**Software Architecture Document**

Tuffy Steering System

Team 4

Nathan Marcos,   
Dylan Nguyen,   
Lorena Macias,   
Phuc Le,   
Kenneth Truong,   
Omar Ramirez,   
Grant Marsh,   
Jacob Powell

02/14/2021

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 02/25/2021 | 1.0 | Added: Project name, Team Name, Team Members, | Dylan Nguyen |
| 03/11/2021 | 1.1 | Added: References, Architectural Goals+Constraints, Introduction, Logical Overview, | Omar Ramirez, Nathan Marcos, Lorena Macias, Dylan Nguyen, Kenneth Truong |
| 03/12/2021 | 1.2 | Edited: Logical Overview | Kenneth Truong |
| 03/16/2021 | 1.3 | Edited: Use-Cases, Goals and Constraints | Nathan Marcos |
| 03/18/2021 | 1.4 | Edited: Introduction, G&C, References, Component Diagram | Lorena Macias, Omar Ramirez, Nathan Marcos |
| 03/20/2021 | 1.5 | Added: Process View | Grant Marsh |
| 03/21/2021 | 1.6 | Added: Architecturally Significant Design Packages, Purpose, Scope, Definitions Acronyms, and Abbreviations | Kenneth Truong, Phuc Le, Dylan Nguyen |
| 3/22/2021 | 1.7 | Edited:Architecturally SIgnificant Design Packages Added: Size and Performance and Quality | Kenneth Truong,  Nathan Marcos |
| 03/23/2021 | 1.8 | Added: Deployment View and Appendix, 4+1 model for Architectural Representation, Class Diagram Edited: D.A.A, broke sections into different pages, formatted text | Dylan Nguyen,  Jacob Powell |
| 4/29/2021 | 1.9 | Edited: Figure 5 Class Diagram to be vertical (based on comments left by professor)  Edited: Figure 6 Implementation View and added specification to Description | Jacob Powell,  Lorena Macias |
| 5/2/2021 | 2.0 | Edited: Process View to include Traditional and functional Process view | Grant Marsh |
| 5/3/2021 | 2.1 | Editing: Figure 8: Deployment/Physical View. Added Physical Specification to Diagram and edited description (based on comments left by the professor)  Edited: Updated Tab | Dylan Nguyen |

Table of Contents

[**1. Introduction**](#_heading=h.qm5rbqx8cl38) **5**

[1.1 Purpose](#_heading=h.grqxsyqlh7g) 5

[1.2 Scope](#_heading=h.phm5r12uakms) 5

[1.3 Definitions, Acronyms, and Abbreviations](#_heading=h.ddapo2xcl679) 5

[1.4. References](#_heading=h.4olr78tg6drw) 6

[1.5. Overview](#_heading=h.mhc92yfw9oei) 6

[**2. Architectural Representation**](#_heading=h.u1n0rtvtsdsc) **7**

[**3. Architectural Goals and Constraints**](#_heading=h.jcbi328lhkdc) **8**

[4. Use-Case View](#_heading=h.nsnukblrlynm) **9**

[4.1. Use-Case Diagrams](#_heading=h.vq7a115vyonw) 15

[**5. Logical View**](#_heading=h.fom0wjxcxibs) **16**

[5.1. Overview](#_heading=h.pwv1dxjhtlz1) 16

[5.2. Architecturally Significant Design Packages](#_heading=h.cimqcdtk0i2x) 17

[5.3. Architecturally Significant Design Classes](#_heading=h.uubmjkbeyjoe) 17

[**6. Implementation/Development View**](#_heading=h.59n0d16b1rjr) **20**

[6.1 Development Diagram](#_heading=h.hbrv8hbzfvd4) 20

[6.1.1. HMI](#_heading=h.nqmx2l6daqf2) 20

[6.1.2. Application](#_heading=h.40whrmwz3mld) 20

[6.1.3. Infrastructure](#_heading=h.u9aeumsgovnf) 20

[6.1.4 Framework](#_heading=h.tnznt85kf57n) 20

[6.1.5 COTS/OS](#_heading=h.ogd4x0t6so03) 21

[**7. Process View**](#_heading=h.h2irs76jkxmm) **22**

[7.1 Functional Process View](#_heading=h.mkae2plwtb6b) 22

[7.2 Traditional Process View](#_heading=h.16rivymmum8q) 23

[**8. Deployment/Physical View**](#_heading=h.1btj7flh7701) **25**

[8.1.1 System Interface](#_heading=h.atyjt81vsune) 25

[8.1.2 Control Module](#_heading=h.89mmp0on12st) 25

[8.1.3 On-Board Computer](#_heading=h.thavzgwn2mxf) 25

[8.1.4 OBD-II](#_heading=h.ekls008etf19) 25

[8.1.5 Obstacle Avoidance System](#_heading=h.bkx6v6my1kc6) 26

[8.1.6 Speed Module](#_heading=h.e8k23iwigif) 26

[8.1.7 Steering Actuators](#_heading=h.4rqduarssnjm) 26

[**9. Size and Performance**](#_heading=h.7dykejr84mc5) **27**

[**10. Quality**](#_heading=h.kty44n452r5b) **28**

[**Appendix A: Architectural Design Principles**](#_heading=h.m2o3zn8f8zun) **29**

