Navigation and Steering System

Software Architecture Document

Version <1.2>

**Team Hot Wheelz**

**Authors:**

**Andy Nguyen  
Miles McCloskey**

**Jonathan Boe**

**Arshi Singh**

**Chase Moynihan**

**Tan Tran**

**Austin Draper**

Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| <03/03/2018> | <1.0> | First Draft of the Software Architecture Document |
| <03/18/2018> | <1.1> | Second Draft of the Software Architecture Document |
| <04/26/2018 | <1.2> | Third Draft of the Software Architecture Document |
|  |  |  |

Table of Contents

[**Introduction**](#_wcpypzgk8mqu) **4**

[Purpose](#_l69fr93eibyw) 4

[Scope](#_jmkywtewnp5d) 4

[**References**](#_vxwno7yj56gc) **4**

[2.1 Customer Supplied Documents](#_w9s3o0z2rcfo) 5

[2.2 Government Documents](#_jwlkajx7bxvp) 5

[2.3 Other Documents](#_euy80jka8odq) 5

[**Software Architecture**](#_2s8eyo1) **5**

[**Architectural Goals and Constraints**](#_w2jn5il47cev) **7**

[**Logical Architecture**](#_1d2oxi2s1bb) **8**

[**Process Architecture**](#_pd2nkfrsph8q) **9**

[**Physical Architecture**](#_6xc9ygueq3sf) **11**

[**Development Architecture**](#_lclxjjyyhrwh) **11**

[**Scenarios**](#_a5g3001zo4x0) **12**

[**Non-Behavioral Requirements**](#_461arm3vddsi) **13**

[**Appendix**](#_e0v4s3ehorl2) **13**

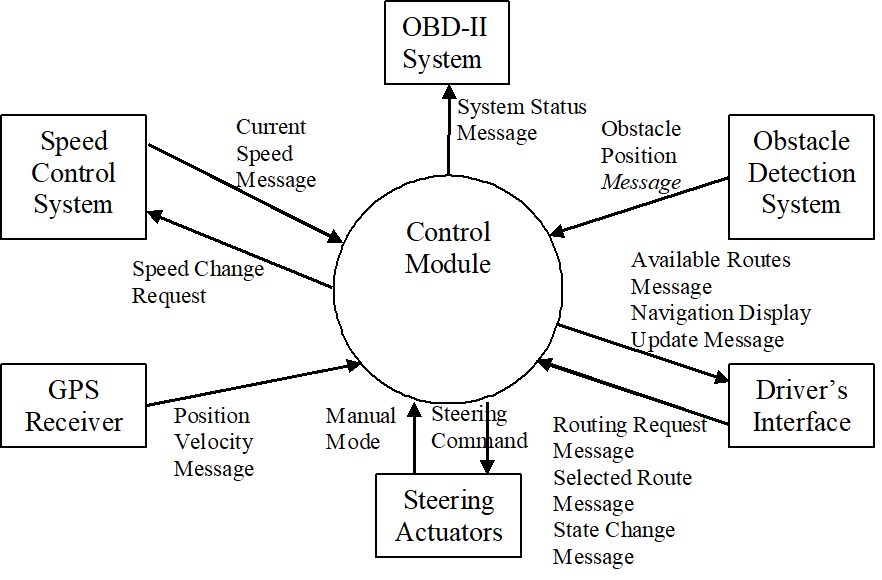
NSS Software Architecture Document

# Introduction

This report explains the requirements for the Navigation and Steering System (NSS) for the Sonny Motor Company’s automobile lines. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the system.

## Purpose

The Navigation and Steering system will guide Sonny Motor Company automobiles on city streets and highways from their current locations to user supplied destinations. The system consists of a computer and two steering actuators controlled by the computer. The system will interact with other systems computerized and mechanical, as part of the overall system that is each automobile. Figure 1.1-1 shows the Navigation and Steering system block diagram.



## Scope

This document describes the Navigation and Steering system’s requirements. The behaviors of the other systems are described in their respective requirements documents.

