DRIVERLESS INTELLIGENT PARKING SYSTEM

By

Capt Syed Irfan Tasneem (Group Leader)

Capt Muhammad Ehsan Ullah Khan

Capt Hussain Abdullah

Supervisor:

Asst Prof Athar Mohsin Zaidi

Submitted to the Faculty of Computer Science

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

May 2014

CERTIFICATE

Certified that the contents and form of project report entitled **"DRIVERLESS INTELLIGENT PARKING SYSTEM"** submitted by 1) Capt Syed Irfan Tasneem, 2) Capt Muhammad Ehsan Ullah Khan, and 3) Capt Hussain Abdullah have been found satisfactory for the requirement of the degree.

Supervisor: ____________________

Asst Prof Athar Mohsin Zaidi

ABSTRACT

Due to rapid increase in cars on the road, car parking is becoming a major issue in congested areas. DIPS focuses on equipping cars with intelligent controllers which shall enable automated parking without any driver's assistance. This project involves development of such embedded controller design for the cars along with an accompanying mobile side application to communicate with the controller. Here is typical scenario:

A car is driven to the start of the car park area by the driver. The driver then leaves the vehicle and instructs the car to park itself through the mobile application. The embedded controller on the car receives this instruction and requests a free parking slot from the central parking controller application. Upon receiving free parking slot, the vehicle controller will then drive the car towards the parking slot automatically, avoiding any obstacles encountered en route.

Similarly, once the driver wants his car back, he will do so by sending a command from the mobile application to the car. The car will request permission from the central parking controller and will then start to move towards the exit point of the parking area. The embedded controller shall ensure safe and smooth ride avoiding any contacts with pedestrians or other cars.

Due to budgetary constraints, the system is developed on low cost, small scale (1 / 10) cars. However, the implemented system could be installed in all modern cars.

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

To Our Parents and Teachers for their continuous support

ACKNOWLEDGMENTS

We are grateful to our parents for their support and prayers. Their faith kept us going.

We are extremely grateful to our project supervisor Asst Prof Athar Mohsin Zaidi for his support and guidance without which we could not have moved on with the research. We are thankful to our teachers and instructors here at Computer science department, MCS, all of them have guided us have made it possible for us to complete both this degree and the project.

Table of Contents

1. Introduction

a. Background

Due to an ever increasing number of vehicles on road, congested parking areas have become a major cause of concern for car owners. Busy car parks often result in unnecessary delays in routine and scheduled tasks.

One of the first assistance systems for car parking used four jacks with wheels to raise the car and then move it sideways into the available parking space. This mechanical system was proposed in 1934, yet it was never offered on any production model.One of the first experimental prototypes of automatic parallel parking was developed at INRIA on a Ligier electric car in the mid-1990s. It was extended to an automatic perpendicular parking in the early 2000s.

Automatic parking is an autonomous car maneuvering from a traffic lane into a parking place to perform parallel parking, perpendicular or angle parking. The automatic parking aims to enhance the comfort and safety of driving in constrained environments where much attention and experience is required to steer the car. The parking maneuver is achieved by means of coordinated control of the steering angle and speed which takes into account the actual situation in the environment to ensure collision-free motion within the available space

Automatic parking systems are being developed by several automobile manufacturers. A commercial version of automatic parallel parking was introduced by Toyota Motor Corporation in Toyota Prius in 2003. BMW recently demonstrated its Remote Park Assist system on a 750i. This system initiates parking by keychain remote. Lexus also debuted a car, the 2007 LS, with an Advanced Parking Guidance System. As well in 2007 the Volkswagen Touran debuted with an automatic parking system developed by Valeo, which by June 2009 is also offered on the Passat, Passat CC, Golf, Tiguan, Sharan and Polo.

b. Problem Statement

There is a need for fully automated car parking solution where the driver leaves the car at its own to be parked safely thus saving him/her precious time. Car automation has always had few associated concerns and humans have been reluctant to allow fully automated cars on road.

