Mimir Electromech Iteration 1 Standard Notes

April 29, 2023

Contents

1 Chassis Design

This section covers the design of the Mimir chassis. The chassis includes mounting for all devices and motors. This is a purely mechanical section.

1.1 FIRST DATE IGNORE

1.1.1 Summary

1.2 References/Resources

2 Power Distribution System

This section covers the design of the power distribution system (PDS) including the design of the PCB, electronic component selection, and battery selection.

2.1 4/7/23

2.1.1 Summary

Today we went through early design and component selection of the Mimir PDS.

2.1.2 Roadblock: Multiple Batteries or DC-DC Conversion

The rover generally needs two voltages: 12V and 5V. The current LiPo batteries supply 14.8V, thus a buck module which can step the voltage down from 14.8V to 12V and/or 5V is needed. Alternatively, we can use two new batteries: 12V and 5V. The belief was that two batteries would be less expensive than a buck module, this turned out to not be true. Also, two batteries made the circuit diagrams easier, but they were significantly less compact and centralized.

After determining industrial buck modules were cheaper than 2 new batteries (by almost $ 50), we agreed to the buck modules. Even though the circuit board will be more complex, the physical system will be more compact. Further, the buck modules allow us the ability to swap out the batteries for any other battery (obviously given it's above 12V and has the same connector), so if we design to swap out in the future for higher power deliverance or storage we can do so freely. As a side note, the buck modules we purchased have a variable input voltage for some set output; some devices (not desired) have both set input and output.

The final circuit will convey the battery's power to two variable-to-12V buck modules. The exact model we're looking at right now has a 10A maximum throughput, so we'll probably get 2 and place them in parallel. Both modules will contribute to power delivered to the motors; but, only 1 buck module will (in parallel) feed into yet another buck module that will step the voltage down to 5V. This 5V will feed through to the computers and Stewart platform.

2.1.3 Roadblock: Yet unsolved problems

We need to determine if there's a way to limit how many amps the motors are allowed to pull. The buck modules can deliver 20A, and the motors will likely pull 6A; but, under some insane condition each motor could draw upwards of 20A (for a total of 80A with 4 motors). If possible, we need to implement

electronics or circuitry that only ever lets the motor pull at most 14A, and save 6A for the computers and Stewart platform; this 14:6 ratio is tentative and definitely subject to change.

We also need to determine the behavior of our Stewart platform servo motors. In a similar fashion to the previous paragraph, servo motors may be able to draw undesirable amounts of current under 'stuck' conditions. The selected servos are quoted for 5V and 3A; does this mean they will pull a maximum of 3A? Or that they always pull 3A and operate full torque, regardless of loading conditions? Either way it's wise to have a system that circumvents high current which isn't a fuse.

2.2 References/Resources

[1] Infineon Technologies, "High Current PN Half Bridge", BTS7960 datasheet, Mar. 2004 [Revised Dec. 2004]

3 Programming

3.1 4/10/23

3.1.1 Summary

Meeting notes with the team, specifically Ryan Miller and Joonsu Park (EE team) on the connections between the electronic components will be laid out.

3.1.2 Roadblock: Determine the connection between NVIDIA Jetson and Arduino

In this meeting we spoke about the pros and cons of how we will connect the NVIDIA Jetson and the two Arduino's that will be used for the motor controllers and Stewart platform. Our main idea is to utilize a USB hub for serial communication between the devices and using the NVIDIA Jetson as the main node that talks to the two Arduino's. The problem we have understood with this design is that USB also provides power as well as serial data, so this may create a circular loop of power. Both the Arduino's being powered by the USB and by the power distribution system. We believe it should be okay, since the other method of using the data pins on the Jetson and Arduino's would consider us to redesign the already built ROS serial libraries.

3.2 4/12/23

3.2.1 Summary

This section talks about the progress made to create a website that is hosted off the NVIDIA Jetson device and provides us with a Flask App and using an HTML frontend. With a little bit of help you are able to integrate a Flask app and a HTML frontend, test and deploy it, and create a powerful web based interface for controlling a rover.

3.2.2 Methodology

Step 1: Install and Set Up the Jetson Nano First, you need to install and set up the Jetson Nano. The Jetson Nano is a small but powerful computer that can run ROS (Robot Operating System) and other applications. You'll need to connect the Jetson Nano to your network, set up the OS and ROS, and ensure that it's running smoothly before proceeding to the next steps.

Step 2: Install Flask and ROS on the Jetson Nano Once you've set up the Jetson Nano, you'll need to install Flask and ROS on it. Flask is a lightweight web framework that allows you to build web applications using Python, while ROS is an open-source framework for building robot software. You'll need both of these tools to create a web-based interface for controlling your robot.

