2.12: Introduction to Robotics
Lab 1: ODrive Motor Drivers*

Spring 2020

Instructions:

1. **When your group is done with each task, call a TA to do your check-off.**
2. **After the lab, zip your files and make a backup using online storage/flash drive.**
3. **No need to turn in your code or answers to the questions in the handout.**

1 Objectives

In this lab, you will learn how to interface with **Harmonic Drive motors** using **ODrive motor driver boards**. You will primarily be using the terminal on an Ubuntu 16.04 machine. At the end of the lab, you will edit a Python script to run your specific motor.

Harmonic Drive motors are brushless DC motors with **zero-backlash** gear units. This is done using strain wave gearing which, in addition to zero-backlash, allows for high gear ratios and excellent repeatability.

ODrive motor driver boards are open source motor driver boards built to handle brushless DC motors. They have tons of features including handling of high peak current, a straight forward python interface, and several different control protocols. It should be noted that other brushless DC motor driver boards do exist and this will serve as good foundation for learning how to use others.

Batteries used for this lab are high power lithium polymer batteries. As such, they require special safety precautions to be used and are described below.

It should be noted that the installation of odrivetool used in this lab is specific to allow for ROS integration as the default odrivetool is made for Python 3.X but ROS utilizes Python 2.X.

2 ODrive Motor Driver Walkthrough

In this section, we will walk you through the process of connecting an ODrive to your computer and running a motor with it. This will include wiring/assembly, connecting motors to the driver, and writing to the motor driver via your computer. For incorporating an ODrive motor driver into Python and ROS, you will be given example code available on the MIT212 Github.

If you need to review this material later or what an in depth explanation of ODrives, go to the ODrive motor driver website.[1]

---

*

1. Version 1 - 2020:Rachel Hoffman-Bice, Jerry Ng and and Kamal Youcef-Toumi

2.1 ODrive Wiring

Note: Make sure to wire your motor into M1. This will allow you to just copy the commands used in future sections directly.

Begin by screwing in the motor phases into the 3-phase screw terminals. Afterwards, connect the encoders to the corresponding terminals. Next, insert the power cables into the screw terminals. **DO NOT HAVE THESE CABLES PLUGGED INTO THE BATTERY YET. DOUBLE CHECK EVERYTHING.**

You will be hooking up the optical encoder to the motor driver board. The black wire should be connected to ground, and all the other cables will be plugged in series following the black. An example will be provided. After connecting everything, your setup should look similar to the following:

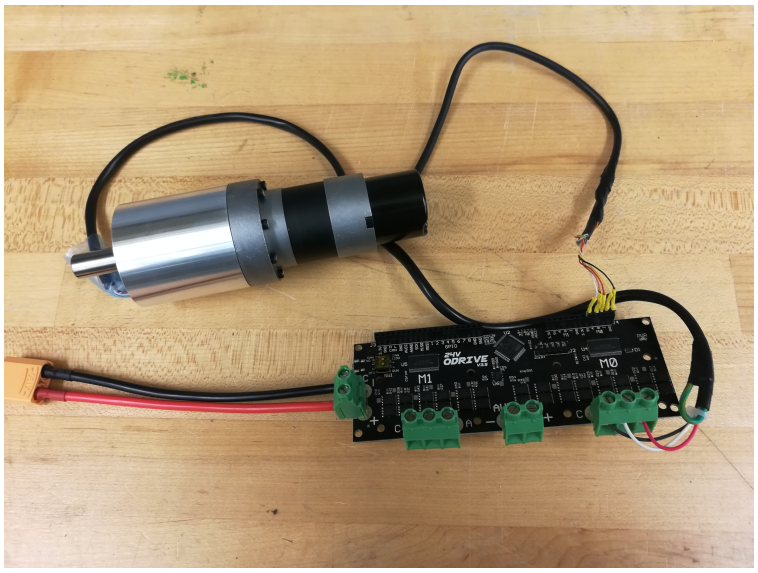

Figure 1: An example of what your wiring setup should look like. You can see this example on the back table in lab.

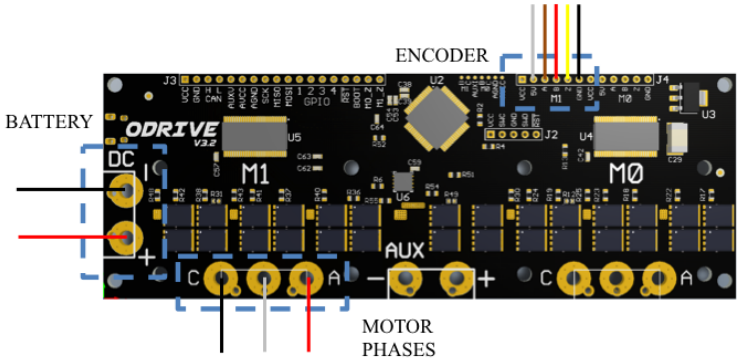

Figure 2: The wiring schematic for M1. Make sure to double check wiring before plugging in the power.

ODrives can handle 12-24V power sources easily. Keep in mind that ODrives do not use the USB as a power source, so the driver will have to be powered through the 24V power line to be able to communicate with your computer.

2.2 Getting Started with ODrive Tool

First, connect the ODrive to your computer via USB. Then, begin by opening a terminal and running the following command:

```
odrivetool shell
```

This will launch the main interactive ODrive Tool in your terminal. The following message will appear in your terminal:

```
ODrive control utility v0.4.12
Please connect your ODrive.
Type help() for help.
```

It will take a bit of time while your computer searches for the ODrive. After it has registered communication with ODrive Tool, the following message will appear:

```
Connected to ODrive 306A396A3235 as odrv0
In [1]: odrv0.vbus_voltage
Out[1]: 11.97055721282959
```

The ODrive Serial number will be shown, in this case it is 306A396A3235. This will come in handy in the future when you implement your robot code in Python. At this point, you can run a command such as:

```
odrv0.vbus_voltage
```

This should return a voltage that corresponds to what your ODrive motor driver is powered by.

