INTELLIGENT CAR DRIVING SYSTEM

By

NC Usama Zafar

NC Amtul Mussawir Hina

PC Hassan Zia

Submitted to Faculty of Computer Software Engineering Department,

National University of Sciences and Technology, Islamabad in partial fulfillment for the requirement of a B.E Degree in Computer Software Engineering

JUNE 2015

ABSTRACT

Intelligent Car Driving System

In today’s world, one of the leading cause of death is road accidents. Failing to follow the road regulations and traffic rules, are one of the main reasons of these accidents, however a more important factor that mainly leads to these road accidents is the human error. Failing to recognize or see a traffic sign, or simply slow reaction times of the human body results in catastrophic accidents, which lead to loss of lives and infrastructure the both. Therefore a need exists to automate the recognition of traffic and the reaction to it.

Our aim is to provide the middle class user a cheaper and an economically viable solution to this problem that will minimize the human error that occurs while driving and automate the car driving in order to avoid road collisions by making intelligent decision based on the changes in the environment the system perceives. Due to the time and budgetary constraints, we have limited the scope of this project and it has been implemented on a model car.

Our project, Intelligent Car Driving System, consists of three major modules: Traffic Light Detection, Traffic Sign Detection and Object Detection. The Traffic Sign Detection Module will identify the three traffic signs (three, in order to limit the scope within the time and budget). The Traffic Light Detection Module will detect any Traffic Signals present and the Object Detection Module will detect if any object is present on the road or not, and whether it is stationary or moving. After successful identification, the system will maneuver the model car, by taking appropriate action of tuning, allowing down, or stopping.

CERTIFICATE OF CORRECTNESS AND APPROVAL

Certified that work contained in this thesis **"Intelligent Car Driving System"** carried out by Usama Zafar, Amtul Mussawir Hina and Hassan Zia under the supervision of Asst. Prof Athar Mohsin Zaidi for partial fulfillment of Degree of Bachelor of Software Engineering is correct and approved.

Approved by

____

(Supervisor Name)

_____ Department

MCS

Dated: ________________

DECLARATION

We declare that the work presented herewith is the result of sole effort of our group, comprising of Usama Zafar, Amtul Mussawir Hina and Hassan Zia; and is free of any kind of plagiarism in part or whole. We also declare that the dissertation has never been submitted previously in part or whole in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our respected teachers whose kind guidance and unfailing support made this mammoth of a task easy for us and to our very dear parents whose unceasing prayers gave us strength and courage to complete the work of this magnitude.

ACKNOWLEDGEMENT

We are very humbly, grateful to The Almighty Allah for bestowing us with the strength and resolve to undertake and complete the project**.**

We owe a special debt of gratitude to our supervisor, Asst. Prof Athar Mohsin Zaidi for putting us on right track and guiding us in understanding the technicalities involved in the field of automated and intelligent car driving systems using latest techniques and for the continuous supervision, motivation and support provided to us right through the project. Without his supervision we would not have been able to complete this successfully. We would like to thank all our other teachers for guiding us in solving our problems related to our project.

We are also thankful to our class mates and our family members for their support throughout the project in every possible way.

Table of Contents

List of Figures

List of Tables

1. Introduction

The purpose of this document is to present a detailed description of the requirements, functionalities and testing of our project, Intelligent Car Driving System. This document will cover each of the system's intended features, what the system will do, the constraints under which it must operate, how the system will behave when operated by user, as well as offer a preliminary glimpse of the software application's User Interface (UI). The document will also cover hardware, software, and various other technical dependencies along implementation and testing details.

1.1 Background

According to a study conducted by the World Health Organization, deaths caused due to the traffic injures are responsible for over 1.25 million deaths worldwide in the year 2013. More than 70% of these accidents are caused due to the human error of failing to perceive and identify the change in the environment of the road.

In America alone. There are 1 billion cars on the roads and their drivers are all vulnerable to some sort of a road accident, either due to their own negligence or human error or due to some other driver's. To avoid this problem, there are solutions available in the market (provided by BMW and Mercedes Benz), that does cater to this problem, by providing a solution to identify the changes in the environment and then either warn the driver or take automated decision to maneuver the car. However, these solutions are expensive and out of reach of a common middle class person and only the 'High End' users can avail these features.

1.2 Aims and Objectives

Our Aim is to provide the middle class user a cheaper and an economically viable solution to this problem that will minimize the human error that occurs while driving and automate the car driving in order to avoid road collisions by making intelligent decision based on the changes in the environment the system perceives. Due to the time and budgetary constraints, we have limited the scope of this project and it has been implemented on a model car.

1.3 Deliverables

Following are the application and document level deliverables prepared during project.

i. Project Synopsis.

ii. Software Requirement Specification.

iii. Architecture and Design Document.

iv. Project documentation and Code.

v. Testing Document.

vii. Embedded system that includes a model car, and all the related hardware components to ensure its working.

2. Literature Review

2.1 Introduction

This chapter will cover the resource material and the literature that was collected for research purposes prior to starting this project. Understanding the hardware components that are to be used is extremely crucial prior to working on it. Some of the information that was collected and gathered is discussed in the subsections 2.2 and 2.3.

2.2 Hardware Equipment

2.2.1 Beaglebone Black (Rev C)

The beagle Bone Black is a small sized and an economical platform that is being used for development and is currently extremely popular within the developers community. The functionality Beagle Bone Black provides is an on board micro HDMI port. 512 MB of DDR3L DRAM, 4GB of onboard flash memory, an AM3358 processor at 1GHz, and making JTAG optional with a user supplied header. In short, Beagle Bone Black is the extremely suitable for physical computing and smaller embedded systems. [1]

Due to the provision of real time analysis by the TI Sitara™ AM3358 ARM® Cortex™-A8 processor, Beagle Bone provides with a great power for processing and I/O. Beagle Bone can also be complemented with a cape plug in boards which further augments the functionality of the Beaglebone Black.

At over 3 million Dhrystone operations per second and vector floating point arithmetic operations, Beaglebone Black is capable of not just interfacing to all of the robotics motor drivers, location or pressure sensors and 2D or 3D cameras, but also running OpenCV, OpenNI and other image collection and analysis software to recognize the objects around the robot and the gestures the user might make to control it. [1]

For our system, the Beagle Bone black will work as a brain, the central processing unit for all the rest of the hardware components. It will also act as a mediator between the sensors and the actuators.

2.2.2 Ultrasonic Range Sensor (HC-SR04)

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
Using IO trigger for at least 10us high level signal, The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back. IF the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning. Test distance = (high level time x velocity of sound (340M/S) / 2,

The Timing diagram is shown below. The user only needs to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .The user can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; it is suggested to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal. [2]

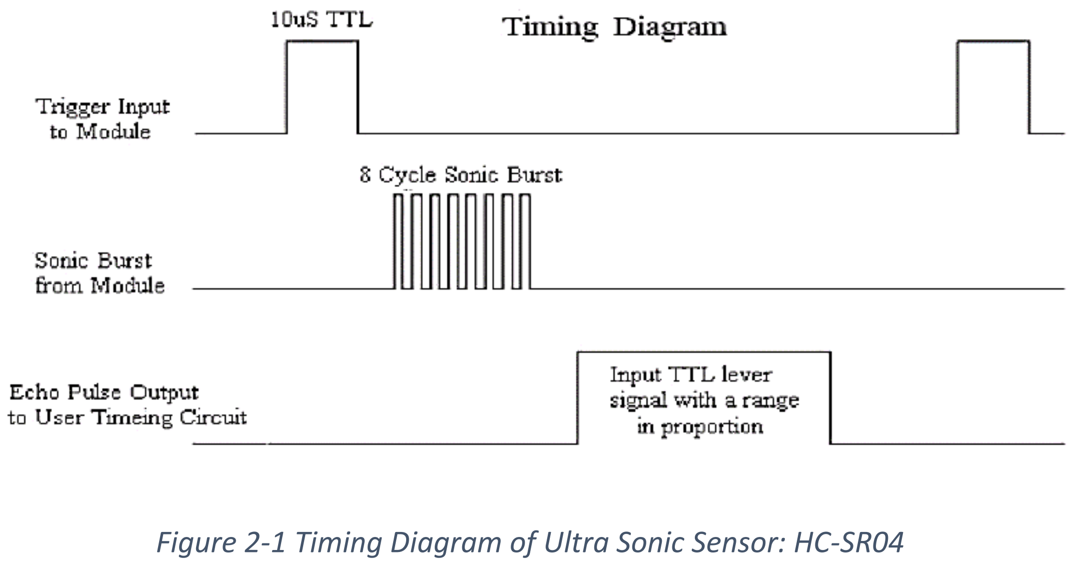

10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timeing Circuit
Input TTL lever signal with a range in proportion

Figure 2-1 Timing Diagram of Ultra Sonic Sensor: HC-SR04

2.2.3 Camera A4Tech (PK835G)

The webcam used is a simple USB plug and play camera to get a video feed from which images will be extracted over a certain frame rate and then they will be analyzed for the evaluation of the detection of changes that may occur in the environment.

2.2.4 WIFI TP-LINK (TL-WN725N)

TP-LINK's 150Mbps wireless N Nano USB adapter, TL-WN725N allows users to connect a desktop or notebook computer to a wireless network at 150Mbps. This miniature adapter is designed to be as convenient as possible and once connected to a computer’s USB port, can be left there, whether traveling or at home. It also features advanced wireless encryption and easy installation. [3]

With its miniature size and sleek design, users can connect the nano adapter to any USB port and leave it there. There’s no need to worry about blocking adjacent USB interfaces or that the adapter may fall out when moving a connected laptop from A to B, with the tiny device flush against the USB port.

When speaking about wireless security, WEP encryptions are no longer the strongest and safest protections against intrusions. The TL-WN725N provides WPA/WPA2 encryptions created by the WI-FI Alliance™, promoting interoperability and security for WLAN, which effectively and efficiently protects users’ wireless networks.
The TL-WN725N comes with a 14-language utility located on its bundled CD that helps users complete the software installation and wireless network settings, including security configurations and wireless connections, quickly and easily, even for novice users new to wireless networking.

For this project, this Wi-Fi module will be used to establish a secure connection between the processing unit and the model car for effective communication for transferring of images and video feed, so that they can be analyzed by the processing unit and the relevant action is determined and commands are sent back to the microcontroller to perform those actions.

Figure 2-2 WIFI TP-LINK (TL-WN725N)

2.2.5 Beaglebone LCD Cape

The 4.3” resistive touch LCD touch from 4D Systems is a cape specifically designed for the Beaglebone Black, and provides a 4.3” primary display for the BBB for direct user interaction and information display. The Beaglebone Black connects directly to the back of the LCD cape, and provides everything the cape requires such as power and display signals. [1]

The Beaglebone Black LCD cape features 7 push buttons below the screen, LEFT, RIGHT, UP, DOWN, ENTER, RESET and POWER, along with 2 LED’s to indicate Power and User Status. Mounting the cape is easy with the 4x 3.5mm mounting holes present on it, enabling standard M3 or #6-32 screws to fasten the the LCD cape as required.

It will be mainly used in this project for debugging and testing to see the output of the processing being carried out on the model car by the beagle bone at run time and to indicate any problems or faults.

2.2.6 Light Dependent Resistor

A light-dependent resistor (LDR) or photocell is a light-controlled variable resistor. The resistance of a photo resistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity. A photo resistor can be applied in light-sensitive detector circuits, and light- and dark-activated switching circuits.
For this system, LDRs in combination with an LED array will be used to follow a dark line over a light surface and or vice versa.

2.2.7 Operational Amplifier (LM324)

The LM 124-N series consists of four independent, high gain, internally frequency compensated, operational amplifiers designed to operate from a single power supply over a wide range of voltages. Operation from split power supplies, is also possible and the low power supply current drain independent to the magnitude of the power supply voltage.
Application areas include transducer amplifiers, DC gain blocks, and all the conventional ap amp circuits which now can be more easily implemented in single power supply systems and easily provides the required interface electronics without additional ± 15V power supplies.
For this project, operational amplifier has been used to enhance the output of the current from the Light Dependent Resistor so as to make the output more readable by the microprocessor in the Beagle Bone Black. [5]

Figure 2-3 shows the schematic diagram of the Operation Amplifier

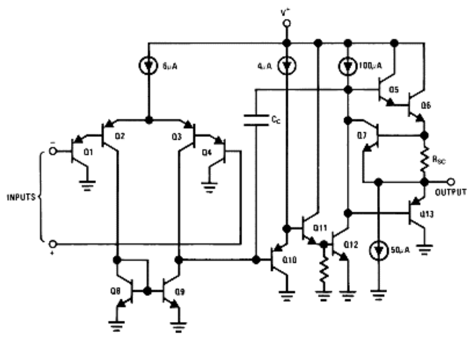


Figure 2-3 Schematic Diagram of Operational Amplifier (LM324)

2.2.8 Potentiometer

A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment. Potentiometers operated by a mechanism can be used as position transducers, for example, in a joystick. Potentiometers are rarely used to directly control significant power (more than a watt), since the power dissipated in the potentiometer would be comparable to the power in the controlled load.
For this system, potentiometer has been used to provide a threshold value to the LEDs that will be used for the movement of the car so as to prevent them from burning due to excessive voltage, or heat buildup. Moreover, it will also provide resistance so as to control the luminance of the LEDs that will be used.

