Carnegie Mellon University

16-681A

MRSD Project I

Individual Lab Report 02
Team C: COBORG

Author: Gerald D'Ascoli

Remaining Team C Members:
Jonathan Lord-Fonda | Yuqing Qin | Husam Wadi | Feng Xiang

Sponsor:
Biorobotics Lab

March 4, 2021

Table of Contents

1 Individual Progress 3
2 Challenges 6
3 Teamwork 7
4 Plans 8

1 Individual Progress

This past week, I redesigned the voice control system to be more scalable with commands and more controllable for tuning for any potential accuracy issues that may arise. The previous system I had worked under my original test cases but was very difficult to scale and was either too sensitive on the trigger word that it heard it all the time or, under different parameters, would not recognize the trigger word at all. My initial plan of creating a custom dictionary to recognize only our commands and keywords caused the sensitivity of recognizing these words to be much too high. Then the next plan to try to add the new word “COBORG” to the existing dictionary resulted in the system still never recognizing the word “COBORG” over other words in the standard US English dictionary. In the redesign, I utilized the “keyword” function built into the pocketsphinx-python package to create a pseudo-custom dictionary to raise the sensitivity for words to be recognized outside of the normal voice commands, such as “COBORG” and “STOP”, without it being overly sensitive. To get the standard dictionary to recognize “COBORG” more accurately, I added the word and the related word segments (sounds interpreted by the NLP model) to the standard dictionary then re-trained the standard US English model using TensorFlow to give priority to the word “COBORG”. The initial training went for over a day and ended in failure because I had set the initial parameters incorrectly. On a second attempt, it trained for a couple of hours and then produced decent results by recognizing the word “COBORG” almost every time it is said. From there I built up the python script as a state machine to switch between a keyword recognition mode and general recognition mode. The script sets up both modes as well as an enum list containing the possible command outputs. It runs continuously to always listen for the trigger keyword “COBORG” from the keyword dictionary, then once recognized, it switches the interpreting dictionary to the standard dictionary so it can recognize every possible word for commands. The command recognition is a switch statement (implemented as a series of if-elif statements in python) that looks for certain command related keywords in the interpreted sentence and then transmits the recognized command. For example, if the user says “COBORG” then “Move to home”, then the word “home” would be recognized in that sentence and the “HOME” command will be transmitted.

In testing, I noticed a pattern of people saying the commands right after the keyword in the form of “COBORG Move to home”. This was causing issues because the script was including the “Move to home” portion in the keyword recognition portion. To fix this, I implemented some audio feedback into the voice recognition system. When the keyword “COBORG” is recognized, a pleasant jingle plays to notify the user that the system is listening for a command. If a command is recognized, it plays a separate success jingle. If no command is recognized, it plays a failure jingle. If an e-stop command is recognized, it plays a slightly more urgent warning sound alerting the user that e-stop actions are being taken.

