Introduction to the BARC vehicle

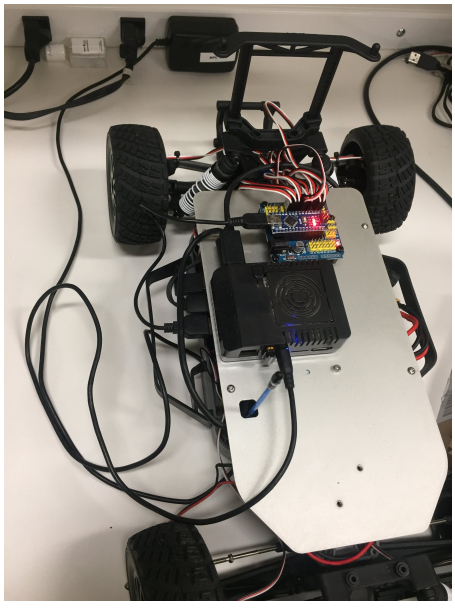
This lab introduces the BARC vehicle, which you will use for the remainder of the course. The BARC (Berkeley Autonomous Race Car) vehicle is a $\frac{1}{10}$ scale RC car with an embedded Linux computer developed in the UC Berkeley MPC lab. It was designed to be a fully open-source development platform for autonomous driving to achieve complex maneuvers such as drifting, lane changes, and obstacle avoidance. The vehicle comes equipped with a suite of sensors and runs on a ROS based framework.

More information, including examples of previous projects using BARC, can be found on the **BARC website** and the **BARC GitHub repo**.

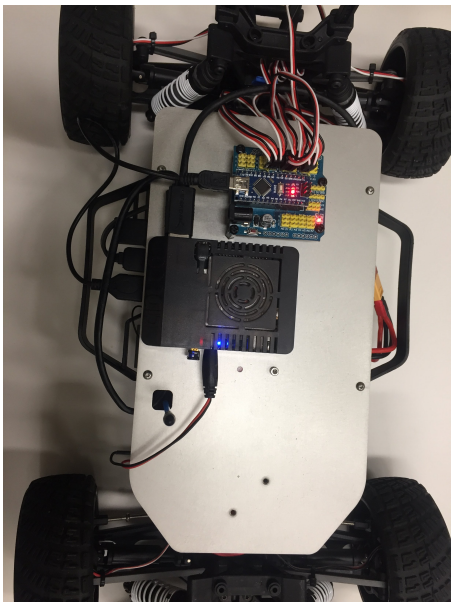
Task 1 Turning BARC on

1. First, power the Odroid with the wall adapter (**make sure to use the correct adapter 5V/4A!!**) (a)
   - a) connecting the Odroid to an outlet with the wall plug-in (see Fig. 12a)
   - b) connecting the LiPo battery output to the Odroid (**ONLY USE BATTERY POWER WHEN READY TO RUN THE CAR ON THE GROUND**) (see Fig. 12b)

   When the Odroid is powered, the red LED will light up and a blue LED will blink. The Arduino will also turn on. (Note that if you are using the BARC battery to power the Odroid, the Odroid LED will only light up after you close the electronics circuit in the next step.)