# References

This section lists all documents referenced in this SAD. If conflicts arise between the documents listed below, the order of priority will be:

1. This document
2. Customer Supplied Documents
3. Government Documents
4. All other Documents

## 2.1 Customer Supplied Documents

None

## 2.2 Government Documents

Federal Motor Vehicle Safety Standards in effect as of 18 February 2004

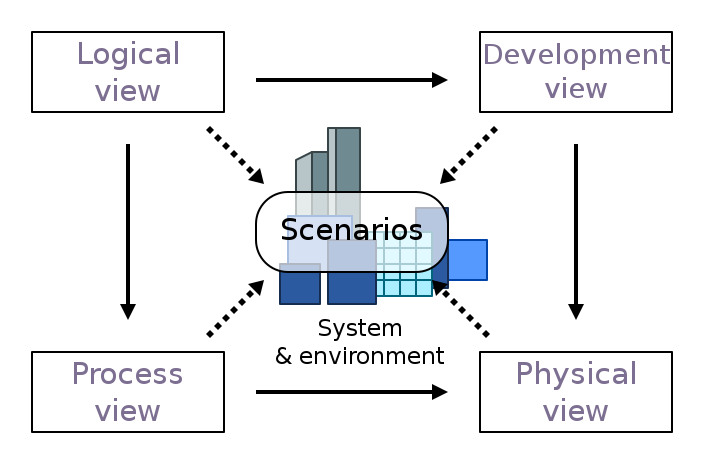
On-Board Diagnostic System Requirements, version 2 (OBD-II)

## 2.3 Other Documents

Check Appendix at the end of this document

# Software Architecture

The architecture of NSS is illustrated using the views defined by the “4+1” view model (Kruchten, 1995). This model includes: Use Case/Scenario View, Logical View, Process View, Physical View, and Development View. The following is a brief description of each view used to document the NSS application:

****

**Logical view**

**Audience:** NSS designers

**Area:** Functional Requirements: describes the design’s object model. Will also describe   
 the most architecturally significant use-cases and requirements of the system.

**Related Artifacts:** [Logical Architecture](#czz3mr2yz1n6)

**Process view**

**Audience:** Integrators

**Area:** Non-functional requirements: describes the design’s concurrency and   
 synchronization aspects. How logical view components are associated to the processes.

**Related Artifacts:** [Process Architecture](#l2k57efp93pd)

**Development view**

**Audience:** Programmers

**Area:** Software components: describes the modules as well as the subsystems of the   
 application. Describes the structural organization of elements used on the system such   
 as object files, libraries, and test codes.

**Related Artifacts:** [Development Architecture](#drd9auv5ac69)

**Physical view**

**Audience:** Deployment managers

**Area:** Describes the mapping of the software onto the hardware. Also shows the   
 distributed aspects of the system.

**Related Artifacts:** [Physical Architecture](#rsxm45rc668a)

**Use Case/Scenario view**

**Audience:** All stakeholders of the system, including the end-users

**Area:** Describes the set of scenarios/use cases which are critical to the architecture

**Related Artifacts:** [Scenarios](#v4n7thwx8km1)

# Architectural Goals and Constraints

The architectural goals listed below were chosen by selecting the architecturally significant requirements from a list containing all current functional and non-functional requirements. The NSS architecture has been designed with the following objectives in mind:

**4.1 Position and Velocity Sensing**

The systems current position and velocity will update upon receiving a valid Position Velocity Message. The system will cease automatic steering if a certain quantity of consecutive invalid or Position Velocity messages indicating errors are received within a certain timeframe.

**4.2 Route Selection**

Receiving a valid Routing Request Message determines multiple potential routes from the

vehicle’s current position to its destination.

**4.3 Steering**

Receiving a valid Selected Route Message will trigger the system into steering the vehicle along   
 the selected route. The system takes the vehicle’s current position and velocity into account as   
 well as the selected route information when determining the timing and magnitude of its steering   
 commands.

**4.4 Obstacle Avoidance**

Receiving a valid Obstacle Position Message indicates a moving obstacle is in the vehicle’s path   
 and submits one more Speed Change Requests to keep the vehicle at a safe distance. The

vehicle will return to its previous speed once the obstacle is no longer a factor. The system may also receive an Obstacle Position Message to dodge a stationary obstacle.

