Propbot Handover Doc

Project :	Propbot Autonomous Robot
Authors :	Team- 50,
	Amr Almoallim (AA) - 83386714,
	Apoorv Garg (AG) - 39485545,
	Cole Shanks (CS) - 54950860,
	Johanan Agarwal (JA) - 29188166,
	Sajjad Al-Kazzaz (SA) - 23401565
Affiliation :	UBC Radio Science Lab

General Status of Propbot

As of April 2022, Propbot is operable in Teleoperation mode, meaning it can reliably send and receive commands from an RC controller and based on those, a user can have fine control over its movements. The autonomy side is set up, and has been validated through simulation, although it is still missing important components like LiDAR and (possible) wheel feedback. The hardware interface between the autonomy and motor controllers only executes limited control of the robot. Powering and wiring of the robot and its components has been implemented and is easily extensible.

Connecting to Autonomy Computer

There are several ways to connect to the Autonomy Computer: serial, ethernet, WiF...

In order to run simulations or Propbot, an ethernet,WiFi, or cellular network needs to be established. Nevertheless, being able to connect serially is helpful for debugging or for setting up other kinds of connections.

Nvidia Jetson NX serial connection tutorial: follow this tutorial for the jetson Nano, replacing each pin with the corresponding one the Jetson NX. (https://www.jetsonhacks.com/2019/10/10/jetson-nano-uart/)

To connect via ethernet, plug the ethernet cable between your laptop and the NX. Use “ifconfig” to verify if a local access network has been established and if your laptop has been given an ip address over the ethernet interface. To view what other devices (like the NX) are connected on that LAN, use the command “*nmap 10.42.0.*”* (replacing the first 3 numbers with the first three from your ip address).

To connect via WiFi, use a router as an access point. On the Nano, connection to a router can be done via the command line using the nmcli command. For example, the command:

sudo nmcli dev wifi connect network-ssid password "network-password"

Can be run to connect to a network. To make the NX automatically reconnect, use:

$ nmcli device set IFNAME autoconnect yes

Where “device” is replaced by the appropriate interface from the output of the command

$ Nmcli device

Note that if you are using a router with no interneti you might need to purchase a WiFi dongle in order to access both internet and the router. Internet is needed for Mission Command Centre viewing of map.

GPS Resources

To setup the Ublox GPS, make sure to download the correct version of U-Center (NOT U-Center2). For some reason, the most recent version of U-Center was not compatible with my version of windows 10, so i had to install U-Center v20.2.

To work with U-Center and the GPS, these resources were helpful:

https://content.u-blox.com/sites/default/files/C94-M8P-Appboard-Setup_QuickStart_%28UBX-16009722%29.pdf	For help with deploying RTK system
https://learn.sparkfun.com/tutorials/setting-up-a-rover-base-rtk-system/all	For general understanding of RTK and specific details on working with U-Center

ROS Helpful Commands

Command	**Description**	**Example Usage**
Roscore	Launches a ROS master server on the current device. A ROS master server must be launched for any communication between nodes to take place	As is

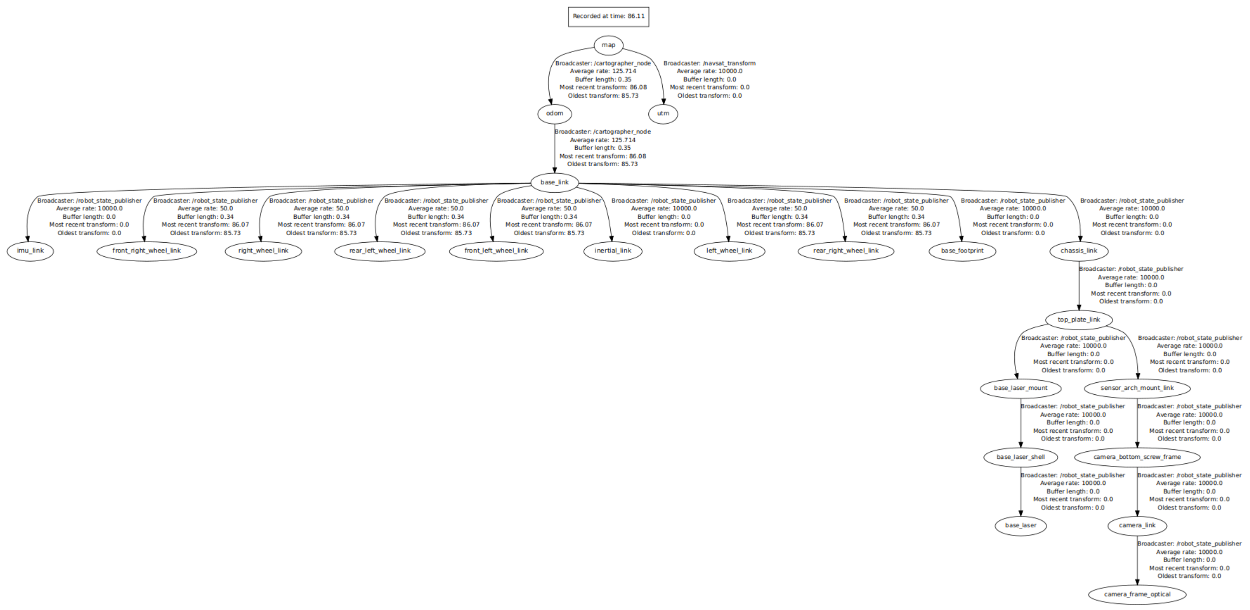
Rostopic	This shows information about ROS topics	$ rostopic <subcommand> <topic name> Subcommands: list, echo, info, and type

