PM Neato

1/29/2024 - Update the paragraph below with a description of what this project is. You can use information from any links to Instructables or other tutorials.	Mr. Burnham Notes

-- >

Date	As part of this description, add the link to the Instructable	Mr. Burnham Notes

Prerequisite & Baseline Knowledge:

< Replace This Text/Paragraph/Section With Your Explanation Of What The User Should Already Know - Provide an Example text here

- know how to do this ← the "this" should be a link to where to learn more
- and this..
- and also this would help

Explain any overarching concept that people need to know about this project.
Have you ever read a tutorial and realized midway through that it's beyond your level of experience? The author forgot to mention the skills you need to complete this tutorial, and now you're frustrated because you've wasted time on something you can't use yet.
-- >

Parts & Software used in this project:

< Replace This Text/Paragraph/Section With Your Overview Or Introduction to the Parts and Software - This section and paragraph is to list out the parts and software, where to get them or how to access them. This paragraph is to say something about any hard to get or hard to access parts or software. You don't want the reader to jump into this project without understanding what parts may be hard to get or software to access.
-- >

Hardware components

	Item	#	Description / Link
1			
2			
3			
4			
5			
6			

Software components

- List the software needed, with an explanation and likes of access, download, or sign up for a free /trial account.
- More links…
- Include and or link to github to point to software and other electronic files.

Additional Electronic Files and Links

- List the links to other files. These are items like 3D Print STL files, Laser cut files, code examples.
- More links…
- Include and or link to github to point to software and other electronic files.

< Replace This Text/Paragraph/Section With Your Explanation of the parts & software, how they work and what to think about when using them - Provide a brief explanation of the actual parts and what they do and how to use them so an explanation of how the particular sensor works or particular motor controller works or particular electromagnetic theories and how they work. **-- >**

Project Overview:

< Replace This Text/Paragraph/Section - Do a deep dive in explaining the purpose and goals of the project, and what the reader will know and be able to do… - This paragraph is an intro to getting started on the specific project. The reader has gotten this far, they are ready to start… get them going with some explanations and then jump into the steps.

Do the following in this introduction and/or in the steps:

- Think of any questions the reader will have, and try to answer them here in the introduction or in each of the steps. Remember, this is your 2-3 time thinking about and trying this lab/tutorial, but your audience may never have seen or learned about this topic. Try to proactively answer any questions they will have.
- On all of your images, make sure to provide a detailed caption that really tells the user what they're seeing and WHY.
- If you are posting code, make sure it is selectable if it is a small snippet, but if not make sure there is a link to a github site so your readers can cut and paste it.
- If a step seems too complicated or has multiple topics, then it probably should be split into multiple steps. It's OK to have lots of steps.
- Make sure you provide a time frame for each step. How long should it take your reader to do?

-- >

Step 1: <Step Title >

< Replace This Text/Paragraph/Section - Give an Overview for the Step - This paragraph is an intro to getting started on the specific project. The reader has gotten this far, they are ready to start… get them going with some explanations and then jump into the steps.

-- >

Overview:

Give an overview for this step. Include a "Why are we doing this step in this order". This can be a short or long or multiple paragraphs.

- This Step will include… < list what they are going to accomplish >

- This Step should take about N minutes to complete.

Tools, Components & Software Used:
List any tools, components or software used in this step. Describe how to prepare these items.

- list them
- here…

Let's Get Started:
Now tell them what they are doing, and how to do it… This is the details of the step. This is probably a few paragraphs or sub-steps. It can be a mix of paragraphs and bullet points. Pictures or videos are also good to have.

- Paragraphs describing the step
- Pictures describing the step
- Blocks and links to code. Your code should be nice to look at. It should look like code, so you could put it in a table cell and color the background and change to a monospaced font like `Courier New`, which is the typical default for a computer terminal font, but you might like other fonts. I currently like `Roboto Mono` or something like Helvetica Neue. See this Google Docs code font tip
-

What Should be Working/Done:
This is just a last paragraph to tell the reader what they should have accomplished, what should be complete, what should be working… AND how they will know. Is there some test to run, is there some mechanical capability the item should have? Just make sure they are done with this step before they move on.

Step 2: <Step Title >

< Replace This Text/Paragraph/Section - Give an Overview for the Step - This paragraph is an intro to getting started on the specific project. The reader has gotten this far, they are ready to start… get them going with some explanations and then jump into the steps.