3. System Requirement Specification:

3.1 Introduction

This chapter defines extensively, the system requirements of this project. The chapter in its initial sections will identify the scope and perspective of the project, followed by the elicitation of functional and non-functional requirements.

3.2 Product Scope

The “Intelligent Car Driving System” is a combination of hardware and software into a packaged system which shall be able to drive a car without any human intervention, keeping in view the parameters such as traffic signs, traffic lights, surrounding objects and roads. The Intention behind developing such a system is to create an automated version of human driver without the faults that are the part of human nature.

The driver shall log into the ICDS and the system keeping the surrounding information in view will drive a car automatically. Limiting humane role will not only be a big step towards automation but also will help save thousands of lives that are ordinarily lost due to traffic accidents caused by human errors. Being the first version of the system its intent is purely towards research based goals that is to explore the possibility of automation in car driving

3.3 Product Perspective

This project is focused to create a system that will minimize human interaction while driving. The system consists of 3 Major modules i.e. Central Control; Sensors Control; Car Controls as shown in Figure 3-1.

The Sensor Control would constantly monitor the environment and the changes within the environment through cameras and range finder sensors, and would notify the Central Control of any changes that are taking place within the environment.
The Central Control System would take input of any changes in the environment from the Sensor Control and would then process this information to determine the

appropriate action that needs to be taken. It would then pass on the required action that needs to be taken to the Car Controls.

The Car Control will take the action that needs to be taken as an input from the Central Controls and would perform the actions of steering (left turn, right turn, left lane, and right lane.), accelerating and break. It would also send a feedback to the Central Control System after performing this action as a verification process.

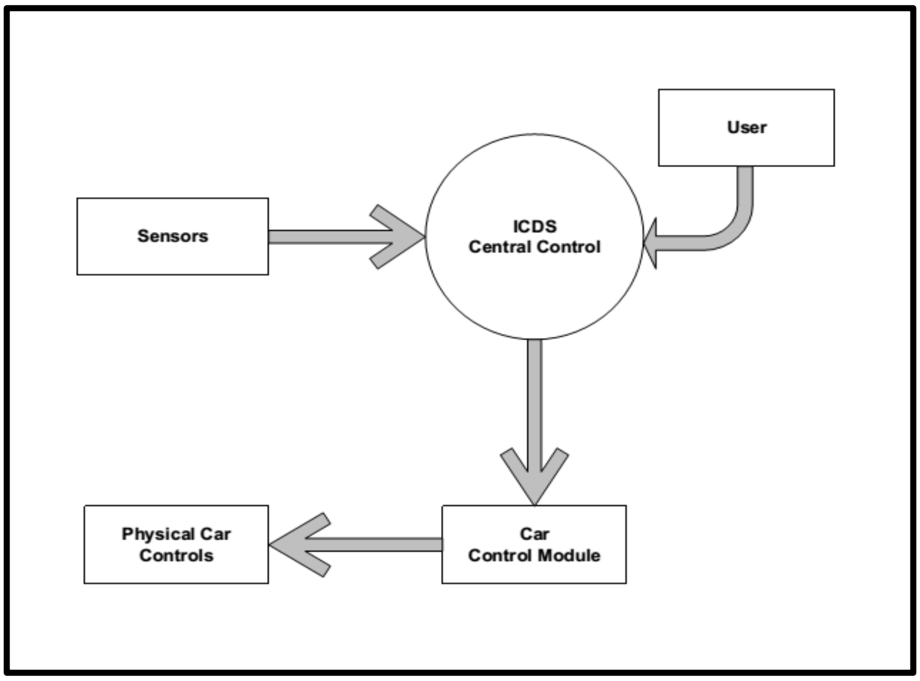


Figure 3-1 Modules of ICDS

3.4 Product Objectives

This section will discuss the objectives that need to be taken into consideration before the start of the development of the project. These objectives are listed as 3.4.1 – 3.4.4.

3.4.1 The project must be completed in one academic year for review by the panel/committee.

3.4.2 The project should not exceed a total amount of Rs. 150'000/-.

3.4.3 Efficient in terms of minimum response time, performance, availability.

3.4.4 Convenient to use and easy to learn in terms of learnability, efficiency and user satisfaction.

3.5 Product Functions

This section will briefly list down the functions that the product shall be able to perform. They are listed as 3.5.1 to 3.5.4.

3.5.1 The system shall be able to perform Steer, accelerate and brake actions on the car

3.5.2 The system shall be able to identify 3 traffic signs due to limited scope of project concept and limited time

3.5.3 The System shall be able to detect traffic lights and perform the necessary action on a traffic signal accordingly.

3.5.4 The system shall be able to perform a surrounding analysis by detecting an object in front of the vehicle and avoid collision with it.

3.6 User Classes and Characteristics

The User for this project will be the driver, the person sitting behind the steering wheel.

3.6.1 Driver (Sole User)

Since this system is a critical system, hence the specification and characteristics of its user are of great importance. Section 3.6.1.1 till 3.6.1.4 will be defining these characteristics that should have in order to operate this system.

3.6.1.1 The user should know how to drive

3.6.1.2 The user should be literate in English language

3.6.1.3 The user should know how to operate a keyboard based text entry system and should know how to use a pointer tool on a screen

3.6.1.4 The user shall be able to enter the user name and password to authenticate itself in order to use the system

3.7 Design and Implementation Constraints

This section will highlight the design and implementation constraints that will be taken into consideration before starting off with this project. Section 3.7.1 till section 3.7.5 will enlist these constraints.

3.7.1 Due to budgetary and cost limitations, the model car that will be bought for this project will not be able to perform functions of gears and the system will therefore not cater to gear changing functions.

3.7.2 Due to the limited standalone processing power units available, laptop will be used to process the inputs and outputs of the system.

3.7.3 Wireless communication would be held between the laptop and the microcontroller and sensors placed on the model car.

3.7.4 The project is dependent on the model car that will be used for this system, therefore the features and functions of steering, breaks and acceleration is dependent on the functions and features available in the model car, for example the degree by which the system can turn the car, depends on the degree of turn the model car can take.

3.7.5 The timings are dependent on the processing power available and the time it takes to transmit the inputs to the processor, the time it takes to transmit the action to the microcontroller from the processor and the time it takes for the action to be performed, therefore a lag in any of these timings may result in an overall delay which will be considered as an unresponsive system, and the control of the vehicle will be returned to the driver.

3.8 External Interface Requirements

This section contains the specification of requirements for interfaces among different components of the software and their external capabilities. It presents a high level of abstraction of the basic elements of the Graphical User Interface (GUI) and the display screens of the system. The design and implementation teams may define specific command buttons, seek bars, radio buttons, text fields, and pull-down menus to meet the requirements.

3.8.1 User Interfaces

User interfaces are the graphical interfaces a user will be using to operate the system. The requirements of these interfaces are enlisted from 3.8.1.1 till 3.8.1.3.

3.8.1.1 The system shall have a graphical user interface so that the user can be authenticated with a user name and password.

3.8.1.2 The user shall enter the user name and password using a keyboard and the system will validate the user.

3.8.1.3 The user shall be able to select and connect to an adhoc connection that from the available Wi-Fi connections that are open so that a secure connection can be established between the model car and the processing unit for data transfer.

3.9 System Features

The system features are described as below. Each feature description includes a subsection of Description and priority, Stimulus/Response Sequences, and functional requirements.

3.9.1 User Authentication

This feature will authenticate whether an authorized user is accessing the system or not.

3.9.1.1 Description and Priority

This feature shall facilitate the registered user to request the login by providing the required information (it is assumed that the user is already registered with the system).
The priority of this feature is high.

3.9.1.2 Stimulus/Response Sequences

The feature subtasks will follow the following sequence:

Normal Course:
(User has given the required information to login and using the system.)

3.9.1.2.1 User selects ‘Sign In’ option.

3.9.1.2.2 System asks for required information including user name and password.

3.9.1.2.3 After user provides the information the system verifies the information from database.

3.9.1.2.4 Now user can further access the system.

Alternate Course:

3.9.1.2.5 4.1.2.3.a User entered invalid user name or invalid password.

3.9.1.2.6 4.1.2.3.a.1 System shows an error message and asks the user to re-enter the login and password.

3.9.1.3 Functional Requirements

3.9.1.3.1 The system shall display the option for login and password.

3.9.1.3.2 The system shall acquire the information from the user.

3.9.1.3.3 The system shall be able to match the user information from the database.

3.9.1.3.4 The system shall be able to identify valid user.

3.9.1.3.5 The system shall be able to check if the input data is correct or not.

3.9.1.3.6 The system shall prompt a message for wrong inputs and errors.

3.9.1.3.7 The system shall ask the user to enter password and login id again.

3.9.2 Car Controls

This feature will define the actions that the model car will take when it recognizes a certain sign, light or object.

3.9.2.1 Description and Priority

The feature will control the operations of the car such as steer, accelerate/de-accelerate and brake.

The feature is of High Priority as it will enable a smooth operation of the car drive.

3.9.2.2 Stimulus/Response Sequences

There is no response sequence for this feature as it has no user interaction.

3.9.2.3 Functional Requirements

3.9.2.3.1 The system shall be able to accelerate or decelerate the car as per the commands generated by the Central Control of the system.

3.9.2.3.2 The system shall be able to stop the car using brakes if:

3.9.2.3.2.1 The car has successfully arrived at the destination. Or

3.9.2.3.2.2 The command was issued by the Central Control due any no. of reasons as described in later features.

3.9.3 Traffic Light Detection

3.9.3.1 Description and Priority

This Feature will enable the system to detect any traffic light and take appropriate action based on the information gathered.

3.9.3.2 Stimulus/Response Sequences

There is no response sequence for this feature as it has no user interaction.

3.9.3.3 Functional Requirements

3.9.3.3.1 The system shall be able to detect the traffic lights along it's path

3.9.3.3.2 Once detecting the presence of a traffic light, the system shall identify the light as Red, Yellow and Green.

3.9.3.3.3 If the light identified is red, the system must stop the car slowly and avoid collision

3.9.3.3.4 If the light is identified as yellow, the system shall:

3.9.3.3.4.1 Prepare to move the car if its stationary

3.9.3.3.4.2 Slow down if already moving

3.9.3.3.5 If the light is identified to be green the system shall resume driving the car

3.9.4 Traffic Sign Detection

3.9.4.1 Description and Priority

This Feature will enable the system to detect any traffic sign and take appropriate action based on the information gathered.

3.9.4.2 Stimulus/Response Sequences

Since this feature is solely functions due the response of the system therefore there is no response sequence for this feature as it has no user interaction.

3.9.4.3 Functional Requirements

3.9.4.3.1 The system shall be able to detect and identify three traffic signs and upon successful detection and identification the system shall:

3.9.4.3.2 The system shall take the appropriate action based on the type of the traffic signs.

3.9.5 Surrounding Analysis

3.9.5.1 Description and Priority

This feature is designed to enable the smooth operation of the car. The feature will enable the system to detect object in front of the car and take appropriate actions depending upon the nature of the object.

3.9.5.2 Stimulus/Response Sequences

There is no response sequence for this feature as it has no user interaction.

3.9.5.3 Functional Requirements

3.9.5.3.1 The system shall be able to detect objects in front of the car.

3.9.5.3.2 Upon successful detection, the system shall determine if the object is stationary or moving.

3.9.5.3.3 Based on the information acquired, the system shall

3.9.5.3.3.1 Adjust the speed of the car if the object is moving. Or

3.9.5.3.3.2 Adjust the lane of the car if the object is stationary.

3.10 Non-Functional Requirements

Non-functional requirements specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They define how a system is supposed to be. Section 3.10.1 till 3.10.8 enlist these nonfunctional requirements.

3.10.1 Performance Requirement:

This section will enlist the performance requirements of the system. The performance requirements will ensure that the system performs to certain threshold level. The sections 3.10.1.1 till sections 3.10.1.6 enlist these requirements.

3.10.1.1 This section is intentionally kept at the level of performance criteria rather than response times.

3.10.1.2 Commands shall be acknowledged in a positive or negative way before the occurrence of the corresponding action within given response times.

3.10.1.3 In standard workload, the CPU usage shall be less than 60%, leaving the rest for background jobs.

3.10.1.4 It would take minimum possible time to get the output at the car controls once an event is detected.

3.10.1.5 The system shall be able to process and perform the car controls within 2-3 seconds after the relevant action has been received by the module.