Rosnode	This shows information about nodes and lists the active nodes	$ rosnode info [node name]
		
		$ rosnode <subcommand> [node name]
		
		Subcommand: list, info

Rosservice	Displays the runtime information about various services and allows the messages to be sent to a topic	$ rosservice <subcommand> [service name] Subcommands: args, call, find, info, list, and type
Rospack list	Lists all ROS packages that can be found on the system	As is
rqt_console	Opens a log console where logs can be filtered and highlighted	As is
rosrun tf view_frames	Creates a pdf showing current tf tree (see "TF Tree" section)	As is
roswtf	Scans all nodes and topics and provides a list of things that might be wrong	As is

TF Tree

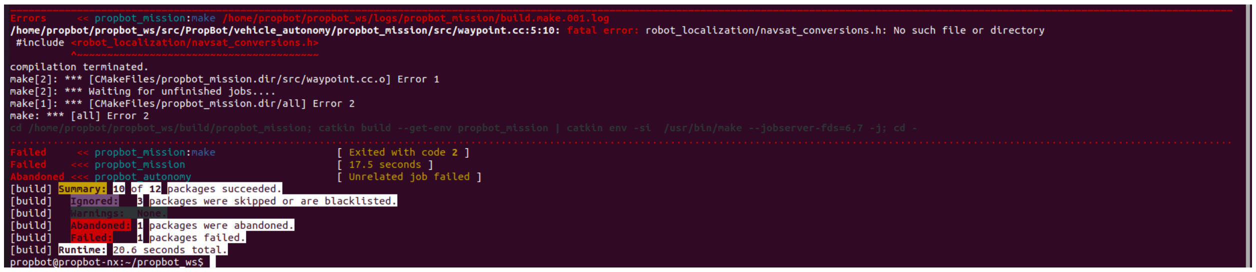
Tf Tree when running simulated tests:



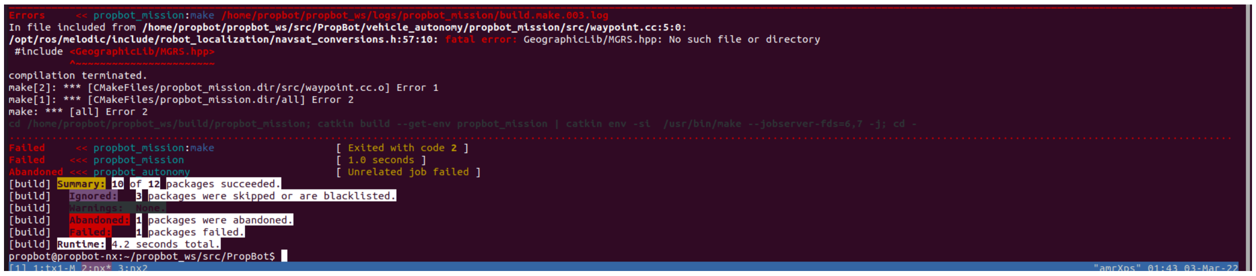
.dot version of tree is uploaded to GitHub under the name "tf_tree_cartographer.dot".

