|  |  |  |  |
| --- | --- | --- | --- |
| **MEMORANDUM** | | A picture containing graphical user interface  Description automatically generated | |
|  |  | | |
| **To:** | Professor Charlie Refvem, Department of Mechanical Engineering, Cal Poly SLO | | |
|  | [crefvem@calpoly.edu](mailto:crefvem@calpoly.edu) | | |
| **From:** | Jack Butler | |  |
|  | [jbutle10@calpoly.edu](mailto:jbutle10@calpoly.edu) | |  |
| **Date:** | September 26, 2022 | | |
| **RE:** | Lab 0x04 – Closed-Loop Motor Control | | |
|  |  | | |

**Program Overview**

Our program is comprised of three main tasks – UI, Motor, and Encoder. UI is the brain of the system, dealing with all of the logic if the system. It sets flags and makes sure user input gets passed to the other tasks if necessary. It also does some simple value fetching, like the encoder position or motor speed. Motor controls the motors – mostly, it sets duty cycles either based off the open or closed loop controller. Encoder manages the encoders and measurement duties – its main jobs are running the step response and the 30-second data collection, and sending the gathered data to UART. The task diagram for our program is as follows:

The shares/queues in the program are as follows:

Table 1: Shares and queues in the program

|  |  |  |
| --- | --- | --- |
| Name | Type | Purpose |
| Start | Share |  |
| Kp | Share |  |
| Des\_vel | Share |  |
| Gain | Share |  |
| Collect | Share |  |
| Mode | Share |  |
| Duty\_flag | Share |  |
| Duty | Share |  |
| Selected\_motor | Share |  |
| Selected\_encoder | Share |  |
| Eff | Share |  |
| Prompted | Share |  |
| Numinput | Share |  |
| Mode | Share |  |
| Collect | Share |  |
| Z\_flag | Share |  |
| P\_flag | Share |  |
| D\_flag | Share |  |
| V\_flag | Share |  |
| G\_flag | Share |  |
| K\_flag | Share |  |
| R\_flag | Share |  |
| S\_flag | Share |  |
| Working | Share |  |
| Collectionpos | Queue |  |
| Colelctionvel | Queue |  |

**Task Details**

*UI*

UI is the only task allowed to use VCP, but is not allowed to use UART. Using VCP, it reads user input and writes any necessary messages to the terminal. It parses user input in order to set the relevant flags for other tasks, and takes care of blocking user input when the system is busy (when recording data or performing a step response). The state-transition diagram for UI is as follows:

*Motor*

Motor sets the motor speeds and a few other jobs. The state-transition diagram for Motor is as follows:

*Encoder:*

Encoder takes care of the heavier data collection jobs, like the step response and the 30-second data collection. It manages writing the data that was collected to UART, and sets a flag to let UI know to block user input while it’s collecting data. The state-transition diagram for Encoder is as follows:

**Conclusion**

This lab laid a lot of the groundwork for things we’ll be doing on our term project – we created a controller class, made a solid user interface, and laid the groundwork for position control. In a sense, a big part of the term project will be scaling up this lab.