OCEANCRUISE

Developed by OceanByte.

Ryan Hartman, Blake Marpet, Luke McEvoy, Dave Taveras

Version 1.08

Table of Contents

1. Executive Summary

We are OceanByte, a diverse and technical software development team that incorporates all aspects of agile software development. We are building a cruise control system that will take advantage of the sensors, embedded computers, and operating systems present in modern vehicles. Our goal is to develop a cruise control system that can be deployed to any modern vehicle by implementing a polymorphic infrastructure. In doing so, our OceanCruise system would be capable of running on virtually any hardware. By implementing a long list of convenience and safety features, enlisting strong cybersecurity practices, and prioritizing reliability, OceanCruise aims to be a versatile and convenient cruise control system deployed in the automation industry.

2. Introduction

The customer requires a cruise control system that can reliably keep a speed set by the driver until either the brakes are engaged or the cruise control is powered off. Furthermore, the customer should have a user interface on their dashboard that displays the set speed for both safety and convenience. Customers will also need the system to consistently maintain the driver's desired speed range as the car encounters obstacles such as potholes, inclines, or declines.

OceanCruise will implement a user interface that shows the current speed indicated by the car's sensors, influencing the cruise control's operations such as toggling modes or setting travel speed. By reading input from the vehicle's internal computers, our cruise control system has the ability to terminate the cruise control system when the brake pedal is engaged by the driver. This feature is essential in preventing accidents. An internal log will be kept to track potentially useful analytics. If permitted by the user, tracking vehicle data will allow OceanCruise to become a better experience for drivers by allowing easier detection of issues, vulnerabilities or potential bugs. Additionally, the cruise control system must be supported by the car battery in the case of an emergency shut off. Should the power supplied to the system ever be interrupted, emergency power off protocol will be initiated, allowing the system to log errors and perform housekeeping before powering down. Coupled with adequate security measures protecting against bad actors, OceanCruise will be a reliable, safe and convenient cruise control system.

Our cruise control system is designed to meet the needs of the consumer, as well as ensuring the safety of the driver and others in the vehicle. Engaging the breaks overrides the system and provides control of the vehicle back to the driver. Our cruise control system is fine-tuned to be responsive and accurate when engaged. In addition to our attention to detail, our system is designed with a solid cyber protection plan. OceanCruise’s software cannot be accessed remotely, is never connected to external networks, and can only be accessed from certified technicians or authorized car dealerships. Further, the cruise control system is recognized as a real-time software with hard, low latency response times that prevent the delay of user input or safety overrides. We have put great thought into our OceanCruise system, making sure to prioritize the safety of the user at all stages of operation.

The cruise control product must work in all conditions, areas and times. It must react to the loss of traction as indicated by the car’s TCS (Traction Control System), reducing speed and prompting user input as warranted. Furthermore, if there is any disruption to the embedded system such as loss of power, the cruise control will ease down to a lower speed instead of stopping the car altogether. If this strategy were not employed, the unexpected termination of the cruise control could lead to a tragic accident. In the case of 4 wheel drive vehicles, our system will work to utilize as little of the vehicle’s fuel as possible to increase fuel economy.

OceanCruise is designed to extend to many use cases. Personal car manufacturers can implement OceanCruise as a convenience feature for drivers, allowing them to set and maintain speed on highways. Commercial cars such as taxi cabs or police cars can similarly be fitted with our cruise control system to provide convenience and lessen driver fatigue. Trucks and other commercial transport vehicles rely heavily on cruise control systems in constant, long-distance and cross-country routes. Even recreational vehicles such as electric skateboards, longboards, and scooters use some form of cruise control, a perfect use case for OceanCruise as it is designed to be used on any hardware.

The team will adopt an agile software development lifecycle allowing for constant input from stakeholders and customers. With frequent revisits to the planning phase, the team ensures the product will be complete and meet initial expectations. Should the project ever deviate from initial goals, the costs to recover will be minimal. With weekly meetings and carefully crafted sprints, the team will work efficiently and meet project deliverables on time. Adequately designed sprints also remove the threat of “death marches”, large amounts of time spent towards a stagnant goal that demolishes team morale. Lastly, using weekly peer code reviews, the team ensures the cruise control is a robust, maintainable, and secure software. With careful consideration and review, the emergence of bugs should be minimized, having a very low impact on the mission-critical operations of the cruise control.

After initial production, OceanCruise could be expanded to incorporate more features. Technologies such as adaptive braking, automatic lane changing and driver wake-up calls could be explored as possible features for future versions. Eventually, the cruise control system could evolve into a fully *adaptive* cruise control system, automating the minutia inherent to highway driving. Our system may even be able to be implemented in real-time simulations to predict possible outcomes for autonomous vehicles. While this is a distant goal, it is definitely achievable with the solid fundamental cruise control system of OceanCruise.

3. Requirements

3.1 Functional Requirements

FR 1. The cruising speed set by the driver shall be maintained by the cruise control system, within a 1mph variance (e.g: if the speed is set to 35 mph, acceptable speeds are in the range of 34-36mph).

FR 2. The driver shall be able to terminate the system at any time by pressing the appropriate button inside of the vehicle or depressing the brake pedal.

FR 3. The cruise control system shall suspend operation while the driver is actively accelerating. After the driver has finished accelerating, the system shall continue to maintain the previously set speed.

FR 4. The cruise control system shall receive information from the EMS (Engine Management System) to monitor the vehicle's speed.

FR 5. The cruise control system shall receive information from the TCS (Traction Control System) to monitor potential traction hazards.

FR 6. The cruise control system shall terminate operation in the event of a warning issued by the TCS or EMS (blown tire, loss of traction, etc.).

FR 7. The cruise control shall send requests to the EMS to maintain speed and track statistics such as trip length, fuel mileage and distance traveled.

FR 8. The cruise control system shall display information to the driver in the car's dashboard such as current speed, trip duration or distance traveled while activated.

FR 9. The system shall require the vehicle to be moving at least 15mph before allowing the driver to enable the cruise control. If the vehicle is moving above 15mph, the system shall not allow the driver to set a cruising speed below 15mph.

FR 10. The cruise control system shall deactivate in the event of a collision. A collision is alerted by the EMS, which monitors collision sensors around the vehicle for potential crashes. The cruise control system shall subscribe to alerts from the EMS,

FR 11. The cruise control system shall allow the user to increment or decrement the current cruise speed in either one or five unit increments while cruising.

FR 12. If the cruise control is engaged and the car is turned off (see NFR #4), the system shall perform housekeeping operations such as flushing memory or updating logs before powering off.

FR 13. The cruise control shall periodically communicate with the vehicle's sensors, fetching current measurements such as speed, tire pressure, etc.

FR 14. The cruise control shall accept inputs from sensors after the engine is ignited.

3.2 Nonfunctional Requirements

NFR 1. The cruise control system shall reach 99.999% reliability (5 nines).

NFR 2. The cruise control system shall implement an internal clock for diagnostic and logging purposes.

NFR 3. The cruise control system shall use the car battery as a source of power while the engine is on.

NFR 4. The cruise control system shall exhibit hard response times to user input below 20ms.

NFR 5. The cruise control system shall be capable of being deployed to any hardware.

NFR 6. The cruise control system shall be supported by the vehicle's battery. This will allow for cleanup operations in the event the vehicle is turned off while the system is enabled (see FR 12).

NFR 7. Environment sensors capable of supplying the vehicle speed and brake pedal state.

3.3 Security Requirements

SR 1. The cruise control system shan't be accessible from any external network (WI-FI, Bluetooth, etc.).

SR 2. The cruise control system shall only be accessible by certified technicians or OceanByte developers.

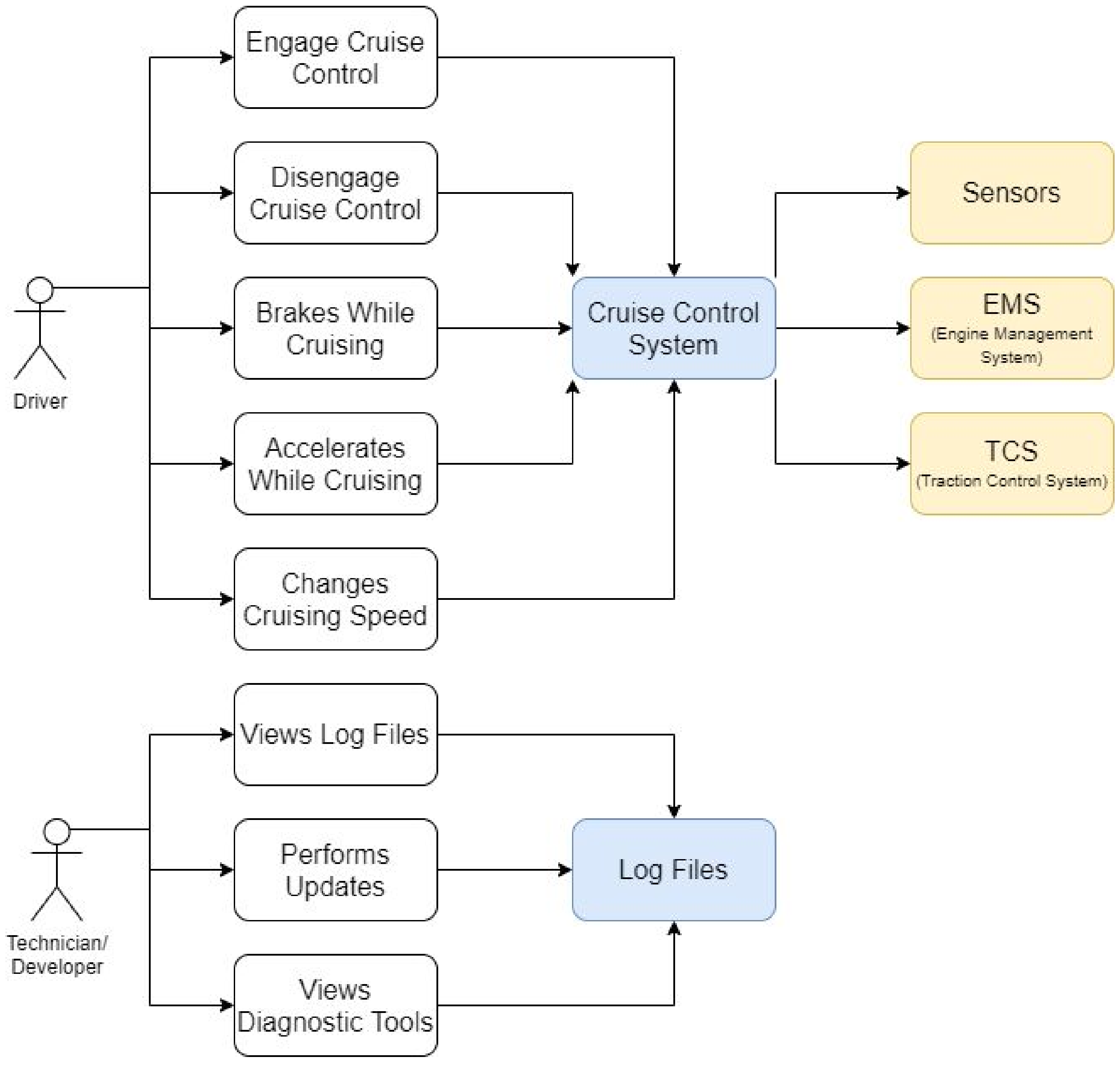
SR 3. All of the data generated or communicated by the cruise control system shall be encrypted.