(a) Using wall plug-in



(b) Using BARC battery output

Figure 1

2. Next, power the vehicle's actuators by connecting the power cables together.

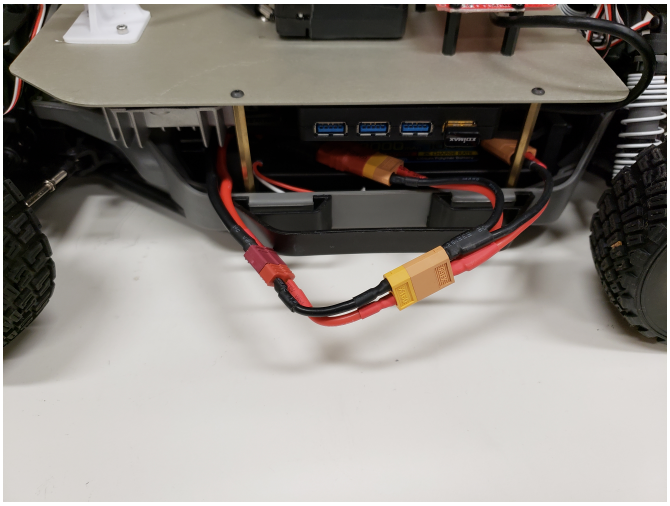
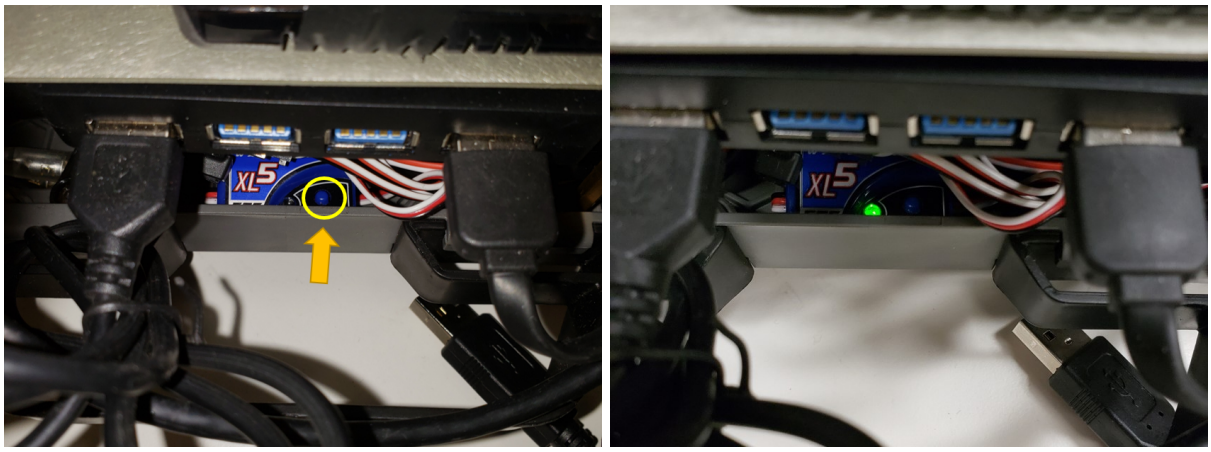


Figure 2

3. Locate the BARC's ESC (Electronic Speed Control), the blue device mounted on the right side of the vehicle's chassis. Turn on the servo and motor by **pressing the small button** (circled in yellow) for two seconds and then letting go. The LED should light up solid green.



(a) ESC off (b) ESC on

Figure 3

4. Connect the voltage monitor with the black wire on the outermost pin. It will beep loudly and some numbers will flash. Right after it says "ALL", a number around 7-8V will appear. Then, "-1-" followed by a 3.5-4.2V. Finally, "-2-" followed by a 3.5-4.2V and the loop repeats.

   FOR SAFETY PURPOSES, DO NOT LET THE TOTAL VOLTAGE GO UNDER 7V (3.5V PER CELL) AND MAKE SURE EACH CELL IS WITHIN 0.1V OF EACH OTHER!

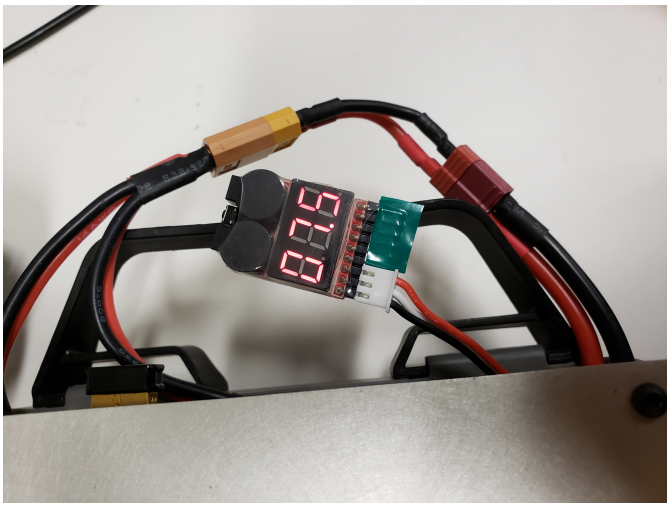


Figure 4

   If the cells are not balanced, that can damage the battery.