Table of Figures

[Figure 1: 4+1 View Model](#_heading=h.xokd4s9ab7et) 8

[Figure 2: Use case diagram](#_heading=h.ih0vdsk1e70j) 16

[Figure 3: Logical View Overview](#_heading=h.wruzi7fy1pn8) 17

[Figure 4: Package Diagram](#_heading=h.jb8hr8obdr1z) 18

[Figure 5: Class diagram](#_heading=h.2z4cl981rq4k) 19

[Figure 7.1: Functional Process diagram](#_heading=h.7t9z956org0t) 23

[Figure 8: Deployment Diagram](#_heading=h.2j8farsu1s77) 26

# 1. Introduction

This document gives a high-level overview of the Tuffy Steering System software architecture and the goals of the architecture. The software architecture of the Tuffy Steering System is based on the system requirements and use cases from “Navigation and Steering Requirements Version 12.”

Tuffy Steering System is designed to guide automobiles on the roads from their current location to a user supplied destination.

## 1.1 Purpose

This document provides a comprehensive architectural overview of the Tuffy Steering System software, by using the use-case and logical views to depict how the system helps with steering.

## 1.2 Scope

The Tuffy Steering System will guide Tuffy mobiles on the street from their current location to a provided destination. The system receives the users’ input for a desired destination and the system will lead them there. Additionally, the system can steer the wheel automatically and avoid obstacles on the way.

## 1.3 Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| Terms | Definition |
| 4+1 Model | Use-Case View, Logical View, Implementation/ Development view, Process view, Deployment/Physical View |
| COTS | Commercial- off- the- shelf |
| GPS | Global Positioning System |
| NSS | Navigation and steering system |
| SN | Function Requirements |
| SRS | System Requirement Specification |
| UC | Use Case |

## 

## 

## 1.4. References

1. We referenced “Navigation and Steering System Requirements Version 12” in order to get a brief overview of how our Steering System would function.
2. Example: Software Architecture Document, Rational Software Corporation , www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm.

## 1.5. Overview

This Software Architecture Document provides an architectural overview of the Tuffy Steering System. This document is organized using a “4 + 1” view model to present different features and characteristics of the Tuffy Steering System. The use of different architectural views helps provide a better idea of the products overall goals and functionalities.

# 2. Architectural Representation

Figure 1 contains a brief overview of the different views we will be using to convey the architecture of the Tuffy Steering System. In it, we see the Logical View, the Process View, the Development View, and the Physical View. The Logical View contains all of the main functionalities of the Tuffy Steering System, the Process View shows how all of the separate processes communicate with each other, the Development View shows how the system will be constructed from the perspective of a programmer, and the Physical View lays out all of the physical aspects of the system, such as the software components topology, as well as physical hardware and connections.

**

#### Figure 1: 4+1 View Model

# 

# 3. Architectural Goals and Constraints

The following sections outline the goals and constraints that have major architectural impact on the system.

Hardware:

* The Control Module must be able to record positioning and velocity constantly via off-the-shelf high precision sensors placed around the vehicle and communicate this data with the Steering Actuators in real time to perform specified use-case specifications.
* At least 16GB of RAM and 64GB of disk space will be needed for the system to function.
* The system will run on purchased ARM-based microboards with cellular internet capability and an ARM64 CPU.
* The Driver’s Interface will be displayed on a touchscreen with accurately labeled buttons.
* Purchased GPS sensors will be used to gain GPS data from satellites.

Software:

* The system must be able to integrate 3rd party GPS software systems to allow route selection.
* Legacy navigation and steering systems are to be reused and improved to integrate into current and new vehicle models by using the development environment and language from those systems if possible.
* QNX Neutrino will be the operating system used and can be updated via the Driver’s Interface when updates are available.
* Software will work on ARM64 architecture natively
* The system must support optional encryption of saved route selections, activity, and user information.
* No virtualization will be required
* Will require internet for updates
* Error reports will can be generated but not sent to any third parties
* Configurable logging of activities

# 

# 4. Use-Case View

**4.0.1 Position and Velocity Sensing**

***[UC-001]: Update current position and velocity (SN-001)***

**Actors**: Driver, Control Module, Driver’s Interface

**Brief Description**: The driver initiates automatic steering. While automatic steering is active, the system will update its current position and velocity upon receipt of valid Position Velocity Messages.

***[UC-001]: Suspend steering upon invalid Position Velocity messages (SN-002)***

**Actors**: Driver, Control Module, Driver’s Interface

**Brief Description:** The driver initiates automatic steering. While automatic steering is active, if 3 (adaptable) consecutive invalid Position Velocity messages are received within 2 seconds (adaptable), the system shall cease automatic steering.