**4.5 Driver Controls**

The system will cease automatic steering if a State Chance message indicating “Suspend” or   
 “Off” is received. The vehicle will not steer or generate Speed Change requests until the vehicle is   
 back on the selected route OR the user selects a new valid route. The Driver can manually steer   
 without the system fighting for control of the vehicle.

**4.6 Emergency Actions**

The system ceases automatic steering if Navigation and Steering system produces an error   
 which prevents it from functioning as expected.

**4.7 Diagnostic Information Generation**

If the system ceases automatic steering other than from State Change Messages, at least one   
 System Status Message will indicate what caused the automatic steering to cease. The system   
 accepts no Routing Request Messages until the vehicle has been shut-off and restarted.

**4.8 Reliability**

The system will not fail more than once every twelve months.

**4.9 Availability**

The system will generate Navigation Display Update messages displaying the vehicle’s current   
 position 99.9999% of the time the vehicle is operating. Also be able to generate routes for 95% of   
 user inputs.

**4.10 Portability**

The software is designed to maximize portability. This will ensure the software is able to be   
 hosted in later model years without needing to significantly redesign it.

The major design and implementation constraints for the system are:

1. Must comply to the Federal Motor Vehicle Safety Standards in effect as of 18 February 2004.
2. Must adhere to the On-Board Diagnostic System Requirements, version 2 (OBD-II).

All functional and non-functional requirements will be maintained over time as the architecture continues to be developed as later detailed in the Use Case/Scenario model.

# Logical Architecture

This section describes the logical structure of the system as well as any definitions or acronyms. It starts from the overview of the architecture and then presents its key structural and behavioral elements such as usage and dependency. The Logical view of the SMC navigation system is separated into 3 main cases - user interface, application layer, and subsystems.

User interface - It contains classes for each of the actors use to communicate with the system. Classes would include to GPS routes, Speed calculations, Redirecting routes due to obstacles.

Application Layer - This case would have the main applications performed by the system. The application would send and receive requests from the users.

Subsystems - Subsystems such as Speed Control System, Driver’s Interface, GPS Receiver, Steering Actuators, Obstacle Detection System.



# Process Architecture

The process view will describe the system’s decomposition as well as the forms of communication between processes, like message passing, activity between components, and message sequencing.

6.1 **UpdatePositionAndVelocity**

Upon receiving a valid Position Velocity Message (PVM), this process will update the vehicle’s current position and velocity.

6.2 **CeaseAutoSteering**

This process will check if one of the following conditions is true:

1. The system receives 3 consecutive invalid messages or PVMs indicating GPS signal integrity errors within 2 seconds.
2. The system cannot safely avoid an obstacle that is reported within 100ms.
3. The system receives a State Change Message (SCM) indicating Off.
4. The system detects an error that prevents it from safely performing its functions.

If so, this process will force the system to cease automatic steering.

6.3 **MakeNDU**

Upon receiving a valid PVM, this process will check if it is one of every 4th such PVMs sent so far. If so, this process will send a Navigation Display Update (NDU) with the vehicle’s current position.

6.4 **DetermineRoutes**

Upon receiving a valid Routing Request Message (RRM), this process will select from 3 routes that start at the vehicle’s current position and end at the destination.

6.5 **MakeARM**

This process will check for available routes to the destination. If at least one such route is available, this process will then send one Available Route Message (ARM) for each available route. If no routes are available, this process will send an ARM indicating that no routes are available.

6.6 **ActivateAutoSteering**

Upon receiving a Selected Route Message (SRM), this process will force the system to begin automatic steering.

6.7 **CalculateSteering**

This process will load the vehicle’s current position and velocity and data for the selected route, then adjust the timing and magnitude of the current steering command according to that data.

6.8 **MakeSCR**

This process will check if one of the following conditions is true:

1. The vehicle cannot safely make a required turn at its current speed.
2. The system receives an Obstacle Position Message (OPM) indicating an obstacle in the vehicle’s path.

If so, this process will then send a Speed Change Request (SCR) indicating the necessary reduction to the vehicle’s current speed.

Once all of the above conditions are no longer true, this process will send a SCR indicating that the system will resume control and the vehicle will return to its previous speed.

