Software Design Document

ADAS SYSTEM

Autonomous Self-parking Car

VERSION: 1.1

History of Changes			
Revision	**Issue Date**	**Description**	**Author**
Ver 1.0	05/11/2019	Creation of the first draft of the SDD.	BELGHITI KAOUTAR/ ELKARAKHI SOUKAYNA
Ver 1.1	08/11/2019	Update the SDD template and add some content.	BELGHITI KAOUTAR/ ELKARAKHI SOUKAYNA
Ver 1.2			

Table of Contents

-

1.Glossary:

IHM……...…….....….....Interface Human Machine
CAN..……...…….....Controller Area Network
OpenCV……...….....Open Computer Vision
AUTOSAR……....……..….Automotive Open System ARchitecture
MISRA C……...…….....Motor Industry Software Reliability Association
SDLC….……...……...……..…..…Software Development Life Cycle
SRS….……...……....….....……..Software requirements specification

2.Introduction

This document is the Software Requirement Specification(SRS) for the self-parking car task (SPC). We will describe in detail the system requirements, desired behavior, and all the functionalities of the system. Using the human-machine interface in the vehicle, the SPC system can be activated by the driver when the vehicle's speed is under 2 km/h. The system will then scan its surroundings using distance sensors then the parking process shall start only if an empty spot is found and the user confirmed the displayed spot to start parking using the IHM interface. Once one is selected, the system will take control of the vehicle and steer itself into the spot and shift into park. The system will also keep track of obstacles while undertaking this maneuver and simultaneously brake if one is detected. The motivation behind this system is to prevent injury to both passengers and outside pedestrians, as well as provide ease of use for the driver.

3. Scope

3.1 Subject :

The goal from this document is to provide a detailed and precise representation of our self-parking car system. It will give descriptions of this system and its components both software and hardware. The system interactions and reactions depend on various external and internal conditions that will also be included in this document.

3.2 Project overview

The goal of this system is to make the parking process easier for drivers and reduce accidents resulting from traditional parking maneuvers. This system will be configured to ensure that the vehicle can automatically park itself in either a parallel or perpendicular parking space. The system will identify parking spots, allow the user to accept or reject each spot as it is presented, and once a spot is selected, it will control the gear, speed, and steering of the vehicle in order to maneuver it into the parking spot. The distances sensor will be used to detect any obstacles and warn the drivers in case he is getting closer to an obstacle. The parking process shall start only if an empty spot is found by the system and the user agreed to start parking on the IHM interface, but if an obstacle is detected in the way during the maneuver, the system will stop the vehicle and cancel it, returning control to the driver. It will not be able to determine if it is currently safe to begin parking but instead will rely on the driver's judgment.

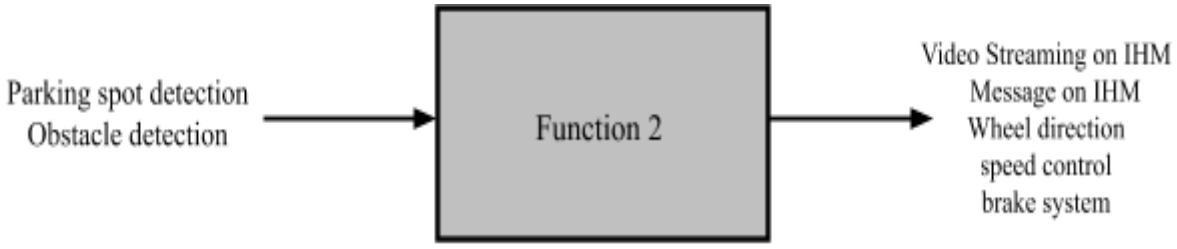

Figure 1: Block functionality

3.3 Document Overview

This document gives

- Description of software and hardware environnement.
- Design and implementation of High level diagram of the system
- Design and implementation of data flow diagram of the system.
- Detailed description of differents blocks of the self parking system

3.4 Definitions, acronyms, and abbreviations

Term	Definition
Self-parking Car (SPC)	The name of the system being designed, meant to park a car automatically within certain constraints.
Controller Area Network Bus (CAN Bus)	Serves as the main method of communication via electronic signals between the various systems and subsystems of a vehicle.
Driver	The operator of the vehicle, able to interact with the controls of the vehicle and the HMI.
Human-machine Interface (HMI)	The touchscreen display set in the middle dash of the car which serves as the main point of interaction between the driver and the system for initiation.
Parallel parking	Parallel parking is a method of parking a vehicle parallel to the road, in line with other parked vehicles.
Perpendicular parking	Perpendicular parking is a type of parking that requires cars to be parked side to side, perpendicular to an aisle or curb. This type of parking takes less space than parallel parking and is commonly used in parking lots and car garages.
Sensors	Ultrasonic sensors on all sides of the vehicle, and visual sensors(cameras) on the front and rear of the vehicle.