***[UC-001]: Suspend steering upon GPS signal integrity errors (SN-003)***

**Actors**: Driver, Control Module, Driver’s Interface

**Brief Description:** The driver initiates automatic steering. While automatic steering is active, if 3 (adaptable) consecutive Position Velocity messages indicating GPS signal integrity errors are received within 2 seconds (adaptable), the system shall cease automatic steering.

***[UC-001]: Generate and display Navigation Display Update (SN-004)***

**Actors:** Driver, Control Module, Driver’s Interface

**Brief Description:** The driver initiates automatic steering. While automatic steering is active, the system shall generate a Navigation Display Update message indicating the vehicles current position after every 4th (adaptable) valid Position Velocity Message

**4.0.2 Route Selection**

***[UC-002]: Generate up to 3 valid Routing Request Message upon receiving a destination (SN-005)***

**Actors:** Driver, Control Module, Driver’s Interface

**Brief Description:** A Driver inputs a destination via the Driver’s Interface, which is then translated into a Routing Request Message and sent to the Control Module. Upon receiving the Routing Request Message, the Control Module will then determine up to 3 possible optimal paths from the Driver's current location to the specified destination.

***[UC-002]: The Control Module has found up to 3 optimal path for a destination (SN-006)***

**Actors:** Driver, Control Module, Driver’s Interface

**Brief Description:** The Control Module has finished calculating up to 3 optimal paths for a Driver-given destination. The Control Module will then pack the path information into up to 3 Available Routes Message depending on how many routes were found.

***[UC-002]: The Driver inputs an invalid or unavailable destination (SN-007)***

**Actors:** Driver, Control Module, Driver’s Interface

**Brief Description:** The Driver inputs a destination in the Driver’s Interface which is then sent to the Control Module. The Control Module determines that the destination is unreachable at the current location. As a result, an Available Routes Message is generated as normal, however this message will contain a message notifying the Driver that the specified route is unavailable.

**4.0.3 Steering**

***[UC-003]: The Control Module receives a Selected Route Message (SN-008)***

**Actors:** Driver, Control Module, Driver’s Interface, Steering Actuators

**Brief Description:** The Driver has selected a route to their destination and as such has sent a Selected Route Message via the Driver’s Interface to the Control Module. Upon receiving this message, the Control Module will be in constant back and forth communication with the Steering Actuators as the vehicle drives along the chosen route towards the destination.

***[UC-003] The system shall take into account the vehicle’s current position and velocity, and the selected route information when determining the timing and magnitude of steering commands. (SN-009)***

**Actors:** Driver, Control Module, Driver’s Interface

**Brief description:** The vehicle shall adjust the timing and the magnitude of steering wheels based on the vehicle’s current position and velocity and route information.

**[*UC-003] If a required turn cannot be made safely at the vehicle’s current speed, the system shall generate a Speed Change Request indicating the necessary speed reduction. (SN-0010)***

Note: The safe speed for a 180° turn is 5 mph. The safe speed for a 90° turn is 15 mph. The safe speed for a 45° turn is 35 mph.

**Actors:** Driver, Control Module, Driver’s Interface

**Brief description:** Vehicle determines it needs to turn base on the driver destination. system will generate speed change requests to the control module. From there the control module shall reduce the speed to appropriate mph. Speed shall reduce to 5 mph when doing 180° turn. 15 mph for 90° and 35 mph for 45°.

***[UC-003] The system shall generate one or more Speed Change Requests to return the vehicle to its appropriate speed after a turn has been completed.(SN-0011)***

**Actors:** Driver, Control Module, Driver’s Interface

**Brief description:** System will be able generate one or more speed change requests to the control module after the vehicle has finished a turn. Control module will receive the message and shall initiate the appropriate speed change for the path.

**4.0.4 Obstacle Avoidance**

***[UC-004] Upon receipt of an Obstacle Position Message indicating a moving obstacle in the vehicle’s path (e.g., another vehicle that is moving slower than the host vehicle), the system shall generate one or more Speed Change Requests to reduce the vehicle’s speed to keep the obstacle at a safe distance. (SN-0012)***

**Actors:** Driver, Control Module, Driver’s Interface

**Brief description:** When the system generates and receives an obstacle position message which notify that there is a moving obstacle that is going to block its driving path, the system will generate a speed change request to the control module. The control module then changes the vehicle speed appropriate amount to keep the moving obstacle at a safe distance. The process shall repeat if the moving obstacle distance to the vehicle is in unsafe distance.

***[UC-004] Detect when to return to a safe speed (SN-0013)***

**Actors:** Control Module, Obstacle Detection System, Speed Control System

**Brief Description:** The Obstacle Detection System will send an Obstacle Position Message to the Control Module when it determines there is no longer an obstacle in factor. The Control Module will send multiple possible Speed Change Requests to the Speed Control System. The Speed Control System will accept the most recent Speed Change Request and the vehicle will increase in speed.

***[UC-004]******Drive around stationary objects (SN-0014)***

**Actors**: Obstacle Detection System, Control Module, Steering Actuators

**Brief Description:** The Control Module is given an Obstacle Position Message indicating a stationary obstacle is near the vehicle. The Control Module communicates with the Steering Actuators to drive around the stationary obstacle.