3.10.1.6 The system shall be able to detect the traffic light and signs in real time, calculate the appropriate action to be performed and send the action to be performed to the car controls module within the minimum possible time.

3.10.2 Availability

This section will enlist the availability requirements of the system. The availability requirements will ensure that the system is available to the user at all times. The sections 3.10.2.1 till sections 3.10.1.4 enlist these requirements.

3.10.2.1 The application will be available 24/7 provided the user has installed this application and has all the required sensors. Since the system is an offline one so availability can only be hampered by hardware failure (sensors).

3.10.2.2 Automatic application shutdown will only occur in case of the hardware failure.

3.10.2.3 Fault recovery, exception handling, fail-safe checks, etc. should be used to ensure availability.

3.10.2.4 Less than 1 minute will be needed to restart the system after a failure, 99.90% of the time.

3.10.3 Maintainability

This section will enlist the maintainability requirements of the system. The maintainability requirements will ensure that the system maintenance criteria is defined for any future updates. The sections 3.10.3.1 till sections 3.10.1.2 enlist these requirements.

3.10.3.1 Not more than 3% of the existing functionality shall be affected by adding the new functionality. Since the system is a modularized on so any changes to a single module will be contained within the module itself and shall not affect the whole system.

3.10.3.2 Installation of a new version shall leave all personal settings unchanged.

3.10.4 Portability

This section will enlist the portability requirements of the system. The portability requirements will ensure that the system portability criteria is defined. The section 3.10.4.1 enlist these requirements.

3.10.4.1 The system shall mainly be developed in C or C# so it will be deployable on all windows based systems and devices.

3.10.5 Reliability

Reliability is defined as “The duration or probability of failure-free performance under state conditions” or “The probability that an item can perform its intended function for a specified interval under stated conditions”. Sections 3.10.5.1 till 3.10.5.2 enlists these requirements.

3.10.5.1 The system defect rate shall be less than 1 failure per 1000 hours of operation.

3.10.5.2 The values returned by the application through different sensors would be reliable as it would have passed through different testing processes even in the presence of test drivers.

3.10.6 Usability

This section will enlist the usability requirements of the system. The usability requirements will ensure that the usability of the system criteria is defined for any future updates. The sections 3.10.6.1 till sections 3.10.1.2 enlist these requirements.

3.10.6.1 It would take less than 3 minutes for a new user to completely understand the application.

3.10.6.2 For most of the links or buttons, help would be provided to the user for knowing the main functionality of that particular link or button for his better understanding.

3.10.7 Efficiency

The 3.10.7.1 section will highlight the efficiency of the system.

3.10.7.1 The system shall be able to handle all the data inputs from the three specified modules and use it effectively for the required results.

3.10.8 User Satisfaction

User satisfaction of the system ensures that the user is comfortable to use the system. The section 3.10.8.1 enlists the user satisfaction requirement.

3.10.8.1 At least 70% of candidates shall rate their satisfaction with the system after using it at 7 or more on a scale of 1 to 10 else the system shall be improved until the aforementioned user satisfaction is achieved .

3.11 Conclusion

Through extensive requirement definitions, the main functionality of the system has been identified, so that the development of this system is done on these lines. This is a crucial step before the actual development starts so that the development remains on track and conforms to a limited scope that has been identified in the previous sections.

4. Design and Development

4.1 Introduction

This Chapter will cover the system architecture of our project. First we will discuss about the modular level breakdown of the system. In the second chapter we will discuss the system architecture and the architecture pattern that we have used in our system. In the next section, we will be looking at the design pattern that has been used to design the low level view of the system followed by the UML diagrams that define other features of the system.

4.2 Overview of Modules/Components

The ICDS consists of 4 main modules divided in two subsystems. The first subsystem consists of the user interaction of the system where the user is authenticated and the user initializes the system by connecting the front end application with the model car. The user then initializes the ICDS Central Control, which is the second subsystem of our overall system.

This subsystem consists of a single module and is further divided into two components: Login and Connect to Beagle
The Login module authenticates and verifies a user and Connect to Beagle will initialize the second subsystem by establishing a connection between the front end application and the model car

The second subsystem consists of the ICDS Central Control, where the system manages the driving of the car by monitoring the changes in the environment and taking appropriate actions based on the decisions designed in the system.The second subsystem consists of 3 main components: Traffic Sign Detection, Traffic Light Detection and Object Detection.

The Traffic Sign Detection successfully detects and identifies 3 Traffic Signs and maneuvers the car accordingly.

The Traffic Light Detection successfully identifies the traffic lights and performs the appropriate action based on the traffic light that has been detected.

The object detection module successfully detects an object lying on the path, determines whether it is moving or not and then takes appropriate measures to avoid collision.

4.3 System Architecture

The main structure of the system is based on the Environmental Architectural Pattern which is an extension of the Model View Controller Architectural Pattern. The Environmental Pattern is used on those systems in which the software controls the operation of the equipment based on the stimuli of the system's environment. As is shown in the figure 4-2:

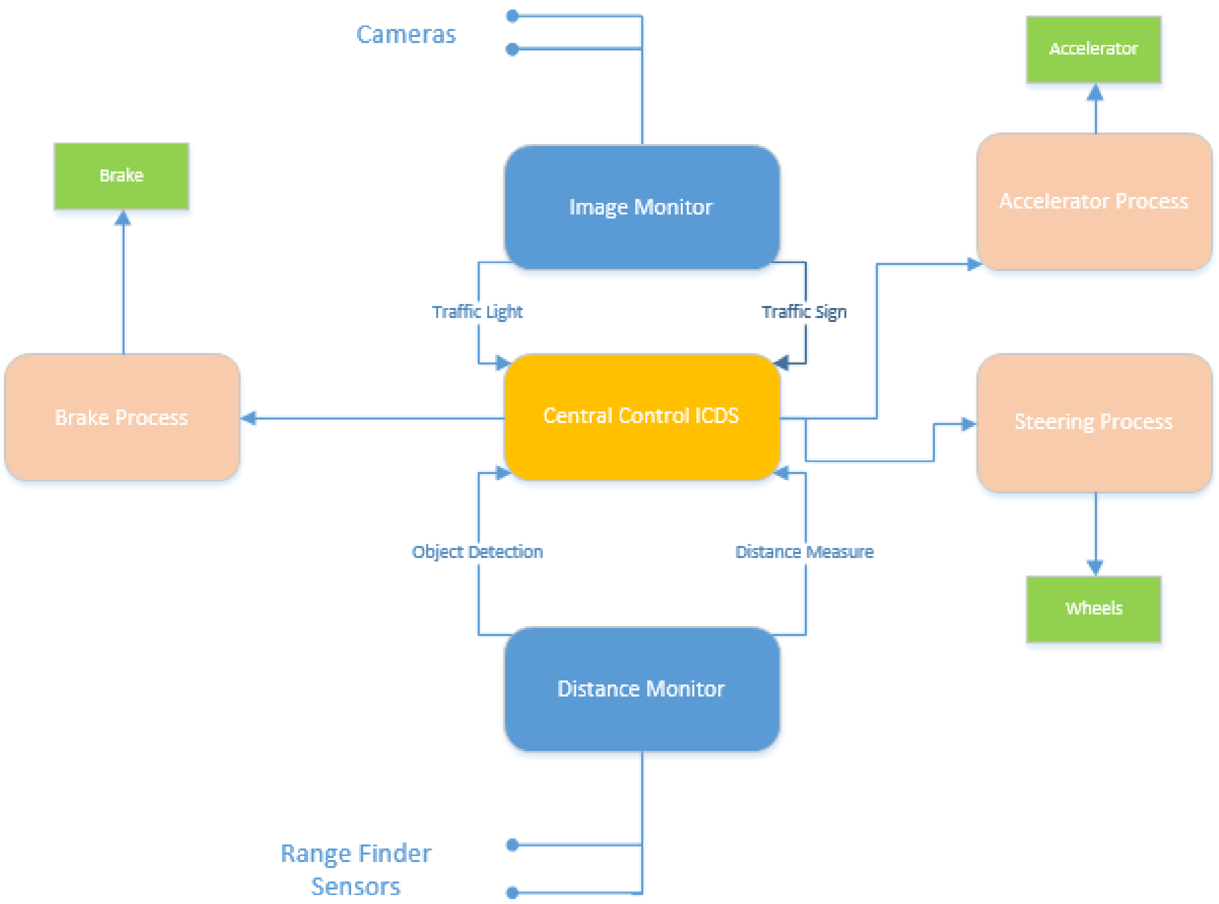


Figure 4-1 Architecture Diagram for Environmental Architectural Pattern

Here the concept of sensors, process monitors and actuators are used. The system analyzes information from a set of sensors that collect data from the system's environment. Further information may also be collected on the state of the actuators that are connected to the system. Based on the data from the sensors and actuators, control signals are sent to the actuators that then cause changes to the system's environment.

In our system, the diagram in figure 4-2 is based clearly on these concepts. The cameras and range finder sensors act as sensors, whereas the accelerator, brake and wheels act as the actuator. The monitor processes (shown in blue) then process the input from the sensors and classify it accordingly. The central control process (shown in yellow), then uses this data to determine the action that needs to be performed. The Control Processes (shown in pink) are then activated to maneuver the vehicle accordingly.

This architecture can also be considered as an extended form of the MVC Model where the Sensors act as the Controller as they send in the data to the Model (The control Processes) to update the Model's state. The model then determines the appropriate action that needs to be taken and notifies the View (The Actuators) so that the necessary action is taken by them. This relationship is depicted in the figure 4-3 below.

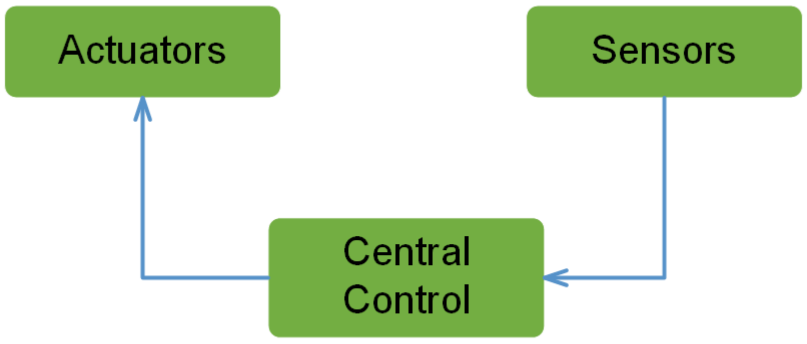


Figure 4-2 MVC Model

4.4 Detailed System Design

The design pattern used for our project is the Observer Pattern. It is due to the fact that our system is an event driven system in which most of the work being done is based on the Observing and Reporting state change and the observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. It is mainly used to implement distributed event handling systems. The Observer pattern is also a key part in the familiar model–view–controller (MVC) architectural pattern. The observer pattern is implemented in numerous programming libraries and systems, including almost all GUI toolkits.
The participant classes in this pattern are:

Observable - interface or abstract class defining the operations for attaching and de-attaching observers to the client. In this context there are two such classes i.e. Observable_RangeFindor, Observable_Camera.

ConcreteObservable - concrete Observable class. It maintain the state of the object and when a change in the state occurs it notifies the attached **Observers**. In this case Camera & Range Finder are the two classes of this category.

Observer - interface or abstract class defining the operations to be used to notify this object. The Central Control Class falls under this category.

The flow is simple: the main framework instantiate the ConcreteObservable object. Then it instantiate and attaches the concrete observers to it using the methods defined in the Observable interface. Each time the state of the subject it's changing it notifies all the attached Observers using the methods defined in the Observer interface. When a new Observer is added to the application, all that is need to be done is to instantiate it in the main framework and to add attach it to the Observable object. The classes already created will remain unchanged.