4. System environment

4.1 Hardware environment

In this section, The equipment used to realize this task "Self-parking car "will be described:

- **Distance sensor:**

HC-SR04 is a low-cost sensor, it operates with a supply voltage of 5 volts, has a measurement angle of about 15 ° and allows to measure distances between 2 centimeters and 4 meters with an accuracy of 3 mm.

- **Car chassis:**

We will develop this chassis using our own components, to be fit our requirements.

- **Motor DC:**

DC series motor there is a linear relationship between the applied voltage and the speed, given a certain load. The higher the voltage, the higher the rpm. This means that speed and torque can simply be controlled by changing the applied voltage. Also, it doesn't need complex electronics to be controlled. Finally, the DC motor allows for quick start-stop acceleration.

Tension	3v to 12v
Couple	800g/cm
Reduction Report	1/48

Table 1: Characteristics of the Motor CC

- **Servomotor:**

To control the direction of the wheels we will use MINI TOWER PRO SG90, this servomotor has the technical specifications presented below in table 2

Dimension	22 x 11,5 x 27 mm
Weight	9g
Speed	0.12 sec/60° sous 4.8V
Couple	1.2Kg/cm sous 4.8V
Tension	4.8V – 6V

Table 2: Characteristics of the servomotor

❖ **Raspberry pi 3:**

To control the vehicle and various sensors, a raspberry pi prototyping board was chosen.

❖ **Arduino UNO**

is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header, and a reset button the. We will be used to converts the analogical signal to numerical ones.

❖ **Camera 8MP:**

To realize the trajectory prediction system, a camera will be used. This camera has the technical specifications presented below in table 3:

Sensor	8 MP
The sensor dimension	3280 x 2464 pixels
Video resolution	1080p30, 720p60 et 640x480p90
The pixel dimension	1,4 µm X 1,4 µm with technology OmniBSI

4.2 Software environment:

In this section, we will describe the software environment :

❖ **Matlab software:**

used to develop the logic of automated parking and simulate the system.It also will be used to calculate the necessary calculations for the planning of the trajectory.For the first time, the simulation will be created in the Matlab environment, C or Python language will be used as a programming language to develop the algorithm of the system

❖ **CANopen library :**

The CAN Bus module counts with a C++ library that lets you manage the CAN Bus module in a simple way. This library offers an simple-to-use open source system.In order to ensure the same code is compatible in both platforms (Arduino, Raspberry Pi and Intel Galileo) we use the ArduPi libraries which allows developers to use the same code.

❖ **the OpenCV library:**

It provides many very useful features for image processing such as face recognition, the creation of depth maps (stereo vision, optical flow), text recognition or even for machine learning. In addition, OpenCV (Open Source Computer Vision) can be integrated into both its C ++ files and its Python scripts. Especially in terms of feature recognition in images taken by the Raspberry Pi, OpenCV is very helpful.

5. System architecture

5.1 Architectural design

This is a high-level diagram of our self-parking car system is an architecture that provides an overview of the entire system, also identifies the main components that would be developed,
The main objective of this part is to identify each high-level subsystem and the roles assigned to it also describe how these subsystems interact with each other in order to attain the desired application.