Overview:
Give an overview for this step. Include a "Why are we doing this step in this order"
Tools, Components & Software Used:
List any tools, components or software used in this step. Describe how to prepare these items

Let's Get Started:
Now tell them what they are doing, and how to do it… This is the details of the step. This is probably a few paragraphs or sub-steps. It can be a mix or paragraphs and bullet points

What Should be Working/Done:
This is just a last paragraph to tell the reader what they should have accomplished, what should be complete, what should be working… AND how they will know. Is there some test to run, is there some mechanical capability the item should have? Just make sure they are done with this step before they move on.
-- >

Software:

Related and "What's Next?"

< Replace This Text/Paragraph/Section With Your "Dig Deeper" Plan - This where you provide a paragraph on what you think your or reader should do next. Describe what your plan to do or what they should do to learn more, dig deeper, or how to expand their learning on this topic.

- List links
- list another…

-- >

< Delete this before you are done > Project Parking Lot - "What's Next", Stuff To Save, Delete or Reorganize:

Each day you work on this project, you should spend some time here moving and organizing topics you have put here… Think of this as the "What am I doing next" section

- Save lots of Photos - Choose images that will be useful to illustrate your instructions

-

-- >

Daily Blog Of Work and Project Progress:

Mr Burnhams "blog"

5/15/2024 - updated the "top NEATO page"
https://sites.google.com/view/steam-clown-mechatronics/robot-club/robot-cars/neato-robots
added neato build scripts for the Raspberry Pi and the crap top.

https://raw.githubusercontent.com/jimTheSTEAMClown/neatoRovers/main/Neato-RaspberryPi-Ubuntu-Build.sh

https://raw.githubusercontent.com/jimTheSTEAMClown/neatoRovers/main/Neato-Craptop-Ubuntu-Build.sh

But I'm testing the linux command code
sudo apt install build-essential
sudo apt install ros-humble-xacro
sudo apt install python3-rosdep2
mkdir neato
cd neato
mkdir ros
cd ros
mkdir src
git clone https://github.com/cpeavy2/botvac_node.git
git clone https://github.com/cpeavy2/neato_robot.git
git clone https://github.com/kobuki-base/cmd_vel_mux.git
git clone https://github.com/kobuki-base/kobuki_velocity_smoother
git clone https://github.com/stonier/ecl_tools

Lexi ✨ & Lijia ✨

★ Day 1: Repair and Pinout

- Today we repaired our neato by resoldering the wires that were previously cut because there was a mistake with which wires to cut so they had to be resoldered. Then we had to cut holes to make way for wires to be soldering to

our battery connector and we also found this reddit page that seemed to have the pinout for the battery connector and it also had a video that showed the circuitry in the battery so this should help with figuring out how to interface with the battery.

★ Day 2: Battery Pain

- We spent the entire day trying to fix our neato's battery connector as it got messed up and we had to take a while trying to resolder some jumpers to the pins in an attempt to repair but we still have a bit to go in getting it actually hooked up to a real connector that will work with the battery.

★ Day 3: It has been Restored!

We spent the entire day fixing up our neato after doing a lot of the work for the battery yesterday and after having to swap batteries due to our's having a faulty connector it worked! So then we sealed it up and FINALLY our Neato was working. So now we need to actually work on the connector stuff.

★ Day 4: Wires

- This class period was spent creating the connection between the Neato and and raspberry pi. We started by preparing many wires by stripping the ends and then soldering them to the proper ends. Once that was done 5 of the wires were screwed into the proper place. The last part was soldering the rest of the wires to the power converter to ensure that we were delivering the appropriate amount of current. Unfortunately our first switch had to be replaced.

★ Day 5: Craptop

- After booting up the raspberry pi, we got a laptop and got a Ubuntu image onto a sd card for the raspberry pi. We replaced the sd card, and then got the new one to boot up as well despite running into issues. We also managed to get all the necessary VIAM stuff onto the devices.

★ Day 6 : SSH Pain

- As we attempted to follow Camp Peavy HomeBrewed robot pdf, we ran into a couple issues. Although we got Turtle and previous installations to work,a certain packet was unable to be found by the raspberry pi leading to issues. Despite it being active we could only get the raspberry pi to connect to the craptop. Despite that we made substantial progress.

★ Day 7: SSH Working!

- We talked to Camp Peavy who has done this before, and he was able to help us by allowing us to realize that we were messing up the ssh command because the first part of it is actually a username field which made it so we couldn't log in properly since we had "ubuntu" in there like how it did in his book. He also helped us install openssh-server on the pi which we were able to do just by doing sudo apt update and upgrade and then attempting to install. Now we just have to go through the rest of the book and hopefully all goes well, however we'll likely get stuck on installing prerequisites for Camp's github package because Mr. Burnham has talked about one of them failing.