**[UC-004]** **Decrease the speed (SN-0015)**

**Actors:** Control System, Speed Control System

**Brief Description:** The Control System will send a Speed Request Change to the Speed Control System to lower the speed of the vehicle if the maneuvers require it.

**[UC-004] Change speed and direction [SN-0016]**

**Actors:** Control System, Speed Control System, Steering Actuators, Obstacle Detection System

**Brief Description:** The Obstacle Detection Systems detects that an obstacle is no longer a factor and sends an Obstacle Position Message to the Control System. The Control System will send at least one Speed Change Request to the Speed Control System depending on real time calculations. The Control System will send steering commands to the Steering Actuators.

**[UC-004]: Detect when to cease automatic steering (SN-0017)**

**Actors:** Driver, Control Module, Driver’s Interface

**Brief Description:** The driver requests to enable automatic steering. The system activates auto steering mode. Upon driving into an unavoidable object, the system will detect the object and cease automatic steering controls within 100 ms of detection. Once the driver breaks or drives around the object, the system will re-enable automatic steering within 100 ms of detecting the object is no longer there.

**[UC-005]: Suspend and resume automatic steering (SN-0018)**

**Actors:** Driver, Control Module, Driver’s Interface

**Brief Description:** The driver requests to enable automatic steering. The system activates auto steering mode. The driver requests to Suspend automatic steering. The system suspends auto steering mode and ends Speed Change requests. The driver requests to Resume automatic steering. The system activates auto steering mode and accepts Speed Change requests.

1. If the vehicle is still on the selected route, the system **shall [SN-0019]** resume steering and the generation of Speed Change Requests.

***[UC-005]: The system shall generate a Speed Change Request of “0” to apply a brake. (SN-0036)***

**Actors**: Driver, Control Module, Driver’s Interface

**Brief Description**: The system will first detect an object (unavoidable) through its Obstacle Detection System. Then, the system will cease automatic steering within 100ms. The system shall initiate a Speed Change Request of “0” to apply a brake. After braking, the system will stop and notify the driver to take manual control.

***[UC-005]: The system shall notify the driver to take manual control steering the vehicle. (SN-0037)***

**Actors**: Driver, Control Module, Driver’s Interface

**Brief Description**: After applying a brake, the system will ask the driver to take manual control steering the vehicle before stopping. Driver can initiate the system again if no object is detected within 100ms of detection.

***[UC-005]: Suspend and resume automatic steering when on a route (SN-0019)***

**Actors**: Driver, Control Module, Driver’s Interface

**Brief Description**: The drive requests to follow a route. The system begins following the selected route. The driver requests to enable automatic steering. The system activates auto steering mode. The driver requests to Suspend automatic steering. The system suspends auto steering mode and ends Speed Change requests. The driver requests to Resume automatic steering. The system activates auto steering mode, accepts Speed Change requests, and follows the currently selected route.

***[UC-005]: Turn off automatic steering (SN-0020)***

**Actors:** Driver, Control Module, Driver’s Interface

**Brief Description:** The driver requests to enable automatic steering. The system activates auto steering mode. The driver requests to turn Off automatic steering. The system turns off auto steering mode.

**4.0.5 Driver Controls Use Cases**

***[UC-007]: Switch from automatic driving to Manual Mode automatically (SN-0030)***

**Actors**: Driver, Control Module, Steering Actuators

**Brief Description:** When the Driver takes manual control over the vehicle during NASS, a message is sent to the Control Module. Upon receiving the message, the Control Module will disengage cruise control and the driver will be able to resume manual driving.

***[UC-007] Switch from Manual Mode driving to automatic steering (SN-0031)***

**Actors:** Driver, Control Module, Steering Actuators

**Brief Description:** When the Driver indicates they are no longer steering the vehicle, a Manual Mode message will be sent to the Control Module. Upon receiving the message, The Control Module will re engage automatic steering.