SR 4. All firmware shall be digitally signed to prevent forgery or malicious interference.

SR 5. All attempts to access or modify parts of the cruise control system shall be logged in a log file with read only permissions.

4. Requirements Analysis Model

4.1 Use Case Diagram



4.2 Use Cases

Case 1. Driver engages cruise control

Iteration: 2 Last modification: February 29 2020 by R. Hartman

Actor: Driver

Goal: Start cruise control system and maintain the speed of the vehicle, or accelerate/decelerate to the speed set by the driver

Preconditions: The vehicle must be on and traveling at a minimum speed of 30mph.

Trigger: The driver presses the appropriate set speed or maintain speed button inside the vehicle

Sequence:

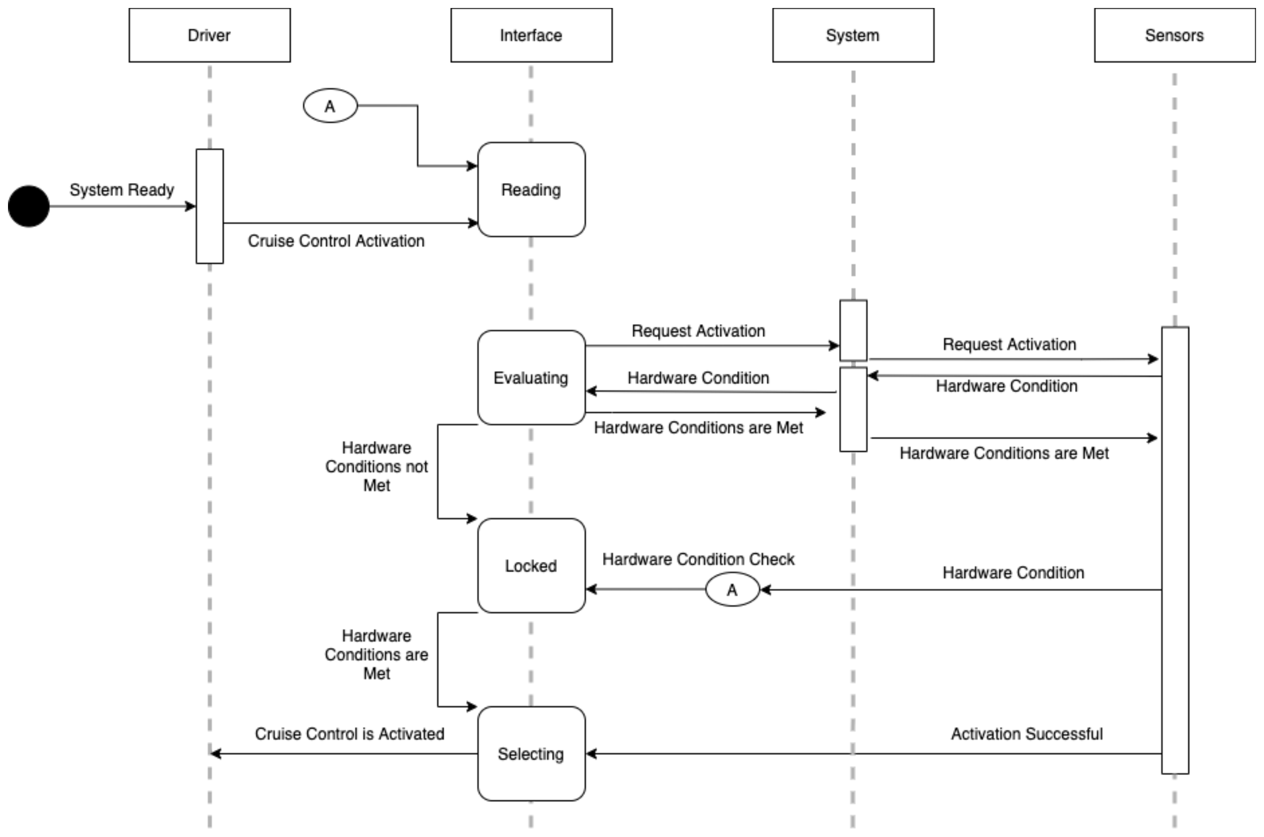
1) The driver requests the cruise control to be engaged by pressing the appropriate on/off button inside the vehicle
2) The cruise control system checks the status of the brake and accelerator pedals
3) The cruise control system requests the vehicle's current speed from the EMS
4) If the vehicle's speed is above 30mph and neither the brake nor accelerator are pressed, the system engages.
5) The system displays to the driver that the cruise control is "on" in the vehicle's dashboard
6) The cruise control system requests to set the speed of the vehicle to the speed set by the driver

Exceptions:

1. The driver attempts to engage the cruise control while the vehicle is traveling below 30mph. If so, the system must reject the driver's command upon receiving the current speed from the EMS, and display an error message on the dashboard interface.
2. A warning is published by a sensor or system. For example, if the TCS warns that the vehicle has lost traction, the cruise control system will disengage and notify the driver through the user interface.
3. The car is shut off while the cruise control is engaged. If notified

by the EMS that the car is shut off, the system will perform cleanup operations and log potential errors to its log file.

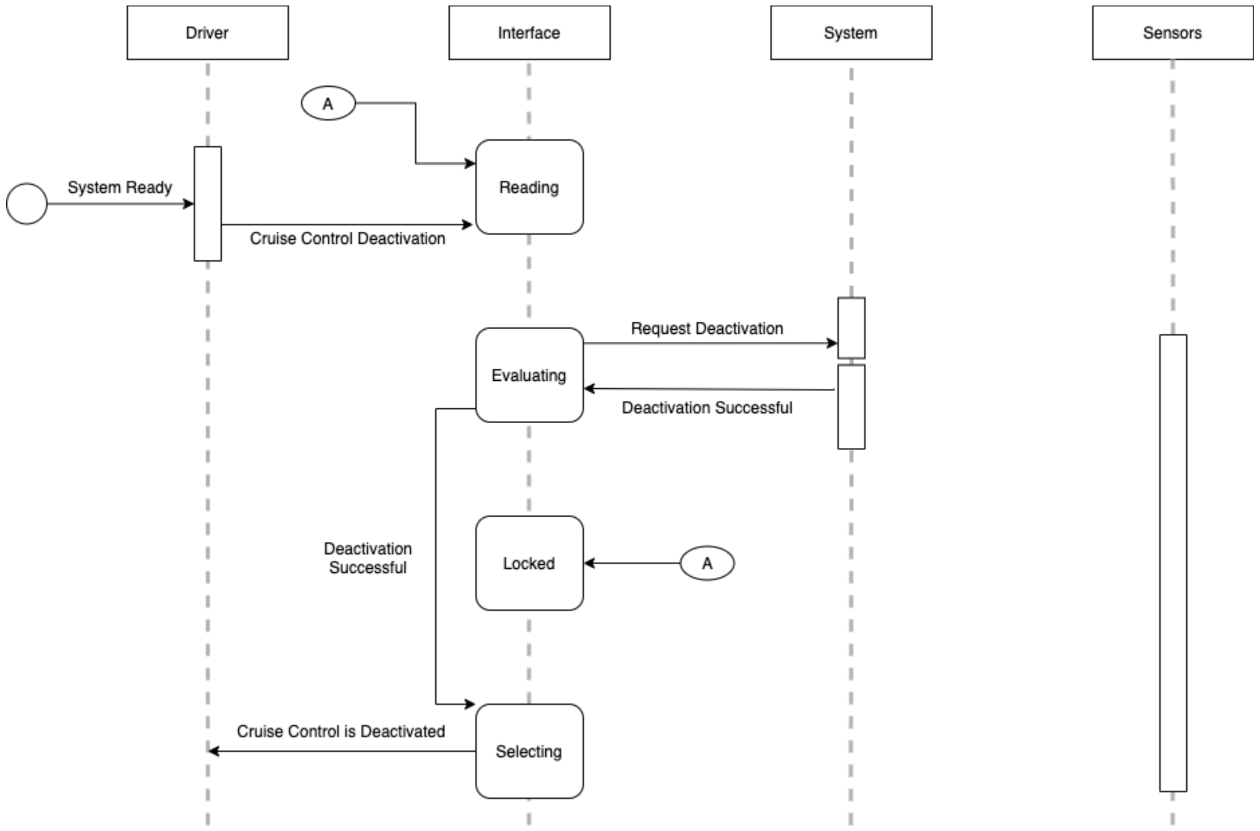
4. The driver attempts to engage the cruise control while pressing the accelerator, brake pedal, or both. If so, the system will not engage and instead display an error message to the driver in the user interface.
5. The driver attempts to maintain a speed greater than 90mph. In this case, the cruise control system will ignore the driver's request and instead display an error message in the user interface, suggesting the driver lower their speed before engaging the cruise control.



Case 2. Driver disengages cruise control using "off" button

Iteration: 2 Last modification: February 29 2020 by R. Hartman

Actor:	Driver
Goal:	Shut off cruise control system
Preconditions:	The cruise control system must be engaged
Trigger:	The driver presses the appropriate “off” button to disable the cruise control inside the vehicle.
Actions:	1) The driver presses the “off” button to disable the cruise control system. 2) The cruise control system immediately switches to a disengaged state. 3) The system requests the EMS return to normal operation and not maintain a set speed. 4) The system notifies the user through the dashboard of the vehicle that the cruise control has been disengaged.
Exceptions:	1. Driver presses the button to disable the system while the cruise control is not engaged. In this case, the system simply does not respond to the request, and silently logs the driver’s input in its log file.