5/18/2024 - it's important to document the exact commands, so someone like me a=can try it and check to see if it works. Can you document the actual ssh linux commands you are using.

Mr. Burnham Notes

★ Day 8:

- ssh [username]@[ipadress]
- * username and ipaddress field should be replaced
- We made more progress by beginning implementing the github stuff from the book however we ran into some issues when building some programs specifically when building nav2_util, neato_driver, neato_node packages. (nav2_util is only on the workstation whereas neato_driver and neato_node are on both the Pi and the Workstation and have errors on both) So we'll need to figure out what's up with those tomorrow.

★ Day 9: Colcon pain

- Today we attempted to install the Colcon package multiple times to no avail. Currently we are now working on installing all the dependencies in an attempt to fix this issue. This involves messaging Camp Peavy once again for advice, and possibly the creation of a new script to install all of the necessary material.

★ Day 10: Dark Magic and More Colcon pain

- This time we're deciding to take matters into our own hands since we figured out we weren't actually missing any dependencies so we looked up the error message outline and found a small fix that almost allowed the build for nav2_util here's the link to that fix. Basically go into service_client.hpp located in <ws>/src/navigation2/nav2_util/include/nav2_util/ go down to line 49 and add `.get_rmw_qos_profile()` after `SystemDefaultQoS()` but before the comma and that should allow you to build to about 70%. Make sure to save the file before building again to make sure it works.

- We then did something a little risky, we weren't able to find any other help from the internet so we went ahead and thought outside of the box, we saw an error message was complaining about a function being private, because of this we then had to edit another package (tf2_ros) to make the function public. Even though the script was read-only we forced the text editor to be able to edit the file by using the following command to open a text editor instance that could edit the file needed `sudo gedit /opt/ros/humble/include/tf2_ros/tf2_ros/transform_listener.h`. Then we moved the function `subscription_callback` in the private section of the TransformListener class defined on line 158 and the 2 lines above it, as they are also part of the function into the public section on line 107 underneath the last function of the public section `~TransformListener`
- For the next fix we did something similarly risky and used the same command from last fix (`sudo gedit /opt/ros/humble/include/tf2_ros/tf2_ros/transform_listener.h`) to edit the same function that we moved into the public section and we added the virtual keyword to the beginning of the function (so now it looks like virtual void subscription_callback(...) which then allows the override keyword to work properly since that keyword requires the original function to be virtual.
- We then had some errors when it came getting header files for nav2_util, it seems for some reason that humble's instance of nav2_util doesn't have all the files needed so we ran another risky command `sudo nautilus /opt/ros/humble/include/nav2_util` in order to open a file browser at humble's copy of the package so that way we can copy the missing header files to the package. However we still appear to be missing things so we'll have to diagnose what went wrong tomorrow.
- We essentially dealt in programming equivalent of dark magic today

★ Day 11: Chronic Colcon pain

- After updating we got a new error and deleting lines 64 - 74 in <ws>/src/navigation2/nav_2_lifecycle_manager/src/lifecycle_manager.cpp seemed to fix it so now we're back in the same place. The error seemed to be that there were two seemingly unused variables being assigned to the return result of a nonexistent function so just deleting them fixed it
- Underneath the function std::string get_plugin_type_param(around line 154) we wrote some code from the header file from the nav2_util package we built that seemed to be missing in humble's version of the same header which was causing another error the command we used to open the file was `sudo gedit /opt/ros/humble/include/nav2_util/node_utils.hpp` and the code we wrote was
  ```
  /**
   * @brief Sets the caller thread to have a soft-realtime prioritization by
   * increasing the priority level of the host thread.
   * May throw exception if unable to set prioritization successfully
   */
  void setSoftRealTimePriority();
  ```
- The previous fix didn't work on it's own, I suspect that for whatever reason the copy of nav2_util was outdated or cut off somehow such that the complete package wasn't there since I had to take the compiled binary that was made when we managed to finish building nav2_util ourselves and put it into opt/ros/humble/lib I used the command `sudo nautilus /opt/ros/humble/lib` in order to be able to copy the compiled binary that we had located at <ws>/install/nav2_util/lib/libnav2_util_core.so to that location so it could replace the old binary humble has.
- We now had a similar error to the first error we encountered on this journey, so now we just opened <ws>/src/navigation2/nav2_behavior_tree/include/nav2_behavior_tree/bt_service_node.hpp we went down to line 75 and added `.get_rmw_qos_profile()` after `SystemDefaultQoS()` but before the comma just like the first error.
- Also for some reason our terminal seems to inconsistently crash and sometimes the computer also freezes when the craptop is building the libraries, not sure why this is happening other than the computer is bad.
- Also for some reason after the latest crash the computer is going through the process of rebuilding most of the packages I think maybe stuff got corrupted or something and so now it has to be rebuilt
- Now for some reason we are seeing the same errors as before this could be due to us using apt update and upgrade after modifying libraries that would be affected by them so while building I guess DO NOT update or upgrade when making these changes