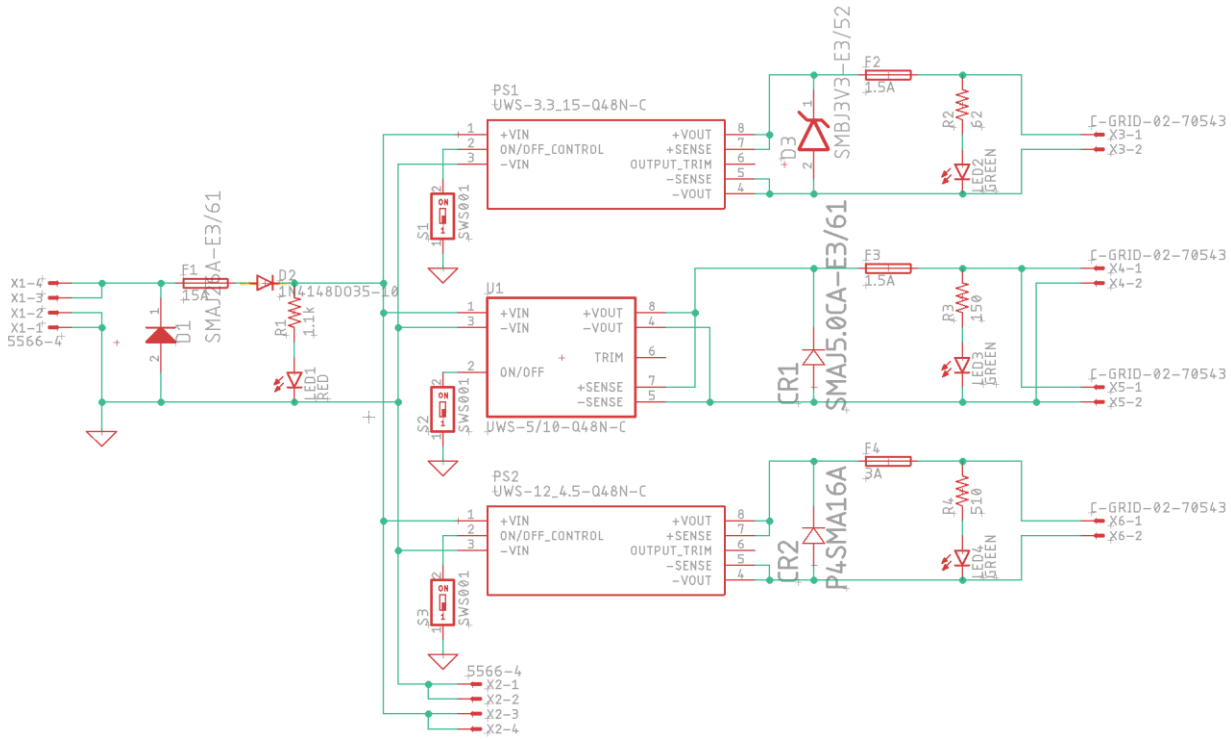

Figure 1. Power Distribution Board PCB Schematic

Apart from work directly on the COBORG project, I handled the PCB assignment for our team. With help from Yuqing and Jonathan, I designed the schematic given the parts list on the assignment. From there I designed the PCB layout and generated the base CAD model. I decided to use the buck-boost (switching) DC-DC regulators, as shown in Fig 1, because they are dramatically more efficient than the linear regulators. For the ON/OFF enable inputs for the regulators, I decided to use DIP switches as they are small in scale but still easily manipulated by hand. For overcurrent protection, I used fuses rated to break at 150% the expected current at the input connector and the output of each regulator. For the input that meant limiting the current to 15A, then 3A for the 12V regulator output, and 1.5A for both the 5V and the 3.3V regulator outputs. For overvoltage protection, I used the given TVSs rated to breakdown at greater than 10% over the desired voltage at the input and each of the regulator outputs. The input required the SMAJ26A-E3/61 TVS rated to breakdown at 28.9V, ~20% overvoltage. The 12V regulator output required the P4SMA16A TVS rated to breakdown at 15.2V, ~27% overvoltage. The 5V regulator output required the SMAJ5.0 TVS rated to breakdown at 6.03V, ~21% overvoltage. The 3.3V regulator output required the SMBJ3V3-E3/52 TVS rated to breakdown at 4.1V, ~24% overvoltage.

For the PCB layout, shown in Fig 2, I organized the components such that the parts related to each regulator (DIP switch, LED, fuse, TVS, output connector) were in line with the corresponding regulators. The input 24V connector is on the opposite side of the board as the regulator outputs. The motor output connector is on this same side as that output is not regulated by any of the regulators.