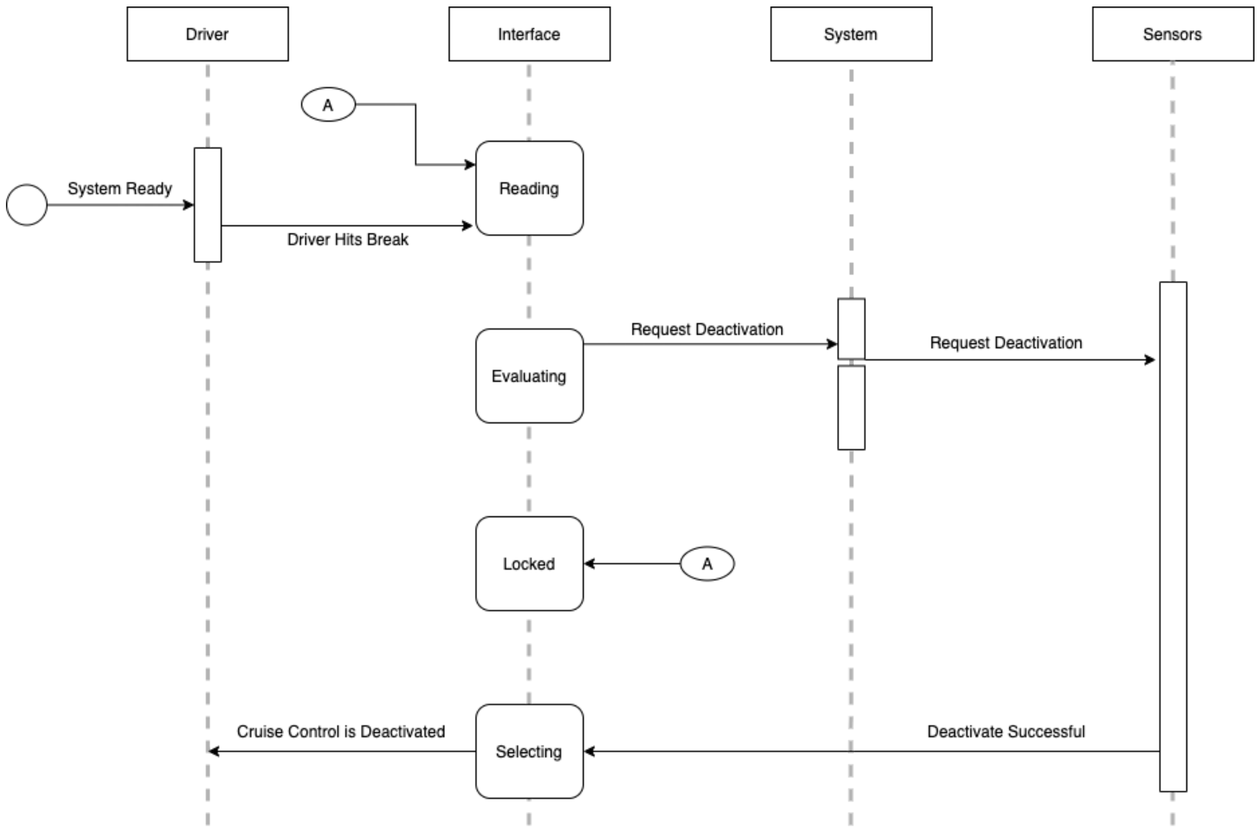


Case 3. Driver brakes while cruising

Iteration:	2 Last modification: February 29 2020 by R. Hartman
Actor:	Driver
Goal:	Shut off cruise control system
Preconditions:	The cruise control system must be engaged
Trigger:	The driver depresses the brake pedal while the cruise control system is engaged.
Actions:	1) The driver depresses the brake pedal while cruising. 2) The cruise control system immediately switches to a disengaged state. 3) The system requests the EMS return to normal operation and not maintain a set speed. 4) The system notifies the user through the dashboard of the vehicle

that the cruise control has been disengaged.

Exceptions:
1. Driver presses both the brake pedal and the gas pedal. In this case, the system carries on as if only the brake pedal was pressed, disengaging and displaying that the system is off in the user interface.



Case 4. Driver changes speed while cruising

Iteration: 2 Last modification: February 29 2020 by R. Hartman

Actor: Driver

Goal: Accelerate/decelerate to newly desired speed and maintain that speed

Preconditions: The cruise control system must be engaged

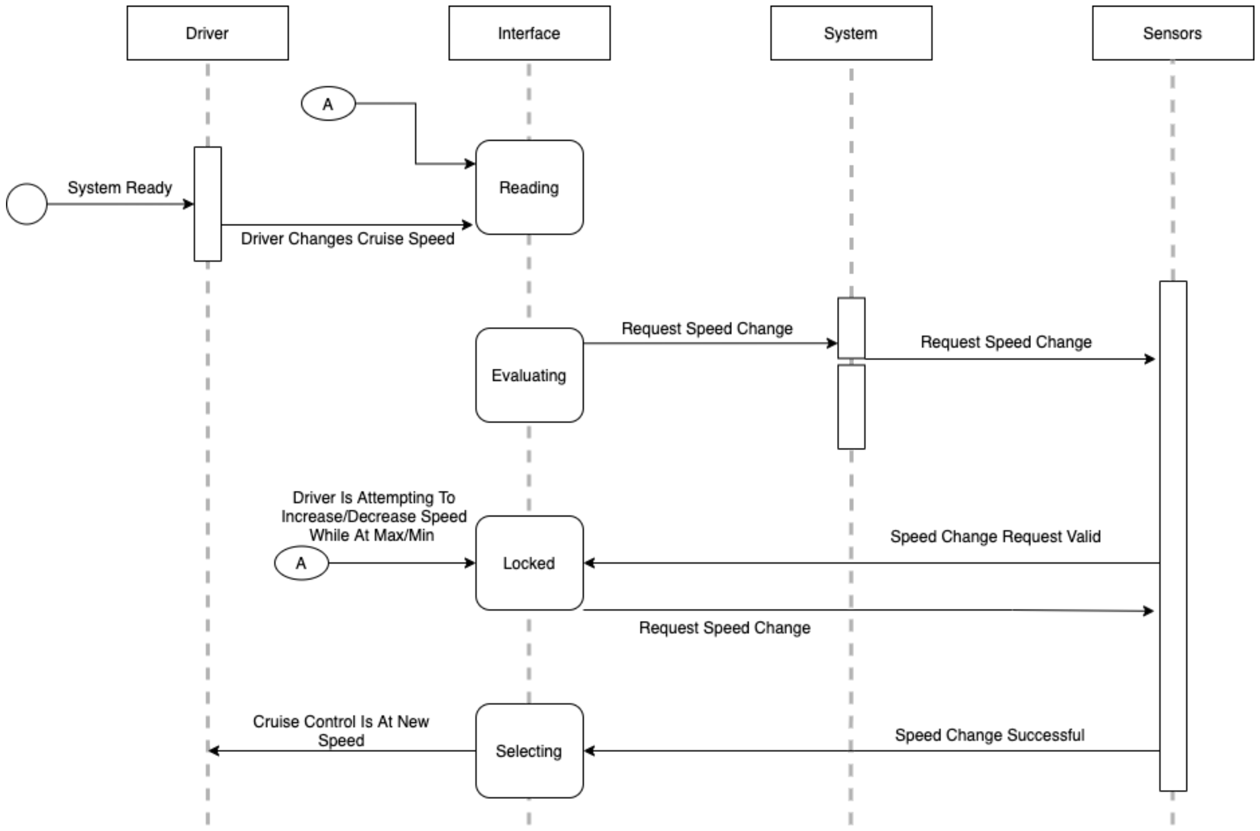
Trigger: The driver increases the cruising speed by 1mph or 5mph increments using the appropriate buttons inside the vehicle

Actions:

1) The driver increments or decrements the cruising speed using the appropriate buttons inside the vehicle
2) The cruise control checks to see if the requested speed is within the valid range (30mph to 90mph)
3) If it is not, the cruise control system ignores the request and keeps the current set speed and displays an error to the driver through the vehicle's dashboard display.
4) If the requested speed is within range, the cruise control system sets the EMS's speed to the newly requested speed
5) The cruise control displays the new speed in the vehicle's dashboard display.

Exceptions:

1. The driver attempts to decrease the speed below 30mph. In this case, the cruise control system will simply ignore the request after checking that it is out of bounds, and keep the cruising speed at 30mph.
2. The driver attempts to increase the speed above 90mph. In this case, the cruise control will simply ignore the request after checking it is out of bounds, and keep the cruising speed at 90mph.

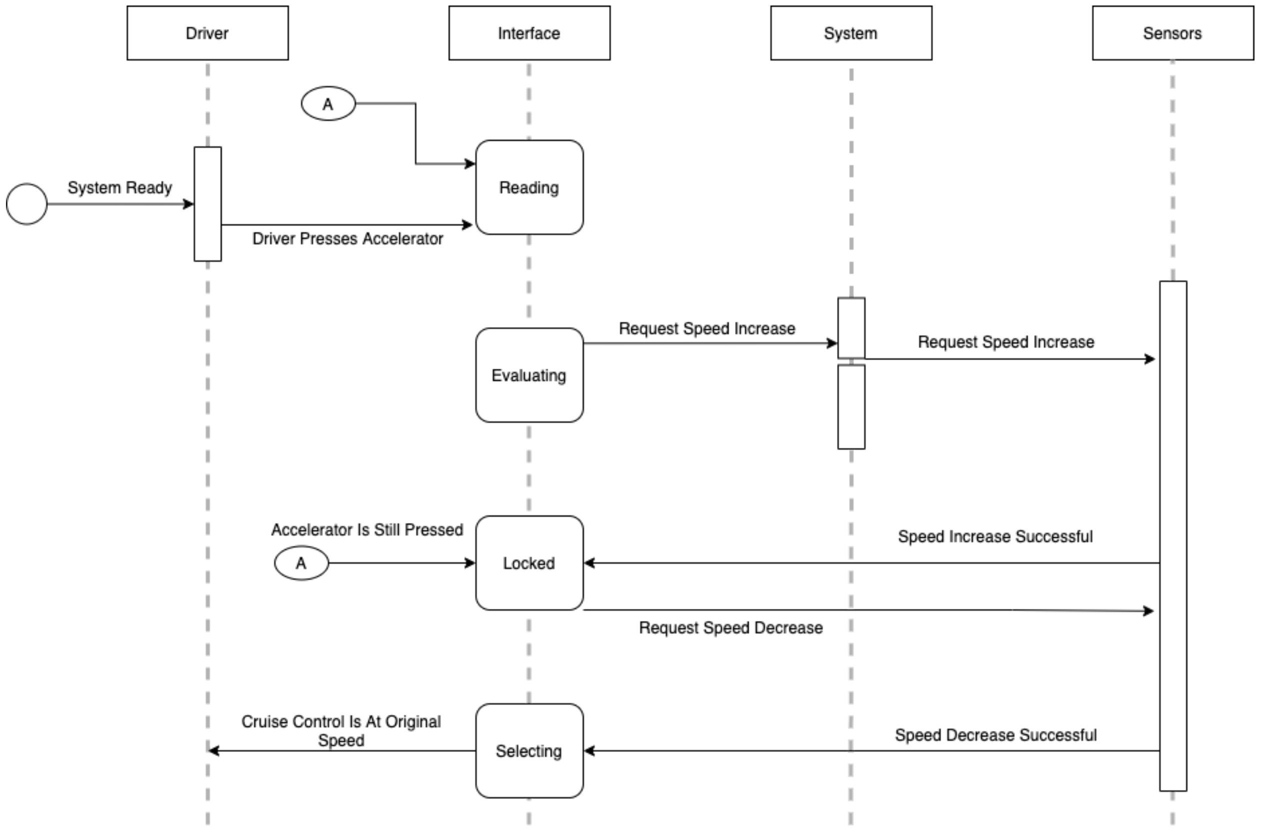


Case 5. Driver presses accelerator while cruising

Iteration:	2 Last modification: February 29 2020 by R. Hartman
Actor:	Driver
Goal:	Allow the car to accelerate, and return to maintaining the cruising speed after the accelerator is released.
Preconditions:	The cruise control system must be engaged
Trigger:	The driver depresses the accelerator while the cruise control is active.
Actions:	1) The driver depresses the accelerator while the cruise control is engaged. 2) The system suspends operation and allows the accelerator to communicate directly with the EMS to change the vehicle's speed. 3) The driver releases the accelerator

4) The system returns to its engaged state
5) The cruise control requests the EMS maintain the speed previously set by the driver.