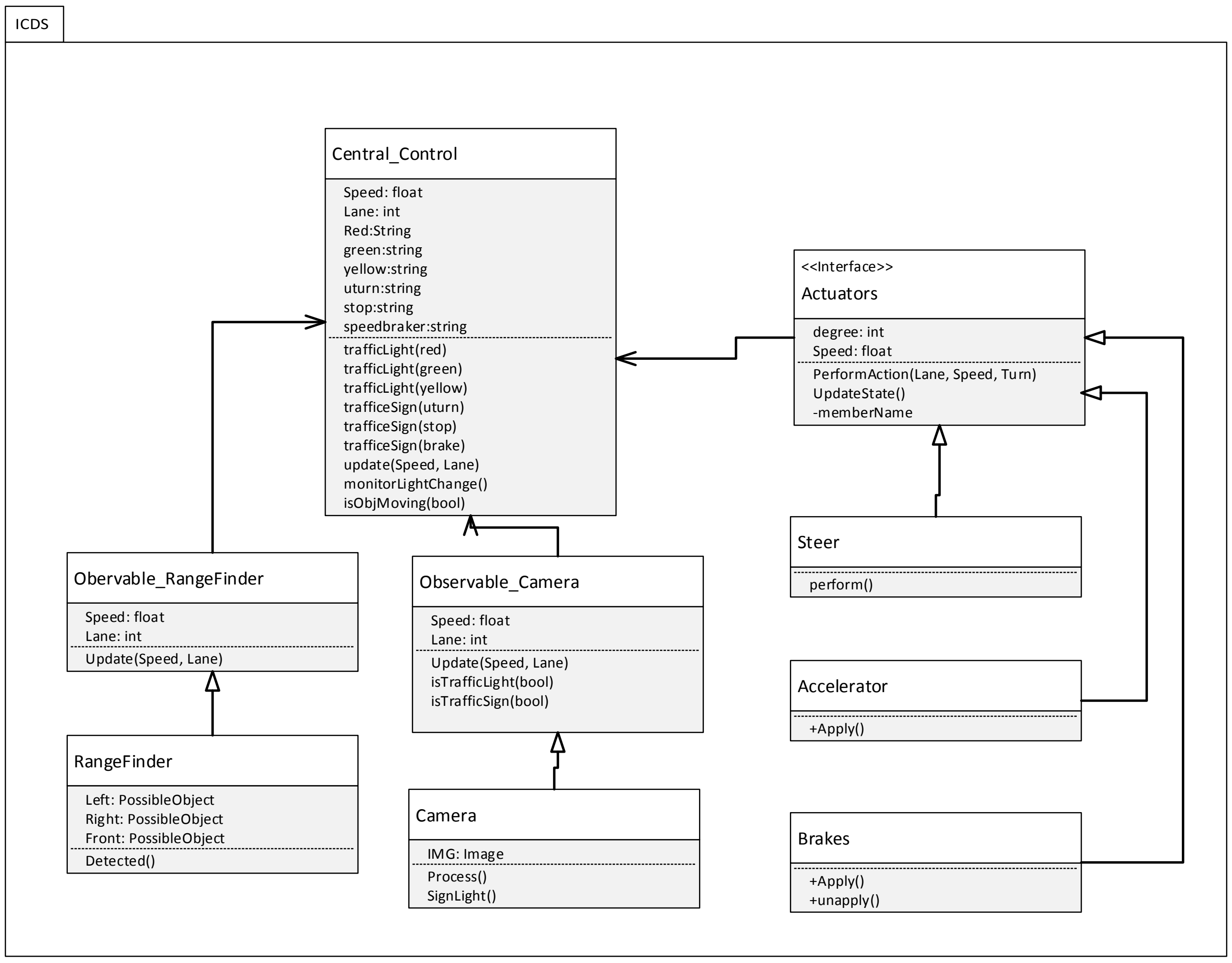


Figure 4-3 Class Diagram

As per the observer pattern any change in the classed such as camera or Range finder is reported to the Observable classes which in turn reports these changes to the Central Control Class. The central control class is responsible for processing these changes and deciding if any particular action needs to be taken based on the results from the processing. Any action needed is taken by notifying the actuator class which oversees all the actuators.

4.5 UML Diagrams

4.5.1 Event Response Table

Use cases and user stories aren't always helpful or sufficient for discovering the functionality that software developers must implement. This is particularly true for embedded and other real-time system. An alternative to eliciting use cases and user stories is to identify the external events to which the system must respond. An event is some change or activity that takes place in the user's environment that stimulates a response from the software system. An event-response table (also called an event table or an event list) itemizes all such events and the behavior the system is expected to exhibit in reaction to each event. We have thereofore created an event response table for our system that will showcase the events, and the response to them by the system.

Table 4-1 System Event Response Table

Event	**System State**	**Response**
IR Sensor detects an object in front of the car	The car is moving at a specific speed	Determine if the object is moving or stationary
	The car is stationary	System does nothing
Detected object is moving	The car is moving faster than detected object	Slow the car down to match the speed of the moving object
	The car is moving slower than detected object	System does nothing
Detected object is stationary	Car is moving in right lane	Change the lane of car to left
	Car is moving in left lane	Change the lane of car to right
Camera detects a RED LIGHT	Car is stopped at the signal	System does nothing
	Car is approaching the signal at a certain speed	Slow down and stop the car at the signal
Camera detects a GREEN LIGHT	Car is waiting for the signal to become green	Start moving the car and continue the journey
	Car is moving and approaching the signal at a specific speed	Continues and cross the signal
Camera detects a TRAFFIC LIGHT	Car is moving	Determine the traffic sign
	Car is stationary	Determine the traffic sign
Sign is STOP	Car is moving	Stop the car

	Car is stationary	Do nothing
Sign is a U TURN	U TURN is suggested in the map	Take the U TURN
	U TURN is not suggested in the map	Do not take the U TURN
Sign is a SPEED LIMIT	Car is moving fast	Slow down the car to the speed mentioned in the sign
	Car is moving slow	Speed up the car up to the speed mentioned in the sign
	Car is stationary	Do nothing
Sign is a SPEED BREAKER	Car is moving fast	Slow down the car and cross the speed breaker
	Car is moving slow	Cross the speed breaker
	Car is stationary	Do nothing

4.5.2 Use Case Diagrams

Following are the use case diagrams for our system.

4.5.2.1 Sub-System 1

The sub system 1 will be the user interaction system where the user will interact and login to the system. Secondly, it will establish a connection with the model car so as to enable communication between the processing unit and the model car.

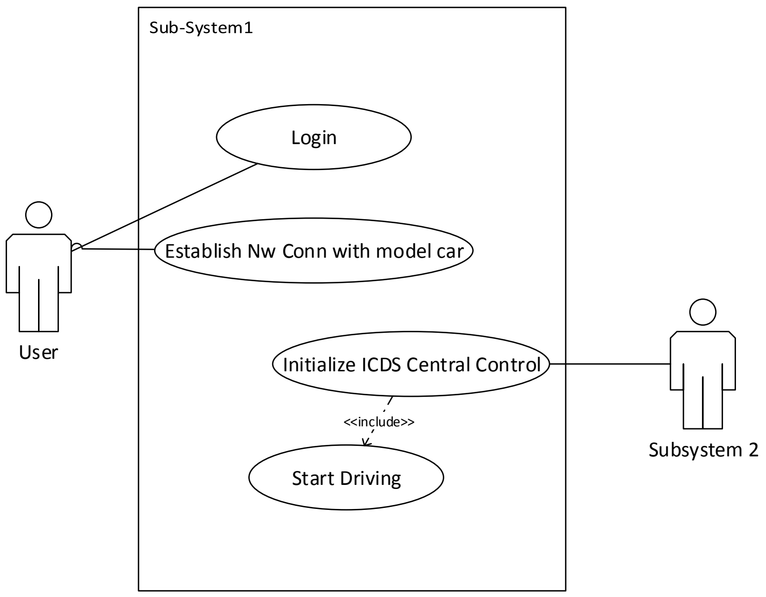

Sub-System1
Login
Establish Nw Conn with model car
Initialize ICDS Central Control
<<include>>
Start Driving
User
Subsystem 2

Figure 4-4 Use Case Diagram Login

4.5.2.2 Sub-System 2:

The Second Subsystem involves the interaction of the sensors, the camera and range finder, with the environment so that the car controls can take appropriate decision.

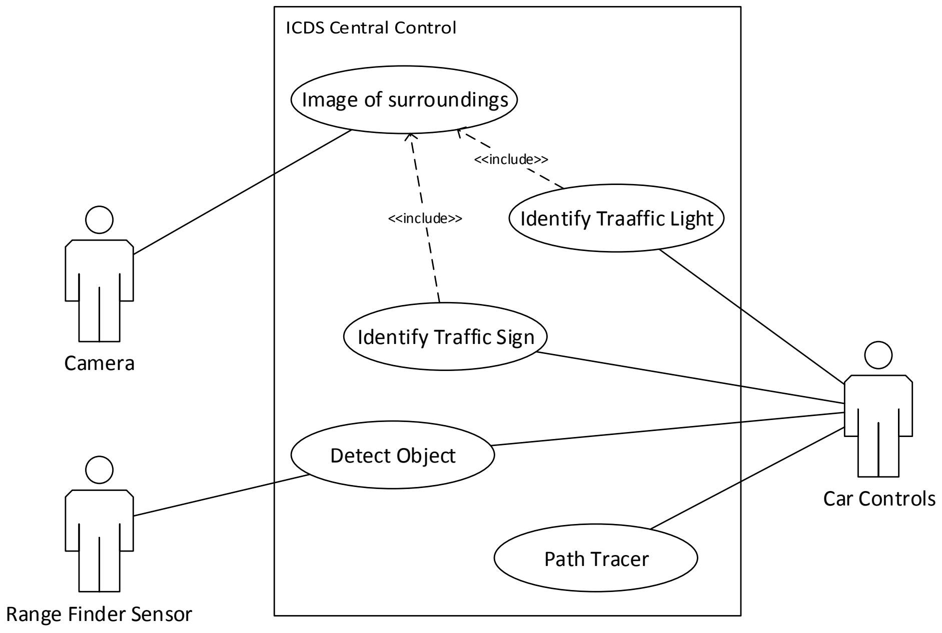


Figure 4-5 Use Case Diagram for Sub-System 2: ICDS Central Control

4.5.3 Fully Dressed Use Cases

Fully dressed use cases were created by Alaister Cockburn in his book "Writing Effective Use Cases". The headings and their descriptions that might be used in a fully dressed use case is as follows:

The Title is used for entering the goal of the use case – preferably as a short, active verb phrase. The description describes the goal and context of this use case. This is usually an expanded version of what you entered in the "Title" field. The Primary Actor is represented as a person or a software/hardware system that interacts with your system to achieve the goal of this use case. Preconditions describes the state the system is in before the first event in this use case. Post conditions describe the state the system is in after all the events in this use case have taken place. Main Success Scenario: this field contains the example from the previous post – i.e. the flow of events from preconditions to post conditions, when nothing goes wrong. Extensions is used to describe all the other scenarios for this use case – including exceptions and error cases. And the remaining fields are self-explanatory. [6]

From Table 4-2 to Table 4-5 illustrate the fully dressed use cases for this system

Table 4-2 Fully Dressed Use Case Login

Use Case ID:	1.0		
Use Case Name:	Login		
Created By:	Amtul Mussawir Hina	**Last Updated By:**	-
Date Created:	15.12.2014	**Last Revision Date:**	-
Actors:	Primary Actor: Driver		
Description:	The user enters his credentials that are authenticated so that he is verified as a legitimate user.		
Trigger:	The user starts the system		
Pre-Conditions:	1. User has a user name and password registered within the system 2. The user has the knowledge		
Post-Conditions:	1. User successfully logs into the system 2. Connect to Adhoc Network Screen is displayed		
Normal Flow:	1. User Enters User Name 2. User Enters password		

	3. User clicks Login 4. System displays "connect to adhoc network" screen upon successful login
Alternative Flows: [Alternative Flow 1 – Not in Network]	3a. In step 3 of the normal flow, if the user does not enter correct user name and password 1. System will display the message of incorrect login and prompt the user to reenter the credentials 2. User reenters the user name and password 3. If the username and password is correct, use case resumes on step 5, else it will loop back to 3a
Exceptions:	[Describe any anticipated **error conditions** that could occur during execution of the use case, and define how the system is to respond to those conditions. e.g. Exceptions to the Withdraw Case transaction 2a. In step 2 of the normal flow, if the customer enters and invalid PIN 1. Transaction is disapproved 2. Message to customer to re-enter PIN 3. Customer enters correct PIN 4. Use Case resumes on step 3 of normal flow]
Includes:	
Frequency of Use:	Every time the user starts the system
Special Requirements:	
Assumptions:	The user knows how to enter text in a text box and is aware of his username and password
Notes and Issues:	1. What is the maximum and minimum size of the password that a user can have?

