Oregon State University
College of Engineering

CS Capstone Problem Statement

October 10, 2017

NVIDIA Jetson TX2 Infotainment and Black Box

Prepared for

Kevin McGrath ________ Signature ________ Date

Prepared by

Group 15
Team Wombat

Victor Li ________ Signature ________ Date

Ryan Crane ________ Signature ________ Date

Nicholas Wong ________ Signature ________ Date

Abstract

The abundance of vehicle sensors, necessity of data recording, and prevalence of infotainment systems in modern automobiles creates a new avenue of third-party aftermarket hardware that can combine the functionality of a black box and that of an infotainment system, utilizing a thorough set of modern sensors. To date, no existing aftermarket systems offer such infotainment and logging capabilities of a black box in a complete package.

This untapped avenue of hardware can be capitalized upon by the creation of a software system on a hardware component(s) that merges the data logging capabilities of a black box system with the media, navigation, and vehicle setting management capabilities of an infotainment system, complete in a unit that can be installed in a vehicle and allows the driver to access relevant recorded data. It will make drivers more knowledgeable about their vehicle by giving them easy access to an understandable range of vehicle information that may be useful in a range of situations on and off the road, whilst also offering drivers with older vehicles, coupled with low-volume car manufacturers, a complete piece of hardware that can modernize their vehicles' infotainment, connectivity, and data logging capabilities.

Contents

1 THE PROBLEM

Currently, many companies design and sell aftermarket infotainment systems for install on vehicles, however these systems are largely standalone, and have no control over a vehicle's systems nor widespread access to a modern vehicle's wide array of sensors. Additionally, existing aftermarket systems (as in, installed after a vehicle's manufacture) generally don't have data logging functionality, and do little else than their main purpose of providing basic infotainment functionality.

This presents us with our problem: how to integrate an infotainment system, a device seen in the center consoles of modern vehicles that allows control over media, navigation, and vehicle settings, with an automotive black box: a device that records snippets of a vehicle's data that originates from a modern vehicle's numerous sensors. Doing this will give us a cohesive aftermarket infotainment and black box system never encountered before.

The end solution developed as a solution to this problem must be based on an NVIDIA Jetson TX2 embedded computer, and must be able to be wired into a modern vehicle via a CAN bus or OBD-II port in order to take advantage of its sensors. Additionally, the solution should be much more extensible and powerful than existing infotainment systems in giving the driver/user access to more features that may include extra control functionality or interactivity with their vehicle and its data in a way that most vehicular infotainment systems don't offer.

2 THE SOLUTION

The aforementioned problem involves the development of a software system that will be installed onto the NVIDIA TX2 hardware. The hardware part of this solution shall take the form of a micro-ITX that houses the NVIDIA Jetson TX2 unit and a touchscreen connected by HDMI or embedded DisplayPort connection methods, whichever method proves to be most optimal. The software system developed will run a functional infotainment system that will also have the data logging capabilities of a black box and be able to control various functions such as attached lighting systems (supplementary brake lights, reverse lights, and turn signals). Altogether, the complete deliverable solution is both the necessary hardware and software in the form of a headunit, which can be utilized and marketed as a alternative to current aftermarket infotainment solutions.

2.1 User Interface and Functionality

Developing a software system that can fulfill the purpose of infotainment in turn necessitates the development of a user interface that will allow users to interact with the system via a touchscreen. The user interface should stress ease-of-use and be mass-market oriented in order to make it appeal to as many potential customers as possible. Therefore, the user interface should be simple and minimalistic. It should be designed be fast to respond in order to minimize driver distraction by offering quick access to media, navigation, vehicle data, and settings.

Media, navigation, and vehicle data functionality will utilize open source third party solutions where able. This simplifies the development process by reducing the redundant engineering needed (i.e. utilizing existing open source navigation), increasing the speed of delivery of many elements of the solution, which may have already been thoroughly tested and ready to use.

2.2 Sensors, Data, and Additional Hardware

To improve the info that an infotainment system aggregates and makes visible, the solution developed will use multiple GPS (global positioning system) receivers, an AHRS (Attitude, Heading, and Reference System), and the vehicle data that is pulled from a modern vehicle's OBD-II (On-Board Diagnostics II) port or CAN (controller area network) bus.

In order to give the overall solution control capabilities over elements of a vehicle, the software system on the NVIDIA TX2 hardware will be designed to support some additional connected devices and hardware. This hardware, including supplemental lighting (turn signals, brake lights, and reverse lights), as well as potential dashcams (dashboard cameras that record video of the road while the driver is driving) and backup cameras (a camera mounted on the vehicle's rear that feeds video to the screen of what's behind when reversing) do well to further differentiate our system from existing aftermarket infotainment solutions and make it more capable.

2.3 Legal

Federal guidelines should be followed with regards to anything implemented in this solution, as the automotive industry is highly regulated in terms of safety and implementation. Camera support implemented for additional hardware shall follow federal and state privacy laws regarding the data they record and display.

2.4 Distribution and Copyright

Ideally, the code developed to solve this problem should have an open source license, allowing the software solution to be modified, reproduced, and improved freely by second or third parties. For the overall system, our development team should retain the overall intellectual property in the implementation, however not necessarily all the code.

3 Performance Metrics

We will evaluate our performance on this project by examining whether or not we have fulfilled a variety of requirements that will be labeled ″necessities″ and an additional set of features that will be considered ″superlatives″.

″Necessities″ will outline the essential and core functionality that is expected of our infotainment system and black box requested by our client; not fulfilling these will amount to failure of the project; completion of all necessities amounts to the completion of the project. Essentially, ″necessities″ will be considered our base requirements in software development. An example of a necessity requirement would be: successful implementation of frequency modulation radio (FM Radio) that can receive and tune to US radio frequencies, displaying any radio data system (RDS) data onto the touchscreen for the current radio channel.

″Superlatives″ will outline additional features that are not explicitly required to fulfill the requested functionality of the project, but are desirable and will make our system more capable and versatile.

3.1 Fulfillment

Both metrics of necessities and superlatives will be graded on a yes/no functional criterion, as that is the best way to assess elements of the system. This is because there is little to no middle ground with much of the functionality implemented; for example, navigation cannot ″somewhat work″: it either functions or it doesn't function. This makes it easier to write requirements, test the implementation of said requirements, and then subsequently gauge the completeness of the implementation used to fulfill the requirements, enabling our development process to be much more streamlined and holistic.