Cruise Control Software Document

The Spinning Babingos:
Michael Chunko
Dominick DiMaggio
Ryan Mullens
Alex Heifler

I pledge my honor that I have abided by the Stevens Honor System.

March 31st, 2020

Version 0.08

Contents

1 Executive Summary

The goal of this project is to create an effective cruise control software. The software will be designed so as to run on an embedded system that meets all the stakeholders' requirements. The software will focus on having an intuitive user experience, so that the user will be able to focus on the road as much as possible. Additionally, the cruise control software will include safety features so as to ensure that the driver is completely safe and and to allow the driver to have a functional cruise control system.

2 Introduction

The purpose of our team is to create a cruise control software which runs from an embedded system placed in a contemporary vehicle. There are easily many risks associated with the automation of something so finely managed. Thus the code generated will be rigorously and frequently tested to ensure that there are no unintended or otherwise undesired functionalities in our product. Our cruise control software will be able to assist drivers on the road to drive safely and comfortably. The purpose of our team's software is to do this well, while ensuring the driver first and foremost remains in control of the vehicle. To that end, we want to stay out of the driver's way as much as possible, instead working purely to enhance the driving experience and avoiding anything overbearing that would be considered a hindrance.

Drivers traveling long distances often become fatigued after staying on the road for an extended period of time. Thanks to our cruise control implementation, the driver will be enabled to make the choice to let their car handle its speed, automatically staying in line with the speed set out by the user. As a result, our software will give users the reprieve they need to remain energized during their potentially long commute.

A cruise control system is mission-critical. There is no room for error or unexpected behavior. It is important that nothing is unexpected, lest an accident occur. Our software intends to carry minimal overhead, through use of planned programming practices and optimizations, and leveraging the fact that we are working close to the hardware on this system. During the software development process, our code will be tested thoroughly for fast response times, primarily in response to the driver's input, but also in interfacing with the sensors across the vehicle that we require for accurate operation. Test cases, use cases, and software specifications will be set concretely by an Agile development process and regular scrums between the team and stakeholders, to ensure that the software will closely match the desired specifications and meet all use cases.

We also would like to iterate that responsiveness and effectiveness are not viewed as mutually exclusive. On that thread, we refuse to consider that our software will be working in a pristine environment. While software does not deteriorate in quality, hardware certainly does. Mitigating hardware malfunctions is a major objective throughout our design. Through internal testing and with input from many drivers, we will produce a product that will never leave the driver in a position that will hurt them, regardless of potential hardware failure. More on this will be specified as our process matures.

3 Requirements

3.1 Functional Requirements

3.1.1. The cruise control will activate/deactivate via an input device

3.1.2. The cruise control will deactivate by the user pressing the brakes

3.1.3. The cruise control will temporarily deactivate while the throttle is pressed, then return to the set speed after the throttle is released

3.1.4. The cruise control will have its initial speed set to be the user's current driving speed

3.1.5. The cruise control will have its speed adjusted in 1mph increments via an input device

3.1.6. The cruise control will log the cruise control speed, if it was activated/deactivated, *why* it was deactivated, and the time all of this occurred

3.1.7. The cruise control will use output devices to inform the user of its activation, automatic deactivation, set speed, and if it cannot activate due to too low/high speed

3.2 Non–Functional Requirements

3.2.1. The cruise control will be highly reliable

- 3.2.1.a. It will have a reliability rate equal to or greater than four nines (99.99%)

3.2.2. The cruise control will react to being activated/deactivated, speed adjustments, and the user pressing the brakes/throttle in 100ms or less

- 3.2.2.a. This part of the system is real time and mission critical – even a small delay could result in accidents occurring

3.2.3. The cruise control will stay within 0.5mph of the set speed as anything more could result in a negative user experience

3.2.4. The cruise control will only work between 25mph (residential road speed limit) and 110mph

3.2.5. The cruise control will provide a good user experience

- 3.2.5.a. The user should be able to easily use the cruise control system with no prior experience
- 3.2.5.b. The use should be easily made aware of all the info (3.1.7) that they need to know through the output device(s)