- We had to implement the same fixes we did earlier in order to build previous packages that were undone so really DO NOT update or upgrade when building with all these patches
- While waiting for nav2_behavior_tree to build, we noticed some… weird behaviors. Some of the stuff like the terminal crashing and the computer freezing like mentioned earlier seemed to only happen with this package and at one point the computer even seemed to restart. I have no idea how this happened or if the computer is just wigging out today but… yikes.

★ Day 12: geometry_msgs

- We figured out how to fix the terminal from crashing, at least to an extent. When running colcon build you need to add –parallel-workers <amount> which helps the computer with memory issues from the packages. Obviously if you increase the amount it is more intensive but goes by faster and decreasing it makes it run worse. We personally went with 4 so our command was `colcon build –parallel-workers 4`
- We had an issue where geometry_msgs for some reason didn't have a section of itself, specifically the polygon_instance_stamped bit of geometry_msgs just didn't exist for some reason, so we cloned the repository for it and rebuilt the package and spliced in the stuff that was missing so that way it would work as intended instead of dying.

★ Day 13: Shutdown

- We made a shutdown button for our raspberry pi! We followed the hard option of the tutorial Mr. Burnham gave us however the file directory specified in the service file didn't suit us since we required sudo so instead of `/usr/bin/python3 /home/pi/blink.py` our file directory was `/usr/bin/sudo /usr/bin/python3 /home/<user>/pi/shutdown.py` We also changed the command used in the shutdown script from `/usr/bin/sudo /sbin/shutdown -h now` to `/usr/sbin/shutdown -h now` we did also change the if condition from `if GPIO.input(shutdown_pin) == False` to `if not GPIO.input(shutdown_pin)` since while it is equivalent, using not instead of `== False` is a bit more elegant in my opinion
- When running Colcon build again, we noticed that building nav2_behavior_tree would take up all of our memory, causing the terminal to crash. Although we got up to 51%, we still have a long way to go. In an attempt to solve this issue we used 'colcon build –parallel-workers 8' as we found out our cpu has 8 cores and therefore there should be only 8 parallel workers.
- We then tried to build nav2_behavior_tree by itself since it appears to take a ton of ram to build that package so we used the following command to build that one

by itself before trying to build them all at once again `colcon build –parallel-workers 8 –packages-select nav2_behavior_tree`

★ Day 14: New Packages 🥳 💯!!!

- We now have a command that will allow us to only work on packages that still need to be worked on which is very useful since we constantly run out of memory by rebuilding already built packages so now we have modified our build command to be `colcon build –parallel-workers 8 –packages-skip-build-finished`
- We then ran into an error when building nav2_collision_monitor where we had to change line 84 from `timer_ = this->create_timer(` to `timer_ = this->create_wall_timer(`.
- After a while we noticed that we were missing something from another package and so we went down a change of package dependencies that weren't installed for some reason and had to use the command 'git clone' for the following repos in order to build our own version of rclcpp so that way we can get a function that was missing. The repos we cloned were `https://github.com/ros2/rcutils`, `https://github.com/ros2/rosidl` (specifically rosidl_runtime_c), `https://github.com/ros2/rosidl_dynamic_typesupport`, and of course `https://github.com/ros2/rclcpp`. We then had to use `sudo nautilus` in order to copy the newly built files over, hopefully you know how to copy the binaries and header files by now so that we don't have to go over it in vivid detail.
- In a attempt to fix more missing package dependencies repos were cloned 'https://github.com/ros2/rcl', 'https://github.com/ros2/rcl_interfaces' (specifically service_msgs), 'https://github.com/ros2/rosidl_core' (specifically rosidl_core_generators), We already had rosidl_generator_type_description and rosidl_pycommon from when we cloned the rosidl repo and because of that we didn't need to clone another repo for those 2 packages.
- We found that rosidl_generator_type_description and rosidl_pycommon had some weird file formatting compared to the other packages so we found that for the lib folders the folder inside the local folder that has the name of the package gets put in humble's local folder and the folder inside the lib folder with the package name gets put in humble's lib folder