Exceptions: 1. The driver presses both the accelerator and brake at the same time. If so, the cruise control system will deactivate, as if only the brake pedal was pressed (see *Use Case 3*).



Case 6. Technician or Developer Views Log Files

Iteration: 2 Last modification: February 29 2020 by R. Hartman

Actor: Technician or Developer

Goal: Display log files, and other system components to an authorized user.

Preconditions: The system is disengaged and the user is physically plugged into the system

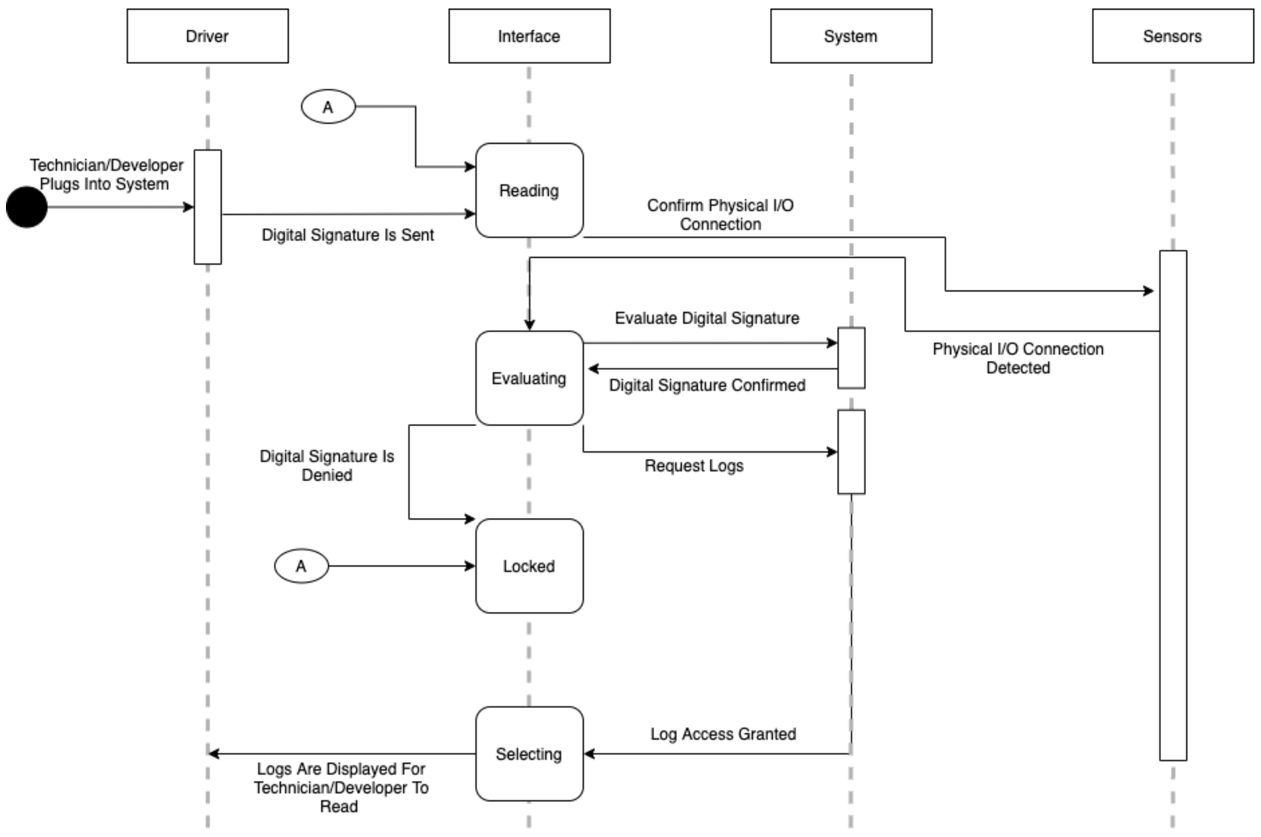
Trigger: A user attempts to access the system through its diagnostic port.

Actions:

1) The user attempts to access the system .
2) The system authenticates the user by requiring a username and password.
3) If the user cannot be authenticated (wrong password or username), the system records the failed login attempt in its read-only log file.
4) If the user is authenticated, the system records the successful login attempt in its read-only log file.
5) The system displays diagnostic tools, access to log files and other system components to the authorized user.

Exceptions:

1. User is not authorized to access the system. In this case, the system will prohibit access to the system for two hours, and log the access attempt to a read only log file.


Driver
Interface
System
Sensors
A
Technician/Developer Plugs Into System
Reading
Digital Signature Is Sent
Confirm Physical I/O Connection
Evaluate Digital Signature
Evaluating
Digital Signature Confirmed
Physical I/O Connection Detected
Digital Signature Is Denied
Request Logs
A
Locked
Log Access Granted
Selecting
Logs Are Displayed For Technician/Developer To Read

Case 7. Technician or Developer Performs An Update

Iteration: 2 Last modification: April 17 2020 by D. Taveras & L. McEvoy

Actor: Technician or Developer

Goal: Update the software and firmware of the Cruise Control system.

Preconditions: The system is disengaged and the user is physically plugged into the system

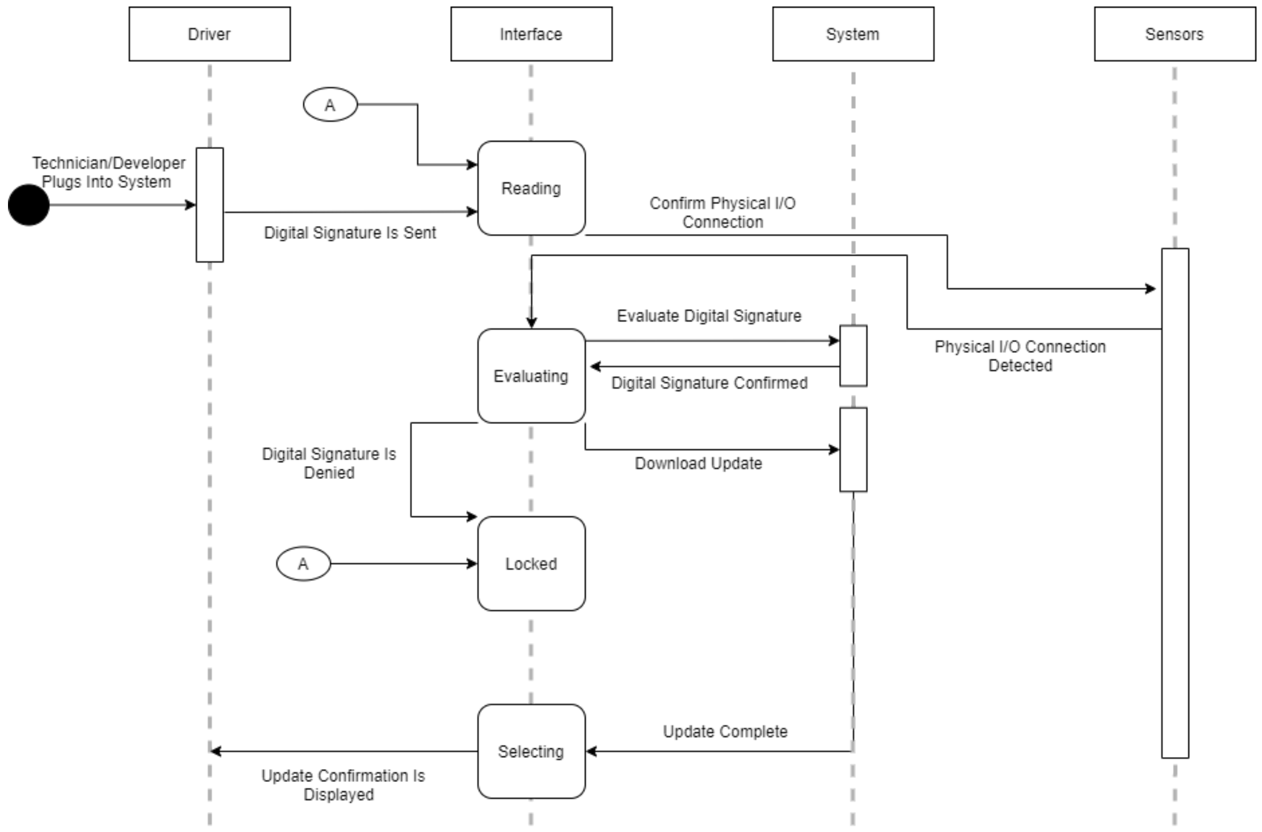
Trigger: A user attempts to access the system through its diagnostic port.

Actions:

1) The user attempts to access the system .
2) The system authenticates the user by requiring a username and password.
3) If the user cannot be authenticated (wrong password or username), the system records the failed login attempt in its read-only log file.
4) If the user is authenticated, the system records the successful login attempt in its read-only log file.
5) The system displays a review and confirmation of the update that just took place to the user.

Exceptions:

2. User is not authorized to access the system. In this case, the system will prohibit access to the system for two hours, and log the access attempt to a read only log file.


Driver
Interface
System
Sensors
A
Technician/Developer Plugs Into System
Reading
Digital Signature Is Sent
Confirm Physical I/O Connection
Evaluate Digital Signature
Physical I/O Connection Detected
Evaluating
Digital Signature Confirmed
Digital Signature Is Denied
Download Update
A
Locked
Update Complete
Selecting
Update Confirmation Is Displayed

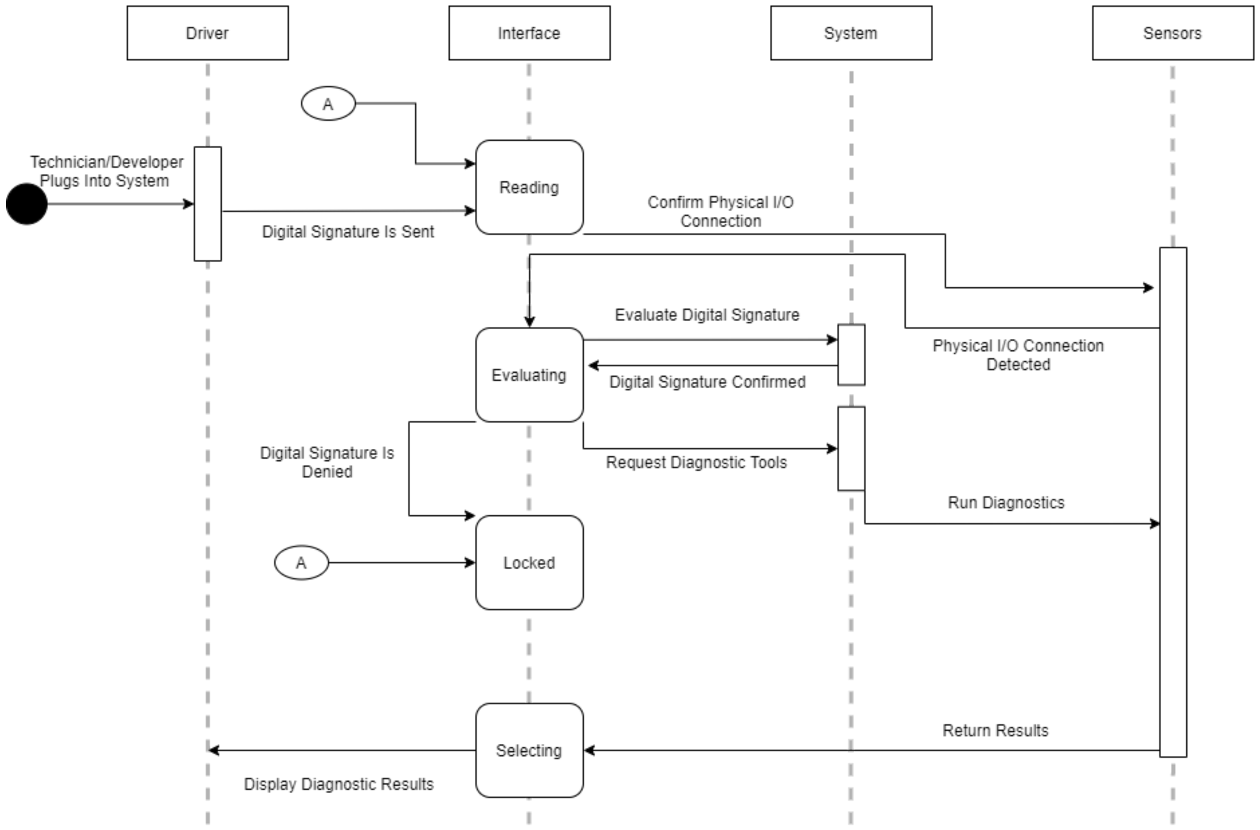
Case 8. Technician or Developer Views Diagnostic Tools

