CSCE 274 Section 001 Fall 2019 -- Project 2 -- A. Staton, M. Ziemer, R. Carff

Description:

The goal of project two was to implement the "random move" function for a roomba. To accomplish this, we augmented our project one interface by adding in the ability to read the Create2's bump, wheel drop, and cliff sensors. We wrote most sensor reading capabilities in the robot's interface ('RobotInterface.py') to allow for greater simplicity within our motion-control logic in the main function. In our main function, we created a single thread that calls each sensor reading method. We chose to do this to reduce the excessive reading of sensor data from the Create2. We found that if many threads were reading sensors simultaneously, the Create2 would regularly return erroneous data. A limitation of the iRobot Create2 is that it sensors every 15 milliseconds. We had to develop solutions around that limitation.

After deciding to thread our sensors successively, rather than in parallel, and after verifying its correctly returned data, we were able to begin writing our main function to control the iRobot's motion. Our robot drives forward until the left, right, or both of these bumpers are pressed. Once pressed, the robot turns accordingly. We chose to check a bitwise 'and' of leftBump and rightBump first, because if we decided to check solely the left or right bumper, it would always evaluate to true even if both were pressed.

Within this control loop, we also check for presses of the Clean button. If the Clean button is pressed, we stop the roomba and wait for the Clean button's press to resume. We implemented a longer wait time in our method for button reading to allow for debouncing. Our song method is written in the interface and is called whenever the wheel drops or cliff sensors are triggered.

We were able to implement two of the three of the extra credit problems for this assignment. We implemented threading and using python's logging library to output distance changes, angle changes, unsafe events, and button presses to a file. This file is called 'execution.log'.

Evaluation:

Our program works well, except for when the cliff sensors are read. Implementing the reading of the cliff sensors seemed to be the culprit in scrambling our data and fundamentally destroying the roomba testing process. Our two most likely reasons for this were: the reading of too much data and bits were getting scrambled; or, we were unpacking the return byte wrong. The latter reason is unlikely considering all other sensor data was returned effectively. Besides the cliff sensors, we were satisfied with the responsiveness of our roomba software. We continued to use GitHub to manage our code, which helped us to successfully be able to revert back to the most recent known version of functional code when the cliff sensors proved fatal.

Allocation of Effort:

This project's group (#4), has three members: Miles Ziemer, Robert Carff, and Austin Staton. Each member of the group made significant contributions to the development, advancement, and completion of this project. Ziemer provided many of the mathematical calculations and gave structural input to ensure the project's completion; Carff wrote many functions of the code and guaranteed their effective internal documentation; Staton also contributed to the programming functionality and wrote the technical documentation. This group proved to be both effective and successful in their completion of Project 2.