★ Day 15:

- Today we are having to undo the rest of our work, since a section of the book is wrong. After contacting Camp Pevy on LinkedIn we managed to identify the source of our issues. ***Nav2 must not be compiled from source!!!!*** This is because when compiling by source the default branch does not necessarily match the distro of ros you're using and therefore when compiling from source it will create a bunch of errors if you have a different distro than the default branch so compiling from source was the source of all our problems that `sudo apt install

ros-humble-navigation2` and `sudo apt install ros-humble-nav2_bringup` fixed. Then updating Ubuntu we went on to follow all of the commands from'https://github.com/cpeavy2/botvac_node' on both laptop and raspberry pi4.
    - We started from page 143 and connected the raspberry pi 4 to the neato using a short USB type A to Micro USB cable, and a short USB-A to USB-C to connect the battery to the Pi4 Power supply

★ Day 16:

- We attempted to follow the book's instructions.***You don't need to create a new user .***

★ Day 17:

- We attempted to fix some of the issues that came up when trying to run the commands needed to get the neato up and running. We weren't able to fully diagnose why the usb connection wasn't working but in the meantime we fixed the issue with velocity_smoother. We edited the file at `~/ros2_ws/install/botvac_node/share/botvac_node/launch/include/velocity_smoother.launch.py` and we added the import `from ament_index_python.packages import get_package_prefix` and then we changed line 42 (after adding the import it looks like this `executable = exe_name`) to `executable = get_package_prefix(pkg_name) + "/lib/" + pkg_name + "/" + exe_name`. Which fixed the error by specifying to the computer where the executable is.

★ Day 18:

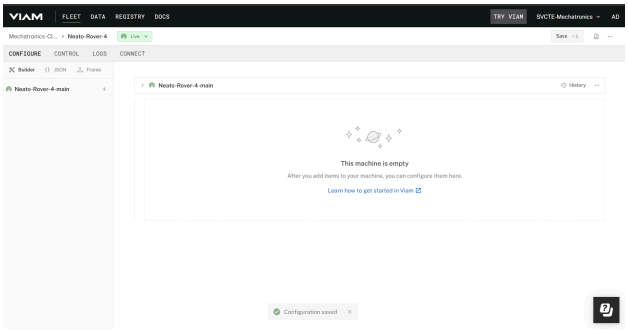
-

Alex & Charley:

- Day 1: Disarmament
    - We resoldered the incorrectly cut wires back together, then we drilled a hole for the wires to be put through. Charley suggested using the speaker, so another hole needs to be drilled.

- Day 2: Diconnectent
    - we disconnected the speaker to connect it to the raspberry pie and melted a hole for the speaker wire to go through

- Day 3: Batterrytainement
    - We worked on the battery, which was faulty and not wanting to charge. We got a replacement battery that we were told to “borrow” from another team. We will get our own later on.

- Day 4: Connection
    - 

- Day 5: Wiring
    - We finished the main soldering parts for the Neato Vacuum, which includes the switch and
- Day 6: E
    - There wasn’t much done today, I didn’t get the image onto my raspberry pi. I did work a bit on the Ubuntu laptop. Battery for the rover seems to not want to charge.
- Day 7: Ubuntu
    - I finished up the setup for the ubuntu laptop, and started a bit on the raspberry pi.
- Day 8 - Day 11:
    - Attempting to finish the installation on the raspberry pi.
- Day 12 - 15:
    - Followed the instructions on the github, as the book was wrong. Lexi messed up, but it was more of the book's fault than anything. Now we wait for the battery to charge
- Day 16 - 18?:
    - So Lexi couldn’t get it to work, and time is running out. It is charged, but the connection isn’t working properly.

Reference and Appendix:

---

https://www.bristol.ac.uk/esu/media/tutorials/design-principles/page_08.htm

https://www.reddit.com/r/learnprogramming/comments/yf9zb2/what_makes_a_good_tutorial/

How to Make an Actually Good Tutorial
https://uwaterloo.ca/centre-for-teaching-excellence/catalogs/tip-sheets/key-strategies-effective-tutorials

https://www.webdew.com/blog/tutorial-video-examples

https://dailyblogtips.com/11-essential-tips-to-writing-the-ultimate-tutorial/

https://codingwriter.com/how-to-write-better-tutorials-part-1/

https://dev.to/savvasstephnds/in-your-own-opinion-what-makes-a-tutorial-beginner-friendly-mg4

https://medium.com/@keshidong.dev/how-to-format-code-in-google-doc-833e28b304f1

Notes Mr. Burnham Notes