It consumes a lot of time in our daily life to safely park our vehicles. This time could be saved if cars were intelligent enough to recognize empty parking spaces and steer themselves towards that at their own.

c. Objectives

The objective is to develop an automated parking system that allow drivers to leave the car in the parking area and proceed on doing their job without worrying about parking the cars

d. Deliverables

(1) Project Synopsis
(2) Software Requirements Specification
(3) Mobile Application
(4) Desktop Application

(5) Vehicle Controller

(6) Scaled down model of a vehicle equipped with the vehicle controller

2. Literature Review

a. Previous Work

Modern cars are equipped with Drive By Wire system where the central computer is responsible for controlling the movements of the car. For example, a common car nowadays i.e Toyota Prius is equipped with this system where it can park itself automatically without driver's intervention just by activiting auto park mode.

Semi-automatic parking is also practised nowadays where the driver is informed about free parking slots at the start of parking area which allows him to directly proceed to the desired parking slot without wasting time.

b. Shortcomings

The Existing parking systems got relatively high equipment and operating costs (although usually less than building additional structured parking). They are only suitable in parking structures with attendants. They increase time required to park and retrieve vehicles and are unsuitable for many types of vehicles e.g vans.

c. Issues solved by this "Driverless Intelligent Parking System"

Our system is fully automatic and will be available for all type of vehicles installed with our vehicle controller module.

It solves the following issues:

(1) Relieves operators present in traditional parking lots.

(2) There is no need of extra space for mechanical equipment installed in traditional parking lots.

(3) No extra time is spent in queues for parking, instead our system saves driver's time.

3. Design and Development

a. Introduction

This part of the document provides a detailed description of the system. The main idea of the project is to build an Intelligent Parking Space where cars can drive at thier own to an allocated, empty parking space, without the driver. A central parking controller will be responsible for all the necessary coordination and communiction between the vehicles and the base station.

b. Scope

The idea of the project is to build a Driverless Intelligent Parking System enabling cars to drive at their own to an empty parking space without any assistance from the driver. A central parking controller will be responsible for all the necessary coordination and communications between the vehicles and the base station.

Due to rapid increase in cars on the road, car parking is becoming a major issue in congested areas. Almost all the modern cars are controlled by a computer onboard. A very common feature nowadays is 'Cruise control' which allows the car to cruise at a constant, fixed speed without involving the driver. Such control is achieved thanks to the Drive by Wire (DBW) technology that allows a computer to take control of the car.

This proposed system will be designed to take control of the car in a similar fashion and enable it to park itself automatically. It will consist of three major components i.e.

(1) Mobile Client Application
(2) Vehicle Controller
(3) Central Parking Space Controller

Due to budgetary constraints, we will develop the system on low cost, small scale (1 / 10) cars.
However the implemented system could be installed in all modern cars supporting drive by wire technology (DBW) with, of course, some minor changes.

c. Product Perspective

The DIPS is a self-contained software system intended for use on modern vehicles along with a mobile device. While automated parking is the main focus of the project, there is also a computer side application which will be responsible for database and synchronization services. The scope of the project encompasses both mobile and desktop applications as well as the vehicle controller itself so all aspects are covered in detail within this document. Figure 1 shows an overall view of the complete system and the interactions between different components.

