**Design Specification**

1. **Project Overview**

Automobiles is a primary mode of transportation in the US. One of the issues for car drivers is checking for cars approaching from blind spot area. Some modern cars, especially self-driving car, have already fixed this problem by integrating the detection system into the car itself. However, many old models or cheaper models still do not have this system yet. The goal of this project is to create a system for detecting an approaching car in one’s blind spot area and alert the driver with sound.

1. **User Requirements**

Sponsor requirements:

The sponsor requires that the system is able to detect objects when they get into detection zone. The system will then warn the driver through sound only. Visualization will not be used in this system in order to not distract the driver from driving. The sponsor also requires that the system is a mobile app that connects with the sensor through wireless technology. The app should have a good UI and a mechanism to auto start-up after setting up. The sensor also needs to be low power consumption. That is because it will be convenience for the user to not have to replace the battery of the sensor often. The sponsor also states that the system should emphasize on accuracy, reliability and usability in other to compete again other products in the market. This will be achieved through both hardware choices and software design.

User Action:

For this system, there will be only one type of user, which is the automobile driver. In order to use the system, a driver needs to mount the sensor to a place on the car (the detail of how to set up is not yet decided). The user will need to do a connection setup with the sensor and then enable the app. After the app is enabled, it will start up automatically whenever the user enters the car and will continuously check for objects moving into its detection area in the background, warning the user through sound.

1. **Development Environment**

Hardware

* An Android mobile device
* An AWR1642 Obstacle Detection Sensor with Wide Field-of-View Antenna Evaluation Module
* A SimpleLink Wifi CC3220SF Wireless Microcontroller Launchpad Development Kit.

Software

* Android studio for developing Android native app
* Ti mmWave SDK

Explanation for Environment Choice

Native application is chosen instead of hybrid(progressive web app) or web app for this system. There are two reasons for this decision. First, the sponsor requires a real-time reliable system for detecting approaching cars in blind spots. Since cars might travel at high speed, it is important that the application runs quickly to detect the cars approaching blind spots in time. Therefore, native app would be a better choice than hybrid or web app because native app has better performance. Second, the sponsor wants a wireless connection to the sensors. This wireless connection might be Bluetooth or Wi-Fi or both. Native app would allow more control over both of these. Web app is not capable of controlling wireless technology and hybrid app would have control over Bluetooth but not Wi-fi.

In the automotive field, there are several popular proximity sensors: LiDAR, RADAR, ultrasonic, and camera. In this system, it is decided that RADAR will be used. Compared to camera and LiDAR, RADAR works better in different environmental condition (like rain, dust and smoke) and it is also cheaper compared to LiDAR. Compared to ultrasonic, RADAR has a higher range of detection and has much faster time of transit for their signals. This means that RADAR would be more reliable and is a better choice.

1. **Deployment Environment**

The deployment Environment should be very similar to the development environment. The possible changes lie in the hardware parts. The current sensor is a good choice but if better sensors are found, then it can be replaced. Furthermore, for development environment, an evaluation board of the sensor is used to develop the system. For the final product, the sensor board will be customized to reduce the cost, energy consumption and size of the sensor.

1. **Architecture**

The sensor

Code for the Sensor Processor

Wireless communication

The mobile application

SensorProcess Class

SensorReceiverService Class

Controller class and other necessary classes for mobile software

The code for the sensor processor is a demo application for obstacle detection that is developed by TI company. The code is written in C and is used to simplify the work to test the sensor. Customized application will need to be developed for deploying the system.

The SensorReceiverService Class is an extension of Android Service class. A Service is a special kind of thread that can continue to run even though the app is in the background. This Service is responsible for receiving data from the sensor, send it back to the main thread for further processing.

The SensorProcess class would receive the data when the data is passed back to the main thread. It is responsible for further processing of the sensor data and alerting the user through sound.

The ViewController and other file are responsible for the basic configuration of an Android app, user interface and initialization of other class.

1. **Implementation Strategies**
2. **User Interfaces**

User stories:

* When the user opens the app, the user will be shown a Home Page.
* The Home Page will show the status of the sensor, whether or not it is connected, a enable button and a set up connection button.
* When the start application button is clicked/tapped, the app will start scanning for objects. When an object is in the detection zone, an alarm will sound warning the user.
* When the set up connection button is clicked, The app will allow the user to set up the connection with the sensors (how this is implemented is currently undecided).

1. **Test and Integration plan**

There are several tests requires for the system. First, the wireless connection needs to be tested to make sure the connection has been established so that the device can connect to the sensor (after first time set up) without any action from user. Second, we need to test the accuracy of the system to see how well it detects. Third, we need to test to see the response time of the system to measure the reliability of the system.

1. **Project timeline**

Sep 21 – Deadline for deciding on the sensor a

Sep 30 – Deadline for getting the sensor and done with the basic UI

Oct 7 – Deadline for connect the sensor and the mobile device through wireless

Oct 14

Oct 21

Oct 28 – Deadline for object detection

Nov 4 – Initial Presentation and various documents submit

Nov 11

Nov 18 – Deadline for perfecting the system.

Nov 26 – Final presentation

1. **MVP**

The MVP of the system is an Android application that is able to detect objects and warns the user through sounds. The app should connect with the sensor through wireless connection. A simple visualization will be given to observe the result of the sensor. There are focus on hardware in this project that might take time from developing a better software.

1. **Bibliography**

* Referencing several online websites for comparing different type of sensors.
* Referencing several Blind Spot Detection Systems to survey existing products.
* Refencing Sensor companies’ website while looking for industrial sensor and also referencing some of the knowledge in programming sensor
* Referencing websites to choose the sensor and reference their documents to program the sensor if needed
* Referencing Stack Overflow, google and other sites to complete the Android Mobile App