5. Your BARC vehicle should now be powered and ready for use! **The LiPo battery lasts 1-2 hours at full charge depending on power consumption so make sure to keep an eye on the voltage level and charge it!** See later section on proper charging practices.

Task 2 Connecting to the BARC vehicle

via Direct Connection

1. Connect the Odroid to a monitor using an HDMI cable. Fig. 5 shows the Odroid's HDMI port.

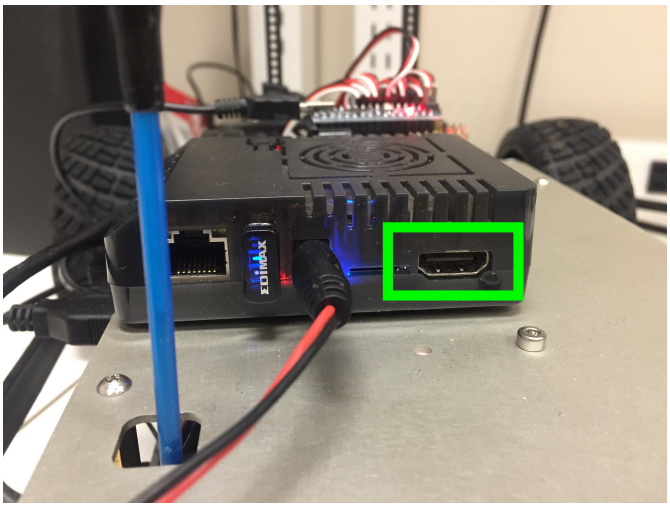


Figure 5

2. Connect a portable mouse and keyboard to any of the BARC's available USB hubs.
3. You should now be able to see the Odroid's operating system on your monitor, and be able to interact with the repositories using your mouse and keyboard (like you would with your own computer).

via Wireless Connection

- After approximately 1 minute of powering on (the wifi dongles should light up with a blue LED), you should be able to see your BARC wi-fi network (`barc-traxxas-01`) on your laptop's list of nearby wi-fi networks.
- Connect to the `barc-traxxas-01` wi-fi, using the password 1234567890
  Note that it may take a while to connect.
- Next, you need to download software that will allow you to visualize the Odroid's list of available networks.
  - For Mac users: install the XQuartz dependency on your machine, available for download **here**.
  - For Windows users: install MobaXTerm on your machine, available for download **here**.
- SSH (using MobaXTerm for windows, or terminal ssh for Mac) to barc: ssh X odroid@10.0.0.1
  This should automatically launch XQuartz in the background.
  - For Mac users: Open a terminal, and type
    `ssh -X odroid@10.0.0.1`
  - For Windows users:

1. Open your new MobaXTerm application.
2. Begin a new session by clicking "Session" in the top left corner of the window.
3. Select the "SSH" tab.
4. Enter `10.0.0.1` into the remote host field.
5. Click the checkbox next to "Specify username", and enter `odroid`
6. Click "OK"
7. This will launch an `ssh` connection. When you are prompted for the password, enter `odroid`

If you receive an error that the `host is down`, re-connect to your BARC wi-fi network and try again.

You can use `ssh-keys` to automatically connect to the Odroid (as opposed to typing in your credentials each time you log on). Enter the following commands in the terminal **on your machine** (i.e. NOT within `ssh`):

```
$ ssh-keygen
$ ssh-copy-id odroid@10.0.0.1
$ ssh-keyscan -H odroid, 10.0.0.1 >> ~/.ssh/known_hosts
```

Check that your `ssh-keys` were correctly set up by connecting to the Odroid using the following command

```
$ ssh -X odroid@10.0.0.1
```

You should not be prompted for a password. The "-X" tag enables X11 forwarding, which enables you to view graphics likes plots and figures.

- VNC Viewer

  VNC Viewer is a remote desktop access software that will allow you to better visualize the Odroid's processes and control algorithms on your laptop, rather than relying solely on the terminal's command line interface.