Iteration:	2 Last modification: April 17 2020 by D. Taveras & L. McEvoy
Actor:	Technician or Developer
Goal:	Display diagnostic tools, and other system components to an authorized user.
Preconditions:	The system is disengaged and the user is physically plugged into the system
Trigger:	A user attempts to access the system through its diagnostic port.
Actions:	1) The user attempts to access the system . 2) The system authenticates the user by requiring a username and password. 3) If the user cannot be authenticated (wrong password or username), the system records the failed login attempt in its

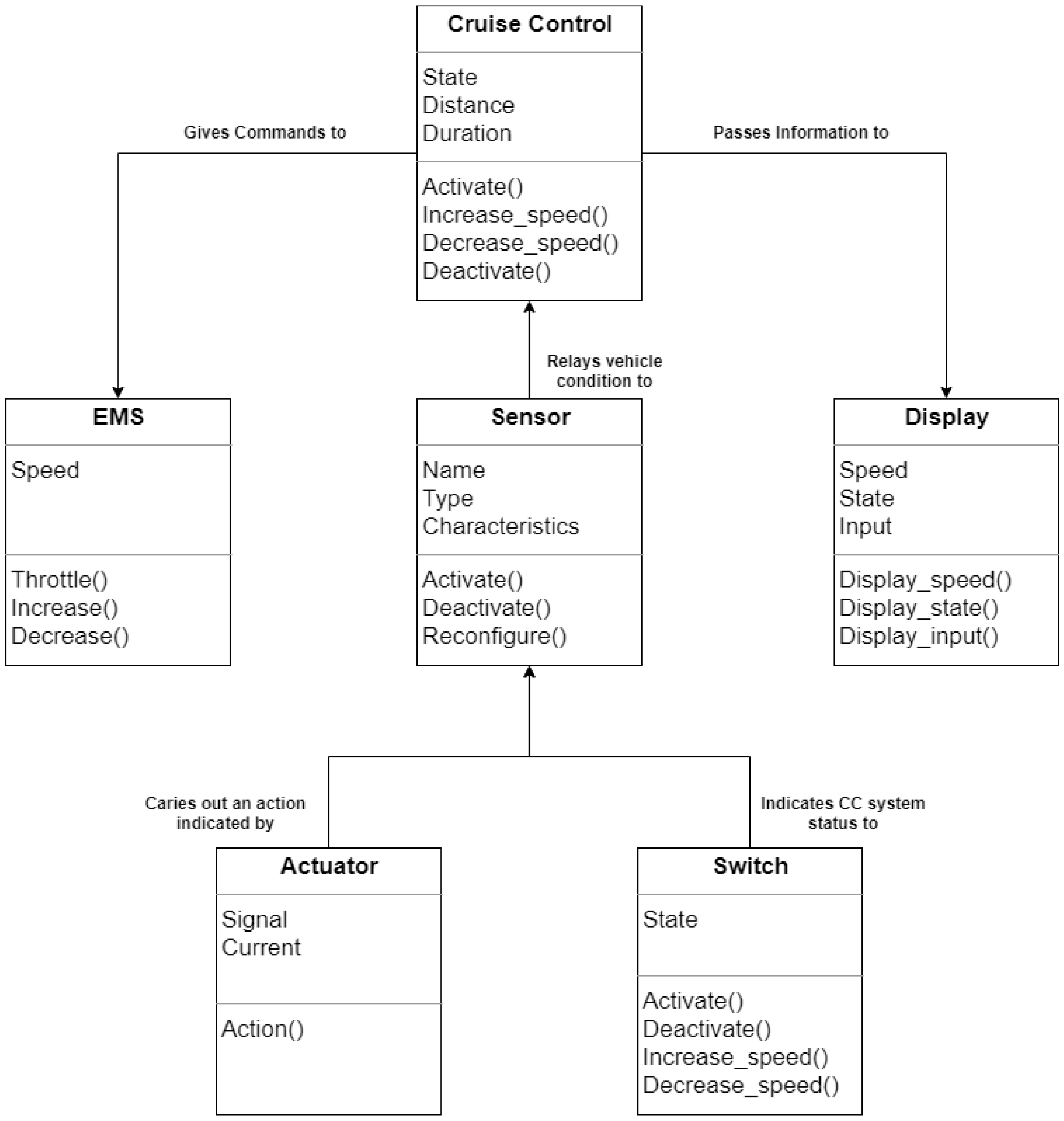
read-only log file.

4) If the user is authenticated, the system records the successful login attempt in its read-only log file.
5) The system displays diagnostic tools and other system components to the authorized user.

Exceptions: 3. User is not authorized to access the system. In this case, the system will prohibit access to the system for two hours, and log the access attempt to a read only log file.


Driver
Interface
System
Sensors
A
Technician/Developer Plugs Into System
Reading
Digital Signature Is Sent
Confirm Physical I/O Connection
Evaluate Digital Signature
Physical I/O Connection Detected
Evaluating
Digital Signature Confirmed
Digital Signature Is Denied
Request Diagnostic Tools
Run Diagnostics
A
Locked
Return Results
Selecting
Display Diagnostic Results

4.3 UML Class Diagram



4.4 UML CRC Model Index Cards

Class: Cruise Control	
Manages the operations of the vehicle's cruise control system.	
Responsibility	**Collaborator(s)**
Display current state of the system	Display
Store the current set speed	EMS
Set desired speed	EMS, Sensors
Change state of the system	EMS, Sensors

Class: EMS	
Endpoint for the Engine Management System, where throttle adjustments are requested by the Cruise Control	
Responsibility	**Collaborator(s)**
Accelerate/decelerate to a set speed	Sensors, Cruise Control
Maintain a set speed	Sensors

Class: Sensor	
Various devices that monitor input or communicate measurements to the Cruise Control	
Responsibility	**Collaborator(s)**
Send data to Cruise Control	Cruise Control

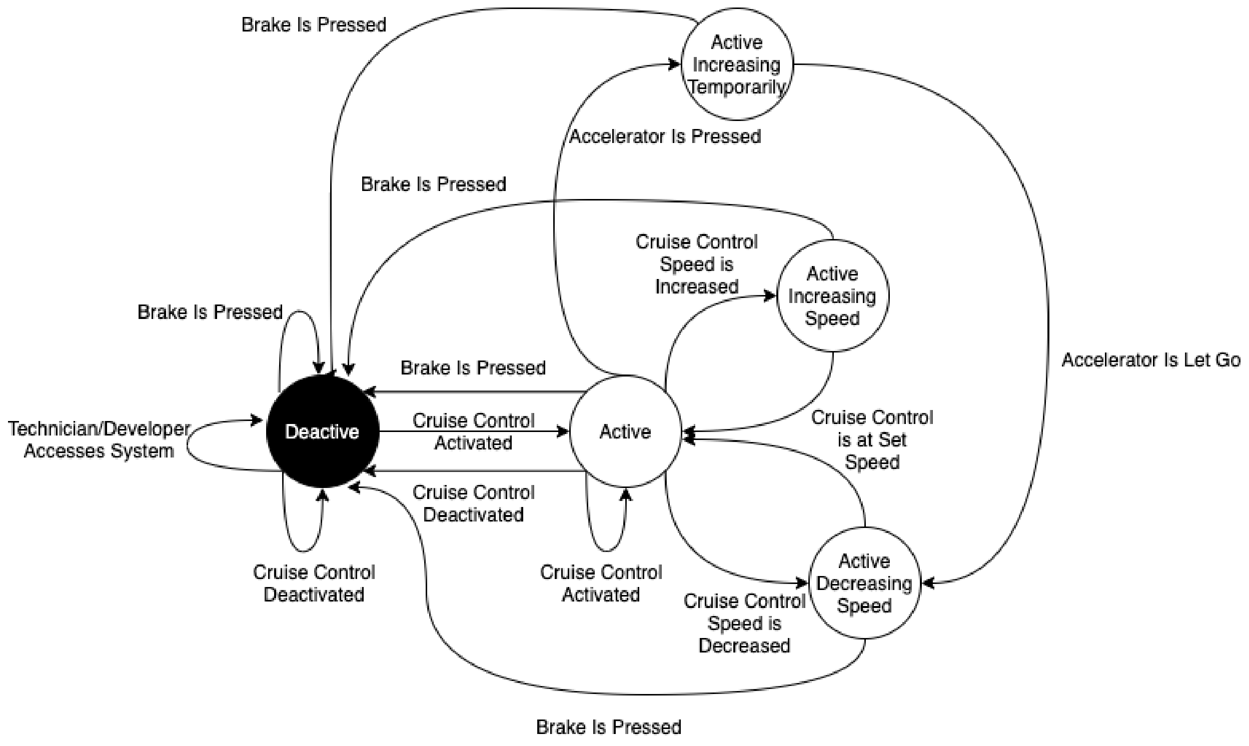
Class: Actuator	
A sensor that can be set to a range of values, e.g: Accelerator pedal	
Responsibility	**Collaborator(s)**
Send data when read	Cruise Control, Sensors

Class: Switch	
A sensor that can be in two states, on or off.	
Responsibility	**Collaborator(s)**

