Software Requirements Specification

(Autonomous vehicle for mission challenge)

Team: SmartRide

Haorui Hao	U1510022	Group: 15-6
Bekzodjon Norkuziev	U1510405	Group: 15-6
Shokhrukh Aminov	U1510423	Group: 15-6
Abdulaziz Abdukakhkharov	U1510301	Group: 15-1

Table of Contents

1. Introduction

This section gives a scope description and overview of everything included in this SRS document. Moreover, the purpose for this document is described and a list of abbreviations and definitions is provided.

1.1 Purpose

The purpose of this document is to give a detailed description of the requirements for the project called “Smart Car Project”. The goal of this project is to make an autonomous self-driving car, capable of maneuvering around bends, avoiding obstacles and following traffic signals and road signs.

The tools used in this project and described in this document are:

- OpenCV library

The hardware used in this project and described in this document are:

- RC Car serving as the actual self-driving car
- Raspberry Pi 3B used for controlling the car speed and turning
- Ultrasonic sensors to detect objects in front of the car
- IR Senors to detect the line
- Camera to detect and analyze road and signs using image conversion of OpenCV library

1.2 Scope

The “Smart Car” is hardware-equipped autonomous car that is built for mission challenge, thereby the car is required to do start controlling itself autonomously until finish based on some regulation sign and obstacles on the road. Road safety has been an issue for as long as cars have been in existence. Car has several advantages, as described below:

- Better road safety: Machines are not prone to human-error and distractions, leading to swift and appropriate responses in real-time road conditions.
- Solution to parking problem: Most of the modern cities face parking problems and which can be resolved by this solution.
- Better traffic discipline: Better law enforcement can be achieved and traffic can be managed by capping speed in various regions.

1.3 Definitions, acronyms and abbreviations

Term	Description
System	The full package which takes an input and gives an output to run the car
Tags	Attached label to road which shows extra information
Function	External function calling to do some calculation or analyzing

1.4 References

IEEE Standard 830 – 1998 Recommended Practice for Software Requirements Specifications.

1.5 Overview

The remainder of this document includes three chapters and appendixes. The second one provides an overview of the system functionality and system interaction. This chapter also introduces different types of stakeholders and their interaction with the system. Further, the chapter also mentions the system constraints and assumptions about the product.

2. Overall Description

2.1 Product Perspective

This system consists of following components:

- RC Car: resembles real-world car
- Raspberry Pi: along with motors, driver form the car’s hardware
- PiCam: collects visual data
- Ultrasonic Sensor: collects information regarding distance
- IR sensors: detect the color, used to detect line

RC Car acts as an analog to the real world car with basic motion capabilities. It is battery operated.

Raspberry Pi controls the actual car’s hardware. It runs the car’s motors and collects distance from the ultrasonic sensor. Thereby, performs the driving decisions using sensors and camera.

2.2 Product Functions

- Autonomous motion
- Obstacle avoidance
- Tracking road line
- Following traffic sign and regulations

2.3Operating Environment

Thesystem with the following minimum specifications:

- Operating System: Linux distribution, Raspberry OS
- Processor: 1.2 GHz ARM
- Network: 802.11n Wireless LAN
- Memory: 1GB or more
- Ability to interface with a camera
- Ability to interface with a distance sensor and IR sensor

2.4 Assumptions and Dependencies

The assumption is that the roads are ideal with no environmental variations like potholes, rash drivers, and changing weather conditions etc.

All traffic signals are clearly visible and not malfunctioning.

3. External Interface Requirements

3.1 User Interfaces

Since this an autonomous self-driving car, the user does not directly interact with the car. Rather, the car may have some ad-hoc computer system, like smart cars. The smart systems provide navigation aids like autonomous driving. They, thus, act as an interface between the user and the car.

3.2 Hardware Interfaces

The embedded software in Raspberry Pi will itself communicate with the car. It will will run the car by generating PWM signals sent to driver to run the motors.

The Raspberry Pi Camera Module connects onto the Raspberry Pi via a non-standard parallel interface.

The ultrasonic distance sensor communicates with Pi using a regular GPIO interface.

