Readme File

System

The algorithm is intended to simulate and put into practice an Object Oriented Approach for a driverless car. Several functionalities are achieved through this program.

The Setup

The system is fed by a csv file extracted off a Microsoft (2019) Excel file, containing 100 rows and 7 columns with 7 variables:

- Time: starts from 0 all the way to 10 seconds in timesteps of 0.1 seconds. A smaller timestep would have achieved more accurate results, but in light of faster execution this timestep is presented. The system if fed a smaller timestep will respond positively, but please note, the frequency or attribute "*rate*" must be modified from 0.1 to 0.01 or whichever timestep preferred.
- Acceleration: simulating input from an IMU module at 100Hz, the acceleration function chosen is a relatively achievable function for a normal car, calculated with simple math:

$$a = -\frac{4}{25} * x^2 + \frac{40}{25} * x$$

  Note how the acceleration is manipulated on the csv to yield 0 as soon as the brake column displays True and then the software inverts the value during brake operation.

- Orientation: the car object is set to move in the x_y domain, basic trigonometry used to determine orientation within the program.

- Latitude and longitude: latitude and longitude inputs are placeholders opening way for the functionality to be implemented. For future implementation, latitude and longitude positions will be fed as initial position instead of 0,0 and then check for discrepancies between calculated coordinates and the ones given by the GNSS module.

- Obstacle: simulates input received by the obstacle detection module. In reality, this one would be a combination of lidar, radar and camera sensor technologies to determine whether obstacle should be set to True or False.

- Speed: does not serve specific function rather than comparison useful for debugging as system speed is already calculated on the software internally.

Updates

The system was updated to the latest feedback and deeper understanding of the system. Please, do feel free to refer to the attached document "*Feedback Applied*".

Functionality

The car system achieves several functionalities. Given how oversimplified it is, this system setup allowed the car to successfully:

- Increase speed: prints current speed and notifies the user that the vehicle is moving.
- Reduce speed: prints current speed and notifies the user that the vehicle is braking.
- Move in the x_y domain: the system successfully travels through the 2D space, given a numeric input in degrees, 0° being "*east*" and 180° being "*west*".
- Allows the ProcessUnit to log errors and count them on the hardware side, like a normal car would, like errors in light colour, blink rate, wheel size(only wheel sizes of less than 17 allowed), reverted velocity in electrical motor and invalid braking reading if it does not receive signal (True or False). This functionality can be further explored, the system is not passed any faulty input on the proposed csv, but the algorithm successfully checks and calls the error functions on runtime.

How to use the system

The autonomous car system is an oversimplification of the functionalities that a car can achieve, as well as the input is an assumption, contrary to having real reading from hardware components.

When inserting the destination, the software will let the user know if the input added is invalid, as it is really important for the code to work correctly. Please also note that the passed input will make the car move around an already specified area, if the destination added goes off these limits it will not arrive to destination.

Disclaimer

The system accuracy can be improved with a smaller step and more accurate positioning functioning. However, the functionality of the system will not be affected.

Thanks for using the software and I hope it added value to your understanding about the topic as it did to me.

References

Microsoft Corporation, 2019. Microsoft Excel. Redmond, WA: Microsoft.