installed if desired. A shell script is a set of shell commands

**Part 4. ROS Tutorials:**

Work through the ROS tutorials for Beginner Level here:<http://wiki.ros.org/ROS/Tutorials>. Specifically,

You and your partner shall run the tutorial on a host computer and the Jetson device, respectively. Make sure you go through Parts 1-5 of the tutorial:

[1. Installing and Configuring Your ROS Environment](http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment): This tutorial walks you through installing ROS and setting up the ROS environment on your computer and shall be done in Part 2 of Lab 1.

2. [Navigating the ROS Filesystem](http://wiki.ros.org/ROS/Tutorials/NavigatingTheFilesystem): This tutorial introduces ROS filesystem concepts, and covers using the roscd, rosls, and [rospack](http://wiki.ros.org/rospack) commandline tools.

3. [Creating a ROS Package](http://wiki.ros.org/ROS/Tutorials/CreatingPackage): This tutorial covers using [roscreate-pkg](http://wiki.ros.org/roscreate) or [catkin](http://wiki.ros.org/catkin) to create a new package, and [rospack](http://wiki.ros.org/rospack) to list package dependencies. We use catkin in our course.

4. [Building a ROS Package](http://wiki.ros.org/ROS/Tutorials/BuildingPackages): This tutorial covers the toolchain to build a package.

5. [Understanding ROS Nodes](http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes): This tutorial introduces ROS graph concepts and discusses the use of [roscore](http://wiki.ros.org/roscore), [rosnode](http://wiki.ros.org/rosnode), and [rosrun](http://wiki.ros.org/rosrun) commandline tools.

Your report shall demonstrate/state that both team members work through the tutorials. Make sure to also answer the following questions.

Q4.1. What are the three layers of ROS hierarchy? What are the contents in each of the layers in the turtlesim example?

The three layers of the ROS hierarchy are nodes, packages, and workspaces. Nodes are programs that run as a single process and have a particular purpose. Packages are groups on nodes. Workspaces are directories. They are called catkin workspaces in ROS. (Referenced Module A-4 slides)

Q4.2. How are ROS filesystem tools different from Linux filesystem tools?

Navigating the filesystem uses similar but different commands. For example, instead of ‘ls’ and ‘cd’ as in Linux, to navigate the ROS system ‘rosls’ and ‘roscd’ are used. There is another tool called ‘rospack’ which can be used to find information about ROS packages. In the tutorial we used the ‘find’ function of ‘rospack.’

Q4.3. What is a topic, a message, a publisher, or a subscriber in the turtlesim example?

According to (Module A-4 slides), the four items are defined as follows. A topic is a channel where messages can be passed between nodes. A message is the data passed. A publisher sends messages, and a subscriber receives messages. Both publishers and subscribers are nodes.

In the turtle example, the topic allows the nodes ‘turtlesim\_node’ and ‘turtle\_teleop\_key’ to communicate. The ‘turtle\_teleop\_key’ is the publisher because it sends data containig keystrokes to move the turtle. The ‘turtlesim’ is the subscriber which receives this data. The keystrokes themselves are the message.

Q4.4. How many terminals does the turtlesim example use? What are they? How to exit a ROS process without closing the terminal?

The examples uses \_\_\_\_ terminals. They are: \_\_\_\_\_. Pressing Ctrl+C will exit the process without closing the terminal.

Q4.5 In which directory would you run catkin\_make? What are the files and folders generated by catkin\_make in the beginner\_tutorials example? What is a CMakeLists.txt? What is the difference between the CMakeLists.txt in ROS and a make file used for compiling C++ objects?

Q4.6. What are the $source commands in the tutorials? What is the significance of sourcing the setup files? (or What does source do? What happens if not sourcing? )

Q4.7. What problems did you encounter working through the two tutorials? Did you watch the tutorial video and/or work through the original ROS tutorials? If so, report your experience: how are they helpful to your learning?

We had trouble getting the plot to work using the command given in the tutorial, but found a solution in the ROS forums. https://answers.ros.org/question/59251/rqt\_plot-not-plotting-data/

Screen capture a turtlesim run in a video and submit it as your code submission for Part 4 of Lab 1.