# Summative Assessment: System Implementation

I began the development of this program with the previous module, available [here](https://github.com/turbits/essex-m2a1) as an essay. The overall premise was to design a partial autonomous vehicle system that contained three processes present in autonomous vehicles. I chose [Lane-keeping Assist (LKA)](https://tc.canada.ca/en/road-transportation/driver-assistance-technologies/lane-keeping-assistance), [Automatic Emergency Braking (AEB)](https://tc.canada.ca/en/road-transportation/driver-assistance-technologies/automatic-emergency-braking), and [Adaptive Cruise Control (ACC)](https://tc.canada.ca/en/road-transportation/driver-assistance-technologies/adaptive-cruise-control). These three systems I felt represented separate concerns as far as autonomy in a vehicle goes, with LKA controlling the direction of the vehicle, ACC controlling cruising speed, and AEB representing one of many critical safety systems designed to assist or take over in the event of an emergency detection.

Unfortunately, due to time constraints, the program is obviously not a full design and implementation of an autonomous vehicle or the aforementioned systems but serves as a representation of what one could look like. The backend generated information is basic and the functions of the vehicle are heavily simplified, i.e., they are immediate and do not take into account factors that would be exhibited on a real-world equivalent. In other words, they ignore physics. For example, if the vehicle is going 40km/h and the AEB detects an emergency stop is required, the vehicle's speed is set to 0 and the stop is immediate.

As part of the process of developing this program, I implemented threading to be able to run the backend data generation while the user was able to use the frontend command line interface. This is important as otherwise only the front or the backend could be used at any given time. The backend, once the simulation is started, is constantly generating the user vehicle, a traffic vehicle, and a road. It also tracks current statistics that can be pulled up by the user via the frontend.

Had I more time outside of work, one thing I would have liked to implement would be the Observer pattern. Fleshing out the detection of the vehicle to be more of a "real-world" equivalent, albeit simplified, along with a detection matrix. The subsystems of the Vehicle (LKA, AEB, ACC) would subscribe to the detection event and react accordingly. I regret not having the time to do this as I believe it would have added a level of autonomy to this program that is currently not present.

Another system that I began to work out but did not complete was the idea of system error codes and error handling. I worked out a simple, but in my opinion, affective and intuitive way of presenting an error to the user or to another system. For more context, see below Error Codes section and the “c\_err” function in the utility module.

In closing, I also created a REQUIREMENTS.MD file to list and provide context of the assignment requirements, it felt necessary to provide something like it given the complexity of the project.