  1. Download the VNC Viewer program corresponding to your laptop's operating system by clicking **here**.
  2. In VNC Viewer, click **File** – **New Connection**.
  3. In the VNC server prompt, enter your Odroid's IP address: 10.0.0.1 and click **Ok**. You should now see the BARC's Odroid screen on your laptop, and be able to control it using your regular mouse and keypad. Make sure that you are connected to your car's wifi.
  4. If there is an error such as "Incorrect RGB values", ssh into the car and launch a fixing script by typing
     `$ sudo bash ~/barc/scripts/fix_vino.sh`
     When prompted, answer `Y` to agree.

  For future tasks, you will simply need to connect to your BARC's wi-fi network, open VNC Viewer on your laptop, and resume your session.

  Note that you are free to connect to the Odroid using whichever connection method (`ssh`, VNC Viewer, an appropriate alternative method) you prefer.

Task 3 Turning BARC off

1. Shut down the Odroid's operating system by typing
   $ sudo shutdown now
   in your terminal.
2. When the lights have turned off, disconnect your Odroid's power source (either wall adapter or the BARC battery).
3. Turn off the ESC by shortly pressing down the small button. Ensure that the LED has turned off.
4. **Disconnect the battery power cable and the voltage monitor (Fig. 6).** Your battery should be disconnected whenever you are not using the BARC car. The voltage monitor should be disconnected as well since it will drain your battery when plugged in.

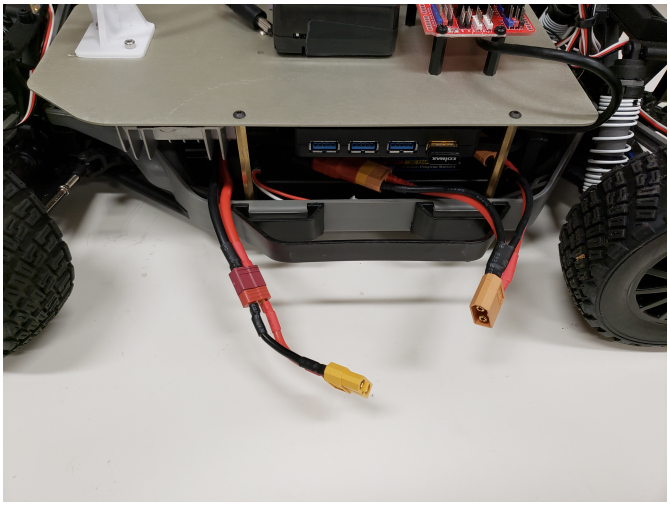


Figure 6: You should **ALWAYS** disconnect the battery power cable when you are not using your BARC car.

Task 4 Cloning your forked GitHub repository (Optional)

The barc repository is already cloned on the Odroid. If you would like to replace it with your own forked version so you can upload changes, you will need to repeat the steps you took in Lab 2 to clone your forked `barc` GitHub repository.

1. In your terminal, enter
   $ rm -rf barc
   This will remove the current, outdated `barc` folder on the Odroid.
2. Clone your forked repository onto the Odroid (as in Lab 2):
   `$ git clone your-forked-http-address`

3. Once again set the original `MPC-Berkeley/barc` repository as the upstream master, so you will be able to pull any updates we publish to the code. **First, move into the new `barc` repository you just cloned using `cd barc`.**
   `$ git remote add upstream https://github.com/MPC-Berkeley/barc`
4. Move into the `barc/workspace` folder, and compile your cloned repository using
   `$ catkin_make`
   Close your terminal in order for the changes to take effect.

Task 5 Changing your wi-fi name

Next, you need to change the name of the wi-fi network your BARC car creates to a name unique to your group's vehicle.

1. In a new terminal, type
   `$ sudo pico /etc/rc.local`
   This will open up the `rc.local` file in the `pico` text editor. (We will use `pico` on the Odroid, as Sublime is not installed.) When prompted about your root folder, answer YES.
2. In the `rc.local` file, edit the line
   `/usr/bin/create_ap g 10.0.0.1 wlan1 wlan0 barc-traxxas-01 1234567890` to
   `/usr/bin/create_ap g 10.0.0.1 wlan1 wlan0 barc-traxxas-YOURGROUPNUMBER 1234567890`
   Please enter your group number as an integer. Do not write YOURGROUPNUMBER.
3. Type `Ctrl-X` to exit out of the `pico` editor. Select YES to save your changes, and press enter.
4. In the terminal, enter
   `$ sudo shutdown now`
   This will initiate a safe shutdown sequence, and you should run this command every time you plan to shut down the Odroid.
5. When the lights have turned off, unplug and replug the Odroid's power cable.