**4.0.6 Emergency Actions Use Cases**

***[UC-007] The System will stop automatic steering if an error is detected in the systems (SN-0022)***

**Actors:** Control Module, Steering Actuators

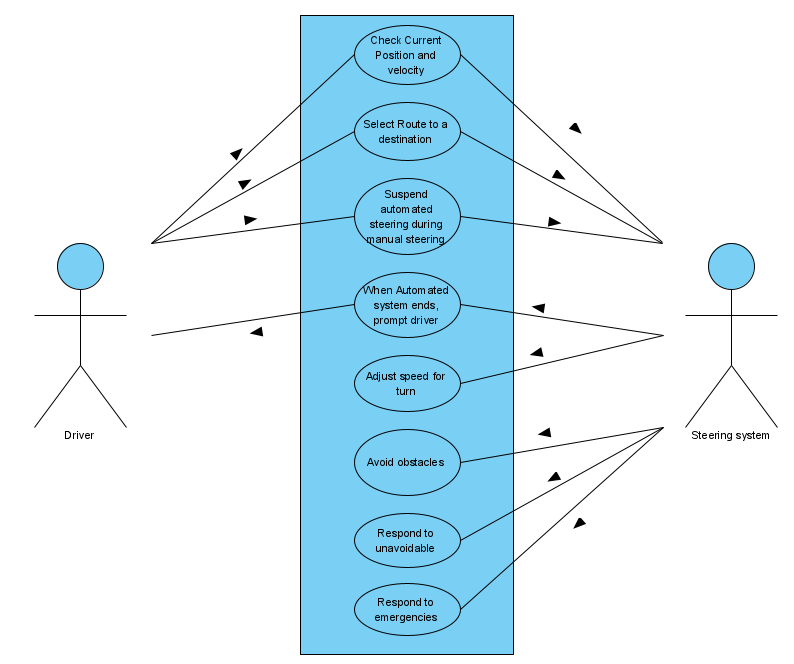
**Brief Description:** When the Control Module receives a message from the Navigation and Steering system that indicates the system cannot perform safely, the Control Module will disengage automatic steering.

***[UC-007] When automatic steering is stopped due to safety reasons, notify the driver (SN-0035)***

**Actors:** Control Module, Driver Interface

**Brief Description:** When the Control Module disengages automatic steering, send a warning message to the Driver Interface. Upon receiving the warning message, the driver interface will notify the driver to manually steer the vehicle until automatic steering is reengaged.

## 4.1. Use-Case Diagrams

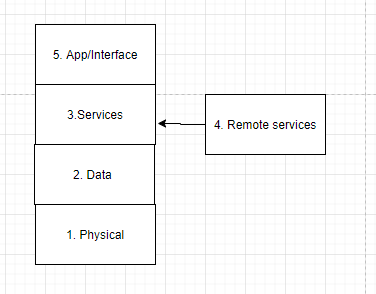


### Figure 2: Use case diagram

# 

# 5. Logical View

## 5.1. Overview

**

#### Figure 3: Logical View Overview

1. **Physical layer/hardware**

As shown in figure 3, The physical layer is below the Data layer. This layer represents the physical part of the system which includes electrical, cable, voltage, sensor, connectors, hub, control module. When there network problem or error, usually people check the physical layer if all cable and other components are properly connected and is power,

1. **Data**

From figure 3, Data layer is below Service layer. This layer handles error correction from the physical layer and data transfer between nodes. This layer contains Media Access Control and Logical Link Control. Where the MAC helps make transfer packets easier with no collision. LLC provides error checking and flow control. This layer is where most switches operate. Handles error and system activities.

1. **Services**

From figure 3, this layer handles the service for the automotive car such as automatic steering, and GPS routing. Notify the user if an error has occurred such as automatic steering has stopped. This layer relies on a data layer to operate services.

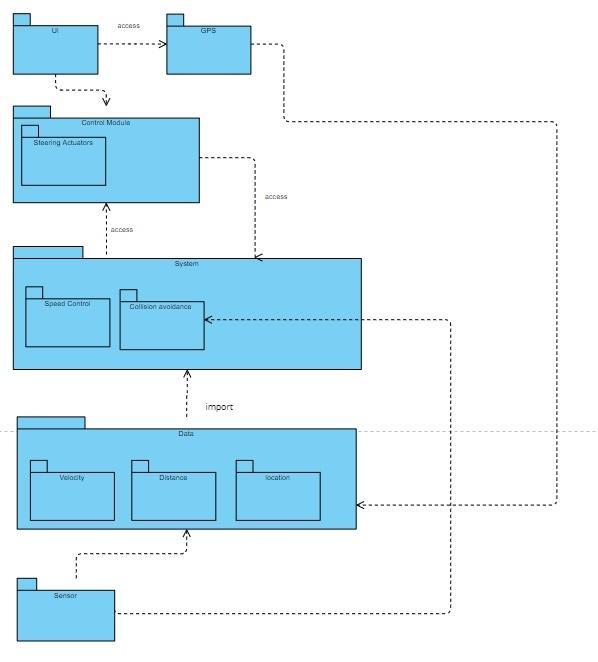
1. **Remote services**

From figure 3, this layer allows feature updates for the system for better performance.

1. **App/ interface**

From figure 3, This layer allows the user to interact with the control module allowing them to input route and automatic steering control on or off.

## 5.2. Architecturally Significant Design Packages

**

#### Figure 4: Package Diagram

## 5.3. Architecturally Significant Design Classes

## 

#### Figure 5: Class diagram

**5.1 GPS Class**

The GPS Class aggregates methods from three sub-classes, which themselves read directly from the GPS Sensor. These three sub-classes then provide the respective information to the GPS class, which can then provide the GPS information to the Driver Interface Class. The Control Module Class never directly speaks with the GPS class, it only receives information through the Driver Interface.

**5.1.2 Speed Module Class**

The Speed Module Class aggregates methods from two sub-classes, which themselves read directly from the Speed Sensor and the Engine. These two sub-classes then provide the respective information to the Speed Module Class, which can then expose a few methods pertaining to the car’s speed to the Control Module Class.