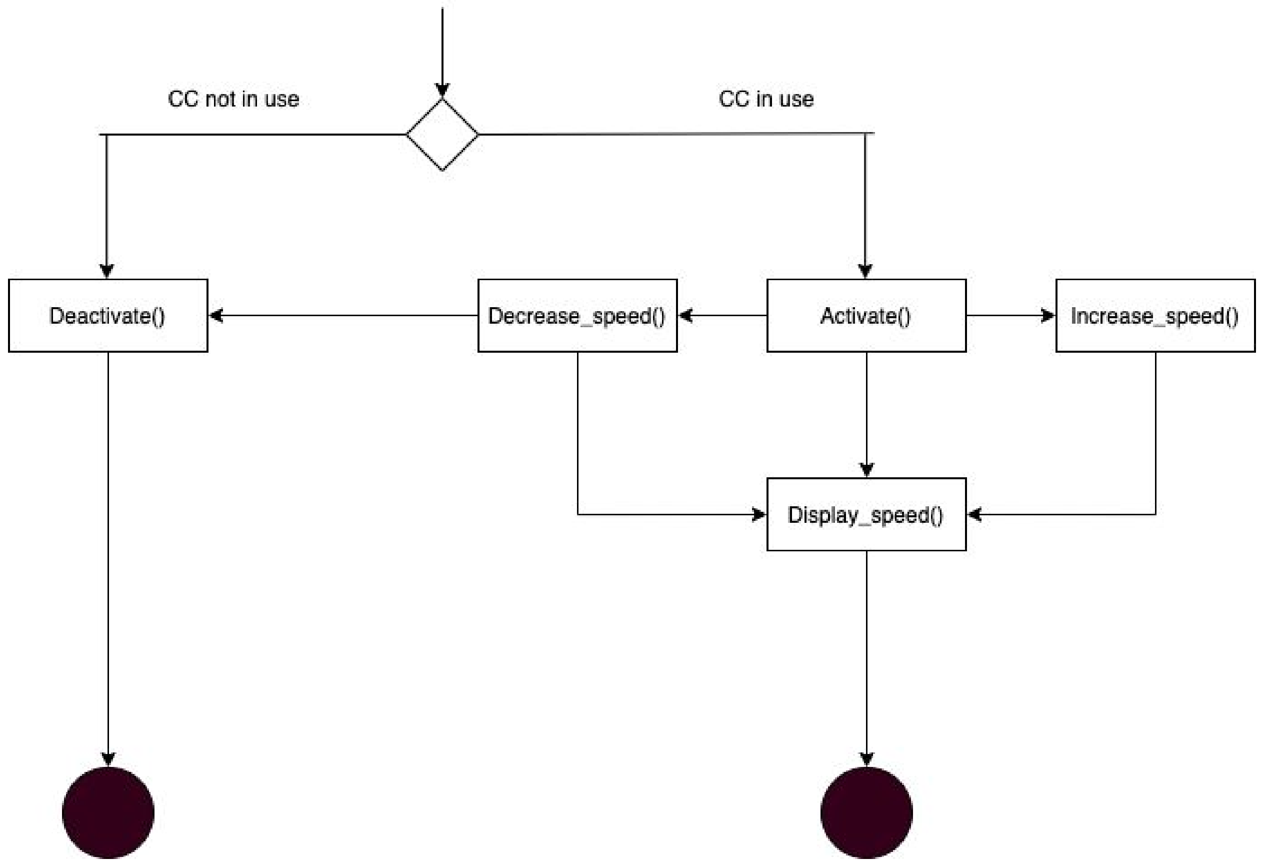
Send data when state is changed	Cruise Control, Sensors

Class: Display	
Display located in the vehicle's dashboard to show status messages to the user.	
Responsibility	**Collaborator(s)**
Report current cruising speed	Cruise Control
Report current state of the system (enabled or disabled)	Cruise Control

4.5 UML State Diagram



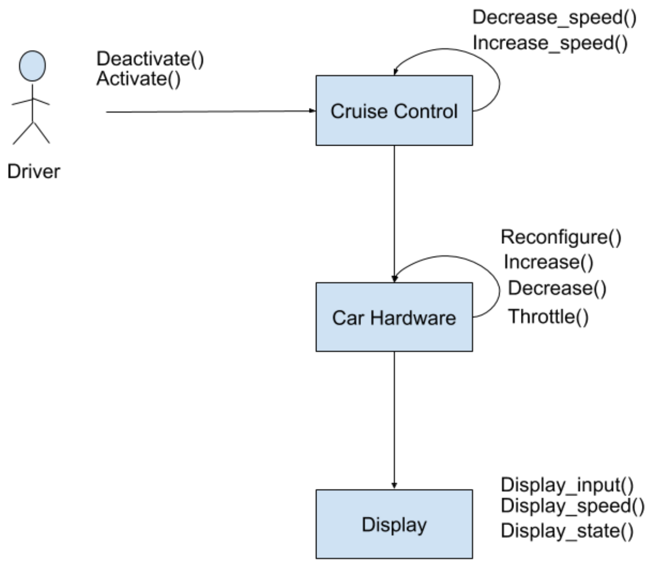
4.6 UML Activity Diagram



5. Software Architecture

5.1 Architecture Style

Considering the pros and the cons of each we felt that our cruise control system should utilize an Object Oriented Architecture because one of the biggest benefits of an OOA is code reusability and modularity. This will allow us to maintain our agile development process while reducing risk management. Additionally, an object oriented architecture closely models real world architecture, making it easier to visualize how to carry out our tasks while maintaining a relatively user friendly style. Object Oriented Architecture for the cruise control system would be optimal for the cruise control system because the CC system is manipulated by operations of the objects encapsulated within it. Data from the sensors is accessed through these operations and all of the components of the system pass messages to each other.



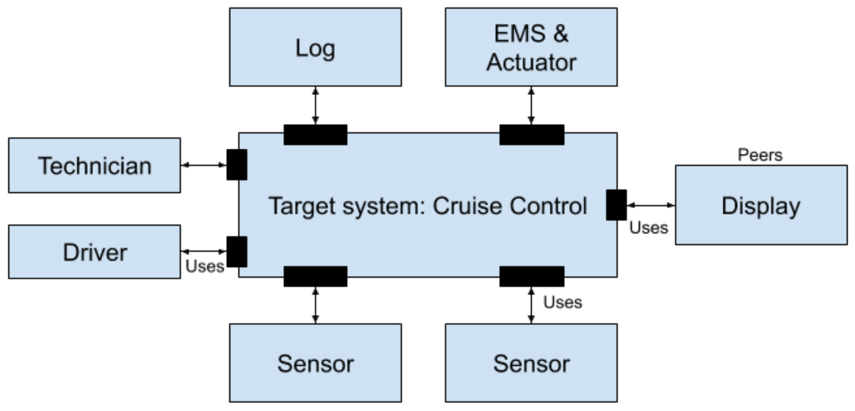
5.2 Components, Connectors and Constraints

The components for a system are all the parts that perform a function required by the system. In the cruise control system, the set of components are the controls for the system that appear on the steering wheel and the visual display on the dashboard. The connectors of a system are all the parts that allow the components to work together. In the cruise control system, the set of connectors are the different sensors, like the switches and the actuators, and the engine management system. The constraints for the cruise control system are items such as outside parameters such as weather that can alter sensor readings.

- Economy - The cruise control system will be easy to use and understand to provide drivers with a very user friendly system. There will not be any unnecessary information provided to the driver, and the driver will only see what is needed in order to control the system.
- Visibility - The cruise control system will be designed in such a way that anyone reviewing or editing the system will be able to tell that it is using an object oriented architecture. This can be done through how we structure our system and code for the system.
- Spacing - The cruise control system will be utilizing an object oriented architecture. This means that the system will be dependent on any previous computations that occur.
- Symmetry - The cruise control system will be designed in such a way that there is consistency throughout the whole system. In order to do that, it should be clear what each aspect of the cruise control system is meant to do.
- Emergence - The cruise control system will be a self-reliant system that is not dependent on any outside factors. In order for the cruise control system to run, all that will be needed is the driver, the system, and a few sensors on the car.

5.3 Control Management

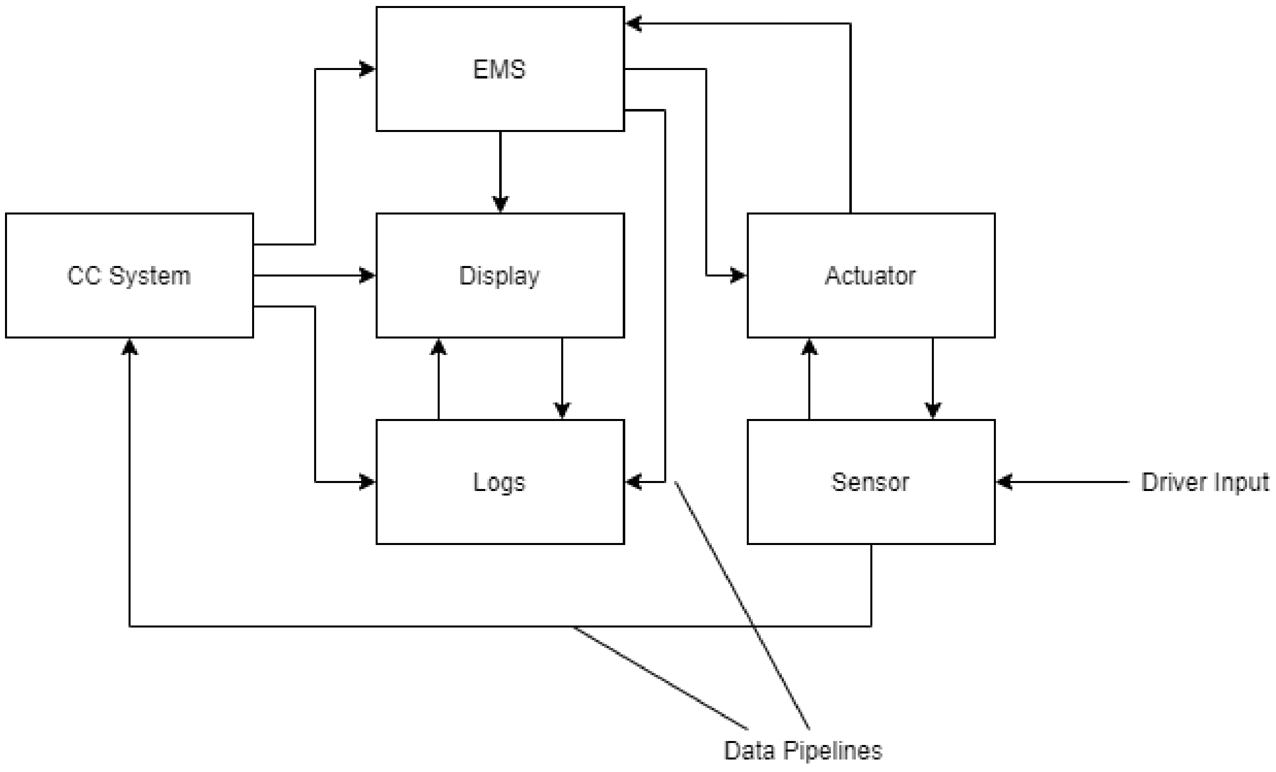
Control is managed within the architecture by having systematic unit tests throughout system hierocracy. Base cases established in prior sections of CC documentation will allow the CC system to see if its input and output are correct and realistic. Also, because the design process often leaves you with a number of architectural alternatives, it is important to establish a set of design criteria that can be used to assess an architectural design that is derived.