3.3 Software Interfaces

The software system on the Raspberry Pi has the following interfaces:

- Camera input using a built-in library
- Speed and turn data output using RPi.GPIO library
- Direction and speed data input using built-in functions
- Output to motors using built-in libraries

4. System Features based on Use-cases

Here is the list of possible use-case while smart car is running:

Case	Description
Encountering clear-straight road	Front of the car is straight and clear.
Encountering left side	The road in front of the car is curving the left (may be up to 90 deg.)
Encountering left side	The road in front of the car is curving the right (may be up to 90 deg.)
Encounter "red" traffic signal	Traffic signal in front of the car with a red light.
Encounter "yellow" traffic signal	Traffic signal in front of the car with a yellow light.
Encounter "green" traffic signal	Traffic signal in front of the car with a green light.
Encountering "Crosswalk" stop sign	There is a " PEDESTRIAN " line in front of the car.
Encounter obstacles	There is an obstacle in front of the car.

4.1 Case: Clear straight road

4.1.1 Description

There is a clear, straight road in front of the car.

4.1.2 Response

Algorithm determines whether the path in front of the car to be a clear, straight road. The car, in response, runs the motors at their full speed, accelerating the car to its full speed.

4.2 Case: Encounter left curve

4.2.1 Description

When encountered with a left curve the car slows down and starts to turn until a new stimulus is introduced into the environment.

4.2.1 Function Response

The computer vision data is sent to the algorithm model for determining. When the algorithm classifies the road ahead as a left turn, the car slows the forward motors down and start the turning the front wheels to the appropriate degrees in the left direction.

4.3 Case: Encounter right curve

4.3.1 Description

When encountered with a right curve the car slows down and starts to turn until a new stimulus is introduced into the environment.

4.3.1 Function Response

The data is sent to the algorithm model for determining further action. When the machine learning model classifies the road ahead as a right turn, the Arduino slows the forward motors down and start the turning the front wheels to the appropriate degrees in the right direction.

4.5 Case: Encounter red traffic signal

4.5.1 Description

The car encounters a red traffic signal right in front of it on the road. The car comes to a halt. The car remains stationary until signal reverts back to green, then resumes its motion, depending on the environment and the stimulus.

4.5.2 Functional Response

The computer vision data recived post processing results in alerting the Raspberry Pi of the presence of a red traffic signal. The Raspberry Pi instructs to halt the car's motion. The Raspberry Pi makes car to stay in stationary position as long as traffic signal is not green. Once green, Raspberry Pi instructs to accelerate and waits for the next stimulus.

4.6 Case: Encounter yellow traffic signal

4.6.1 Description

The scenario is very similar to 4.5 with the difference that the car encounters a yellow traffic signal right in front of it on the road. The car slowly decreases it's speed and comes to a halt. The car remains stationary until signal reverts back to green, then resumes its motion, depending on the environment and the stimulus.

4.6.2 Functional Response

The vision data received post-processing results in alerting the Raspberry Pi of the presence of a yellow traffic signal. The Raspberry Pi instructs the Arduino to slowly decrease the car's speed and halt its motion. The Raspberry Pi makes the car to stay in stationary position as long as traffic signal is not green. Once green, Raspberry Pi instructs to accelerate and waits for the next stimulus.

4.7 Case: Encounter green Signal

4.7.1 Description

There is a traffic signal in front of the car with a solid green light. If a GREEN Signal is encountered, the car accelerates straight until a new stimulus is introduced in the environment.

4.7.2 Functional Response

The vision data received post-processing results in alerting the Raspberry Pi of the presence of a green signal. The Raspberry Pi instructs to accelerate and waits for the next stimulus.

4.10 Case: Encounter Pedestrian Line

4.10.1 Description

There is a "Pedestrian" sign board in front of the car. If a sign is encountered, the car will stop for 5 seconds, then move forward until a new stimulus is introduced in the environment.

4.10.2 Functional Response

The computer vision data received post-processing results in alerting the Raspberry Pi of the presence of a Pedestrian sign. The Raspberry Pi instructs to halt the car's motion. The Raspberry Pi waits for a fixed time seconds, before instructing to accelerate and waits for the next instruction.

4.12 Case: Encounter obstacle (Case of Normal Braking)

4.12.1 Description

The car encounters an obstacle right in front of it. The car comes to a halt, and waits for the obstacle to be cleared before resuming action.

4.12.2 Functional Response

The computer vision data received, along with the data from the distance sensor (ultrasonic), alerts the Raspberry Pi of the presence of an obstacle. The Raspberry Pi instructs to halt the car's motion. The car remains stationary so long as the obstacle is present. Once cleared, the Raspberry Pi instructs to accelerate and waits for the next instruction.