Task 6 Understanding the BARC ROS Architecture

We will be using ROS to interact with and control the BARC vehicle. This section will help you familiarize yourself with the existing BARC source code, so you know what tools are available for you to use without needing to write them yourself.

1. Connect to your BARC using your preferred method (`ssh` or the VNC Viewer)

2. In a terminal, first update the source code on your Arduino by running the command:
   $ flash_nano
   If this gives you an error about unsynchronized clocks, update the date using:
   $ date + %Y%m%d –s "YYYYMMDD"
   replacing the YYYYMMDD with today's date. Then, try the $ flash_nano command again.

3. In a terminal, initiate ROS using the `roscore` command.

4. In a new terminal, initiate communication with the Arduino by typing:
   `$ roslaunch barc arduino.launch`
   This command initiates the node `serial_node.py` from the package `rosserial_python`, and tells the system where the Arduino is located on the operating system. If it says it could not find the file `/dev/ttyUSB0`, re-plug the Arduino's USB into the hub and try again.

5. We can use ROS utilities to explore the topics on the node you launched. In a third terminal window, use the `rostopic list`, `rosnode list` and `rqt_graph` commands. Which sensors are integrated through the Arduino? What is the direction flow of information (i.e. publishing vs. subscribing)?

6. The `imu.launch` file launches the BARC's IMU sensor node. The IMU records linear and angular accelerations. Read through the `imu.launch` file, and make sure you understand what it is doing. Initiate communication with the IMU by typing:
   `$ roslaunch barc imu.launch`
   Use the ROS commands from the previous task to understand the ROS nodes and topics associated with the BARC's IMU sensor.

7. ***Deliverable**: Draw a flowchart of the nodes, topics, and messages types associated with the BARC's IMU and actuators. This will be very helpful moving forward.

Task 7 Sending simple control commands

Now that you have an understanding of how the ROS nodes and topics already available as part of the source code are connected, you can use ROS to send simple control commands to the real BARC vehicle. Because we will only test simple commands for this task, we will not go through the typical process of simulating the behavior in the ROS BARC simulator first. Instead, **please ensure that all four BARC vehicle wheels remain off the ground** for the entire duration of this task (i.e. having a teammate hold the vehicle up or suspending the vehicle's chassis on a box). **Also make sure that the LiPo battery is connected and the ESC is turned on (solid green)**

The BARC has two actuators: a motor and a servo, which control vehicle speed and steering angle, respectively. The motor can rotate continuously forward or backward, or brake. An unattached servo can rotate to various positions between 0 and 180 degrees, but when the servo is attached to the BARC steering system, it is limited to approximately $\pm 45^{\circ}$. Trying to turn beyond this limit may damage the servo, the steering system, or both, so **please keep these limits in mind**.

Both the servo and motor are controlled through pulse width modulation (PWM), and we send actuation commands by writing the corresponding microseconds in the range of $\{1000, 2000\}$.

- Servo: We install the servo such that a command signal of $u_s = 1500$ corresponds to the tires pointing approximately straight ahead. Note there will most likely be a small, vehicle-dependent offset for your straight-ahead direction which you will determine in future labs. Values above 1500 correspond to the wheels turning one direction, and values below correspond to the tires turning the other direction. Since the attached servo is limited to a $\pm 45^\circ$ actuation signal range, we limit our command signal to $u_s \in [1200, 1800]$.
- Motor: For the motor, an input of $u_m = 1500$ corresponds to no torque (i.e. no motion). An input $u_m > 1500$ corresponds to positive torque (i.e. forward rotation), and higher values mean larger torque (i.e. faster forward rotation). An input $u_m < 1500$ corresponds to braking, and lower values mean harder braking. Reverse direction driving is achieved by sending a hard brake signal, but for this lab we will restrict ourselves to forward and neutral modes.