3.3 Sensor & Hardware Requirements

As for sensor requirements:

3.3.1. The cruise control will have a brake sensor (to determine when the user pushes the brakes)

3.3.2. The cruise control will have a throttle sensor (to determine when the user pushes the throttle)

3.3.3. The cruise control will have access to the speedometer (to properly maintain speed)

3.3.4. The cruise control will have a clock (for logging)

And as for hardware requirements:

3.3.5. The cruise control will have an input device capable of sending data for activation/deactivation

3.3.6. The cruise control will have an input device capable of sending data for speed adjustments

3.3.7. The cruise control will have output device(s) capable of receiving data for info to display to the user (3.1.7)

3.3.8. The cruise control will have a processor capable of doing the needed computing

3.3.9. The cruise control will have digital storage with a capacity of 25MB to store the logged information

3.3.10. The cruise control will have access to the engine management system to properly control the car's speed

3.3.11. The cruise control will have access to both the car's battery and alternator to use as a power source

3.4 Security Requirements

In order to ensure that the cruise control system is secure it will have the following requirements:

3.4.1. The cruise control log file will only be able to be accessed through a hardware port

3.4.2. Read access to the cruise control log file will be password protected

3.4.3. Only the cruise control system will have write access to its log file

3.4.4. The cruise control log file will keep track of all read requests and whether or not access was granted

3.4.5. The cruise control shall have no connection to the internet, Bluetooth, or other similar networks

3.4.6. The only source of input/output for the cruise control will be through the sensor and hardware requirements enumerated in Section 3.3 (3.3.1-3.3.11)

4 Requirements Analysis Modeling

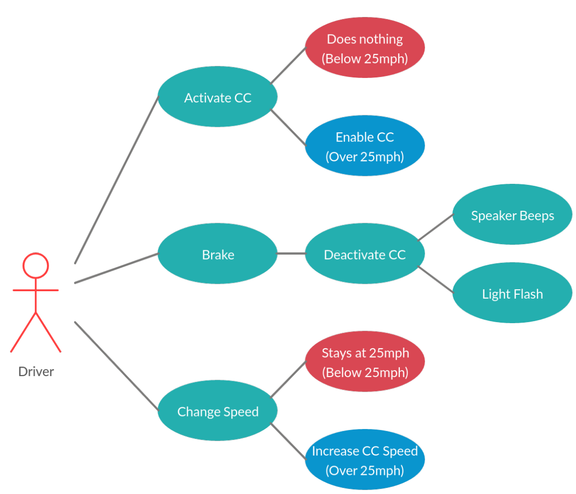
For brevity's sake, "CC" is occasionally used in place of "cruise control" throughout this section.

4.1 UML Use Cases & Diagrams

Note that the use cases here refer to input/output devices such as a display screen or a button. The cruise control software is only required to provide proper input and output, devices are not specified. We use these device names here for ease of reading.

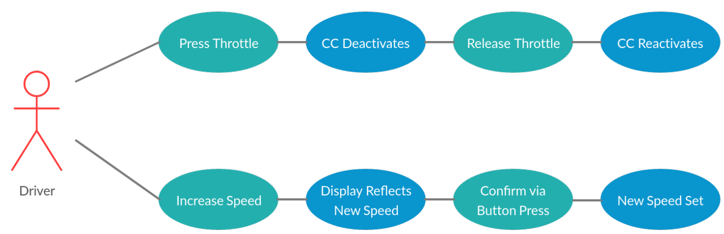
4.1.1 Use Case 1: User's first time using the cruise control system:

