Name: Hansel Dsilva, **Roll No.** 9535, **TE COMPS B, Batch-A**

CASE STUDY 1 – SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

1. Abstract

We aim to manufacture an 'Obstacle Detection Car System', that autonomously detects obstacles in its path, and accordingly chooses an alternate path. The obstacles are detected by the car, with the help of a Ultrasonic distance Sensor. The ultrasonic distance sensor, mounted on the front of the car, detects the obstacle, which signals the servo motor, via the motor driver which is the sole component that powers the car with the Arduino UNO. The distance sensor gets alarmed when the emitted signal is received within less than a set threshold time. The servo motor rotates the distance sensor which characterises a camera, and aids in alternate path detection.

2. Introduction

2.1 Purpose

The purpose behind our Obstacle Detection Car system was to minimize the accident cases that might be caused due to the negligence of the driver. This project is a very much scalable concept, as with the implementation of Deep Learning and Neural Networks this project can be taken on to a whole different level for real-time data analysis.

2.2 Scope

The idea of the Obstacle Detection Car System has a lot of scope. With a dataset of all the hospitals and health centres in the locality, we can find the nearest centre and design a routing algorithm to that centre, by selecting the appropriate centre based on the facilities of the centre, and also the severity of the case of the patient, having conceptualised the Obstacle Detection Car System, as an ambulance; all by the virtue of the real-time location of the car system.

Although the car system that we build aims at being driven via hand detection and gestures; we imagine a bigger picture of cars being able to detect road signs and pedestrians and act accordingly.

2.3 Definitions, Acronyms, Abbreviations

– Not applicable.

2.4 References

[1] https://ieeexplore.ieee.org/Xplore/home.jsp

[2] https://youtu.be/I2RNq90l2nc

[3] https://youtu.be/svJwmjplm4c

[4] Blobscanner library for processing environment under GNU GPL v3.

[5] Arduino development board user manual, sparkfun electronics, 2009.

[6] Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems, 2013.

[7] Ashton K. That ‘‘Internet of Things’’ thing. RFiD Journal; 2009.

[8] P. D. Minns, C Programming For the PC the MAC and the Arduino Microcontroller System. Author House, 2013.

[9] M. Banzi, Getting Started with Arduino. " O'Reilly Media, Inc.", 2009.

[10] Feng J B, 2018. Design of curtain control system based on PIC MCU. Digital Communication World.

[11] Li X Z, 2018. Brushless DC motor control system based on PIC MCU. Technology Wind.

[12]http://mafija.fmf.unilj.si/seminar/files/2007_2008/MEMSaccelerometers-koncna.pdf

[13] http://en.wikipedia.org/wiki/Accelerometer

[14] Honeywell Sensing and Control Catalog. (2000) Series 940-942 Ultrasonic Sensors. [Online]. Available: www.honeywell.ca/sensing/.

[15] Philtec Product Data Sheet. (2000) Series D6-D171 Fiber Optic Displacement Sensors. [Online]. Available: http://www.philtec.com/datasheets.htm.

2.4 Developer’s Responsibilities

The developer is responsible for -

(a) developing the system and putting the parts all together, **(b)** installing the software on the client’s hardware in this case downloading and running Arduino IDE, **(c)** conducting any user training that might be needed for using the system, and **(d)** maintaining the system for one year after installation.

3. General Description

3.1 Product Functions Overview

Some of the main parts are described below:

1. Ultrasonic sensor: An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves and measuring the time for the wave to return. Ultrasonic waves travel faster than the speed of audible sound. Ultrasonic sensors have two main components: the **Transmitter** (which emits the sound using piezoelectric crystals) and the **Receiver** (which encounters the sound after it has travelled to and from the target).

2. Arduino Uno: The Arduino Uno is a microcontroller board. The board is equipped with sets of digital and analogue input/output (I/O) pins that may be interfaced with various expansion boards (shields) and other circuits. The board has 13 digital I/O pins (six capable of PWM output), and 6 analogue I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable for Arduino UNO.

3. Motor: 4 small motors, that work on a 9V dc battery, are used to provide mobility in the system. They help the car to move forward, backwards, left or right.

4. Servo Motor: A motor used to mount the Infrared Sensors and the Ultrasonic sensors.

5. Motor Driver: The Arduino Motor Shield allows you to easily control motor direction and speed using an Arduino. By allowing you to simply address Arduino pins, it makes it very simple to incorporate a motor into your project. It also allows you to be able to power a motor with a separate power supply of up to 12v.

4. Specific Requirements

4.1 Inputs and Outputs

The main component of our project is the Arduino UNO. All the other components are linked to this component in one way or another. The Arduino UNO is mounted on a cardboard base. The Motor Driver Shield which regulates the speed and direction of the motors is mounted on top of the Arduino UNO.

The 4 motors are connected to the Motor Driver Shield by jumper wires. The Left motors act as one unit and the Right ones act as another. By doing so, the car can turn on its own spot without needing to move forwards or backwards. The wheels are attached to the 4 motors.

The code on the Arduino UNO is written in the Arduino IDE and is uploaded via a special USB cable. The servo motor, on which the ultrasonic sensors are mounted, is attached at the very start of the car. This helps in detecting obstacles easily and efficiently. The battery supply needed is a 9V battery which supplies voltage to the board, the Motor driver, the motors, the Ultrasonic Sensors and the servo motor. The input is the directions we show the car by the obstacle. The output is the direction the car moves after seeing the obstacle.