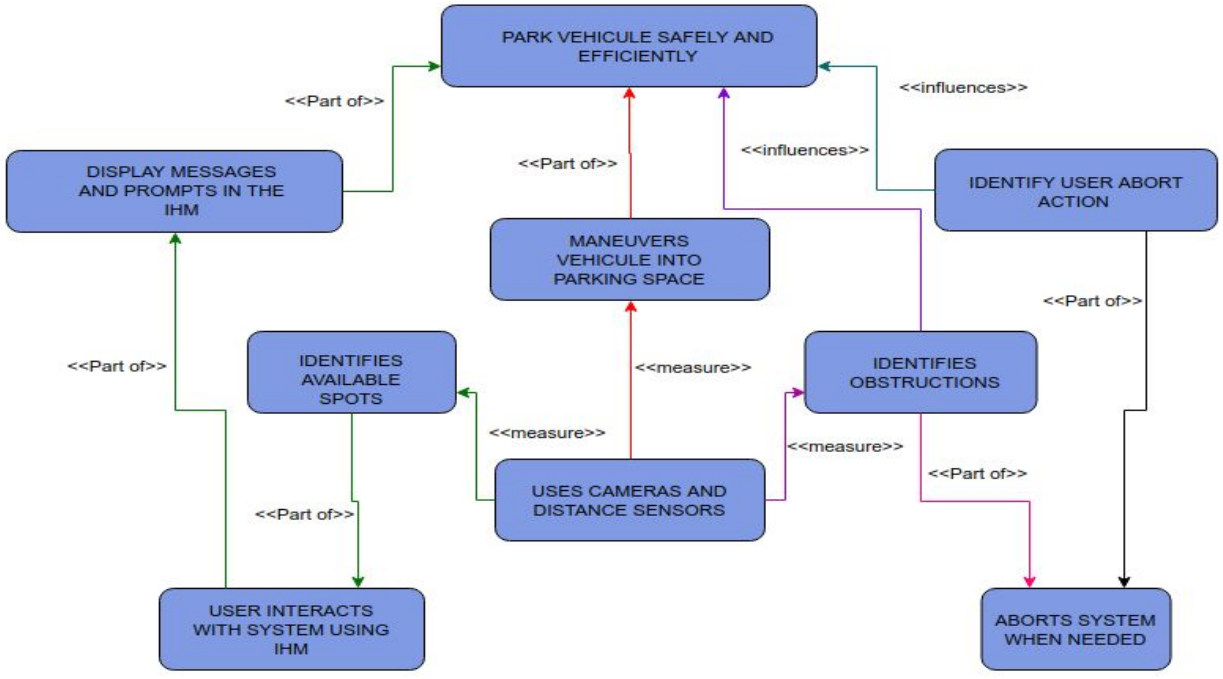

Figure 2: High-level diagram of Self-parking car system

In general, they are 9 subsystems :

- **Park vehicle safely and efficiently**: we have to make sure that the vehicle doesn't cause any damage to the vehicles around him or to any passenger.
- **Display messages and prompt in the IHM**: The system shall be able to show messages to the user to inform them of the advancement of the parking process.
- **Maneuvers vehicle into park space:** The wheels are monitored using the measures of the sensors to safely take the adequate moves and maneuvers to fit in the parking spot.
- **Identifies available spots**: By using sensor distances and cameras, the system starts searching for an appropriate spot to park. If the car doesn't find an empty spot then the user can decide to stop the parking process using the IHM interface.
- **Identifies Obstructions:** When the obstacle is detected the system shall be aborted automatically.
- **Uses cameras and distance sensors:** collects data and measures to allow the vehicle to detect its position vis-a-vis obstacles and to detect any available parking spot and it's size but also to calculate the speed and the position of the wheels during the parking maneuvers.
- **User interacts with the system using IHM:** The system shall allow the user to choose the parking type, the parking spot, or the abort system.
- **Identify user abort action:** if the user decides that he no longer decides to park and wants to take over the car and continue driving he can do so using the IHM.
- **Abort system wheel needed:** in this case, two major conditions to abort the system, when the user chooses to abort system via IHM interface or when the obstacle is detected.

5.2 Decomposition Description

SPC system is a feature of the smart card system. it's a level 5 (full automation) on the ADAS scale, which means once the user selects the data input using the human-machine interface to launch or abort system or to choose the parking type, but also to confirm the parking spot sent by the park control system. The parking spot is detected using the data gathered by the sensors and the camera and processed using the vehicle position entity once the spot is confirmed the wheels control entity starts a series of maneuvers to safely park the car at the same time the vehicle position entity keeps an eye on obstacles to launch an emergency brake procedure via the wheel control entity