Table 4-3 Fully Dressed Use Case Traffic Sign Detection

Use Case ID:	3.0		
Use Case Name:	Traffic Sign Detection		
Created By:	Amtul Mussawir Hina	**Last Updated By:**	
Date Created:	16.12.2014	**Last Revision Date:**	

Actors:	Camera
Description:	The system successfully detects the traffic sign and appropriate action is taken according to the type of sign
Trigger:	
Pre-Conditions:	**The system is in the driving state mode**
Post-Conditions:	The system successfully identifies the traffic sign
Normal Flow:	1. The camera detects a traffic sign 2. The System identifies the traffic sign as a U-Turn 3. The car slows down and makes a U-turn
Alternative Flows:	2a. The System identifies the traffic sign as Stop 1. The car slows down and comes to a halt before the stop line 1a If there is another car ahead, avoid collision while coming to a halt 2. The system resumes driving if the road ahead is clear of any obstructing object 2b. The System identifies the traffic sign as a speed breaker. 1. The car slows down and crosses the speed breaker 2. The system resumes acceleration after crossing the speed breaker
Exceptions:	-
Includes:	-
Frequency of Use:	Every time the camera successfully detects the traffic sign
Special Requirements:	
Assumptions:	The system camera successfully identifies every traffic signal on the road

Table 4-4 Fully Dressed Use Case Traffic Light Detection

Use Case ID:	4.0		
Use Case Name:	Traffic Light Detection		
Created By:	Hassan Zia	**Last Updated By:**	Amtul Mussawir Hina
Date Created:	16.12.2014	**Last Revision Date:**	17.12.2014
Actors:	Primary Actor: Camera		

Description:	The camera will detect traffic signals. After detecting the signal it will analyse the color of the light and it will send this information to control unit.
Trigger:	User clicks start driving button.
Preconditions:	1. User must be logged into the system. 2. User must have the essential training for using the system. 3. Connection is established between the processing unit and the car.
Post conditions:	1. The system send the results to the control unit.
Normal Flow:	1. Camera searches for traffic signal. 2. Traffic signal is detected. 3. Color of the light is detected as red. 4. Results are sent to the control unit. 5. Control unit monitors the distance between the nearest object through range finder 6. The car stops at an appropriate distance from the nearest object 7. The system waits for a change in the traffic light to yellow.
Alternative Flows: **[Alternative Flow 1 – Not in Network]**	3a. The color of the light is detected as yellow 1. Results are sent to the control unit 2. Control unit monitors the distance between the nearest object through range finder 3.If the car was in motion, the car slows down maintaining distance from the nearest object 4. If the object is stationary, the car slowly accelerates, iff the car ahead of it moves, maintaining distance from it 3b The color of the light is green 1. The car begins accelerating (iff the car ahead of it moves) maintaining appropriate distance from the object ahead
Exceptions:	-
Includes:	-
Frequency of Use:	After the start driving button is clicked. This module will be used after every 5 sec.

Special Requirements:	The camera shall not miss a traffic signal.
Assumptions:	The camera will never miss a traffic signal and it will detect traffic light colors correctly.
Notes and Issues:	If a camera misses a traffic signal, accident is a possibility.

Table 4-5 Fully Dresses Use Case Object Detection

Use Case ID:	5.0		
Use Case Name:	Object Detection		
Created By:	Amtul Mussawir Hina	**Last Updated By:**	
Date Created:	16.12.2014	**Last Revision Date:**	
Actors:	Primary: Range Finder Sensor		
Description:	The system successfully detects an object lying in the path, determines whether it is stationary or moving and then takes appropriate action		
Trigger:	The range finder detects an object		
Pre-Conditions:	1. User must be logged into the system. 2. User must have the essential training for using the system. 3. A connection is established between the processing unit and the car 4. The car is in a driving state		
Post-Conditions:	1. Successful detection of an object 2. Successful determination of whether the object is stationary or moving 3. Object avoided through appropriate decision taking		
Normal Flow:	1. Object detected by range finder sensor 2. System determines the object to be moving 3. System slows the speed down and maintains distance with the moving object to avoid collision		
Alternative Flows:	2a System determines the object to be stationary 1. System changes the car lane to avoid collision		
Exceptions:			
Includes:			
Frequency of Use:	Every time an object in the path is detected		
Special Requirements:			

Assumptions:	The range finder will successfully detects an object every time an object come into the path
Notes and Issues:	What if multiple objects lie ahead in the path

4.5.4 Sequence Diagrams:

Following are the sequence Diagrams of the system.

4.5.4.1 Login:

The figure 4-6 illustrates the Sequence Diagram of the login module

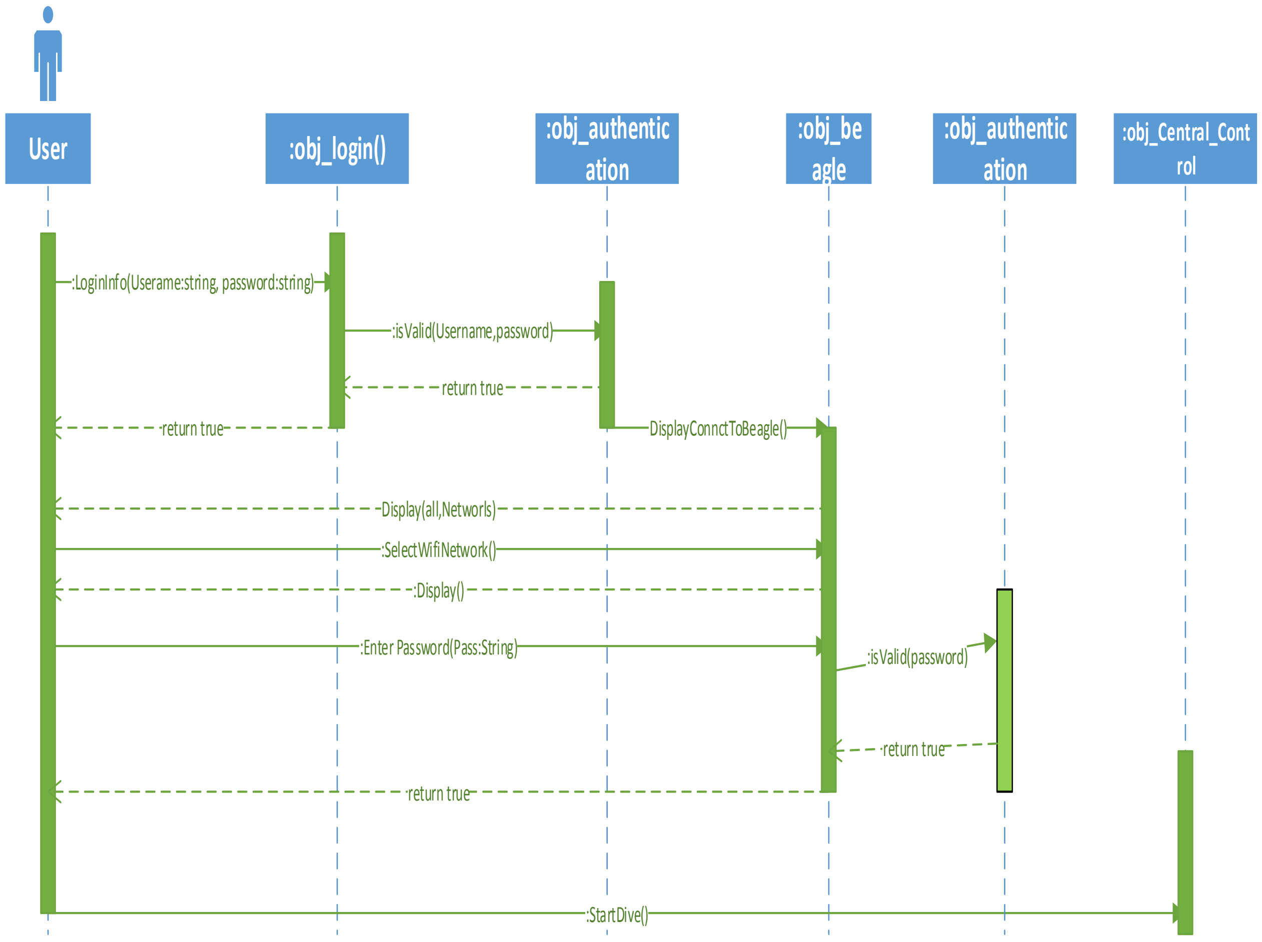


Figure 4-6 Sequence Diagram: Login

4.5.4.2 Traffic Sign Detection:

The figure 4-7 illustrates the Sequence Diagram of the Traffic Sign Detection module

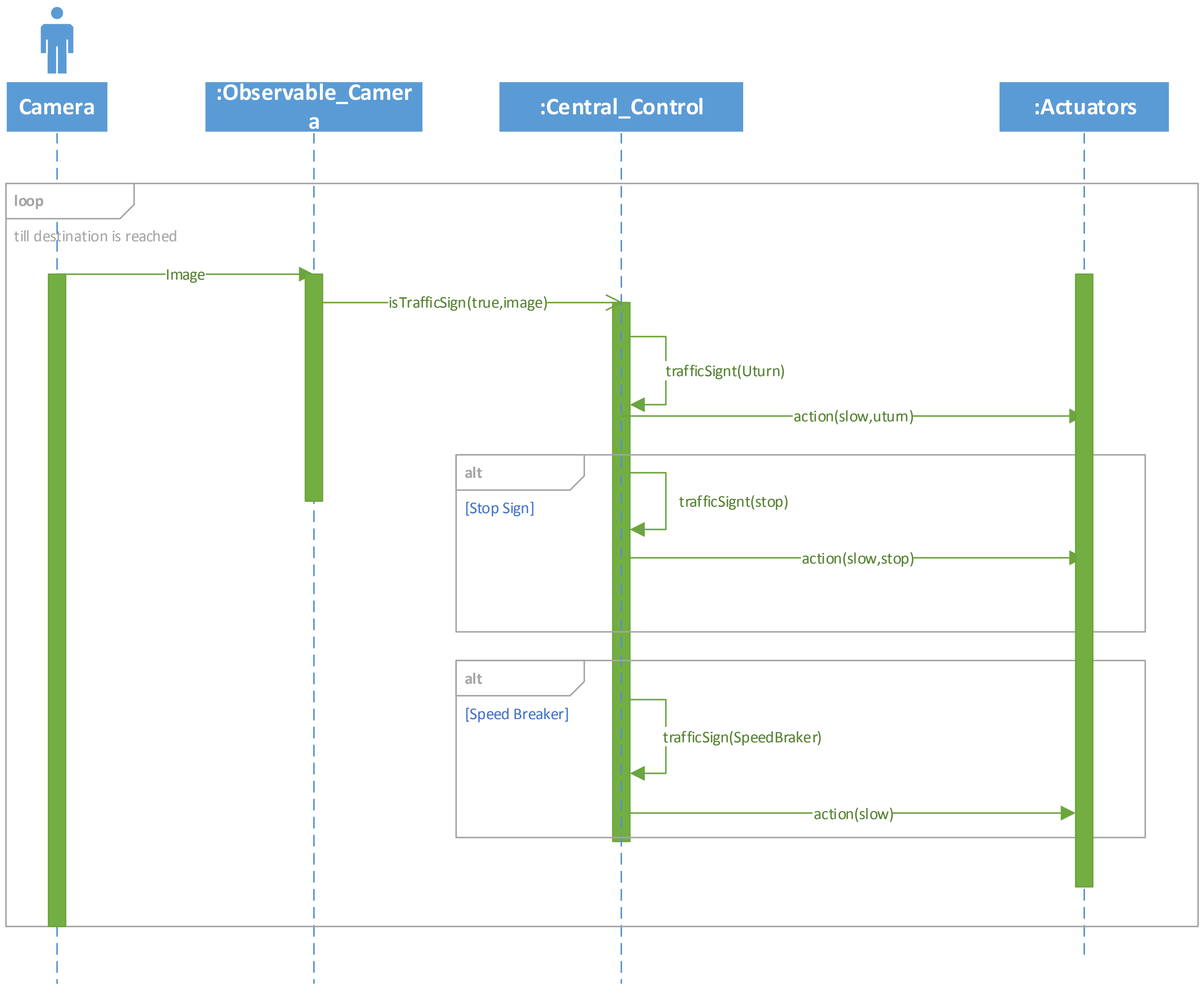


Figure 4-7 Sequence Diagram: Traffic Sign Detection

4.5.4.3 Traffic Light Detection:

The figure 4-8 illustrates the Sequence Diagram of the Traffic Light Detection module