Step 3: Build the Flask App Next, you'll need to build the Flask app. This involves creating a Python script that uses Flask to handle HTTP requests and responses. Your Flask app should be designed to control the motors of your robot and publish data from the lidar.

Step 4: Create the HTML Frontend With your Flask app up and running, you can create the HTML frontend for your website. The frontend is the user interface that your visitors will see, and it's what they'll use to control the robot. You can use HTML, CSS, and JavaScript to create the frontend.

Step 5: Integrate the Flask App and HTML Frontend Once you've created the Flask app and HTML frontend, you need to integrate them. This involves linking the frontend to the backend by creating HTTP routes in your Flask app that handle requests from the frontend. You'll also need to ensure that your Flask app is capable of hosting publisher nodes for ROS.

Step 6: Test and Deploy Finally, you'll need to test your website and deploy it to the Jetson Nano. You can use a web browser to test your website and ensure that it's functioning properly. Once you're satisfied with your website, you can deploy it to the Jetson Nano so that it can be accessed from any device on your network.

3.3 4/17/23

3.3.1 Summary

This section talks about this weeks progress on implementing the website. All the code created can be seen in the hackrover/rover GitHub repository. This provides us with a insight on how the webserver works. With this weeks update we worked on a way to choose between the different outputs of the LIDAR camera that we used (RealSense L515). We used a slider in HTML and Javascript backend to make it work. Then with the backend on Flask. Examples of how it looks are shown in the repository [2].

3.4 4/25/23

3.4.1 Summary

This section talks about this weeks progress on adding improvements to the webserver. Mostly talking about how we were able to create distance data from the LIDAR heatmap output. The LIDAR heatmap provides us with a heatmap of both blue and red results. Now what does that blue and red mean? Well it

is just like the child game you played as a child where blue meant cold and red meant hot. The closer an object is to the LIDAR then the more red the output image will be. And vice versa with the blue where the more blue the object the "colder" the object is away from the LIDAR camera. Using this data we can call on some built in functions from the pyrealsense import for the realsense camera from Intel and we utilize the getWidth, getHeight, and getDistance functions to calculate the distance. This gives us a rough estimate from the LIDAR for the distance of an object. Keep in mind that this only provides distance measurements of objects RIGHT in front of the LIDAR camera, and employs a mask that ignores the blue around the central object.

3.5 4/27/23

3.5.1 Summary

This section talks about this weeks progress on the thought process and information surrounding this rover project. Currently we are still hosting the server directly on the Jetson itself.

3.5.2 Methodology

Motor Controls: The motor controls aspect of the project involves controlling the movement of the robot, which can be accomplished by sending signals to the motor controllers. To do this, you can use GPIO pins on the Jetson Nano to connect to the motor controller board, and then use a Python library like RPi.GPIO to control the pins. By designing a user-friendly interface, you can allow users to control the movement of the robot through the website.

Lidar Output: The lidar output aspect of the project involves collecting data from the lidar sensor and displaying it on the website in real-time. To accomplish this, you'll need to connect the lidar sensor to the Jetson Nano and then use ROS to process the data. Once the data has been processed, you can use Flask to serve the data to the HTML frontend, where it can be displayed in a visually appealing and user-friendly format.

Publisher Nodes for ROS: ROS uses a publish-subscribe model to facilitate communication between different nodes in a robot system. Publisher nodes are responsible for publishing data to specific topics, which can then be subscribed to by other nodes. To integrate publisher nodes for ROS into the Flask app, you'll need to use a Python library like roslite to create the publisher nodes and publish data to the appropriate topics.

Deployment: Once you have created the Flask app, HTML frontend, and integrated ROS and motor controls, you'll need to deploy the website to the Jetson Nano. There are several options for deployment, including using a web server like Apache or Nginx to host the Flask app, or using a lightweight server like Gunicorn. You'll also need to configure the firewall on the Jetson Nano to allow incoming connections to the website.

We utilized ROS noetic for this project as there was easier ways to send information between the Jetson device and the Arduino for motor controls. Then we had to figure out all our imports for the Python scripts, especially for the Flask website. We utilized some neat scripts that helped us set everything up, these can be found in the public hackrover/rover GitHub repository, especially in the necessary packages section.

3.6 References/Resources

[1] Schwind, Johan. "Mobile Robot Teleoperation with the Jetson Nano and Ros." Medium, Medium, 22 Feb. 2021, https://johanschwind.medium.com/mobile-robot-teleoperation-with-the-jetson-nano-and-ros-d72b4b57e9be. [2] https://github.com/HackRover/rover