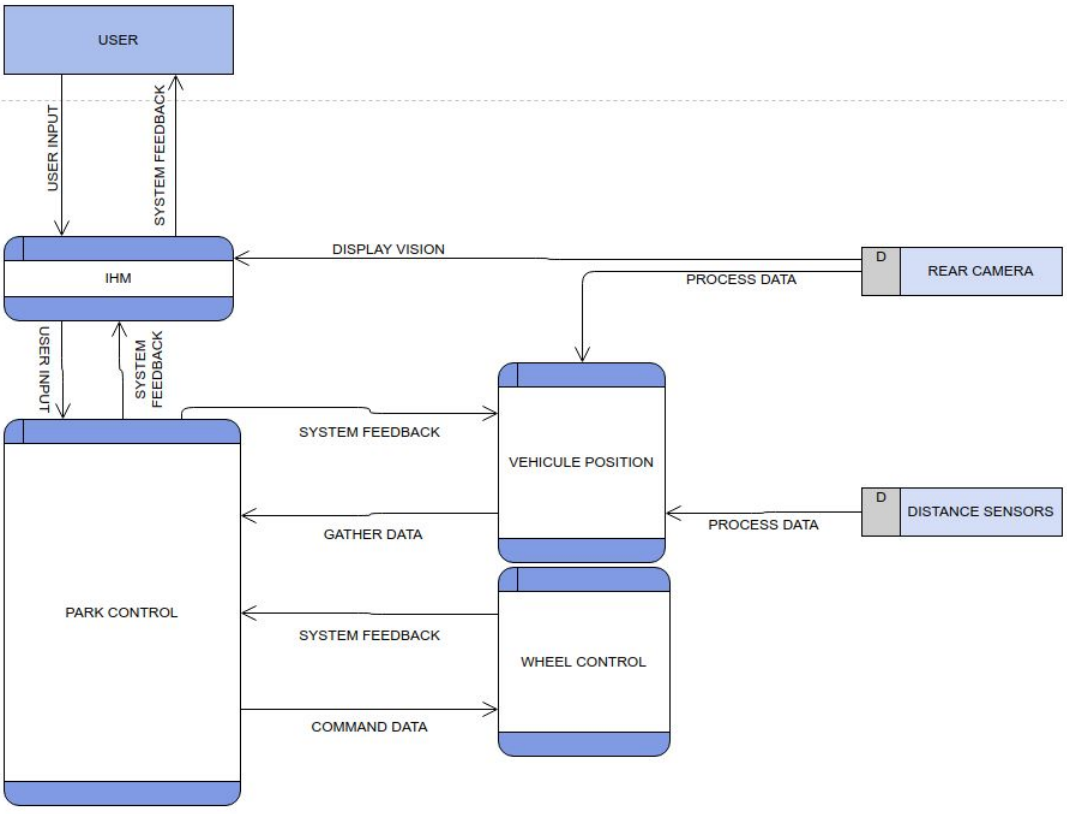

Figure 3: Data flow diagram For self-parking car (SCP)

5.3 Blocks descriptor

Before describing the different blocks of our system, we first need to define the according to the project's requirements. We have redefined those requirements as follow:

PAR_001: The system shall be able to measure different distances necessary for the automatic parking of the vehicle.
PAR_002: The distance sensors shall offer results between cars and obstacles in Cm.
PAR_003: The distance sensors shall indicate the localization of the obstacle vis-a-vis the car.
PAR_004: detecting an empty parking space.
PAR_005: Measuring the length of the empty parking space.
PAR_006: Measuring the width of the empty parking space.
PAR_007: The data collected by the distance sensors shall be sent to the HMI node through the CAN bus.
PAR_008: The data collected by the camera shall be sent to the HMI node through the CAN bus.
PAR_009: The data shall be displayed in an HMI interface of Dashboard.
PAR_010: The system shall be able to work in two types of parking parallel and perpendicular.

Our system shall be divided into four block units as follows:

First unit: System start unit

1. The user shall have access to the IHM interface where he can launch the parking mode.
2. The user shall have access to the IHM interface where he can choose which type of parking to follow parallel or perpendicular.
3. The user can abort the system using the IHM interface if needed.

Second unit: Empty spot detection unit

1. According to the user's choice, the system starts searching for an appropriate spot to park.
2. The car shall keep moving forward until it finds an empty spot or an obstacle in front of it.

3. If the car doesn't find an empty spot then the user can decide to stop the parking process using the IHM interface.
4. The spot shall be empty and wide enough for the car to fit in.
5. The size of the spot is measured using ultrasonic sensors.
6. The collected data of the distances are displayed on the IHM interface.
7. The size of the parking spot depends on the type of parking and the size of the car.
8. The user can abort this unit using the IHM interface if needed.

Third unit: the Parking process unit

1. The parking process shall start only if an empty spot is found by the system and the user agreed to start parking on the IHM interface.
2. The movement of the car wheels depends on the type of parking and the rear camera view.
3. A buzzer sound is launched whenever an obstacle is too close and its frequency increases with the decrease of the distance between the obstacle and the car.
4. The user can abort this unit using the IHM interface if needed.

Fourth unit: Emergency brake unit

1. The car shall stop immediately when the emergency brake the system and the car.
2. The system is aborted after the emergency brake is launched.
3. The user can launch the emergency brake through the IHM interface.
4. The emergency brake can be automatically launched when an obstacle is detected during the parking process.
5. Once launched the emergency brake can not be aborted.