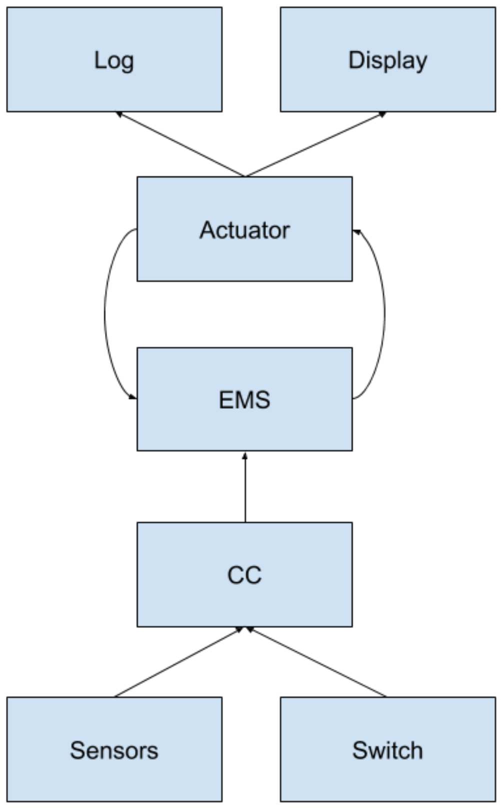
5.4 Data Architecture

Data architecture is the practice of having specifications to describe existing states, define data requirements, guide data integration, and control data assets. Data is communicated between the components through message passing. In order for the cruise control system to function as intended it is essential for the flow of data to be continuous between the components. This allows for our most important features, like disengaging the cruise control when the driver breaks, to take place immediately.

Examples of data architectures are Data-centered architectures and Data-flow architectures. Data-centered architectures: data store resides at the center of this architecture and is accessed frequently by other components that update, add, delete, or otherwise modify data within the store. Data-flow architectures: applied when input data are to be transformed through a series of computational manipulative components into output data.



5.5 Architectural Designs



5.6 Team Review

a) Identify and discuss the quality attributes by walking through the use cases.
   i) Our cruise control system is able to test all of the NFR we have declared in section 3.2. In addition, due to the nature of our CC architecture our quality attributes will be tested effectively

b) Discuss a diagram of the system's architecture in relation to its requirements.
   i) Object oriented architecture. Our CC system will systematically use unit tests as a buffer to reach the next sequence of operation.

c) Identify the architecture patterns used and match the system's structure to the pattern's structure.
   i) Architecture pattern is a linear hierarchy. Our CC system matches the system's structure to the pattern's structure as it also holds a pattern of linear hierarchy.

d) Use existing documentation and use cases to determine each pattern's effect on quality attributes.
   i) We would determine each pattern's effect on quality attributes using the buffer unit tests. An example is the cruise control system shall reach 99.999% reliability (NFR 1) will be examined by using log analytics to guarantee performance reaches the 99.999% reliability (NFR 1) as each failure is recorded. Furthermore, this acts as a baseline to calculate the performance of the CC system.

e) Identify all quality issues raised by architecture patterns used in the design.
   i) Quality issues raised by architecture can cripple the system architecture as the architecture pattern in object oriented. This pattern is dependent on the levels/nodes below it. When a failure occurs below its standing hierarchy, then the system slows down and fails. Furthermore, our CC system's use of unit tests as buffers amplifies this system architecture flaw.

f) Develop a short summary of issues uncovered during the meeting and make revisions to the walking skeleton.
   i) We discovered that our object oriented architecture design is dependent upon previous levels/nodes in the linear hierarchy. With this, we saw that the passing of incorrect output would result in system failure. To combat this, we decided to use systematic unit testing as a buffer to guarantee input to higher level nodes will be correct.

6. Code

```
import tkinter as tk
import os
import math
from datetime import datetime
from tkinter import scrolledtext
from tkinter import *

window = tk.Tk()
window.configure(bg="white")

MAX_SPEED = 120
MIN_SPEED = 15

if not os.path.isdir("./logs"):
    try:
        os.mkdir("./logs")
    except OSError:
        print("Creation of logs directory failed.")
    else:
        print("Succesfully created logs directory.")

logfile = open("./logs/oceancruise_" +
               datetime.now().strftime("%m-%d-%y") + ".log", "a")
if os.path.getsize("./logs/oceancruise_" +
datetime.now().strftime("%m-%d-%y") + ".log") > 0:
    logfile.write("\n")
logfile.write(
    "[" + datetime.now().strftime("%H:%M:%S") + "] Vehicle turned on.")
firstMsg = True
cruiseEnabled = False
currentSpeed = 0.0
setSpeed = 0
state = "Disabled"

actionLog = scrolledtext.ScrolledText(
    window, width=100, state=DISABLED, borderwidth=2, relief="groove")
actionLabel = Label(window, width=45, text="Actions Log",
                    font=("Arial", 15), pady=8, background="white")
```

```
stateLabel = Label(window, width=45, text="Current State: " +
                   state, font=("Arial", 15), background="white")
speedLabel = Label(window, width=45, text="Current Speed: " +
                   str(currentSpeed) + " MPH", font=("Arial", 15),
background="white")

actionLabel.grid(column=0, row=0, columnspan=5)
actionLog.grid(column=0, row=1, columnspan=5, sticky=NSEW)
stateLabel.grid(column=0, row=2, columnspan=5, sticky=NSEW, pady=(12, 0))
speedLabel.grid(column=0, row=3, columnspan=5, sticky=NSEW, pady=12)

brakeBtn = Button(window, width=15, text="Apply Brake",
                  font=("Arial", 12), background="white")
accelBtn = Button(window, width=15, text="Apply Accelerator",
                  font=("Arial", 12), background="white")
enableBtn = Button(window, width=20, text="Enable Cruise Control",
                   font=("Arial", 12), background="white")
upBtn = Button(window, width=5, text="+", font=("Arial", 12),
               background="pale green", activebackground="lime green")
dwnBtn = Button(window, width=5, text="-", font=("Arial", 12),
                background="tomato", activebackground="red")

cpyLabel = Label(window, width=45, text="OceanCruise Simulation Interface -
Developed by OceanByte, Spring 2020",
                 font=("Arial Bold", 10), background="Navy", fg="White")
cpyLabel.grid(column=0, row=5, columnspan=5, sticky=NSEW, pady=(32, 0))


def change_state(new_state):
    global state
    state = new_state
    stateLabel.configure(text="Current State: " + new_state)


def log_msg(msg):
    global firstMsg, logfile
    actionLog.configure(state='normal')
    if not firstMsg:
        actionLog.insert(tk.END, "\n")
    firstMsg = None
    timestamp = "[" + datetime.now().strftime("%H:%M:%S") + "] "
    actionLog.insert(tk.END, timestamp + msg)
    actionLog.configure(state='disabled')
    logfile.write("\n" + timestamp + msg)


def deaccelerate():
    if state == "Disabled":
        global currentSpeed
        if currentSpeed > 0:
            currentSpeed -= 0.05
        else:
            currentSpeed = 0
        speedLabel.configure(text="Current Speed: %.1f MPH" % currentSpeed)
        enableBtn.after(100, deaccelerate)


def update_speed(currentSpeed):
    global setSpeed, cruiseEnabled
    if cruiseEnabled:
        speedLabel.configure(text="Current Speed: %.1f MPH    Set Speed: 
%.1f MPH" % (
            currentSpeed, setSpeed))
    else:
        speedLabel.configure(text="Current Speed: %.1f MPH" % currentSpeed)


def cruise():
    global currentSpeed, cruiseEnabled
    if state == "Enabled":
        if currentSpeed < setSpeed:
            currentSpeed += 0.1
        elif currentSpeed > setSpeed:
            currentSpeed -= 0.05
        update_speed(currentSpeed)

    if cruiseEnabled:
        enableBtn.after(30, cruise)


def disable_cruise(msg):
    global cruiseEnabled, state
    cruiseEnabled = False
    change_state("Disabled")
    log_msg("Disabled cruise control for reason: " + msg)
    enableBtn.configure(text="Enable Cruise Control")
    deaccelerate()


def enable_cruise():
    global cruiseEnabled, setSpeed, currentSpeed, MIN_SPEED
    if state != "Braking":
        if currentSpeed < MIN_SPEED:
            log_msg("Could not enable cruise control, current speed (%.1f
MPH) is less than the minimum of %d MPH." % (
                currentSpeed, MIN_SPEED))
            return
        cruiseEnabled = True
        if state != "Accelerating":
            change_state("Enabled")
        log_msg("Enabled cruise control, maintaining a speed of %.1f MPH."
%
                currentSpeed)
        setSpeed = currentSpeed
        enableBtn.configure(text="Disable Cruise Control")
        cruise()
    else:
        log_msg("Could not enable cruise control, brake is applied.")


def toggle_cruise():
    global cruiseEnabled
    if cruiseEnabled:
        disable_cruise("disable button pressed")
    else:
        enable_cruise()


def accelerate():
    if state == "Accelerating":
        global currentSpeed, MAX_SPEED
        if currentSpeed < MAX_SPEED:
            currentSpeed += 0.1
        update_speed(currentSpeed)
        accelBtn.after(25, accelerate)
```

```python
def brake():
    if state == "Braking":
        global currentSpeed
        if currentSpeed > 0:
            currentSpeed -= 0.25
        else:
            currentSpeed = 0
        update_speed(currentSpeed)
        brakeBtn.after(25, brake)


def toggle_accel():
    global state
    if state == "Braking":
        log_msg("Couldn't apply accelerator, brake is applied.")
        return
    if state != "Accelerating":
        change_state("Accelerating")
        accelBtn.configure(text="Release Accelerator")
        log_msg("Accelerator was applied.")
        accelerate()
    else:
        if cruiseEnabled:
            change_state("Enabled")
        else:
            change_state("Disabled")
            deaccelerate()
        accelBtn.configure(text="Apply Accelerator")
        log_msg("Accelerator was released.")


def toggle_brake():
    global state, cruiseEnabled

    if cruiseEnabled:
        disable_cruise("application of brake")

    if state != "Braking":
        if state == "Accelerating":
            toggle_accel()
        change_state("Braking")
        brakeBtn.configure(text="Release Brake")
        log_msg("Brake was applied.")
        brake()
    else:
        change_state("Disabled")
        brakeBtn.configure(text="Apply Brake")
        log_msg("Brake was released.")
        deaccelerate()


def increase_cruise():
    global setSpeed, cruiseEnabled, MAX_SPEED
    if cruiseEnabled:
        if setSpeed + 1 > MAX_SPEED:
            log_msg(
                "Couldn't increase cruising speed, attempting to increase
speed above maximum speed of %d MPH." % MAX_SPEED)
        else:
            setSpeed = math.ceil(setSpeed + 1)
            log_msg("Cruising speed increased to %.1f MPH." % setSpeed)

    else:
        log_msg("Couldn't increase cruising speed, cruise control is not
enabled.")


def decrease_cruise():
    global setSpeed, cruiseEnabled, MIN_SPEED
    if cruiseEnabled:
        if setSpeed - 1 < MIN_SPEED:
            log_msg(
                "Couldn't decrease cruising speed, attempting to decrease
speed below minimum speed of %d MPH." % MIN_SPEED)
        else:
            setSpeed = math.floor(setSpeed - 1)
            log_msg("Cruising speed decreased to %.1f MPH." % setSpeed)
    else:
        log_msg("Couldn't decrease cruising speed, cruise control is not
enabled.")


brakeBtn.configure(command=toggle_brake)
```

```
accelBtn.configure(command=toggle_accel)
enableBtn.configure(command=toggle_cruise)
upBtn.configure(command=increase_cruise)
dwnBtn.configure(command=decrease_cruise)

brakeBtn.grid(column=0, row=4)
accelBtn.grid(column=1, row=4)
enableBtn.grid(column=2, row=4)
upBtn.grid(column=3, row=4)
dwnBtn.grid(column=4, row=4)

window.title("OceanCruise Interface")
window.mainloop()
if cruiseEnabled:
    logfile.write("\n[" + datetime.now().strftime("%H:%M:%S") + "]
EMERGENCY: Executed emergency shutoff procedure, power was unexpectedly
lost.")
logfile.write(
    "\n[" + datetime.now().strftime("%H:%M:%S") + "] Vehicle turned off.")
```