**5.1.3 Obstacle Detection System Class**

The Obstacle Detection System Class aggregates methods from two sub-classes, which themselves read directly from the LIDAR system and the built in Camera. These two sub-classes then provide the respective information to the Obstacle detection class, which can then provide the ability for the car to detect obstacles and prevent collisions to the Control Module Class.

**5.1.4 Steering Actuators Class**

The Steering Actuators Class aggregates methods from one sub-classes, which itself reads directly from the Steering Wheel. This sub-class then provides the respective steering wheel interfacing functionality to the Steering Actuators class, which can then provide steering wheel control to the Control Module Class.

**5.1.5 Driver Interface Class**

The Driver Interface Class accesses GPS information via the GPS Class and then displays this information among other information, as well as provides a space for users to input a destination, send it to the GPS Class, and then receive a route. All of this functionality is packaged up and then accessed by the Control Module Class.

**5.1.6 Control Module Class**

The Control Module Class is the brains of the Tuffy Steering System. All of the classes provided communicate their respective information and provide their functionalities to this class.

# 

# 6. Implementation/Development View

## 6.1 Development Diagram

|  |  |
| --- | --- |
| HMI | Touch Screen, Emergency Actions, Steering Wheel, Route Selection, Speaker |
| Application | Notifications, Diagnostics, Control Module |
| Infrastructure | Position Velocity Module, Speed Module, Obstacle Detection Module, Steering Module, GPS Module |
| Framework | Position and Velocity calculations, Timing and Magnitude of Steering calculations, Speed calculations |
| COTS/OS | 3rd Party GPS System, Sensors, QNX Neutrino, ARM64 CPU, Steering Actuators |

**Figure 6: Implementation View**

## **6.1.1. HMI**

HMI is where the user can interact with the Tuffy Steering System. A touch screen is provided for simple interaction and steering wheel for manual control. The speaker will announce to the driver of possible routes and actions being taken during self driving. Emergency buttons will also be available for the driver.

## 6.1.2. Application

Application layer makes use of infrastructure and frameworks.

## 6.1.3. Infrastructure

Infrastructurecontains all of the components of the Tuffy Steering System. Each component is separated to easily be replaced or updated when necessary.

## **6.1.4 Framework**

Frameworks include all computational algorithms including position and velocity. Frameworks are separated from the infrastructure to specialize in one area resulting in more accurate calculations.

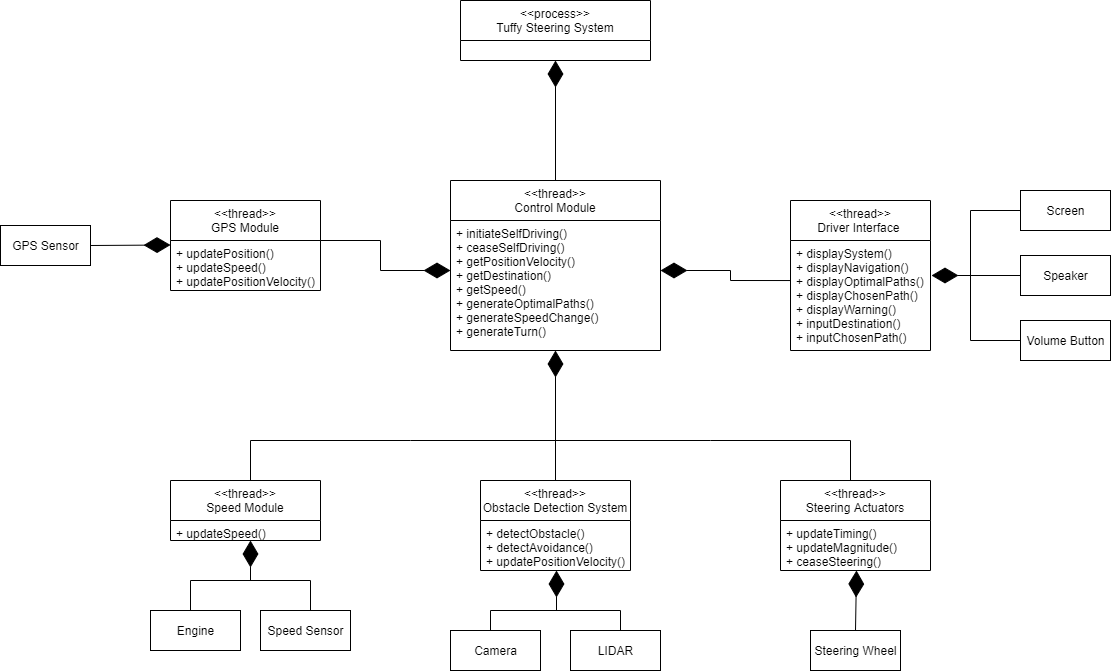
## 6.1.5 COTS/OS

The Commercial-Off-the-Shelf Software incorporated into the Tuffy Steering System is a third party GPS System. All route calculations come from this third party system. In addition, Tuffy Steering System makes use of sensors and steering actuators to detect obstacles and navigate the vehicle safely around them. QNX Neutrino is the operating system being used.

