Driverless Vehicle System Implementation - README File

Having previously completed the planning for my driverless vehicle using UML and deciding on the three functions of adjusting the speed of the vehicle in accordance with speed signs, identifying and reacting to hazards and the ability to recall, input and save destinations within a GPS system, I have written a programme using Python that will do this, utilising OOP techniques such as classes and functions.

While approaching the development of this programme, I wanted to ensure that all variables were considered, with appropriate functions for each of them. For example, while working on the “SpeedSensor” class, I felt it appropriate to ensure that all speed signs that might be found on British roads were written. Although this added additional lines to the code, each function is similar except for how it changes the speed. This allows for each function to be smaller, thus avoiding the need for long and complex functions to run to cover all variables.

In contrast with the number of functions within the “SpeedSensor” class, I intentionally avoided providing additional functions for the “HazardSensor” class outside of one that detects all hazards. This was due to any hazard resulting in a reduction in speed, so separate ones were not necessary. This may potentially cause issues with a full rollout using something like LiDAR, however it should still work in a real world setting.

The “GPS” class operates as a simple list, with the ability to add, search and recall items from this that would act as the destinations for the journey. Along with this, this class contains a function which allows for current locations to be matched against inputted destinations, which in a full rollout should produce messages that confirm the start of the journey and the end of the journey automatically.

As my code grew in size, I wanted to be sure that each function could be called in a timely manner during implementation. This is particularly important with the “TripComputer”, “SpeedSensor” and “GPS” classes, as each of those has a number of functions. To meet this need, I opted to place each function within a dictionary for each class, which should allow for easier recall when required, rather than the system having to write out and execute long functions as they appear.

During the developmental process, I had planned on testing the programme by mimicking the kinds of inputs that may be received from a user interface or a sensor, however I was unable to validate whether these outputs would be accurate. The programme does run fully without any errors, as should be able to slow into a full driverless car system with input and output modules already in place. The intention for this piece of software is to support the function of the driverless vehicle, not to be the complete operating system.

The process to write this programme was, admittedly, rather difficult. I encountered issues with regards to linking the attributes of a particular class with its functions, however I was ultimately able to find solutions to these and was able to implement all of my three required functions by utilising resources such as w3schools.com and the Python 3 Object Orientated Programming (3rd Edition) book by Dusty Phillips. Upon reflection, I believe that this is an area of my skills as a computing professional that I need to continue to work at to ensure I can be an effective software developer in the future.

Word Count: 578