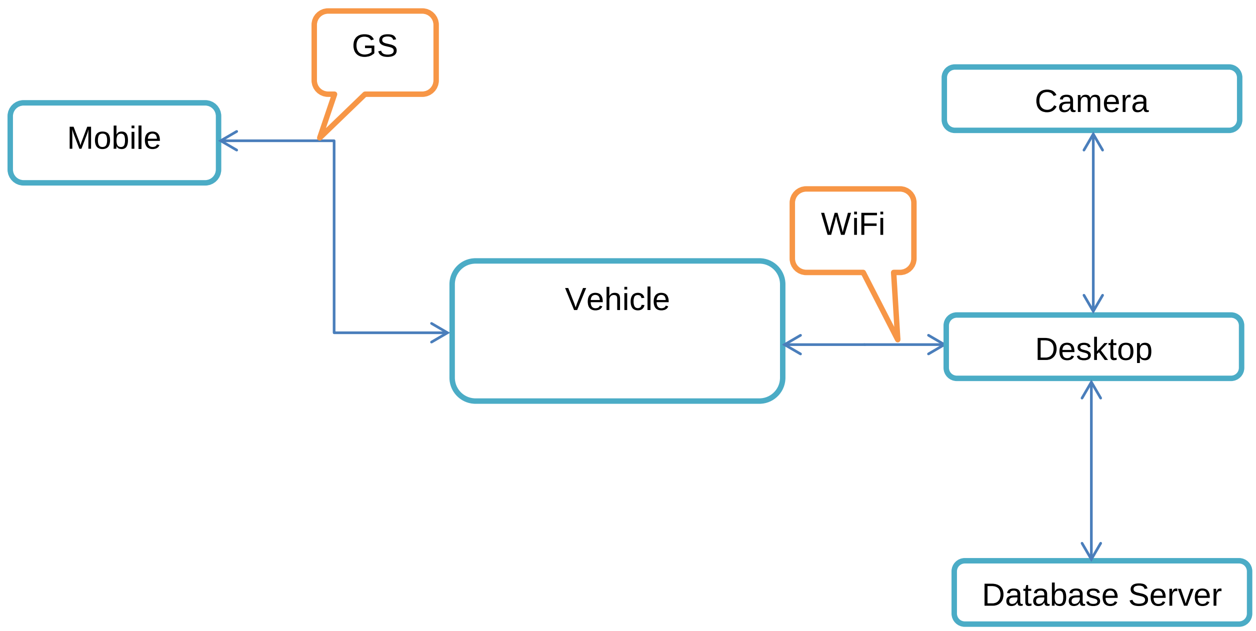

Figure 3.1: Overall View of the System

d. Product Functions

A brief outline and description of the main features and functionalities expected from DIPS are presented in this section. These features are essential to the product operations and shall be implemented for correct functioning of the complete system.

(1) **MOBILE APPLICATION**

(a) VEHICLE REGISTRATION

i. Shall allow the user to register their cars with the DIPS

ii. Shall add information about the drivers and their vehicles in the database

iii. Shall allocate a unique user identity to the driver

iv. VEHICLE DEREGISTRATION

v. Shall allow the user to deregister their cars

vi. The application shall allow removal of information about the drivers and their vehicles from database

vii. Shall release the user identity upon deregistration

(b) INQUIRE PARKING STATUS

i. Shall allow querying information about free parking slots from central parking controller

(c) PARK

i. Shall enable the car to park itself automatically and safely

(d) DEPART

i. Shall allow the car to depart from the parking area and reach the exit point safely

(e) ENABLE / DISABLE AUTO MODE

i. Shall allow switching the driving mode of the car form automatic to manual and vice versa

(2) **VEHICLE CONTROLLER**

(a) GO TO GOAL BEHAVIOR

i. The vehicle controller shall enable the car to reach any parking slot in the parking area without driver’s intervention

(b) AVOID OBSTACLE BEHAVIOR

ii. Shall allow the car to avoid obstacles in the parking area (people, other cars etc.) safely and without slamming into or touching them

(c) POSITION SENSING

iii. Shall get accurate position from an onboard GPS sensor

(d) Shall be able to pass the positional information to parking controller as well as the mobile application when requested

(e) RANGE SENSING

iv. Shall allow the vehicle to sense obstacles around it. This includes obstacles in front as well on the sides and the back of the vehicle

(f) INERTIAL MEASUREMENT UNIT

v. Shall obtain accurate heading data from onboard sensors

(g) OPTICAL ENCODING

vi. The vehicle controller shall be able to get reliable odometery data for the vehicle from onboard optical encoders

(h) ACTUATORS

vii. Shall be able to control the actuators of the vehicle to drive it

(i) WIRELESS COMMUNICATION

viii. The vehicle shall be fitted with enough hardware to communicate wirelessly with the PC application as well as the Mobile Application

(j) ONBOARD DATA STORAGE

ix. The vehicle shall have enough data storage capacity on board in order to hold data while the connectivity is not established or is lost