# 

# 7. Process View

## 7.1 Functional Process View

**

#### Figure 7.1: Functional Process diagram

1. **Control Module**

Is responsible for the overall decision making of the process. Communicates messages to and from threads, altering their behaviors by making decisions based on collected data. These decisions include: Initiating self driving protocol based on user input and non-erroneous conditions, disabling self driving protocol based on position velocity or obstacles, generating optimal routes based on inputted destination, changing direction through turning, and changing vehicle speed.

1. **Speed Module**

Constantly collects vehicle speed data from the engine and speed sensor and sends it to the Control Module to update vehicles speed to appropriate level.

1. **Obstacle Detection Module**

Constantly collects visual data from cameras and LIDAR in order to produce obstacle detection messages to be sent to the Control Module to safely prevent obstacle collisions.

1. **Steering Actuators**

Updates steering angle, timing, and operation based on decisions from the Control Module.

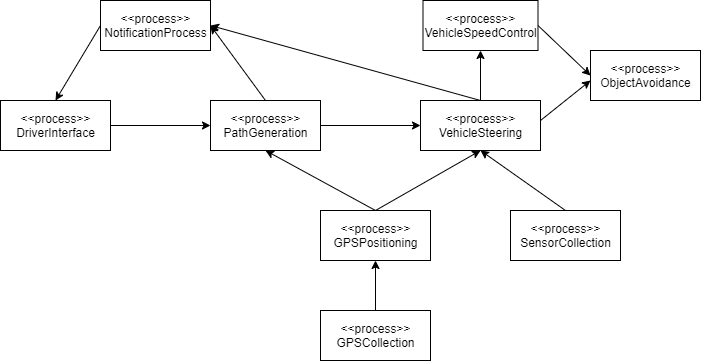
1. **GPS Module**

Constantly collects positional data alongside speed data to accurately calculate positional velocity which is analyzed by the Control Module.

1. **Driver Interface**

Allows the driver to set vehicle destination by interaction with interactive screen display. Control module creates three optimal routes which are then displayed by the driver interface. The driver is then able to choose one out of these three optimal routes for the vehicle to follow. In case of error or cease of self driving protocol, the driver interface will display a visual/audio warning message.

## 7.2 Traditional Process View

****

**Figure 7.2: Process View Diagram**

1. **Driver Interface Process**

This process allows the user to initiate self-driving functionality by inputting destination addresses as well as display important notifications involving path and steering information.

1. **Path Generation Process**

This process creates multiple optimal paths to the user's inputted destination address by using the vehicle's GPS position in relation to the available paths.

1. **Notification Process**

This process sends important notification information to be displayed on the drivers interface. The two times this occurs are first when multiple optimal paths are generated by the Path Generation Process, a notification of the paths are sent to the drivers interface for user selection, and second when autonomous steering is active, vehicle steering information is constantly sent to the drivers interface to be displayed.

1. **GPS Collection Process**

This process uses the vehicle's GPS sensor to constantly receive vehicle and destination GPS data.

1. **GPS Positioning Process**

This process updates the vehicle's internal GPS position based on collected GPS data from the GPS Collection Process.

1. **Sensor Collection Process**

This process uses the vehicle’s camera and LIDAR sensors to collect surrounding environmental visual data to be utilized using sensor fusion.

1. **Vehicle Steering Process**

This process utilizes the vehicles collected camera/LIDAR data from the Sensor Collection Process in order to perform sensor fusion analysis alongside the vehicles internal GPS position from the GPS Positioning Process to steer the vehicle in the correct direction, steer the vehicle at the correct angle, and disengage automatic steering if necessary.