1. ***Deliverable**: Practice sending commands to the BARC actuators. Use what you learned about the ROS architecture on BARC to determine which topic you need to send your command to, and what format your message should be in.
   Recall that we can publish a message of particular type onto a topic using the command
   `$ rostopic pub --once /topic msg_type msg`
   where the `--once` option ensures you only send the actuation command one time. Practice sending messages until you understand the message components – this is critical for future labs. Write down and submit as part of your homework what command you need to type into the terminal in order to steer 30 degrees left at a comfortable forward velocity (use your discretion as to what constitutes a *comfortable* velocity; the exact value does not matter), and describe what each component refers to.

Turn off the ESC and unplug the LiPo battery when you are finished sending commands

---

Task 8 The BARC IMU sensor

We can also use ROS to read sensor signals in real-time. These signals are sent as messages to particular ROS topics from the sensor nodes. Use the appropriate `rostopic echo` command in order to analyze a live-stream of the BARC's IMU signals.

1. ***Deliverable**: Acceleration Measurement: Move your BARC vehicle along the three axes to determine the alignment and positive direction of the x, y, and z axes fixed to the body frame. *Hint:* We highly recommend you use the `rqt` command for visualizing this. Roughly sketch the BARC car, including the acceleration reference frame the BARC car uses, and the acceleration measurements when the car is standing still on the ground. During your experiment, you will notice a large offset in the vertical acceleration measurement. What does this correspond to?
2. ***Deliverable**: Angular Velocity Measurements: Do simple rotations tests and check that roll, pitch, and yaw measured by the gyros in the IMU are relative to the axes you found above. Label their directions on the same plot as your acceleration reference frame. Do the signs follow the right hand rule?

Task 9 LiPo Battery Notes

Lithium ion batteries generate gas during the course of normal operation. This process can be exacerbated by allowing your battery charge to dip below a threshold, so make sure your battery does not ever fully discharge. **DISCONNECT YOUR BARC BATTERY WHENEVER YOU ARE NOT USING YOUR VEHICLE**. If your ESC LED begins blinking, you need to charge your battery.

Eventually, this gas buildup can cause the battery to swell until it no longer looks rectangular, but is bulging on all sides. **If your battery swells up, it is extremely important that you stop using it immediately.** Swollen LiPo batteries are at risk of catching fire. Contact us and we will help you discharge and dispose of it.

Task 10 Checking the LiPo battery charge / Charging the LiPo battery

1. Disconnect the main battery cable from the BARC's electronics circuit. It should now dangle freely off the chassis, as shown in Fig. 6.
2. Power the LiPro charger by connecting it to an outlet.