Errors Encountered

This is a non-exhaustive collection of building errors that we encountered when setting up the autonomy system for the first time. They should be addressed in our setup script, but we showcase them here just in case.



```
Errors     << propbot_mission:make /home/propbot/propbot_ws/logs/propbot_mission/build.make.001.log
/home/propbot/propbot_ws/src/PropBot/vehicle_autonomy/propbot_mission/src/waypoint.cc:5:10: fatal error: robot_localization/navsat_conversions.h: No such file or directory
 #include <robot_localization/navsat_conversions.h>
          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
compilation terminated.
make[2]: *** [CMakeFiles/propbot_mission.dir/src/waypoint.cc.o] Error 1
make[2]: *** Waiting for unfinished jobs....
make[1]: *** [CMakeFiles/propbot_mission.dir/all] Error 2
make: *** [all] Error 2
cd /home/propbot/propbot_ws/build/propbot_mission; catkin build --get-env propbot_mission | catkin env -si  /usr/bin/make --jobserver-fds=6,7 -j; cd -
...............................................................................................................................................
Failed     << propbot_mission:make                    [ Exited with code 2 ]
Failed    <<< propbot_mission                         [ 17.5 seconds ]
Abandoned <<< propbot_autonomy                        [ Unrelated job failed ]
[build] Summary: 10 of 12 packages succeeded.
[build]   Ignored:   3 packages were skipped or are blacklisted.
[build]   Warnings:  None.
[build]   Abandoned: 1 packages were abandoned.
[build]   Failed:    1 packages failed.
[build] Runtime: 20.6 seconds total.
propbot@propbot-nx:~/propbot_ws$
```



```
Errors     << propbot_mission:make /home/propbot/propbot_ws/logs/propbot_mission/build.make.003.log
In file included from /home/propbot/propbot_ws/src/PropBot/vehicle_autonomy/propbot_mission/src/waypoint.cc:5:0:
/opt/ros/melodic/include/robot_localization/navsat_conversions.h:57:10: fatal error: GeographicLib/MGRS.hpp: No such file or directory
 #include <GeographicLib/MGRS.hpp>
          ^~~~~~~~~~~~~~~~~~~~~~~~
compilation terminated.
make[2]: *** [CMakeFiles/propbot_mission.dir/src/waypoint.cc.o] Error 1
make[1]: *** [CMakeFiles/propbot_mission.dir/all] Error 2
make: *** [all] Error 2
cd /home/propbot/propbot_ws/build/propbot_mission; catkin build --get-env propbot_mission | catkin env -si  /usr/bin/make --jobserver-fds=6,7 -j; cd -
...............................................................................................................................................
Failed     << propbot_mission:make                    [ Exited with code 2 ]
Failed    <<< propbot_mission                         [ 1.0 seconds ]
Abandoned <<< propbot_autonomy                        [ Unrelated job failed ]
[build] Summary: 10 of 12 packages succeeded.
[build]   Ignored:   3 packages were skipped or are blacklisted.
[build]   Warnings:  None.
[build]   Abandoned: 1 packages were abandoned.
[build]   Failed:    1 packages failed.
[build] Runtime: 4.2 seconds total.
propbot@propbot-nx:~/propbot_ws/src/PropBot$
[1] 1:tx1-M 2:nx* 3:nx2                                                                                        "amrXps" 01:43 03-Mar-22
```

To fix "robot_localization" / Geographic/MGRS.hpp error when trying to build simulation/autonomy computer, need to build robot_localization from source:

- Sudo apt-get remove ros-melodic-robot-localization
- In 3pp workspace directory, git clone git@github.com:cra-ros-pkg/robot_localization.git
- Change branch to kinetic branch , or use this hash: 9327ee3c7c80ab35abab3bc0172a2d7dc3d52749
- In 3pp directory, run catkin build robot_localization
- In the file: vehicle_autonomy/propbot_mission/package.xml , add the following line : <build_depend>robot_localization</build_depend>
- In the file: vehicle_autonomy/propbot_mission/CMakeLists.txt , add the line in the find_package function:

  Robot_localization

Current Wiring

In this section I will be explaining the power management system and power delivery cabling in detail. Our team purchased a 36V 26.5Ah downtube battery for the purposes of powering Propbot. This battery has an Anderson powerpole end connection. Anderson powerpole are versatile, high voltage and current connections that slide into eachother. On Propbot is a distribution terminal block in the front chassis.

This distribution block has an Anderson powerpole input connection meant for battery input. This distribution block also carries power to the motor controller terminal blocks in the center part of the chassis. In series with the distribution – motor controller terminal block connection is the E-stop and a 40A fuse. The E-stop is connected with higher gauge cabling then the rest of the system (8 AWG) since the E-stop button has to be fairly far away from the terminal blocks for accessibility reasons. The fuse is in place to protect the controller from current overages. In the case that the fuse gets blown or needs to be replaced with a different rating there is a fuse kit with the extra components.