(3) **DESKTOP APPLICATION**

(a) PARKING STATE INFORMATION

x. The application will keep track of free/ occupied parking slots

xi. This state will always be up to date

(b) DATABASE MANAGEMENT

xii. All the activities inside the parking area shall be recorded

xiii. Integrity of the database shall be ensured

(c) IMAGE PROCESSING

xiv. Application shall be able to process digital images in order to find empty parking slots and monitoring the activities inside the parking area

(d) WIRELESS COMMUNICATION

xv. Wireless link shall be established between this application and the cars inside the parking area

xvi. Free slot data shall be sent to the vehicle along with the coordinates allowing it to move towards the free parking slot

e. Assumptions and Dependencies

Following assumption are made for correct functioning of the complete system:

(1) Environmental abnormality e.g. fog and sand storms in the area are not catered for

(2) All the hardware components are working flawlessly

(3) Mobile connectivity is available in the area

(4) Real time monitoring cameras are working correctly

(5) The vehicle is in perfect driving condition

(6) The parking area is free from pedestrian movement

f. Quality Attributes

(1) The system should support all cars having Drive By Wire (DBW) technology

(2) The mobile application will support only Android platform in version 1

(3) The mobile application should support Microsoft Windows and Apple IOS platform in version 2

(4) The system should be available from 5 am in the morning till 11 pm at night. The remaining time will be reserved for maintenance

(5) The vehicle controller should be able to drive correctly without diverting much from the given path. The diversion should be less than 10 inches.

(6) The system shall be maintained for first 6 months by the developer organization

(7) Mean Time Between Failures (MTBF) should be greater than 2 weeks

(8) Mean Time To Repair (MTTR) should be less than 30 minutes

(9) Less than 1 minute shall be needed to restart the system after a failure 90% of the time

(10) The system must be able to handle at least 10 parking slots

(11) The system should switch to backup power supply without shutting down or losing connection in case of a power failure

g. User Classes and Characteristics

As the project involves hardware interactions and is autonomous in nature, it's interaction with the users will be minimal. Nonetheless, there could be as little as two User classes i.e. Drivers and Operators.

Drivers will be driving the cars till the entry point of the parking space. From there, they will simply leave the car and shall ask it to park automatically in the parking area. Similarly, on their way back from the office, they will simply ask the car to depart from the parking slot and reach the exit point from where they can take control of the car. This class is the most important class, therefore, in the system.

An operator can monitor the activities in the parking space through the Desktop Application; however, this application will be designed to work autonomously i.e. without any human intervention. An operator could simply disable the parking space temporarily depending upon the nature of unforeseen circumstances.

h. Core Features and Stimulus Response Model

(1) **Vehicle Registration**

(a) DESCRIPTION AND PRIORITY

Whenever a new car asks for parking permission, it has to first register itself with the system. The driver will use the mobile application to register his car with his own name and identity. This feature has got high priority.

(b) Stimulus/Response Sequences

Stimulus	Response
DIPS application is launched from Android Home screen	Mobile application launches
Driver presses register vehicle button on the application	The application displays an input form to the driver
Driver enters car number, security code and his/her name and presses the register button(The security code will be provided by the parking operator)	(1) The application will forward the data to parking controller application running on the desktop computer
	The desktop application will check for any already existing entries in the database for this vehicle. If no such vehicle exists, it will insert this new driver's information to the database only if the security key and vehicle number are correctly entered and will reply with a success message. If the vehicle already exists in the database or the input data is not verified, an error message will be shown to the user
Driver presses ok button on the results dialog. In case of error message, the	

driver will close the application and will contact parking operator

(c) Functional Requirements

i. One driver can register only a single car
ii. One car can be registered with only one driver
iii. Driver shall have a maximum of 3 retries in case of entering invalid security key
iv. The mobile device shall have a working GSM connection
v. The vehicle shall have a working GSM connection
vi. The vehicle shall be connected to the Wi-Fi network of parking area
vii. The desktop application shall have working data connection with the vehicle
viii. The connection to the database server at the desktop computer shall be alive

(2) Vehicle Registration

(a) Description and Priority

A driver can deregister a car that has already been registered with his name. The car will then be available for new registration by another driver. This feature has got high priority.