2.3 Configuring a Motor

Note: This section requires your motor to be plugged in to Axis 1 (M1). If your motor is plugged in to Axis 0 (M0), then you will need to replace '1' with '0' in all the following commands.

To begin the configuration of the motor for this lab, you'll want to make sure that your ODrive motor driver run the following command:

```
odrv0.erase_configuration()
odrv0.save_configuration()
odrv0.reboot()
```

This will look like the terminal has had an error, but that is just an unhandled break in the connection. If you wait for a moment, the ODrive will reconnect. Should you run into errors with your driver board for whatever reason, you can run those commands in that sequence to reset the configuration to default settings.

In the rest of the lab, you will be updating the ODrive to work with the Harmonic Drive motor that you've plugged in. We recommend using the following settings as they are directly from the motor spec sheet. You should not need to modify the configuration for this course unless you use a different motor. This will correspond to the following commands:

```
odrv0.config.brake_resistance = 0
odrv0.axis1.controller.config.vel_limit = 200000
odrv0.axis1.motor.config.calibration_current = 4
odrv0.axis1.motor.config.pole_pairs = 4
odrv0.axis1.motor.config.motor_type = MOTOR_TYPE_HIGH_CURRENT
odrv0.axis1.encoder.config.cpr = 4000
odrv0.axis1.encoder.config.use_index = True
odrv0.save_configuration()
odrv0.reboot()
```

Rebooting the ODrive allows the save command to be used again after inputting other commands. This is a bug that occurs fairly readily, so keep that in mind. If you have any additional questions about the units/parameters, feel free to ask or search for documentation on the ODrive website.

Next, we will be calibrating the motor. This requires you to run the following command:

```
odrv0.axis1.requested_state = AXIS_STATE_FULL_CALIBRATION_SEQUENCE
```

This calibration sequence is made up of several parts.

1. **System Identification**: A frequency sweep of the motor is completed to determine motor characteristics. The frequency sweep is composed of a sinusoidal voltage of varying frequencies characterize the motor's electrical properties. Based on the motor response it creates a current controller for the system. This is what happens when you hear the beep.

2. **Encoder Calibration**: It does a search to find the pin then it does an actual calibration to find the direction for the commutation process of the motor. This is what happens when you hear a click sound, and see the motor move slightly in both directions.

If there isn't a beep, or you're not sure if the motor calibrated properly, run the following command:

```
dump_errors(odrv0)
```

There will be a readable list of outputs that appears on the screen after that. If an error has occurred, you'll need to redo the process of calibration. If the calibration doesn't run, restart the motor driver board by disconnecting it and reconnecting it to your computer.

Now that the motor has been calibrated and the controller has been set, we will manually update the settings, save the configuration, and once again reboot the driver.

```
odrv0.axis1.motor.config.pre_calibrated = True
odrv0.save_configuration()
odrv0.reboot()
```

Question 1 *Now that the motor has been rebooted, will we be able to control the motor immediately after reconnecting? Why? Look at the next section after discussing this question with your lab partner.*

Since the ODrive has been rebooted, you will need to run encoder calibration again before being able to drive the motor properly. This is required to address the commutation process for the motor, which was described in lecture.

However, the motor has been calibrated, which means that we do not need to run a full calibration. Instead, in this section, you will run commands for only the encoder calibration.

First we will check if there are any errors.

```
dump_errors(odrv0)
```

Next, we will perform the encoder index search:

```
odrv0.axis1.requested_state = AXIS_STATE_ENCODER_INDEX_SEARCH
```

You'll know this process has concluded as you will hear a click sound.

Afterwards we will perform an encoder offset calibration:

```
odrv0.axis1.requested_state = AXIS_STATE_ENCODER_OFFSET_CALIBRATION
```

This offset calibration finds the **offset** between the motor phase and the encoder position. This encoder calibration process must be ran before using the motors every time.

Now that the motor is completely calibrated, you can input the following commands to run position control on the motor.

```
odrv0.axis1.requested_state = AXIS_STATE_IDLE
odrv0.axis1.controller.config.pos_gain = 0.1
odrv0.axis1.controller.config.vel_gain = 0.0001

odrv0.axis1.requested_state = AXIS_STATE_CLOSED_LOOP_CONTROL
odrv0.axis1.controller.pos_setpoint = 100000
odrv0.axis1.controller.pos_setpoint = 0
```

By setting the state to idle, your motor will not move while you are programming in the gains. While these gains will work for this demonstration, you will definitely have to properly tune your controller at a later date. After running these commands, you will see the motor rotate. The setpoint is in counts and does not take into account the gear ratio.

You are likely thinking that this is too clunky of a process to be done manually every time a robot is powered on. Thus, the next section that will be covered is the Python driver that has been created to work with the ODrives for this lab.

2.4 ODrives with Python

You may have noticed that the odrivetool shell syntax is very similar to that of using a class in Python. That is very much intentional as the ODrive package in Python uses the exact same syntax. To get a feel for how this works, run the following commands to download the driver from github.

```
git clone https://github.com/mit212/lab1odrive
```

After the file has completed downloading, run the following commands and follow the instructions in the terminal.

```
cd lab1odrive
python2.7 odrive_demo.py
```

Question 2 *What did you notice about the motions of the motor through the three different trajectories? Can you describe what was happening in the three different modes in terms of control? Try drawing it as a block diagram.*

References

[1] Getting started with odrive. [Online]. Available: https://docs.odriverobotics.com