Figure 7

3. Connect the Deans cable to the red and black leads, as shown in Fig. 8.

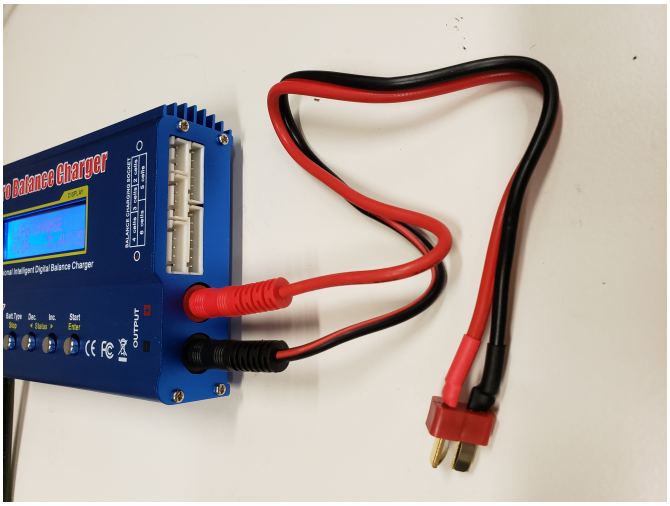


Figure 8

4. Disconnect the "Deans to XT60" cable, as shown in Fig. 9.

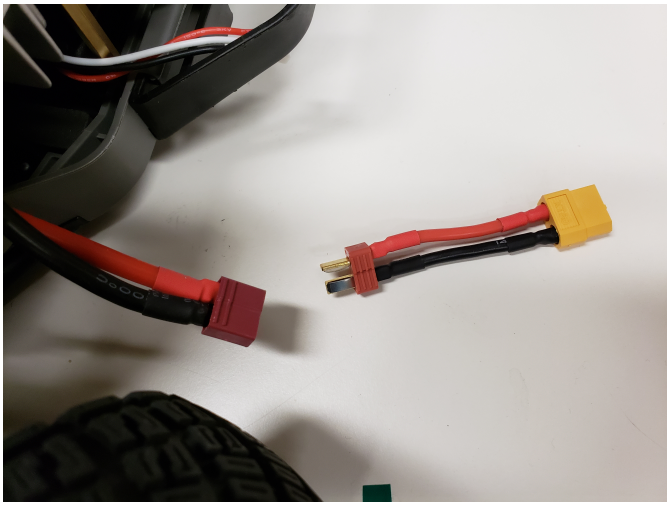


Figure 9

5. Connect the Deans adapter from the LiPro charger to the battery AND the voltage monitor cable to the top right corner of the balance charging sockets (2S for two cell batteries), as shown in Fig. 10.

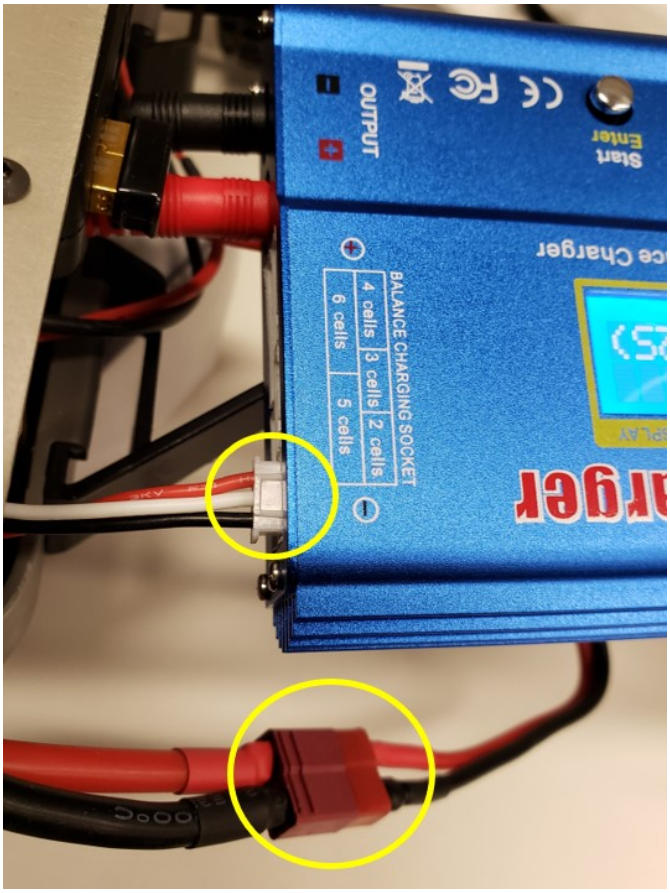


Figure 10

6. When you see the "PROGRAM SELECT" screen, press Start/Enter on the "LiPo BATT" option.

Figure 11

7. If you are using the BARC **IMMEDIATELY** scroll to the right using INC/→ button until you reach the "BALANCE" option. Otherwise if you are saving the BARC for another time, scroll until you reach the "STORAGE" option.

(a) BALANCE charging (b) STORAGE charging

Figure 12

8. Press Start/ENTER once and it will let you modify the charge rate (DEC/← and INC/→ to change the rate) and press Start/ENTER to switch to number of cells (DEC/← and INC/→ to change the number of cells). **Make sure you charge at 2A and 7.4V(2S)**

9. When you are ready to charge, hold the Start/ENTER button until it performs a "BATTERY CHECK".

Figure 13

10. It will ask you to confirm and press START/ENTER again.

Figure 14

11. You will see it start to charge. It will complete around 8.4V total. Make sure to disconnect when it is done charging.

Figure 15

12. Never leave your charging LiPo battery unattended!

13. When you are finished charging (it should say charge complete), disconnect the LiPo battery cable from the LiPro charger. Only reattach the main battery cable to the BARC's electronics circuit if you are going to be using the BARC car immediately afterwards.