(b) Stimulus/Response Sequences

Stimulus	Response
DIPS application is launched from Android Home screen	Mobile application launches
Driver presses Deregister vehicle button on the application	The application displays an input form to the driver
Driver enters car number, security code and his/her name and presses the Deregister button	The application will forward the data to parking controller application running on the desktop computer The desktop application will check for any already existing entries in the database for this vehicle. If any such vehicle exists, it will delete the driver's information from the database after verifying the entered details by the driver and will reply with a success message. If the vehicle doesn't exist in the database or the entered credentials are not valid, an error message will be shown to the user
Driver presses ok button on the results dialog. In case of error message, the driver will close the application and will contact parking operator	

(c) Functional Requirements

i. Driver can have a maximum of 2 retries in case of entering invalid security key

ii. Parking operator will be notified in case of 2 failed retries by the mobile application

iii. The mobile device shall have a working GSM connection

iv. The vehicle shall have a working GSM connection

v. The vehicle shall be connected to the Wi-Fi network of parking area

vi. The desktop application shall have working data connection with the vehicle

vii. The connection to the database server at the desktop computer shall be alive

(3) **Park**

(a) Description and Priority

Driver can instruct a registered vehicle to park itself automatically. This is the most important feature of the complete system. The car will check for free parking slots in the parking area and will try to park itself in one of those. This feature has got high priority.

(b) Stimulus/Response Sequences

Stimulus	Response
DIPS application is launched from Android Home screen	Mobile application launches
Driver presses Park vehicle button on the application	The application forwards the request to the vehicle controller
The vehicle controller will request free parking slot from the desktop application	The desktop application will pick one of the free parking slots and will respond back with the slot number and the route to that slot. In case there is no free parking slot available, it will return a message asking the driver to park the vehicle manually
After receiving the free parking slot message from the desktop application, the vehicle will start to follow the received route to reach the free parking slot avoiding obstacles	After successfully following the route, it will send a success message to the desktop application. In case of any problem(s), the car will send a message to the driver and the desktop application indicating a failure
In case of a parking failure or non-availability of free parking slot, the driver can drive the car manually to park it	After receiving the successful parking message, the desktop application will update its database with the new parking area status

(c) Functional Requirements

i. All the hardware components attached with the vehicle should be in working condition

ii. The route to the free parking slot should be the shortest one without any obstacles in the way

iii. Vehicle shall be able to act upon path corrections from the desktop application

iv. Driver shall not be involved during automatic parking

v. Vehicle shall park itself in only the allocated free slot

vi. The vehicle shall be connected to the Wi-Fi network of parking area

vii. The desktop application shall have working data connection with the vehicle

viii. The connection to the database server at the desktop computer shall be alive

(4) **Depart**

(a) Description and Priority

Driver can instruct a registered vehicle to depart itself automatically. This is the also a very important feature of the complete system. The car will drive away to the exit point of the parking area automatically. This feature has got high priority.

(b) Stimulus/Response Sequences

Stimulus	Response
DIPS application is launched from Android Home screen	Mobile application launches

Driver presses Depart vehicle button on the application	The application forwards the request to the vehicle controller
The vehicle controller request permission to leave from the desktop application	The desktop application will check for any movements in the parking area and in case of no such movements, it will allow the car to depart. If any other vehicle is entering or leaving the parking area, it will ask the car to wait till the time parking area is cleared
After receiving success message from the desktop application, the vehicle will start driving itself towards the exit point of the parking area avoiding obstacles. In case the desktop application asks the car to wait, the car will wait till the time it is allowed by the desktop application	After successfully reaching the exit point, the vehicle will inform the desktop application as well as the driver

(c) Functional Requirements

i. The route to the exit point should be the shortest one without any obstacles in the way

ii. Vehicle shall be able to act upon path corrections from the desktop application

iii. Driver shall not be involved

iv. Vehicle shall depart only when asked to do so from the central parking controller

v. All the hardware components attached with the vehicle should be in working condition

vi. No more than one vehicle can move in the parking area at one time

vii. The vehicle shall be connected to the Wi-Fi network of parking area

viii. The desktop application shall have working data connection with the vehicle

ix. The connection to the database server at the desktop computer shall be alive

(5) **Enable/ disable auto park mode**

(a) Description and Priority

Driver can, at any time, enable or disable the auto parking mode of the vehicle. This is an emergency avoiding feature where the driver can stop the vehicle immediately if he/she feels that the vehicle is going out of control. This feature has got medium priority.