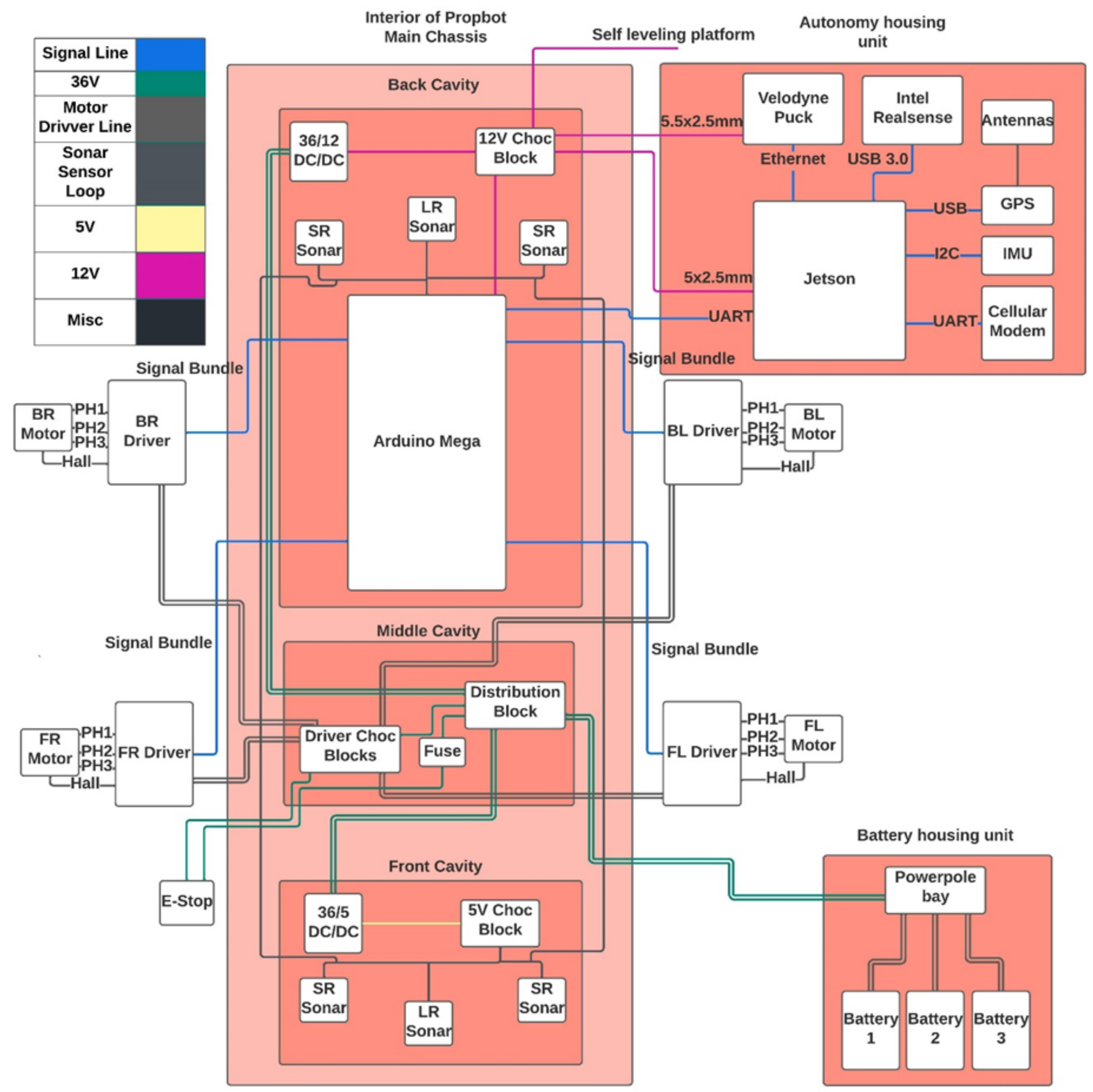
From the motor controller terminal blocks is 4 sets of power connections running to each of the 4 motor controllers in use, all of them are in parallel and delivering 36V from the battery through the distribution block. The cabling used here is 10 AWG and the ends are terminated with male quick connectors to fit into the power input of the motor controllers. In the case where Roboteq motor controllers are purchased and to replace the default controllers, new end connections would need to be fitted (refer to future recommendations).

