CSCE 274 - Section 001 - Fall 2017 - Project 1 - Khalid Salah, Logan Fisher, Julian Hong

Lab Report Project 1

Description:

We wrote three files to run the iRobot Create 2. First we wrote the file Linguist.py, which is a class that handles raw communication between the iRobot and the user. Our second file, Jaguar.py, prepares the robot to be driven. And our final file, Pentagon.py, contains all of our commands to make the iRobot drive in a shape of a pentagon. We decided to make Pentagon.py a different file because it was a file needed specifically for this project, meanwhile Linguist.py and Jaguar.py contain generally information to control the iRobot.

Our file Linguist.py contains our most basic communication with the iRobot. We created a method to make a serial connection with the iRobot first. While having access to it, we want to send commands to our iRobot and also have it return some type data if need be. With our method of sending commands, they have to be sent in as a character. Furthermore, when the iRobot sends the user data to read, it will be sent as the number of bytes wanting to be read. While still within Linguist.py, we have written another method to close the serial connection once the user is done with the iRobot.

Jaguar.py consists of all the general information for the robot. We started with declaring all our magic numbers which includes having to define the hex codes for all buttons, and the OP Code for the states. We wrote a few methods to find two’s complement so we didn’t have to do that by hand. When the iRobot creates an instance, we have it change it’s state to passive. Once the user has clicked the ‘Clean’ button, the method readButton() sends the correct Sensor Data Request Code and the Button Packet ID to have the robot send back bytes specifying if the correct button is hit. If all goes well, we set the iRobot to to drive mode and commands it to drive at a certain speed and radius.

We created Prentagon.py to command the iRobot to drive in the shape of a pentagon. We started Pentagon and Jaguar the same way, by stated the magic numbers in the beginning of the file. We assign how many turns we want the robot to have, and then start the while loop. It checks while turns are still greater than zero, it’ll drive straight at a certain speed for a certain amount of time. Then it checks if that was the second to last turn. If it was, the iRobot will jump out the loop. If not, it’ll rotate at a speed and for period of time. Then it will subtract a turn and continue again in the loop. Once out of the loop, the iRobot has instructions to stop driving. We also have a method to let the iRobot listen and hear if the Clean button was pressed. Then we had to thread to have both listen and driver running at once.

Evaluation:

Our program worked as expected. Linguist.py opened a serial connection with the iRobot, sent commands, read data, and also closed the serial connection. Jaguar.py handled binary conversions, interpreting data, driving the robot and reading buttons. The final file, Pentagon.py, had the driving instructions for our robot. We implemented the listening method into Pentagon.py because the iRobot can listen for a specific button to be pressed thus running the Pentagon.py file. To complete the file, we added threading into the file. When it’s running, it’ll either listen for the sound, or it’ll wait for a set of driving instructions.

Allocation of effort:

Khalid Salah:

* Coded
* Commented code
* Threading Algorithm

Logan Fisher:

* Debugged Pentagon.py
* Driving Algorithm
* Pentagon driving speeds

Julian Hong:

* Commented code
* Wrote README.md
* Wrote report
* Debugged overall