CSCE 274 Section 001 Fall 2019 Project Report 2 (Rene Charles , William Lefler, Ryan Hinson)

Project 2 Description:

Building off the last project we used many of the functions from the first project in the Interface1.py program for project 2. For Task 1, we define def full(): and set it to connection.write(chr(132)) to add the possibility to set the robot in Full mode. We then define def getAngle(): and wrote an OP code to gather and store the sensor package and then stored data to be parsed. We then parsed the data and return the angle for the function which gets the angle turned since function was last used. We define def getDistance(): and wrote an OP code to gather and store this package data, followed by code to store data to be parsed and then parsing the data. We then return the distance in order to get the distance moved since function was last used. In order to collect data from all cliff sensors, we define def collectCliffData(): and then established code to collect data from the left, front left, right, and front right cliff sensors. We then return the left, front left, right, and front right cliff sensors in order to return a list of all the cliff sensor readings. We then define def driveDirect(lVelocity, rVelocity): which takes the velocity of the right wheel and left wheel. We wrote a specific code which passes command to the Robot through serial connection. Towards the end of the code, we define def timedDrive(seconds,lVelocity, rVelocity): to drive forward for a specified amount of time while pending a clean button press. The code endTime = time.time() + seconds which calculates end time and the code driveDirect(lVelocity,rVelocity) followed by cleanButton = False allows the Roomba to drive with given specified speed. We ended this function with a while loop that pends on button press while moving forward We define def createSongs(): which write song and def playWarning(): to play written song. For Task 2, we imported Interface1, time, and random. Interface1.start(), Interface1.full() puts robot in full mode, and Interface1.crreateSongs() creates defined warning sound. We then defined and set sensor variables to not be pressed. We then wrote a while loop while(keepRunning == False): which keeps program continuously running. Within the loop, print “Press Clean to start” to let user know that the robot is ready to start. There’s another while loop which waits for a button or bumper press to start random walk and defined a checkList to take sensor data from the robot and puts it into list for future use. We then wrote some variables to assign changes of sensors to variables to be checked later. We then wrote an if statement for if the clean button was pressed go forward and if bumpers were pressed, to turn around. In order to add the +/-45 degrees, we set velRange = random.randrange(-range,range). We then wrote if else statements like if(rBumper==True), saying if right bumper was pressed, turn counter clockwise and variables to reset both data variables incase both were pressed. The else statement if right bumper was press turn clockwise and variables to reset both variables incase bother were pressed. The variables wheelDrop = [False,False] and cliffData = [False,False,False,False] which adds wheel drop and cliff sensors for on the move instances. In this function, we wrote a while loop which stops movement and goes back to stopped instance if clean button is pressed or wheels drop and another while loop which waits for any of the sensors to trigger. We then wrote an if statement if clean button is pressed of wheels drop it stops the wheels followed by another if statement for if wheels drop play warning sound. If right bumper or right cliff trigger, we wrote an if statement to turn counter clockwise and if left bumper or left cliff sensors trigger, we wrote an else statement to turn clockwise. Within these if-else statements we wrote variables to reset all sensor variables. The code Interface1.driveDirect(Velocity, Velocity) which continue forward. To end our Test1.py, we wrote variables to reset the sensors that would cause the robot to stop if triggered. Evaluation: The Roomba in passive mode and is changed to full mode. It turns 180 degrees (+/- 45 degrees) then moves forward when it reaches an obstacle or when one of the bumpers is pressed. If the left bumper is pressed the robot turns clockwise,. This is also implied with the right bumper. When the cliff sensors are triggered, it causes the robot to turn exactly like the rules of the bumpers. The robot stops completely if the clean button is pressed regardless of the placement The robot also stops if the wheel drop sensors are triggered and a warning song plays when the wheels stop. The program also constantly checks the state of every sensor. In conclusion, the project compiles and runs correctly based on our interpretation of the given instructions.

Allocation:

Rene Charles - Wrote the report. Helped write Task 1 and 2. Rene wrote the use of Drive Direct and made sure the robot was initialized by setting it in passive and safe mode and did part c of Task 2. Ryan Hinson - Helped write Task 1 and 2. For task 1, Ryan wrote the readings of the Bumps and Wheel drops sensor and added the full mode and helped write part b of Task 2. William Lefler - Was the administrator of this project. Helped write Task 1 and 2. William wrote the warning song for task 1. The warning song he chose was a song from JoJo Bizarre Adventure. He was also responsible for part b of Task 2.