4.2 Functional Requirements

The following parts are required to run this system:

1. Arduino IDE	5. Ultrasonic sensor
2. Arduino UNO	6. Communication module (nRF24L01)
3. Servo Motors	7. Arduino NANO
4. Wheels, Connecting jumper wires	8. Infrared sensor

On the software side, only Arduino IDE is required to run the code

4.3 External Interface Requirements

In the gesture-controlled mode of the car system, the deflection of the gyroscopic values (ignoring the changes along the z-axis) guides the car in its course of locomotion.

4.4 Performance Constraints

The sensors must find an alternative route in case of object detection, and must accordingly direct the motors to pivot around the imaginary central axis of rotation, perpendicular to the ground.

4.5 Design Constraints

Not applicable.

POST LAB:

Q1. Evaluate the importance of a well-defined Software Requirement Specification (SRS) in the software development lifecycle and its impact on project success.

Ans: A well-defined Software Requirement Specification (SRS) is critical in the software development lifecycle and plays an important role in project success. The SRS serves as a thorough document that defines the client's needs, expectations, and software product specifications. Here are some of the reasons why a well-defined SRS is critical and how it affects project success:

- Clarity and Consensus: The SRS gives clarity and consensus on what needs to be created. It acts as a link between stakeholders such as customers, developers, designers, and testers. The development team may minimise uncertainty and misunderstandings by having a clear grasp of the requirements, resulting in a more focused and efficient development process.
- Scope Management: A well-defined SRS establishes clear boundaries for the project's scope, outlining which features and capabilities will and will not be included in the product. It aids in the management of project scope creep, which occurs when more requirements are added during development, resulting in cost overruns and delays. With a thorough SRS in place, the project team can limit scope changes and reduce the risks associated with frequent changes.
- Risk Reduction: Ambiguous or incomplete criteria might contribute to project risks and uncertainty. A well-defined SRS aids in the identification and mitigation of risks early in the development process. By addressing possible concerns early on, the project team will be able to make educated decisions and lower the probability of project failure.
- Assessment and Planning: A well-defined SRS is essential for accurate assessment of project durations, resources, and costs. When the development team has a comprehensive grasp of the requirements, they can more efficiently plan and allocate resources. As a result, project management improves and the likelihood of fulfilling timelines and financial limitations increases.
- Quality Control: The SRS serves as the foundation for quality assurance and testing operations. The QA team may verify that the programme fulfils the given criteria and performs as intended by referring to the requirements during testing. This guarantees that the product produced meets the client's expectations, increasing customer happiness.
- Customer Satisfaction: A well-defined SRS integrates the efforts of the development team with the requirements and expectations of the customer. Meeting the needs of the client and delivering a product that meets their expectations increases total customer happiness. Customers who are satisfied are more inclined to offer good comments, recommend the programme to others, and participate in future collaborations.
- Change Management: Projects are subject to changes throughout their lifespan as a result of changing business requirements or external influences. A solid SRS enables effective change management by allowing alterations to be evaluated against the stated requirements. This eliminates uncontrolled modifications and guarantees that any adjustments are in line with the project's objectives.

Q2. Analyse a given SRS document to identify any ambiguities or inconsistencies and propose improvements to enhance its clarity and completeness.

Ans: To spot ambiguities or inconsistencies in an SRS document, go over the criteria carefully to verify they are precise, measurable, attainable, relevant, and time-bound (SMART). Look for ambiguous terminology, contradicting assertions, and gaps in information. To achieve alignment, cross-check requirements against project objectives and stakeholder expectations. Collaborate with stakeholders and subject matter experts to confirm and explain any areas that are unclear. Rephrasing unclear sentences, adding required details, and deleting superfluous material are all suggestions for improvement. provide simple language, minimise technical jargon, and provide examples whenever possible. Ascertain that each demand is distinct and targets a particular feature. Seek input from stakeholders and undertake comprehensive evaluations to improve the SRS document's clarity and completeness.

Q3. Compare and contrast different techniques for requirement elicitation, such as interviews, surveys, and use case modelling, and determine their effectiveness in gathering user needs.

Ans: Techniques for eliciting user requirements are critical for properly gathering user demands. Here's a quick look at three popular methods: interviews, surveys, and use case modelling:

- Interviews: Direct engagement with stakeholders that allows for open-ended inquiries and in-depth conversations. Interviews provide rich qualitative data while also clarifying unclear needs and establishing rapport. They are, however, time-consuming and may not reflect all user viewpoints.
- Surveys: Effective for gathering information from a large number of users. Surveys are inexpensive and can give quantitative information. They do, however, provide limited space for extensive replies and may overlook critical background or complex requirements.
- Modelling use cases focuses on finding interactions between people and the system. It represents user needs and system behaviours in a systematic manner. While it might be effective for describing functional characteristics, it can also ignore non-functional needs or real-world complications.
- Interviews are extremely successful at obtaining complicated and specialised demands. Surveys are appropriate for a wide range of users, whereas use case modelling aids in understanding system behaviour. When these strategies are used, they can give a complete understanding of user demands. The decision is influenced by the project's breadth, user diversity, and available resources.