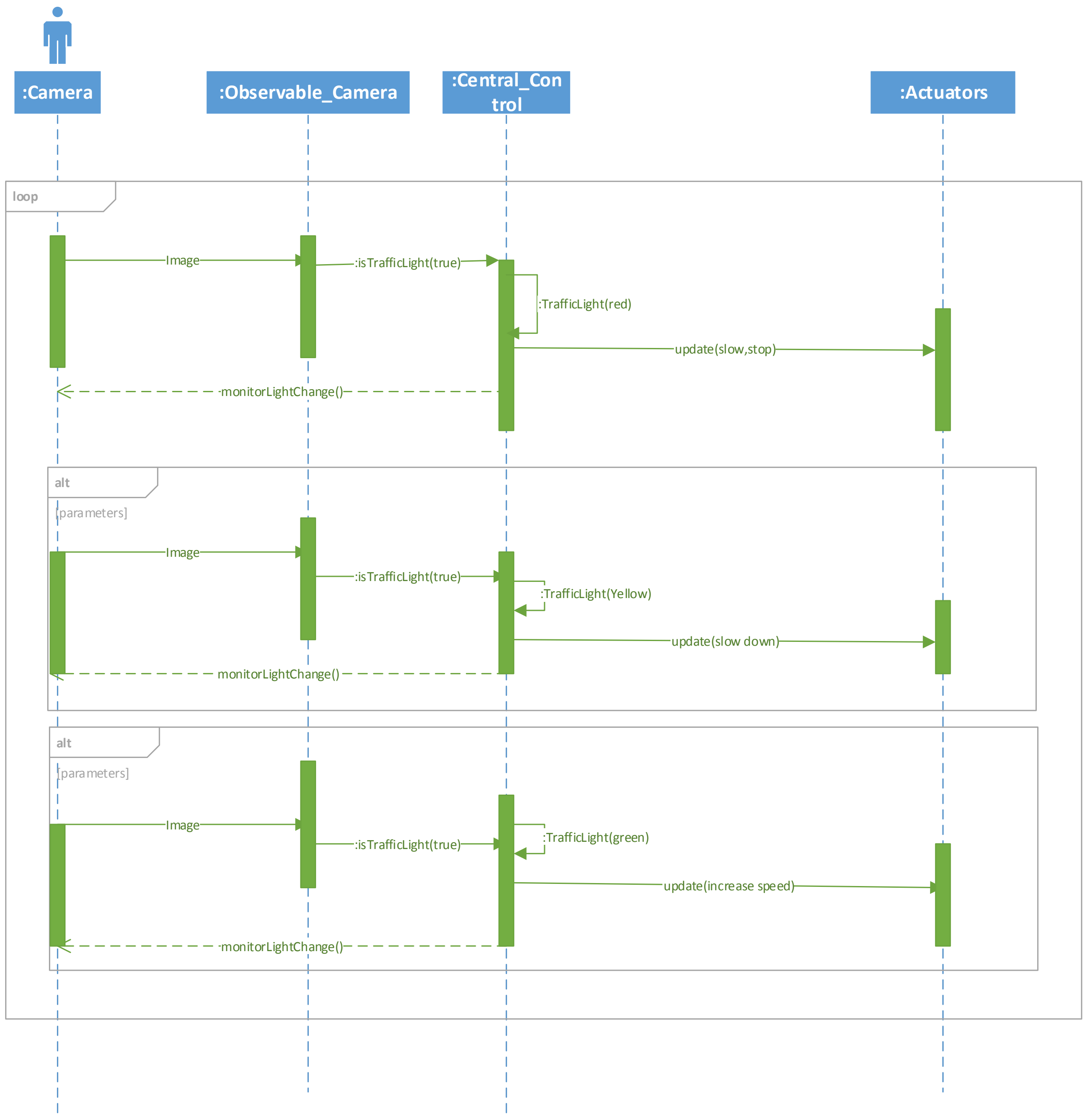


Figure 4-8 Sequence Diagram: Traffic Light

4.5.4.4 Object Detection:

The figure 4-9 illustrates the Sequence Diagram of the Object Detection Module

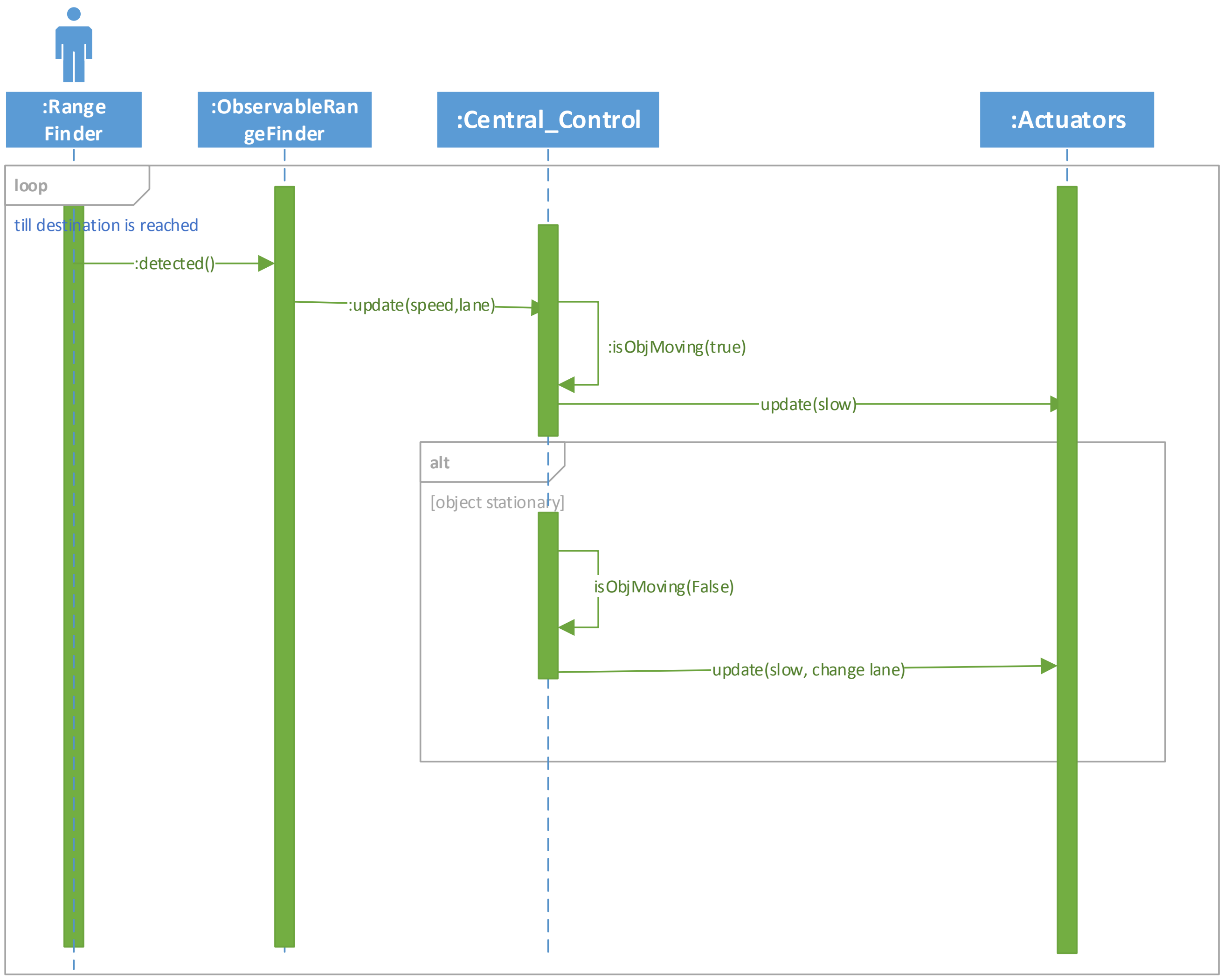


Figure 4-9 Sequence Diagram: Object Detection

4.5.5 Activity Diagram

Activity diagram is another important diagram in UML to describe dynamic aspects of the system.

Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.

So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

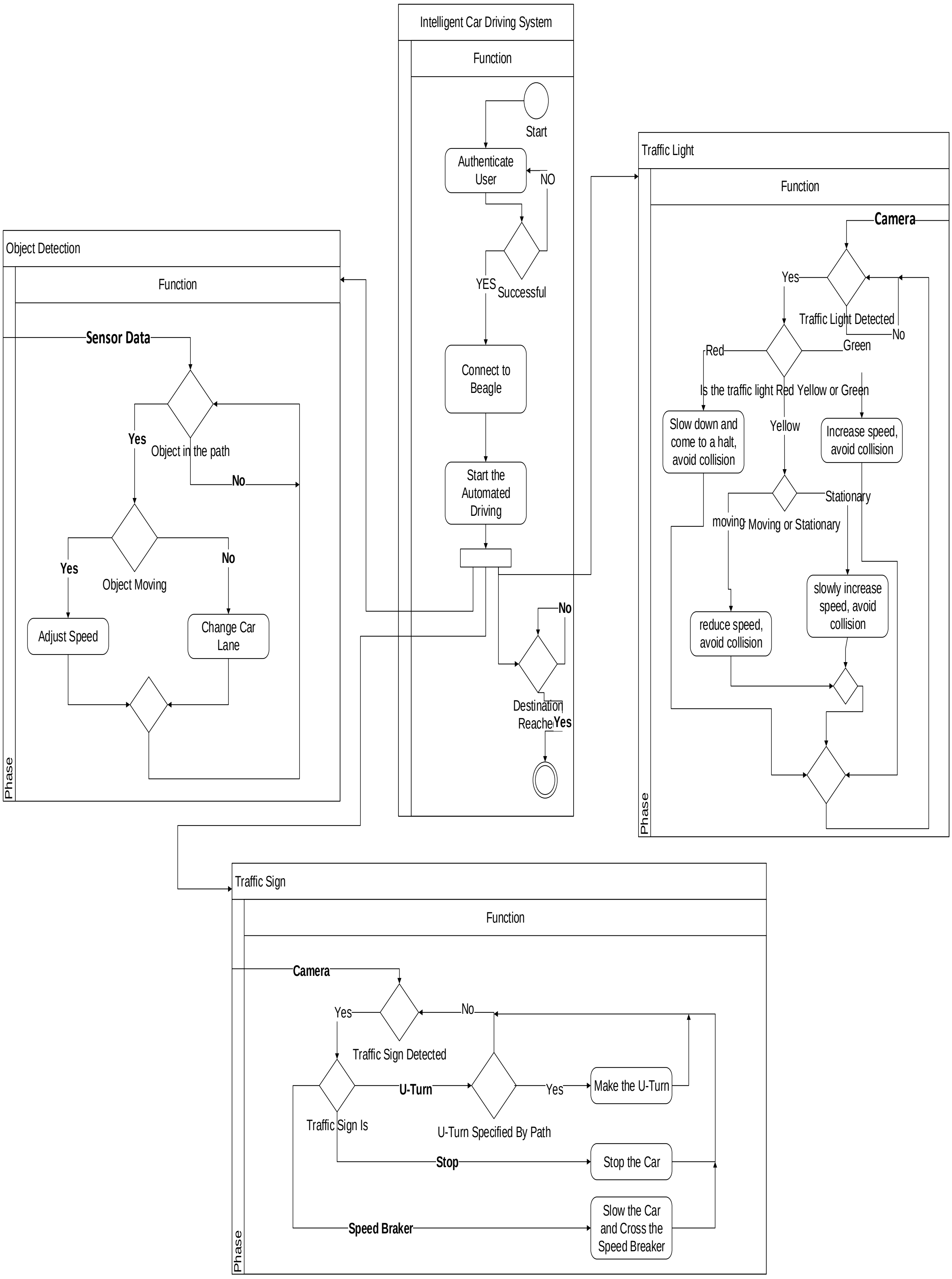


Figure 4-10 Activity Diagram

4.5.6 State Machine Diagram

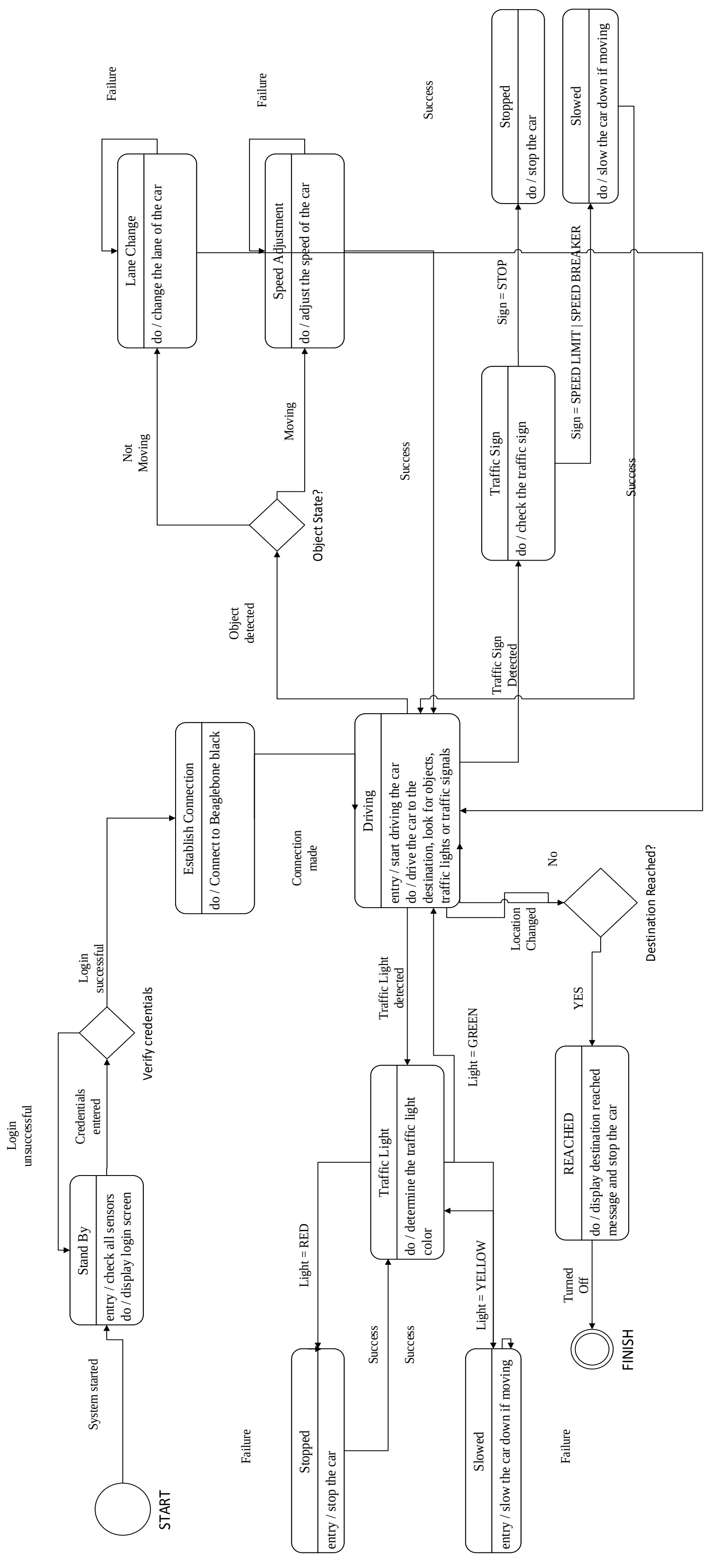


Figure 4-11 State Machine Diagram

4.5.7 Deployment Diagram

Deployment diagrams show the hardware for the system, the software that is installed on that hardware, and the middleware used to connect the disparate machines to one another. This project has a lot of hardware involved so with the help of deployment diagram, hardware and the software that will be install on that hardware will be shown. This diagram also provides the reader with an overview of the system so that developing the system is easy. Figure 4-12 illustrates the deployment diagram of the system.