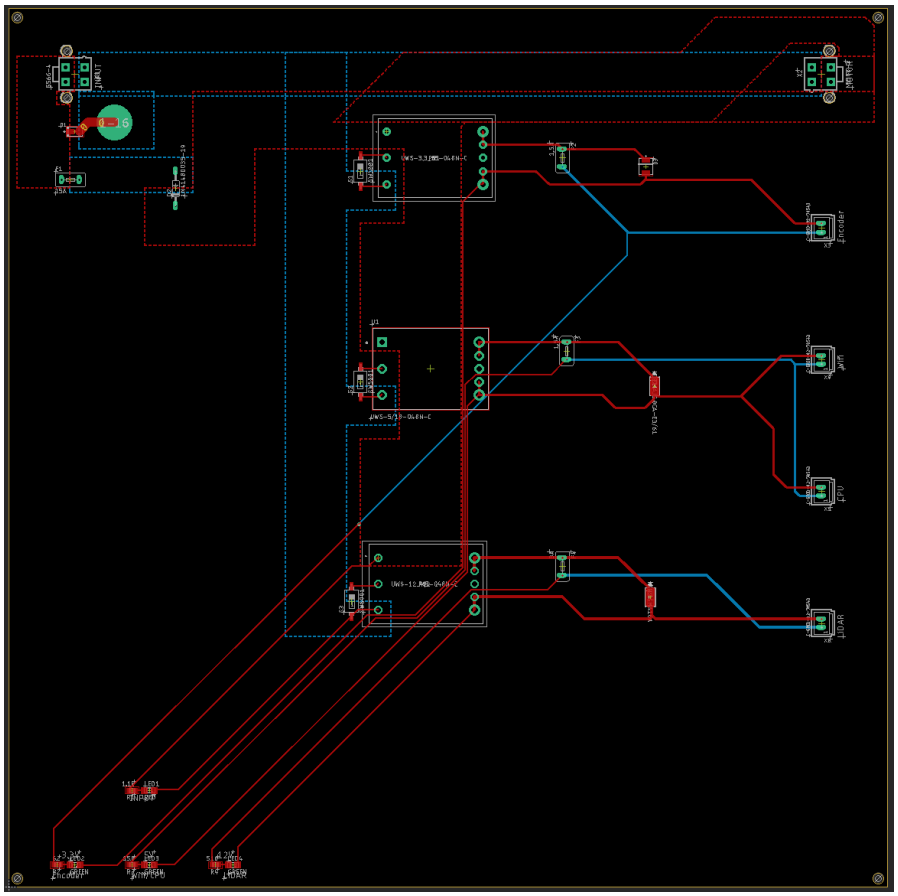

Fig 2. Power Distribution Board PCB Layout

Finally, for the CAD of the PCB, I found a pre-installed ULP in EagleCAD that exported a .brd file as IDF files (.emn and .emp) to port into Solidworks's circuit progam Circuitworks. The generated CAD had the placement of all components but not much detail apart from that, as shown in Fig 3.

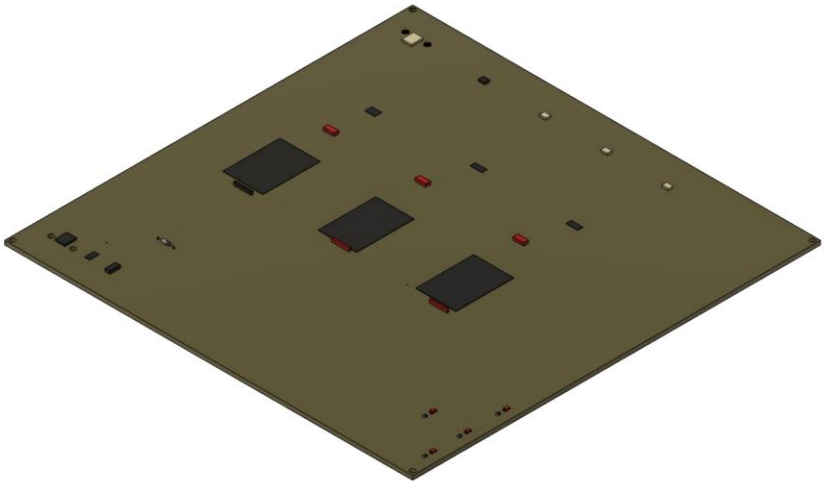

Fig 3. Power Distribution Board PCB CAD

2 Challenges

The challenges that arose in redesigning the voice recognition system mostly centered around trying to decrease keyword sensitivity and trying to increase command word sensitivity. In private testing, the commands were able to be clearly recognized without wearing a mask. Unfortunately, this will probably not be allowable conditions for the spring validation demonstration so tuning is necessary. As for the keyword sensitivity, I reduced some of the probability for false positives without reducing the probability for true positives by running the voice recognition system while my teammates were having conversations in the lab space. Whenever they said a word that triggered as a false positive for "COBORG", I added the word to a list of words that the keyword dictionary should ignore. This showed minor improvement to the oversensitivity while maintaining consistent positive responses to true "COBORG" utterances. Additionally, I had some trouble pushing the voice recognition node to GitHub because some of the NLP model files were too big to be added to the repository. I had to deep-dive into what each file in the model did in order to figure out if these excessively large files were necessary. Luckily, I discovered that they were event logs generated when I was training the NLP model so they could be discarded without any cost to functionality. Another challenge we discovered is that the microphone we got from the MRSD storage that we have been using for tests muffles the audio. This could have been drastically impacting the recognition of words in both keyword and command modes reducing the accuracy of both. We have since started looking for solutions in the form of directional microphones designed to listen to just one user for the microphone that will actually be mounted onto the COBORG.