1. **Vehicle Speed Process**

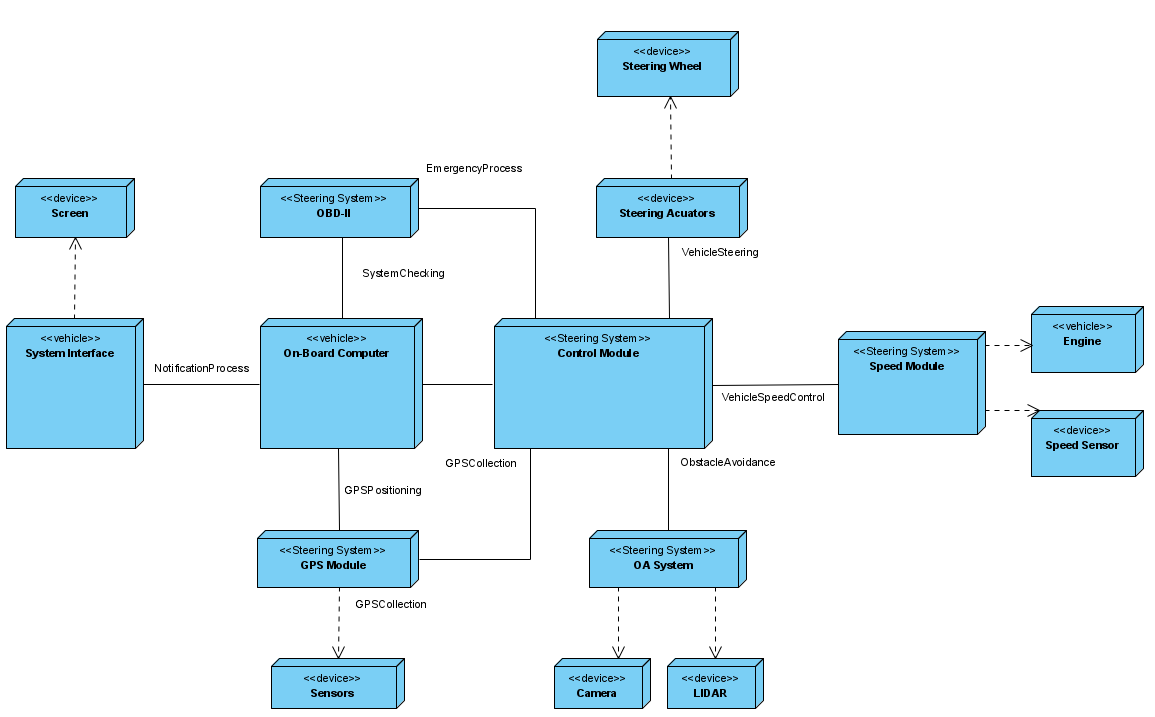
This process utilizes the vehicle's direction and angle from the Vehicle Steering Process as well as internal GPS position to constantly update the vehicle's speed during automatic steering.

1. **Object Avoidance Process**

This process utilizes the vehicle’s direction and angle from the Vehicle Steering Process and the vehicle’s speed from the Vehicle Speed Process in order to make instantaneous speed and direction change decisions in order to avoid potential collisions and hazards.

# 

# 8. Deployment/Physical View

**

#### Figure 8: Deployment Diagram

## 8.1.1 System Interface

System interface serves as host for information sent from the system to the user. All notifications, updates, etc. created from other processes will be displayed here. The interface also controls the Screen which will display all information for the User.(Fig.8)

## 8.1.2 Control Module

Tuffy System is also known as the steering system. This represents the connector between all other component systems. This System also encapsulates the speed module and steering actuators.

## 8.1.3 On-Board Computer

The on-board computer connects information from the steering system to the system interface such that all new information can be displayed quickly. All calculations are calculated here.

## 8.1.4 OBD-II

The OBD-II also known as, the On-Board Detection system regularly checks system functionality and notifies the User immediately if a process or module is not working properly.

## 8.1.5 Obstacle Avoidance System

The Obstacle Avoidance System (OA System) uses the vehicle's LIDAR sensors and cameras to detect all potential obstacles. All of this information is relayed to the Control Module.

## 8.1.6 Speed Module

The speed module uses the vehicle’s sensor to adjust the speed according to safety standards. This module also monitors the engine speed for proper acceleration. This module connects to the Control module and takes data from the GPS module for accurate speed control.

## 8.1.7 Steering Actuators

The Control module uses this device in conjunction with the speed control module to automatically steer the vehicle. This system directly influences the steering wheel.

# 

# 9. Size and Performance

The software architecture will support important sizing and time requirements:

1. The targeted number of users will be around 100 million globally.
2. The number of daily users is estimated to be around 10% of all users (10 million).
3. The system shall provide access to route selection cloud or local databases within 30 seconds of request to all users.
4. The client portion will only use 8GB of RAM and 16GB of disk space.

# 10. Quality

The software architecture will support quality requirements determined throughout this document:

Reliability:

* The meantime between failures for the system must not fail to cease automatic steering or cause any errors for the system to behave in an unsafe manner.
* The meantime between failures must not exceed more than twelve (12) months.

Availability:

* The system must be able to generate Navigation Display Update messages indicating the vehicle’s current position 99.9999% of the time that the vehicle is in operation.
* The system must be able to generate up to 3 optimal routes for 95% of the destinations entered by the user.
* The system will have saved offline maps to navigate when GPS is not available.

Portability:

* The system must be portable so that it may be hosted on vehicles in later models.
  + The system must be able to use any version of the QNX Neutrino operating system starting with the version in which it was first created.

Security:

* The system must prevent any form of hacking or other unauthorized activity.
  + Encryption and security protocols must be integrated throughout the system.

# 

# Appendix A: Architectural Design Principles

* **Built for flexibility:**
  + This system is designed to have maximum portability. This allows modular changes to be made and future modifications will have easier implementation. Also allows future models to host this system without major architectural overhaul.
* **Reuse Assets when possible:** 
  + We make use of our architectural design pattern whenever possible in our design.
* **Components can be changed based on requirements:**
  + To meet customer and business needs, components and services can be completely removed or rewritten as needed.
* **Design Models to be independent to isolate risk**
  + Design architecture with low coupling to make changes easier.