From the distribution block are two more outputs, one for each of the step-down buck converters. These outputs are terminated with female barrel plugs so that the converters can be disconnected easily. The 5V buck converter is in the front part of the chassis and is meant to power the sonar sensors and wireless sensor at the moment. The output of the buck converter goes to the 5V rail distribution blocks where connections can be made in parallel with the converter. On the output of this distribution block is another set of female barrel connectors for the sonar sensor loop, as well as a barrel plug cable meant for the wireless router.

The 12V buck converter is in the back part of the chassis and is meant to power the Arduino(s), the Jetson TX2, and the self-leveling actuators through the 12V rail distribution block. The Arduino and Jetson both require barrel plugs for power delivery, so there are two barrel plug power lines that are screwed into the 12V distribution block. The self-leveling actuators can similarly be screwed into this distribution block as necessary.

Each of the motor controllers needs several signals from the Arduino to be properly controlled. Therefore, the Arduino is placed in the center back of the chassis where there are 4 sets of wire bundles (8x22 AWG each) leading to each of the motor controllers. The connections of the bundles are terminated with JST-SM connections to easily plug into the motor controller pins that are necessary.

Also leading from the Arduino is a connection to the sonar sensor loop. This loop is a set of wire bundles (4x22 AWG) that loop around Propbot delivering power and data signals to each of the sonars. The loop is wired such that each of the sonars receives signal and power in parallel, which is necessary for correct power delivery and I2C signaling. This wire loop is also wrapped in a metallic sheathing to electromagnetically isolate the I2C signal wires and prevent interference since I2C is extremely sensitive to such things.



Future Work & Recommendations

Faster Communication Link

The router we used for communication was not fit for communicating video for long times at high speeds, leading to issues with latency and reliability when viewing camera feed from the Mission Command Center laptop. Thus we recommend purchasing a better router (if still using WiFi).

Optimizing NX Performance

Currently, the NX is not operating at full capacity (only 3/6 CPU cores are enabled) and common operations like changing the power mode (ex *nvpmodel -m 0*) fail. We believe this is due to kernel settings that were setup by the company we bought the NX from, who had pre installed the Nvidia kernel onto the NX's SSD drive. Investigating the issue of enabling the remaining cores or reflashing the SSD are possible endeavors future teams may wish to embark on.

Mechanical Braking

At the moment Propbot uses electric breaking through the motors and motor controllers in order to stop. While this does provide the needed functionality, it is not a completely safe solution since in the case of power failure or excessively steep slopes the electric breaking would not be sufficient in bringing Propbot to a stop. For this reason, we recommend fitting the wheels with mechanical breaks. Since the motors on Propbot are built into the wheels, adding mechanical breaks might be difficult and we recommend consulting with some mechanical engineers or outsourcing the task.

Wheel Encoders to Provide Feedback

Propbots control is currently based on open-loop operation with manual tuning of PWM levels to get the desired speeds. The proposed Roboteq controllers do have a way of reading speed, however, we recommend implementing wheel encoders for more accurate speed reading in order to implement a closed-loop control structure.

3D printed mounts for sonar sensors

Currently the sonar sensors are mounted with the help of industrial grade zip ties and metal wire sheathing. Based on their current orientation they provide accurate reading but can be further made robust by adding 3D printed mounts to the frame of Propbot.

Implement Roboteq Motor Controllers

The current controllers on Propbot are poorly documented, non-programmable controllers that have been difficult to work with and control. Capstone team 2020 recommended switching them out for Roboteq SBL1360a controllers for their rich feature set and documentation. Our team, Capstone 2022, validated the recommendation and attempted to purchase the controllers but were held back by supply chain issues and backorders on the controllers. We pass on our verified recommendation for implementing the Roboteq controllers accordingly.

Integrate LiDAR

Our LiDAR sensor did not arrive, and so we leave this part of the Integration to be completed by subsequent teams.

Waterproofing

Propbot is designed to be capable of operating in an outdoor environment. As such, it is important that the robot is capable of managing any associated issues with operating in this environment. Vancouver has approximately 170 “rainy” days per year which is just below half of the total days. Therefore, we recommend that Propbot be made waterproof so that the maximum utility can be extracted from its operation. Getting a sleeve for the battery, covering all exposed wires, and making the seal between the lower bed of the chassis and the frame watertight are all possible solutions that could help make Propbot waterproof. This is not an urgent piece of the design but we believe that it would be a valuable addition.