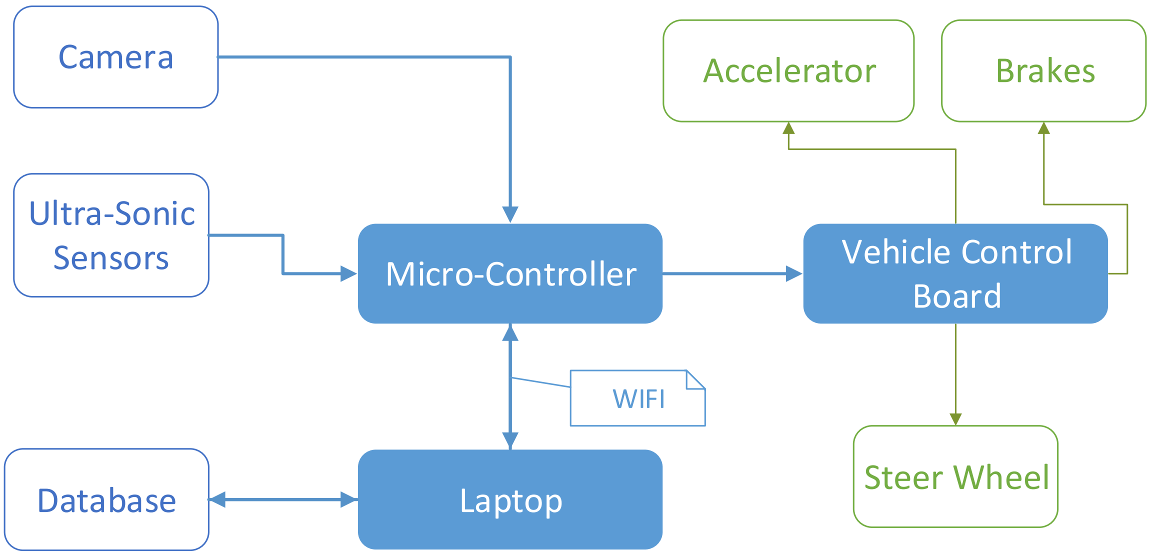


Figure 4-12 Deployment Diagram

4.6 Conclusion

As explained in the previous sections, an extensive design for the system has been developed and proposed, and the development of the project has been done based on this design features.

5. Software Implementation

5.1 Introduction

This chapter is going to cover the implementation aspect of the system and will basically outline the pseudo code in the section 5.2.

5.2 Pseudo Code

This section is going to cover the pseudo code of the system and basically outlines the modules that are going to be implemented and structures it so that it easy for the developer to code and refer back to it

5.2.1 Beagle-bone Black

This section is going to cover the pseudo code that will be implemented on the Beagle Bone Black.

```
activate ultrasonic();          % activates three sensors at the start of the system%
activate camera();              % activates camera at the start of the system%
activate wifi();                % activates WIFI at the start of the system%
activate tracker();             % activates Line Tracking at the start of the system%
create ad-hoc network();                % create ad-hoc to enable device to connect to the
                                          system at the start of the system%
create socket();                % create a socket to connect%
accept connection();            % accept the request to connect%

do parallel                             % this module runs parallel to the other two%

do {
        take video feed();
}while( true );                 % system takes the video feed of the driver at all
                                      times after the system is turned on%

do {
```

```
            process frames();
            send frames();
            receive results();
}while( result = false )                % frames are processed and compared until an event
                                           occurs%
if ( traffic_Sign = true )
do{
            Take_Action()
}
do parallel                                              %this module runs parallel to the other
two%
do {
            observe environment();
}while ( object_detection = false );         % environment is observed(directly in front of the
                                                            car and on either side) until an event
occurs%
if ( object_detection = true )
            adjust_lane();
do parallel                                              %this module runs parallel to the other
two%
do {
            monitor_line();
} while ( on-line = true );


if ( on-line = false )
            adjust_steer();
```

5.2.2 Processing Unit System

This section is going to cover the code that will be implemented on the processing unit.

```
initialize system();
```

```
do {
        get user_name;
        get password;
}while ( user_name = false || password = false );


do {
        connect_beagle();
}while ( connection = false );


do {
        receive_frames();
        process_frames();
        send_results();
} while ( true );
```

5.3 Conclusion

Pseudo-code is to a programmer what a sketch is to an artist or a draft to a writer. It allows him to think and design in terms of required capabilities instead of a specific language's limitations. It lets the developer focus on the big picture rather than slave over details. This in turn lets the developer find relationships that may make completing the details easier. Realizing these insights early reduces the rework you could face when the developer realize them later. The complete code of the system is attached in the appendix

6. Project Analysis and Testing

6.1 Introduction

This section will explain the testing methods that were carried out to ensure that system is functioning according to the parameters that were set in the requirement and design phase. Correctness of the results is ensured so as to avoid any large scale damage as this system is a critical system. In the first section, 6.2, results of the modular testing is discussed in which each module is tested by providing it with the correct form of input and observing the results so as to ensure that the desired results are obtained. Section 6.3 will cover system testing which will cover the results obtained after combining all the modules into a single fully operational system.

6.2 Modular Testing

This section will cover the testing of the system at a modular level. Module Testing will ensure that each module is working properly and is bug free, so that at the time of integration, the system does not fail due to problems based at the modular level. Since our system consists of four main modules, we have developed 4 test cases for each module. They are available for referencing in Table 6-1, 6-2, 6-3, and 6-4.

The table 6-1 briefly illustrates the test case of the Login Module, where the sytem is provided with the correct user name and password parameters, to see if the expected result is obtained. In this test, the system successfully logged in, making this tets case a pass.

Table 6-1: Test Case 1: Login

Project Name: Intelligent Car Driving System	
Login	
Test Case ID: 1	**Test Designed by: Amtul Mussawir Hina**
Test Priority: Medium	**Test designed date: 10th May, 2015**
Module Name: Login	**Test Executed by: Usama Zafar**
Test Title: Verify Login with Valid User Namae and Password	**Test Executed Date: 10th May 10, 2015**
Description: Test the System’s Login Page	
Pre-Conditions: User has valid user name and password	
Description:	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Navigate to Login Page					
2	Provide Valid Username	hassanzia				
3	Provide Valid Password	qwerty123				
4	Click on Login Button		User should be able to login	User successfully logs in and the next screen is displayed	Pass	
Post Conditions: Sys wait for input from sensors						

The table 6-2 briefly illustrates the test case of the Traffic Sign Detection Module, where the system is provided with three traffic signs to see if it successfully identifies them and takes the appropriate car control decision. In this test, the system generates the correct results hence making this test case a pass.

Table 6-2 Test Case 2: Traffic Sign Detection

Project Name: Intelligent Car Driving System	
Traffic Sign Detection	
Test Case ID: 1	**Test Designed by: Amtul Mussawir Hina**
Test Priority: **Medium**	**Test designed date: 10th May, 2015**
Module Name: **Login**	**Test Executed by: Usama Zafar**
Test Title**: Verify Login with Valid User Namae and Password**	**Test Executed Date: 10th May 10, 2015**
Description: **Test the System's Traffic Sign Module**	
Pre-Conditions: **User has logged into the system**	

Step	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes

1	Provide sys image	Stop Sign	The sys recognises the sign and stops the car	The system stops the car	pass	
2	Provide sys image	uturn	The sys recognises the sign and performs uturn		pass	
3	Provide sys image	Speed limit	The sys recognises the sign and adjusts speed		pass	
Post Conditions: System waits for the next image input						

The table 6-2 briefly illustrates the test case of the Traffic Light Detection Module, where the system is provided with three images of traffic lights to see if it successfully identifies them and takes the appropriate car control decision. In this test, the system generates the correct results hence making this test case a pass.

Table 6-3 Test Case 3: Traffic Light Detection

Project Name: Intelligent Car Driving System						
Traffic Light Detection						
Test Case ID: 1			**Test Designed by:** Amtul Mussawir Hina			
Test Priority: Medium			**Test designed date:** 10th May, 2015			
Module Name: Login			**Test Executed by:** Usama Zafar			
Test Title: Verify the Traffic Light detection module			**Test Executed Date:** 10th May 10, 2015			
Description: Test the System's Login Page						
Pre-Conditions: User has successfully logged into the system.						
Step	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Provide sys image	Image of red light	Car stops	Car stops	pass	

2	Provide sys image	Image of yellow light	Car slows down	Car slows down	pass	
3	Provide sys image	Image of green light	Car starts to accelerate	Car starts to accelerate	pass	
Post Conditions: System waits for the next image input						

The table 6-4 briefly illustrates the test case of the Object Detection Module, where the system is provided with two types of objects: moving and stationary, to see if it successfully identifies them and takes the appropriate car control decision. In this test, the system generates the correct results hence making this test case a pass.

Table 6-4 Test Case: Object Detection

Project Name: Intelligent Car Driving System						
Object Detection						
Test Case ID: 1			**Test Designed by: Amtul Mussawir Hina**			
Test Priority: **Medium**			**Test designed date: 10th May, 2015**			
Module Name: **Login**			**Test Executed by: Usama Zafar**			
Test Title**:**			**Test Executed Date: 10th May 10, 2015**			
Description: **Test the System's Object Detection Module**						
Pre-Conditions: **User has valid user name and password**						
Step	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Provide moving object in path	Moving object	System Slows Down, Avoid Collison	System slows down, avoids collision	pass	
2	Provide Stationary object in path	Stationary object	Changes lane, avoid collision	Changes lane, avoid collision	pass	

Post Conditions: Sys waits for next input

6.2 System Testing

System testing is the level of testing which comes when the whole system has been developed and integrated. The complete system was tested in different inputs with different conditions to verify that those conditions do not disrupt the performance of the system. There were no significant findings that need to be documented. All the functional requirements were verified and whole system was analyzed for performance and other attributes (failures, response delays, connection losses etc.).

6.3 Conclusion

Our testing results verified that all the requirements that were collected during the requirement elicitation phase are met and were tested to ensure that the system was conforming to them.

7. Conclusion and Future Work

7.1 Introduction

This chapter will briefly discuss the conclusion of the project along with the future work that the authors would like to propose. In the first section, 7.2, Conclusion will be presented to the reader about the results and learning of this project. In the section 7.3. Future work will be proposed that other students may want to pursue keeping this project in mind.

7.2 Conclusion

To conclude, we were able to complete the project Intelligent Car Driving System successfully. Traffic Signs were identified and necessary action was performed, along with traffic light detection and object detection.

The System is an amalgamation of both hardware and software development. The hardware integration and ensuring that input and output parameters are being met and fully functional was a formidable task. Working in an embedded systems domain required us to push ourselves out of the boundaries of simple software development and enabled us to realize and explore a new dimension of development that we were not familiar with before. Working on microcontrollers, Beagle Bone, enabled us to explore the low level language development in a Linux environment. The user application was developed in C#, using the libraries of WlanApi and OpenCv.

7.3 Future Work

In the future, work needs to be done on identification and recognition of more traffic signs and signals. Moreover, since this system was developed on a model car, work needs to be done on the deployment of this system on an actual car, using actual car controls.

More efficient techniques and algorithms for image processing need to be developed and explored so that the system performance can be increased to work in a real life environment.