- While driving out of their neighborhood at 20mph, the user attempts to activate the cruise control out of curiosity
  - Because the user is below the minimum speed requirement of 25mph the cruise control does not activate
  - The light on the knob flashes and a beep plays to inform them of this
- The user then tries to set the speed using the knob
  - The speed on the display screen will not go below 25mph and will not allow the user to enable cruise control until they reach this speed
  - The user is made aware of the minimum speed requirement
- The user makes it to the highway and activates cruise control by pressing the button
- It works as the user expects (maintains the set speed)
- The user later spots something in the road and suddenly presses the brakes
- The cruise control automatically disables
  - This is to ensure that the user is always in full control of the vehicle
  - The user is again informed of this via a beep and a flashing light



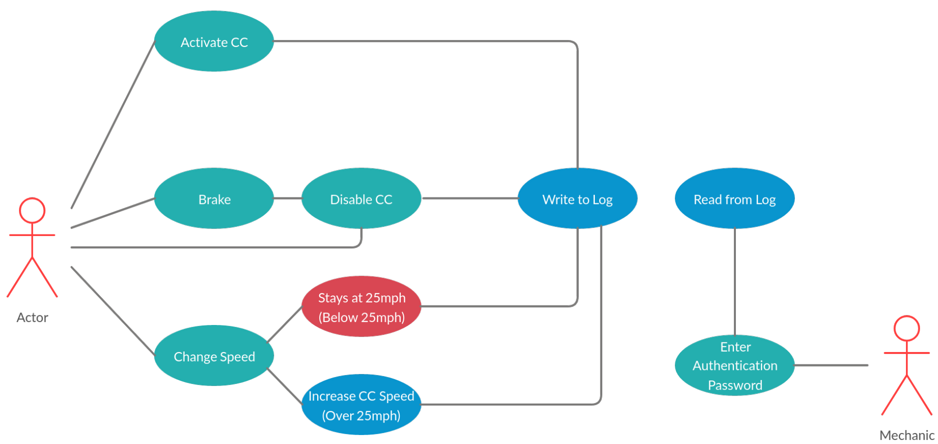
4.1.2 Use Case 2: User is cruising down the highway

- While using cruise control on the highway, the user presses the throttle to overtake another car
  - The cruise control temporarily deactivates to allow the user full control over the vehicle
  - After the throttle is released cruise control is automatically re–activated at the same speed that was previously set
- The user gets on to a stretch of highway with a higher speed limit
- The user adjusts the set cruise control speed to match this new limit
  - The displayed speed changes to reflect the user's new settings and is confirmed by a button press
  - The cruise control system adjusts the car's speed to this new speed at a comfortable rate



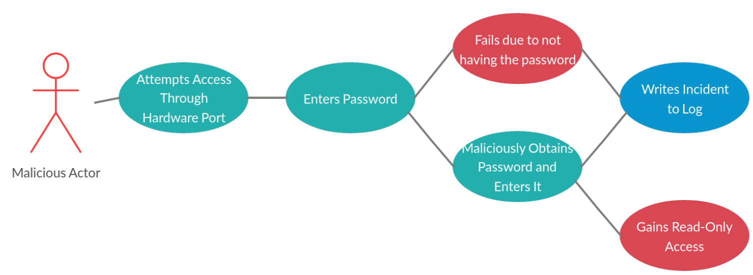
4.1.3 Use Case 3: The car goes in for regular maintenance

- The user brings the car in for regular maintenance
- The mechanic accesses the cruise control's log file through the hardware port as specified in Section 3.4 (3.4.1)
  - The mechanic is prompted for a password
  - They enter the manufacturer–provided password and are successfully granted read–only access
- From here the mechanic can read the contents of the cruise control log file to ensure that there have been no faults
  - The contents of the log file are as specified in Section 3 (3.1.16, 3.4.4)
- The mechanic verifies that the cruise control is working as intended and proceeds with the inspection