6.9 **SuspendAutoSteer**

Upon receiving a SCM indicating Suspend or a Manual Mode Message (MMM) indicating that the driver is steering the vehicle, this process will force the system to suspend automatic steering. (The system will not steer the vehicle or make SCRs.)

6.10 **ResumeAutoSteer**

Upon receiving a SCM indicating Resume or a MMM indicating that the driver is no longer steering the vehicle, this process will perform the following actions:

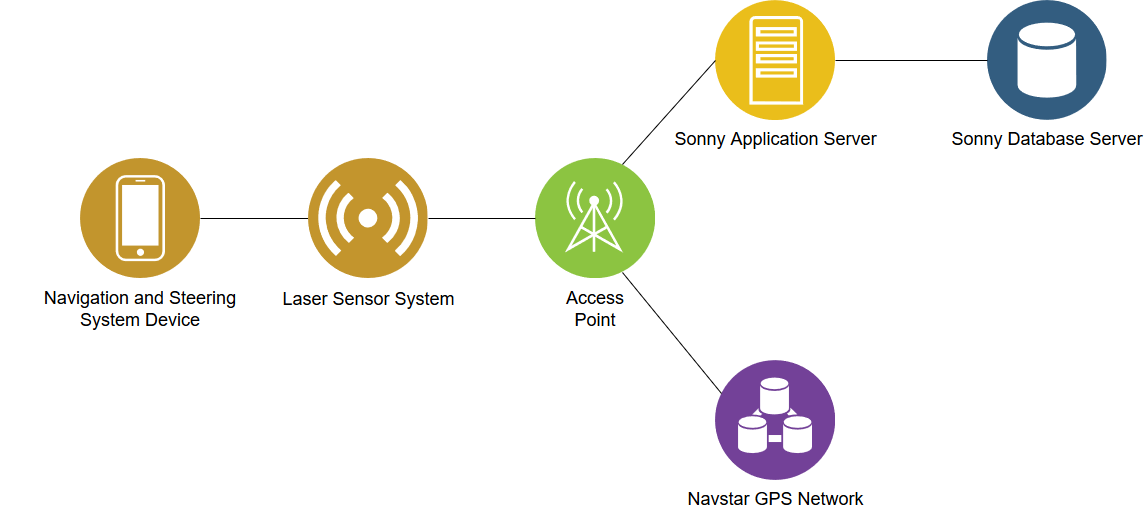
1. If the vehicle is still on a previously selected route, the system will resume automatic steering and make any necessary SCRs.
2. Otherwise, the system will send a new System Status Message (SSM) indicating that the driver must select a new route based on the vehicle’s current position.

6.11 **MakeDiagnosis**

This process will check if the system has ceased automatic steering because of something besides a SCM. If so, this process will perform the following actions:

1. Send one or more SSMs that each indicate the reason automatic steering has ceased.
2. Send a NDU indicating that automatic steering has ceased and the reason why it ceased.
3. Prevent the system from accepting RRMs until the vehicle shuts off and restarts.

# Physical Architecture



The physical architecture consists of a laser signal, access point, and connections to the Sonny servers and Navstar GPS networks. The NSS transmits a laser signal to relays data like error messages and system status to the database and network. Laser signals find an access point closest to the user, which acts as a stronger relay to the Sonny servers and Navstar network. Once the Sonny servers and Navstar networks receive the signal, they respond with up-to-date information like current position and routes for the user.

# Development Architecture

In this we will go over how the system works at the physical level. We have three separate processes that control the access to the sony database, the navstar gps network, and one for the users nav system.

The Database will be implemented in SQL and has a thread that handles database requests and establishes network procedures and connections using TCP(Transmission Control Protocol) network protocol.

The Navstar gps network has a similar thread that also handles connections to their network for user login and updates which will also need to be connected to using a reliable network protocol TCP.