8. References

[1] HobbyTronics, "Hobby Tronics," Sales and Marketing, January 2015. [Online]. Available: http://www.hobbytronics.co.za/p/734/beaglebone-black-rev-c. [Accessed Saturday May 2015].

[2] ROBOTELECTRON, "ROBOTELECTRON," Electronics Parts, Feburary 2015. [Online]. Available: www.ROBOTELECTRON.COM. [Accessed Saturday May 2015].

[3] Tp Links, "TP Links," December 2014. [Online]. Available: http://www.tp-link.com/lk/products/details/?model=TL-WN725N. [Accessed 5 May 2015].

[4] "Tenet Technetronics," nopCommerce, 2015. [Online]. Available: www.tenettech.com/product/3247/beaglebone-black-cape-lcd-43. [Accessed Saturday May 2015].

[5] "VasAutomations," Industrial Automation Solutions, 2014. [Online]. Available: http://vasautomations.com/automation/?tag=lm324. [Accessed 5 May 2015].

[6] M. Shrivathsan, " Accompa, Inc.," Accompa, Inc., 8 October 2009. [Online]. Available: http://pmblog.accompa.com. [Accessed 5 May 2015].

Appendix A: User Manual

USER'S MANUAL

Intelligent Car Driving System

Military Collage of Signals,

NUST

May, 2015

Table of Contents

1. General Information

General Information section explains the system and the purpose for which it is developed.

1.1 System Overview

People have been driving cars for more than a century now, but unfortunately car accidents are a very big cause of death in our society, and where the government is often called upon to improve safety. The fact is that it is the human error of the drivers who are now more at fault. Intelligent Car Driving System is one of the intervention aimed at decreasing the number of crashes that involve drivers.

The aim and novelty of this document is to develop and evaluate Intelligent Car Driving System, is an automated system for driving cars. This system will try to minimize human error.

The system receives the instructions from the user through user interface, which consist of a log in screen, path and destination selection screen and a log out screen.

1.2 Project References

1. Propeller Backpack. (n.d.). Retrieved from: http://www.parallax.com/product/28327
2. F. Barret, S. (n.d.). Arduino Microcontroller: Processing for Everyone! (Second ed.). Morgan & Claypool.
3. GP2Y0A21YK0F Distance Measuring Sensor Unit. (n.d.). Retrieved from http://www.sharpsma.com/webfm_send/1489
4. Jump wire structure. (n.d.). Retrieved from

   http://www.freepatentsonline.com/6899560.html

1.3 Organization of the Manual

The User Manual is divided in three sections.

- General Information
- System Summary
- Interfaces and Functionality

General Information section explains the system and the purpose for which it is developed.

System Summary section gives you information about the system configuration components and different modules of the system.

Interface and Functionality introduce you to the system interface and its functionality.

1.4 Acronyms and abbreviations

ICDS	Intelligent Car Driving System
SRS	Software Requirement Specification
RAM	Random Access Memory
GHz	Gigahertz
Mb	Megabyte
OS	Operating System

2. System Summary

System summary provides the general overview of the system. The summary outline the uses of the system's hardware and software requirement, system's configuration, user access levels and system's behavior in case of any contingencies.

2.1 System Configuration

ICDS (intelligent car driving system) operates on Linux based OS. It shell be compatible with Debian Linux environment.

2.1.1 System Components

Ultrasonic Range Finder:

Provide precise distance measurements with in 2cm to 3m range.

Arduino Due:

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.

Camera:

The PropCAM-DB is a daughterboard that gives black-and-white (grayscale) vision to Parallax Propeller-based systems. With the DB-Expander adapter, it will accommodate other Propeller host systems.

Images captured by the PropCAM can be displayed on a TV monitor (using the Propeller Backpack or other Propeller modules) or on a webpage (using the Spinneret). The images can also be examined and manipulated for use by machine and robotic vision applications.

2.2 User Access Levels

- User can take back control of the vehicle at any time.
- Admin user can change login information

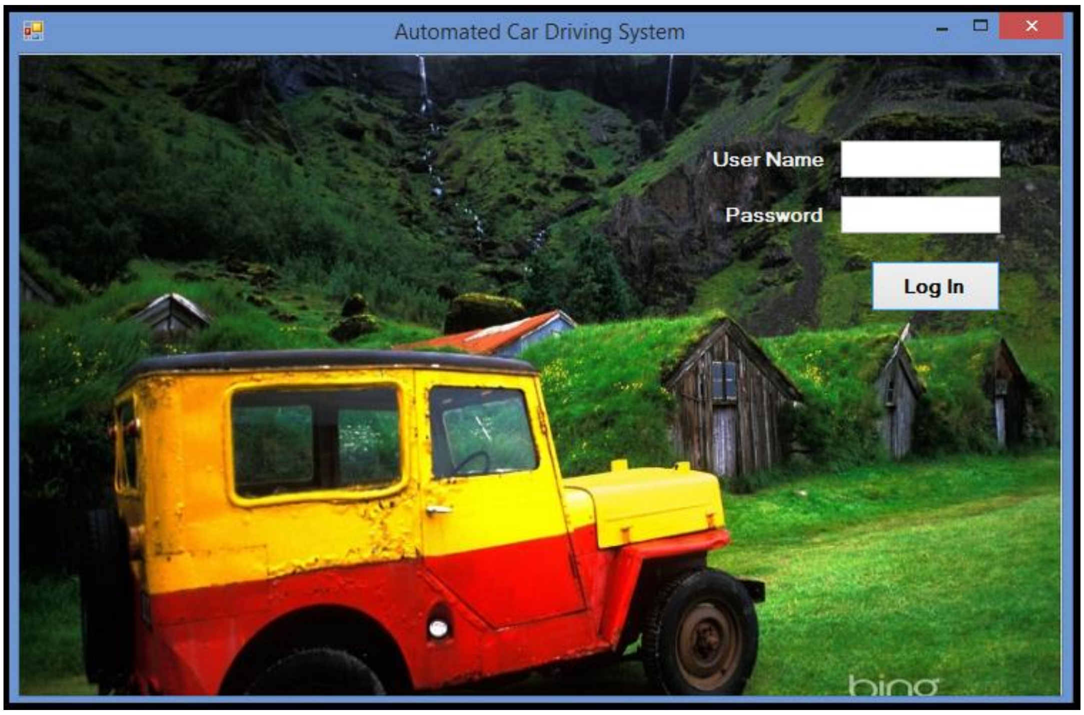
2.3 Contingencies and Alternate mode of Operations

The system shall be available at all time as it does not depend on any kind of connectivity to the internet, any outside assistance. It shall be needing the power source from inside the car.

3. Using the System

3.1 Logging in

To prevent the unauthorized use of the system the user will insert its user name and password to log in to the system.



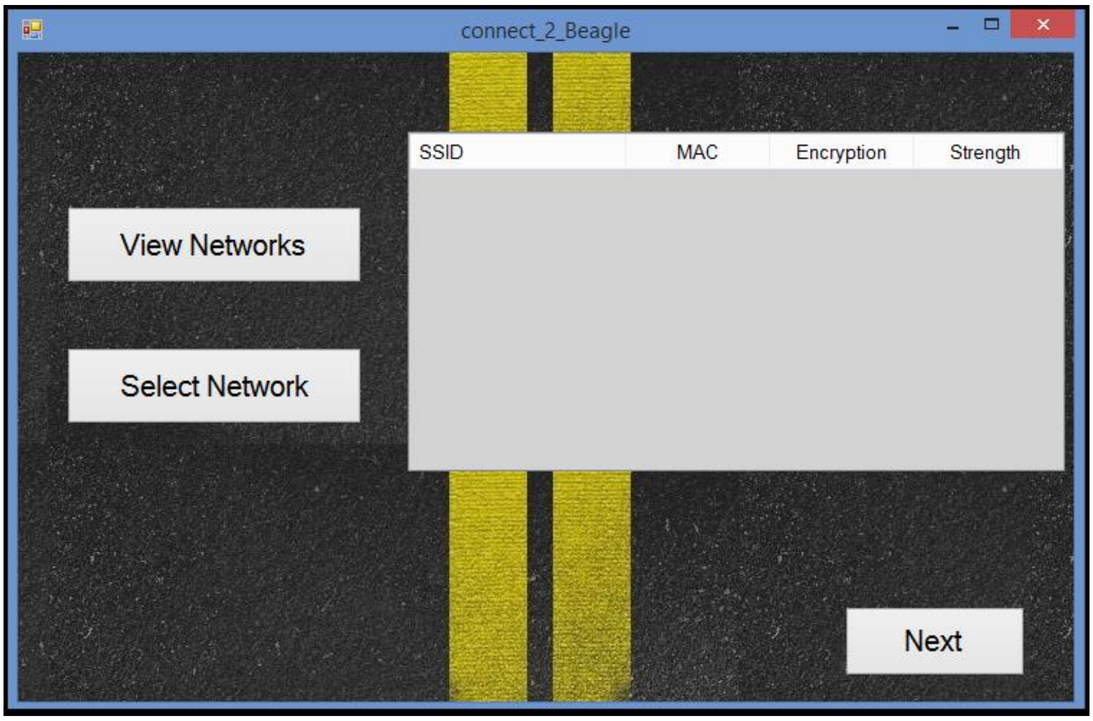
Step1: Enter your specific user name that has been provided with you along with the other reference material

Step 2: Enter your specific password that has been provided with you along with the other reference material

Step 3: Click the Login Button

3.2 Connecting to the Beagle Control Board:

This is a crucial step as this stage will connect your processing unit to the beagle device. Please follow the instructions carefully:



Step 4: Click the View Networks button. A list of Wi-Fi Networks that would be available at the moment would be displayed under the SSID column.

Step 5: Select the Network with the specific SSID that has been given to you along with the other reference material.

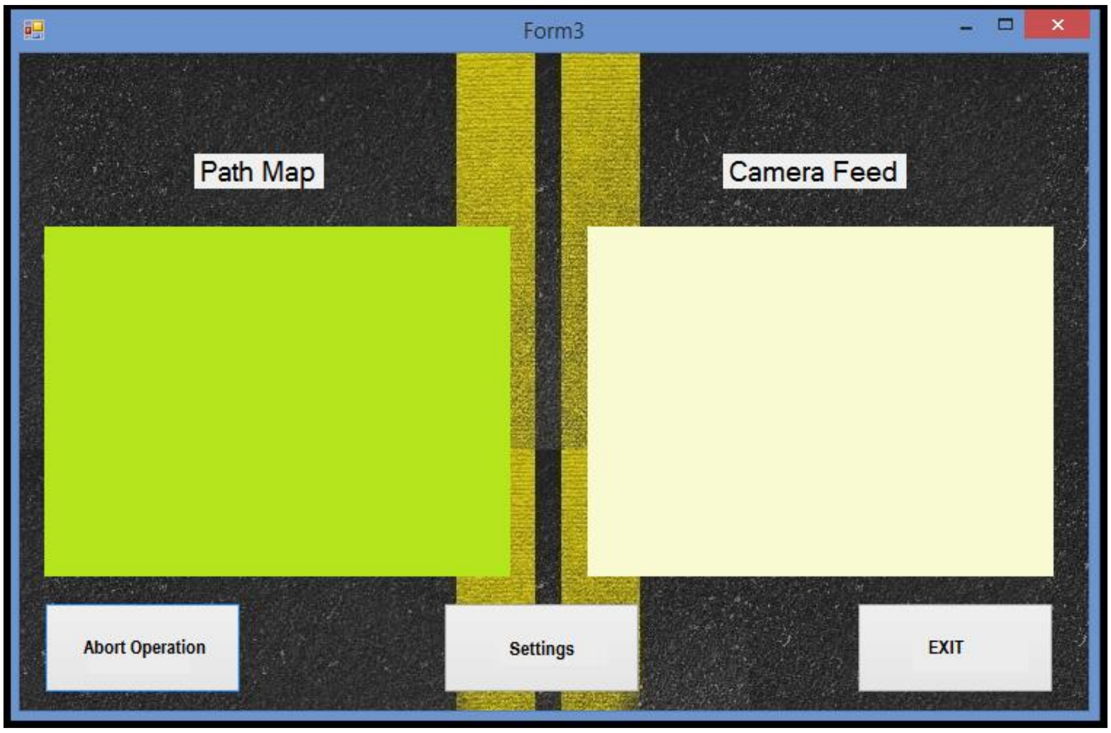
Step 6: The system will prompt you to enter the password. Please enter the specific password that was provided to you along with the other reference material

Step 7: Click Next.

3.3 Successful Login

A screen will display as follows which will indicate that the system is now ready to start and all manual controls will now disengage.

If the screen is not being displayed, please restart the application and revisit the steps mentioned in section 3.1 and 3.2. For further assistance, please contact the developers.



Please click the cross button on the upper right corner if you wish to disengage the automated system and want manual controls back.