4.1.4 Use Case 4: A malicious actor attempts to interfere with the cruise control

- The user has done something to make the malicious actor attempt to damage the car
- The malicious actor starts by trying to damage the cruise control
- The malicious actor tries to access the cruise control through the hardware port
  - If they do not know the password, the cruise control logs this failed attempt and the malicious actor gets no access
  - If they somehow do know the password, the cruise control will log the access grant and the malicious actor will get read–only access
- If the malicious actor has successfully gotten read–only access, they won't be able to do anything harmful
  - Read–only access by definition means that the malicious actor will only be able to read the contents of the log and not modify them in any way
- If the malicious actor instead tries getting access to the cruise control through other systems in the car they will fail
  - According to Section 3.4 (3.4.5, 3.4.6) the cruise control will not have any unnecessary connections
  - As the connected systems are also mission critical, their security should provide that there is no way for a malicious actor to gain access to them



4.2 UML Class–Based Modeling

BrakeSensor
Type : Input Sensor Location : Car Internals Characteristics : Checks if brakes are pressed or not
is_pressed() on_press()

ThrottleSensor
Type : Input Sensor Location : Car Internals Characteristics : Checks if throttle is pressed or not. Activates/deactivates CC accordingly
is_pressed() on_press()

Speedometer
Type : Input Sensor Location : Car Internals Characteristics : Gets the car's current speed
get_speed()

Clock
Type : Input Sensor Location : Car Internals Characteristics : Gets current time in Unix time
get_time()

Input Device(s)
Type : Input Sensor Location : Car Cabin Characteristics : Send data for activation/deactivation. Send data for speed adjustments
set_status() set_speed()

Output Device(s)
Type : Output Actuator Location : Car Cabin Characteristics : Receive data for info to display to the user
display_status() display_speed() display_info()

4.3 UML CRC Model Index Cards

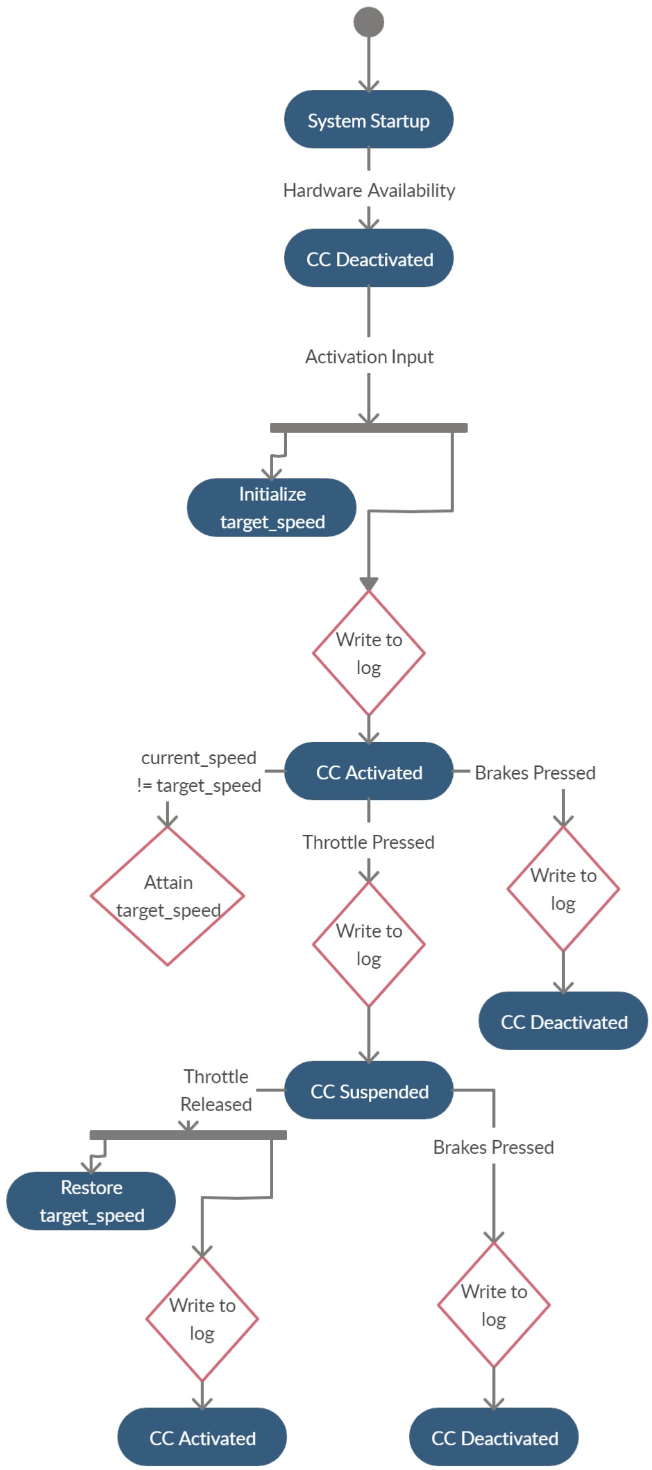
System Startup
State Variables : None
Do : Poll for hardware Availability Then : Enter "CC Deactivated" state

