CSCE 274 - Section 01 - Fall 2017 - Project 2 - Khalid Salah, Logan Fisher, Julian Hong

Lab Report Project 2

Description:

We carried over two files from Project 1, Linguist.py and Jaguar.py. Linguist.py was created to to handle raw communication between the iRobot and the user. Jaguar.py was created to handle the actions of the robot. Our newest file, scout.py, has a thread to drive, a thread for listening to cliffs, a thread for listening to bumps, and a thread for listening to button clicks.. Jaguar and Linguist are files that contain general information for the iRobot while scout is the file that handles the actions of the iRobot.

Our file, Linguist.py, contains our most basic communication with the iRobot. It has a method to create a serial connection with the iRobot first. While having access to it, it has another method to be able to send commands to the iRobot and also have it return some type of 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 needing to be read. While still within Linguist, it also has another written method to close the serial connection once the user is finished with the iRobot.

Jaguar.py consists of all the general information for the robot. We brought over the same files from Project 1 to add on more methods if need be. Within Jaguar, we added four new methods. It has a method to read bumpers and wheel sensors and this tells us if the wheels have dropped or if it got bumped left, right, or both. The next method reads the cliff states of the iRobot. If any cliff packets return a one, we know that cliff is activated. When the cliff is activated, it will play a song and following the readings of cliffs, we have a method to play a warning song. We have also written the new drive method to have each wheel be moving at its own velocity.

Our newest file, scout.py, is the file that executes our program for Project 2. We defined all magic numbers at the top of our code. We set the robot to be driving straight first. While it’s driving, we want it to listen for bumps also. We call for the bump state of the iRobot and check to see if the wheels have dropped or if any bumper was triggered. If the wheels have dropped, it stops and plays the warning song. If it bumped left, it would rotate clockwise 180 degrees and then add in a variance in the range of negative thirty degrees and thirty degrees to it. If it bumped right, it would rotate counterclockwise 180 degrees and add in a variance also. If both bumpers have been triggered, it would pick a random direction and rotate 180 degrees and the variance. To listen for cliffs, we just check the data the iRobot sends to us and check to see if it is activated. We also implemented listening for button clicks into scout to check if they button pressed is the correct button. We ran four threads for this project; driving, listening for cliffs, listening for bumps, and listening for button clicks.

Evaluation:

Our projected 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, reading buttons, reading bump and wheel sensors, reading cliff states, play song method, and our new drive method. The final file, scout.py, had the driving instructions for our robot. We implemented the new drive and rotate functions into the file so that if it bumps into anything, it’ll know to rotate. Within our drive function, we also added in a variance to the rotate so that We also have our listening function in scout, it listens for bumps and wheel drops. In any case of a wheel drop, the iRobot plays the warning song and stops in place.

Allocation of Effort:

Khalid Salah:

* Coded
* Debugged Scout.py
* Cliff States

Logan Fisher:

* Created Drive Algorithm
* Debugged Scout.py
* Created Obstacle()

Julian Hong:

* Created Song Function
* Wrote Report
* Debugged Jaguar.py