For the PCB assignment, the significant challenge was finding the EagleCAD library files for each of the parts on the given list. Luckily, the team cooperated in scraping the internet for the .lbr files so I could move on with developing the schematic.

3 Teamwork

Team Member	Teamwork Progress	Challenges	Future Work
Feng Xiang	• Created URDF model • Integrated URDF with Move-It • Got Robot Arm to Move to a point selected in RVIZ	• Tying T265 camera to URDF model • Developing 3D goal pose update to stabilizer robot arm relative to global point	• Tying T265 camera to URDF model and moving the human on the RVIZ simulator
Jonathan Lord-Fonda	• Updated ROS Node Map with Gains node • Updated ROS Node Map for vision system • Added a speaker output to ROS Node Map • Wrote Main State Node • Wrote up semantic Goal_Setter Node • Helped construct the robot holder/testing structure	• Full-install Linux USB is not Persistent install Linux USB • The Goal_Setter is complex. We're not entirely sure how to match up our images/3d points/transformations and average them • Motion planning will have to be fast and we will also have to adjust our waypoints locally to account for motion	• Connect main_state_machine node to voice_recog node • Semantically write out Gains node • Update/change ROS Node Map as needed • PCB CAD • Work alongside Jason to absorb all of his knowledge • Plan out roadmap for Actuated Manipulation with Jason • Complete at least one task for Jason in Actuated Manipulation
Gerry D'Ascoli	• Redesigned voice recognition system to work with "Coborg" trigger word • Wrote script to translate verbal commands into robot instructions • Added audio feedback for voice control system • Helped construct COBORG demo/testing structure • Designed PCB schematic with Yuqing & Jonathan • Designed PCB layout .brd file • Created CAD model of PCB	• Improving accuracy of command recognition • Decreasing sensitivity of "Coborg" trigger word	• Developing ROS node wrapper to transmit interpreted commands to the main ROS node (State machine) • Investigating new microphone for better single user recognition

Yuqing Qin	• Installed Realsense D435i ROS package • Launched YOLO v3 on hand detection • Wrapped YOLO model with ROS node • Set up GPU to run the YOLO ROS node • Tested D435i with YOLO node	• Need to research on ROS node design for goal's 3D position (goal_setter) • Need to fully understand the intrinsics , depth and 3D location.	• Finding corresponding part location from hand position • Abstracting depth information from D435i • Obtaining the real world 3D position from depth
Husam Wadi	• Cleaned up B512 with the group • Built structure to hold Coborg • Adapted timeline to increased workload • Assisted Jason with intel realsense (T265) • Assisted Yuqing with intel realsense (D435i) • Assisted Jonathan with ROS Node Map • Ensured team used Github	• Discovering how to send an absolute point for the end effector to track to. How do we use /tf topic to translate a depth point from the D435i, transform through the T265, all the way to the end effector of the robot arm.	• Deep dive testing with robot arm and realsense cameras to discover the link between them.

4 Plans

My plans for the next progress review are to have a full functioning ROS node for the voice recognition system that can transmit recognized commands to the main state machine ROS node that Jonathan is making. This will require making a custom publishing format as the standard format using "rospy.spin()" will not work with the current format of the voice recognition node. This is due to the voice recognition script running on an infinite while loop to continuously listen so "rospy.spin()" will hold up that while loop and prevent listening. We will have to fold a publishing structure into the existing command recognition switch statement.

I also plan to find a new microphone for the voice recognition system that better fits our use case. This solution should come in the form of a directional microphone that is easily mountable onto the COBORG and can interpret the voice of the single user.