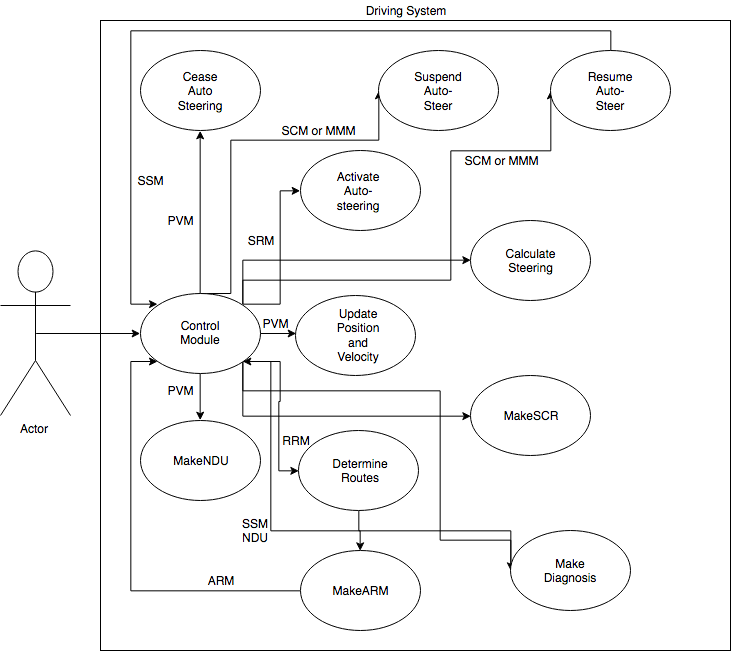
The Lidar laser sensor system on our car sends out lasers and maps out the area around it a million times a second. We take this mapping data and use python’s data analysis library Panda. We like using python for this because Panda can hold a lot of data without slowing our program down. Python is also a good choice because it will be much easier to translate the mechanical engineers specifications to code, along with the fact that the mechanical engineer will have an easier time looking at or modifying the code if need be. Along with that python works well with our SQL based database and TCP protocol.

The user process will have thread is created when a destination is set, this thread will then handle the steering of the car from the current position to the selected destination. This thread will take updates from multiple sources and determine the next correct driving action to take, while also updating the functions that it needs for said information.

A lot of the functions on this thread are dependent on the fact that the user is connected to the internet, but not all.

# Scenarios

The general functionality of the system can be seen in the following diagram:



# Non-Behavioral Requirements

The qualities of the Navigation and Steering System (NSS) include the following quality goals:

1. Reliability
   * The system’s Mean Time Between Failure (MTBF) will be required to be less than twelve months. A failure is defined as any error within the system that causes it to cease automatic steering, or any error that causes the system to behave in an unsafe manner.
   * The system's Mean Time To Recovery (MTTR) will be required to be 2 hours and less time. Recovery is defined as the process of restoring functionality after a system failure.
2. Availability
   * The Navigation and Steering System (NSS) shall be achieve a minimum of 99.5% uptime.
   * Navigation Display Update (NDU) messages indicating current vehicle position shall be generated 99.9999% of the time while the vehicle is operating.
   * Routes will generate at least 95% of test cases supplied by customer.
3. Portability
   * The source code should be designed to run on different processors to improve portability.
   * The source code should be able to compile and execute on major operating systems such as Windows 10, macOS, and Linux.
   * All recording of date and time shall be stored in UTC (Universal Time Coordinated) to maintain consistency.
4. Survivability
   * The system shall backup data at a maximum of every three hours to prevent loss of data.
   * Recovering data from a backup shall not exceed a length of an hour.
   * Customer service support tickets through email shall be resolved 90% of the time within one week.
5. Efficiency
   * All recordings of time shall be accurate up to one nanosecond.
   * Database queries shall take less than five seconds under maximum load.
   * The system should able to process a minimum of 100,000 queries per hour under maximum load.
   * The maximum storage capacity of the server(s) shall scale with the number of users.
   * The storage capacity shall have at least 20 percent of reserved space available under maximum load.

# Appendix

Howell, E. (2013). Navstar: GPS Satellite Network. Retrieved March 18, 2018, from https://www.space.com/19794-navstar.html

Kruchten, P. (1995). *Architectural Blueprints—The “4+1” View Model of Software Architecture*. [online] Cs.ubc.ca. Available at: http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf [Accessed 11 Mar. 2018].

En.wikipedia.org. (2018). *4+1 architectural view model*. [online] Available at: https://en.wikipedia.org/wiki/4%2B1\_architectural\_view\_model [Accessed 4 Mar. 2018].