7. Tests

7.1 Functional Requirement Testing

FR1: CC system maintains vehicle speed

Input:

- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Release accelerator

Desired Result:

- “Current State: Enabled, Current Speed: 30 MPH, Set Speed: 30 MPH”

Actual Result:

- “Current State: Enabled, Current Speed: 30 MPH, Set Speed: 30 MPH”

[TEST CASE PASSED]

FR2: Driver can terminate CC system with break or off switch
Input:
- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Press Break or switch off

Desired Result:
- “Current State: Breaking, Current Speed: (decelerating)”

Actual Result:
- “Current State: Breaking, Current Speed: (decelerating)”

[TEST CASE PASSED]

FR3: CC will allow car to accelerate while active but reduce speed to match set speed when accelerator is released
Input:
- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Allow accelerator to continue
- Release accelerator at desired speed

Desired Result: (After allowing car to decelerate)
- “Current State: Enabled, Current Speed: 30MPH, Set Speed: 30MPH”

Actual Result:
- “Current State: Enabled, Current Speed: 30MPH, Set Speed 30MPH”

[TEST CASE PASSED]

FR4: CC will receive information from EMS system
Input:
- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Release accelerator

Desired Result:
- (The EMS is what actually controls the speed of the vehicle, so if the car is maintaining the speed we know it works)

Actual Result:
- “Current State: Enabled, Current Speed: 30MPH, Set Speed 30MPH”

[TEST CASE PASSED]

FR5: CC will receive information from Traction Control system

Input:

- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Release accelerator

Desired Result:

- (Traction Control is supplementary to the EMS so if the EMS accepts the CC input then we know it works)

Actual Result:

- "Current State: Enabled, Current Speed: 30MPH, Set Speed 30MPH"

[TEST CASE PASSED]

FR6: CC system will terminate in the event of an emergency

Input:

- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Close oceancruise.py program

Desired Log Result: (simulating an emergency)

- "[TIME] EMERGENCY: Executed emergency shutoff procedure, power was unexpectedly lost."

Actual Log Result:

- "[TIME] EMERGENCY: Executed emergency shutoff procedure, power was unexpectedly lost."

[TEST CASE PASSED]

FR7: CC system sends requests to EMS to track statistics

Input:

- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Release accelerator

Desired Result:

- (Similar to FR4, the EMS is what actually controls the speed of the vehicle, so if the car is maintaining the speed we know it works)

Actual Result:

- "Current State: Enabled, Current Speed: 30MPH, Set Speed 30MPH"

[TEST CASE PASSED]

FR8: CC shall display info to the driver
Input:
- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Follow any series of inputs

Desired Result:
- (The display will present the information to the driver following their inputs)

Actual Result:
- Results are displayed

[TEST CASE PASSED]

FR9: CC requires car to be moving 15MPH to be activated
Input:
- From rest, apply accelerator
- Let vehicle accelerate to any speed below 15MPH
- Activate cruise control

Desired Result:
- "[TIME] Could not enable cruise control, current speed (SPEED) is less than the minimum of 15MPH"

Actual Result:
- ""[TIME] Could not enable cruise control, current speed (SPEED) is less than the minimum of 15MPH"

[TEST CASE PASSED]

FR 10: The CC shall deactivate in the event of a collision. A collision is altered by the EMS, which monitors collision sensors around the vehicle for potential crashes. The CC shall subscribe to alters from the EMS.
Input:
- From rest, apply accelerator
- Activate cruise control at 40 MPH
- Close the GUI / program

Desired Results:
- (Subscription alerts are supplementary to the EMS so if the EMS accepts the CC input then we know it works)
- Input in the system log that the system had an emergency shut down

Actual Results:
- "EMERGENCY: Executed emergency shutoff procedure, power was unexpectedly lost."

[TEST CASE PASSED]

FR 11: The CC system shall allow the user to increment or decrement the current cruise speed in either one or five unit increments while cruising.
Input:
- From rest, apply accelerator
- Activate CC at 50 MPH
- Release accelerator
- Increment set CC speed 5 times

Desired Output:
- CC speed is set at 55 MPH
- “Current State: Enabled, Current Speed: 55 MPH, Set Speed 55 MPH”

Actual Output:
- “Current State: Enabled, Current Speed: 55 MPH, Set Speed 55 MPH”

[TEST CASE PASSED]

FR12: If CC is engaged and the car is turned off (see NFR #4), the system shall perform housekeeping operations such as flushing memory or updating logs before powering off.
Input:
- From rest, apply accelerator
- Activate cruise control at 40 MPH
- Close the GUI / program

Desired Results:
- (Subscription alerts are supplementary to the EMS so if the EMS accepts the CC input then we know it works)
- Input in the system log that the system had an emergency shut down

Actual Results:
- “EMERGENCY: Executed emergency shutoff procedure, power was unexpectedly lost.”

[TEST CASE PASSED]

FR13: CC shall periodically communicate with the vehicle’s sensors, fetching current measurements such as speed, tire pressure, etc.
Input:
- From rest, apply accelerator
- Let vehicle accelerate to 30MPH
- Activate cruise control
- Release accelerator

Desired Results

- (Subscription alerts are supplementary to the EMS so if the EMS accepts the CC input then we know it works)

Actual Results

- Log takes in every input data points and documents it.

[TEST CASE PASSED]

FR14: CC shall accept inputs from sensors after the engine is ignited.

Input:

- Press the break
- Release the break
- From rest, apply accelerator
- Let vehicle accelerate to 16 MPH
- Activate cruise control
- Release accelerator

Desired Result

- Reported in log that the break was used. Next the CC will reach 16MPH and stay there after the accelerator was released.

Actual Result

- “Current State: Enabled, Current Speed: 16 MPH, Set Speed 16 MPH”

[TEST CASE PASSED]

7.2 Nonfunctional Requirement Testing

NF 1: The cruise control system shall reach 99.999% reliability

Result: Passed

NF 2: The cruise control system shall implement an internal clock for diagnostic and logging purposes

Result: Passed. System log takes in user entry time and enters data.

NF 3: The cruise control system shall use the car battery as a source of power while the engine is on.

Result: Passed.

NF 4: The cruise control system shall exhibit hard responses times to user input below 20ms.

Result: Passed. System is programmed to output results in 2.5mms.

NF 5: The cruise control system shall be capable of being developed to any hardware.
Result: Passed.

NF 6: The cruise control system shall be supported by the vehicle's battery. This will allow for cleanup operations in event the vehicle turned off while the system is enabled (see FR 12).
Result: Passed. When this occurs, an emergency shut off is logged along with time.

NF 7: Environment sensors capable of supplying the vehicle speed and brake pedal state.
Result: Passed. Speedometer, brake and accelerator are integral parts of system architecture.

7.3 Security Requirements

SR1: The cruise control system shan't be accessible from any external network (WiFi, Bluetooth, etc.)
Result: Passed. The only external port in the system is connection to log when technicians extract its content during maintenance.

SR2: The cruise control system shall only be accessible by certified technicians or OceanByte developers.
Result: Passed.

SR3: All of the data generated or communicated by the cruise control system shall be encrypted.
Result: Passed.

SR4: All firmware shall be digitally signed to prevent forgery or malicious interference.
Result: Passed. GUI and system are watermarked with "OceanCruise Simulation Interface - Developed by OceanByte, Spring 2020"

SR5: All attempts to access or modify parts of the cruise control system shall be logged in a log file with read only permissions
Result: Passed. Log file is local to computer and solely has read permissions.

8. Issues

Error: No logging of errors in Architecture Style diagram
Solution: Whenever CC activated or deactivated, data of time, speed, and action will be logged.

Error: What data type will log hold?
Solution: The CC log will be a txt file of time, speed, and action used by drive.

Error: How will the time of use of CC be logged?
Solution: When the CC system is activated, the time will be zeroed out. This means that if an activity commences 20 minutes after the CC system is activated, that activity will be located with a time of 20.

Error: Max or Min in class or architecture
Solution: Create a min_speed variable and a max_speed variable to keep track of the min and max speeds

Error: Log does not get display data
Solution: log_msg() function will be declared to log the information that is displayed.

Error: How does EMS know where speed is temporarily increased/decreased or if it is permanently increased/decreased.
Solution: bool temporary_speed_adjustment will be declared. This book will tell the EMS if the speed increase/decrease should be added upon the CC set speed or if it should be ignored.
Example: An example is a drive with a CC speed of 60mph. If the driver accelerates to 65mph using the gas but does not change the set speed on the CC system, the car will eventually coast back down to 60 mphs after the gas if let go.

Error: Sensors do not dictate data types

Solution: Declare an integer variable for the Speedometer speed

- Speedometer → int

Error: Sensors do not dictate data types
Solution: Declare a boolean variable for the Switch to determine if active or not

- Switch → bool

Error: Sensors do not dictate data types
Solution: Declare integer variables for the Display to report

- Display → int

Error: Sensors do not dictate data types
Solution: Declare string variables to write them in the Log

- Log → txt?

Error: Sensors do not dictate data types
Solution: Declare an integer variable for the requested speed of the EMS

- EMS → int

Error: Sensors do not dictate data types
Solution: Declare an integer variable for the Actuator to take an actio

- Actuator → int

Error: Sensors do not dictate data types
Solution: Declare a boolean variable to determine if the Brake is being pressed

- Brake → bool

Error: Class diagram, cruise control needs speed function
Solution: Initialize a speed function for the cruise control class to keep track of current speed.

Error: Sensor is not working correctly and has no output
Solution: Display a warning message and prevent the cruise control system from turning on to prevent any further issues.

Error: Actuator is not working correctly and has no output
Solution: Display a warning message and prevent the cruise control system from continuing to prevent any further issues.

Error: EMS is not working correctly and has no output

Solution: Display a warning message and prevent the cruise control system from continuing to prevent any further issues.

Error: Cruise control system doesn’t keep track of maximum speed
Solution: Make a max_speed variable and compare it with the current cruise control set speed to see if the set speed is too high. If the set speed is too high, lower it to the value that max_speed is set to. Also, if the speed of the cruise control is above the max_speed on activation, prevent the activation of the cruise control system unless the speed increases.

Error: Cruise control system doesn’t keep track of minimum speed
Solution: Make a min_speed variable and compare it with the current cruise control set speed to see if the set speed is too low. If the set speed is too low, raise it to the value that max_speed is set to. Also, if the speed of the cruise control is below the min_speed on activation, prevent the activation of the cruise control system unless the speed increases.