CC Deactivated
State Variables : current_speed
Do : Poll for activation input Then : Enter "CC Activated" state and initialize target_speed with current_speed

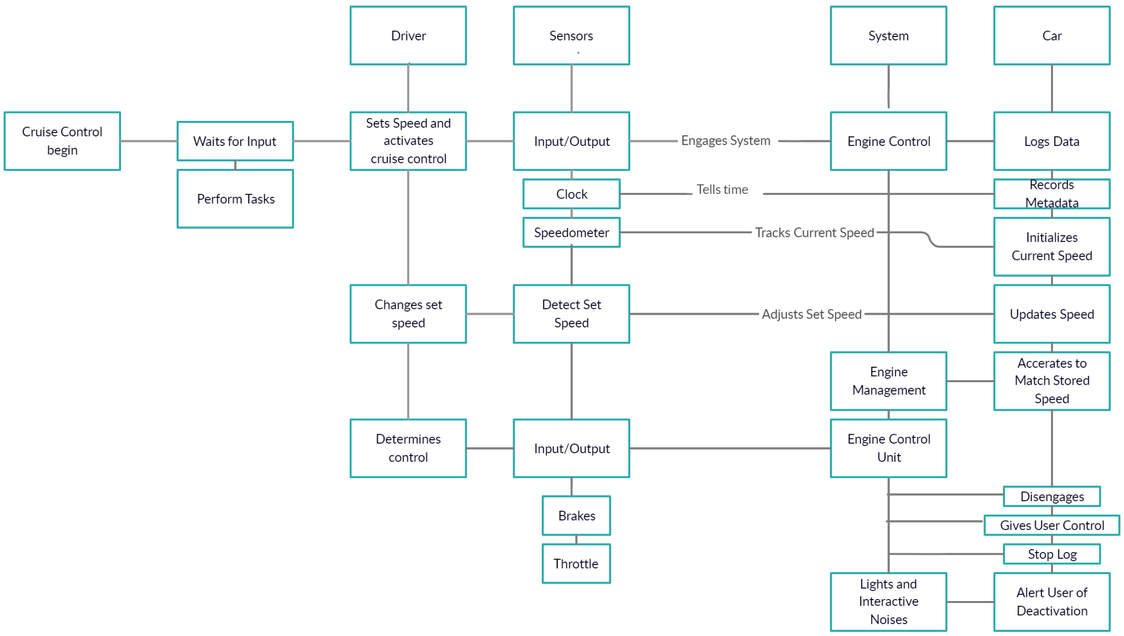
CC Activated
State Variables : current_speed, target_speed, log_fp
On Startup : Write to log file Do : If current_speed does not equal target_ speed, progressively attain it Do : If brake is pressed, write to log and enter "CC Deactivated" state Do : If throttle is pressed, write to log and enter "CC Suspended" state

CC Suspended
State Variables : target_speed, log_fp
Do : If throttle is released, enter "CC Activated" state and restore target_speed Do : If brake is pressed, write to log and enter "CC Deactivated" state

4.4 UML Activity Diagram



4.5 UML Sequence Diagram



4.6 UML State Diagram