(b) Stimulus/Response Sequences

Stimulus	Response
DIPS application is launched from Android Home screen	Mobile application launches
Driver presses Enable / disable auto mode toggle button	The car stops immediately suspending all ongoing operations if the driver has pressed disable button. It resumes the suspended operations if driver has pressed the enable

	button. The status of the vehicle is also forwarded to the desktop application

(c) Functional Requirements

i. Desktop application shall maintain a log of the mode of vehicle

ii. The mobile device shall have a working GSM connection

iii. The vehicle shall have a working GSM connection

iv. The vehicle shall be connected to the Wi-Fi network of parking area

v. The desktop application shall have working data connection with the vehicle

vi. The connection to the database server at the desktop computer shall be alive

i. Use Case Diagram

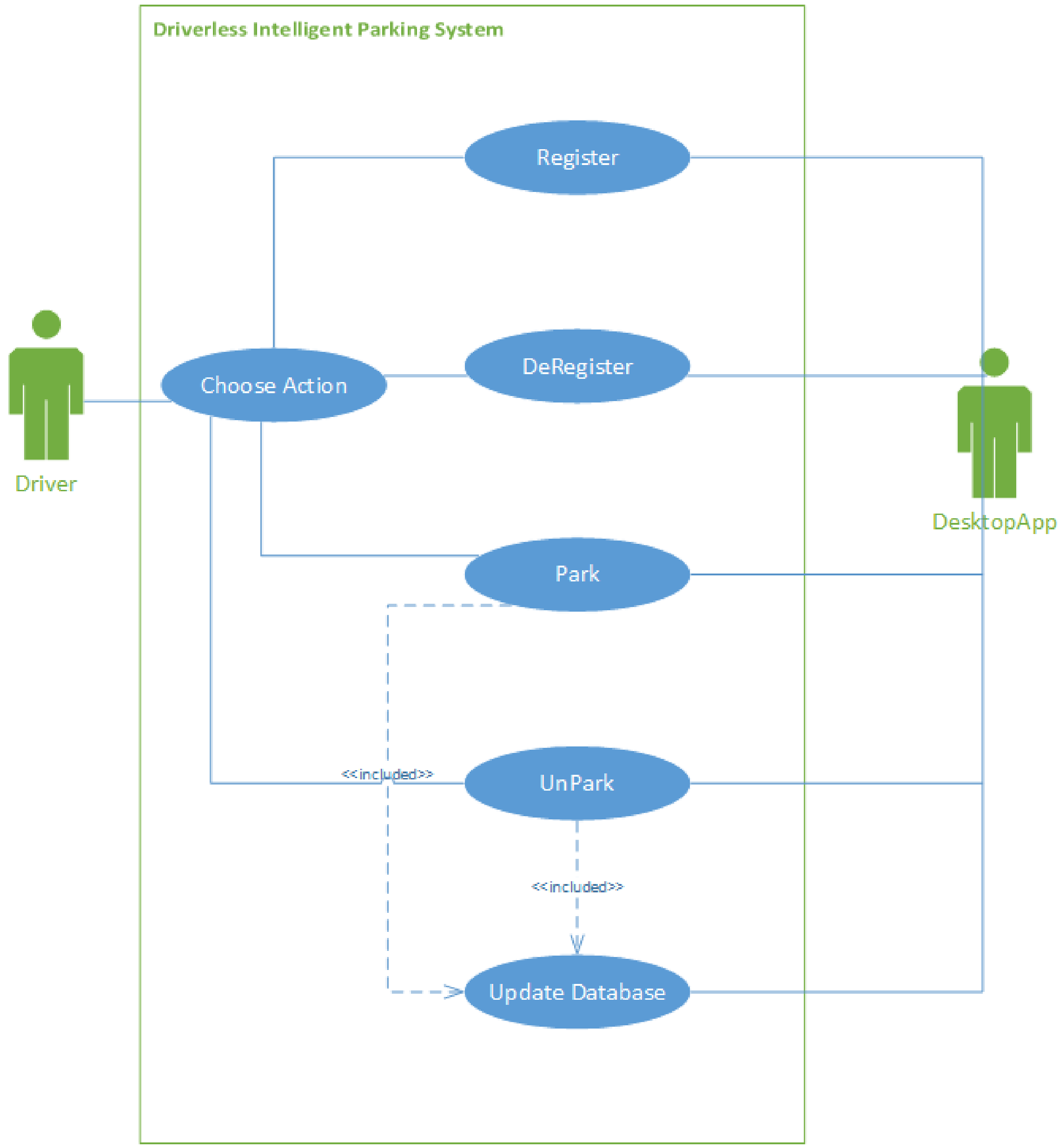

Figure 2Use Case Diagram

j. Sequence Diagram

(1) Registring a Vehicle

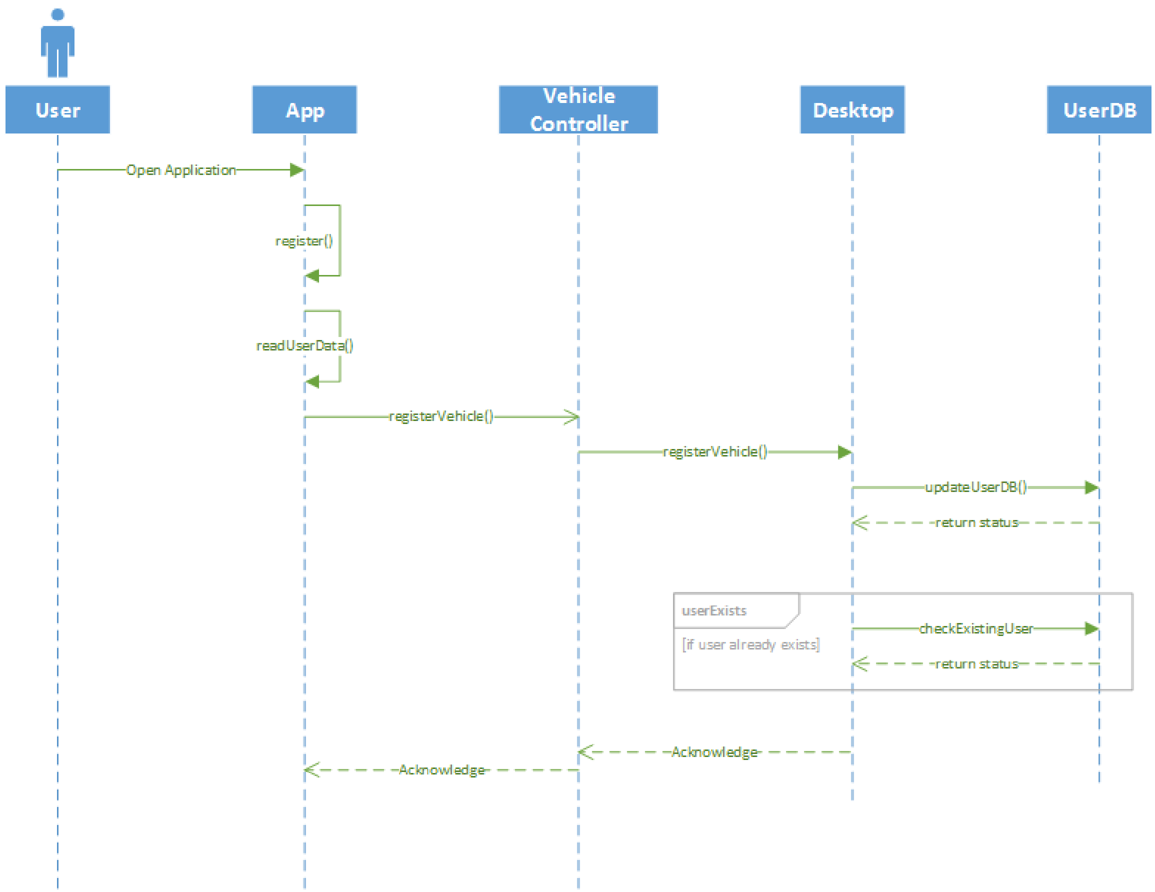

Figure 3 Register a vehicle with Parking System

(2) De-Registring a Vehicle

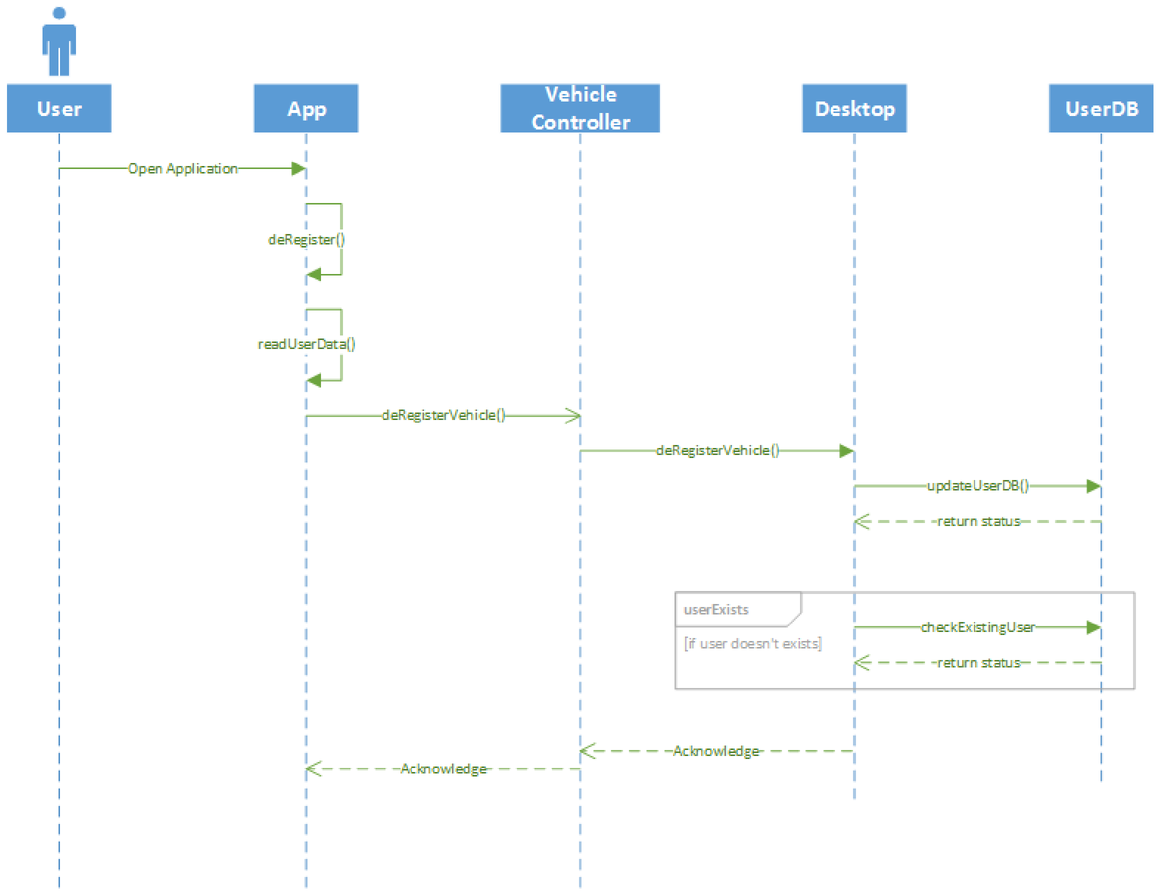

Figure 4De-Register a vehicle

(3) **Park Vehicle**

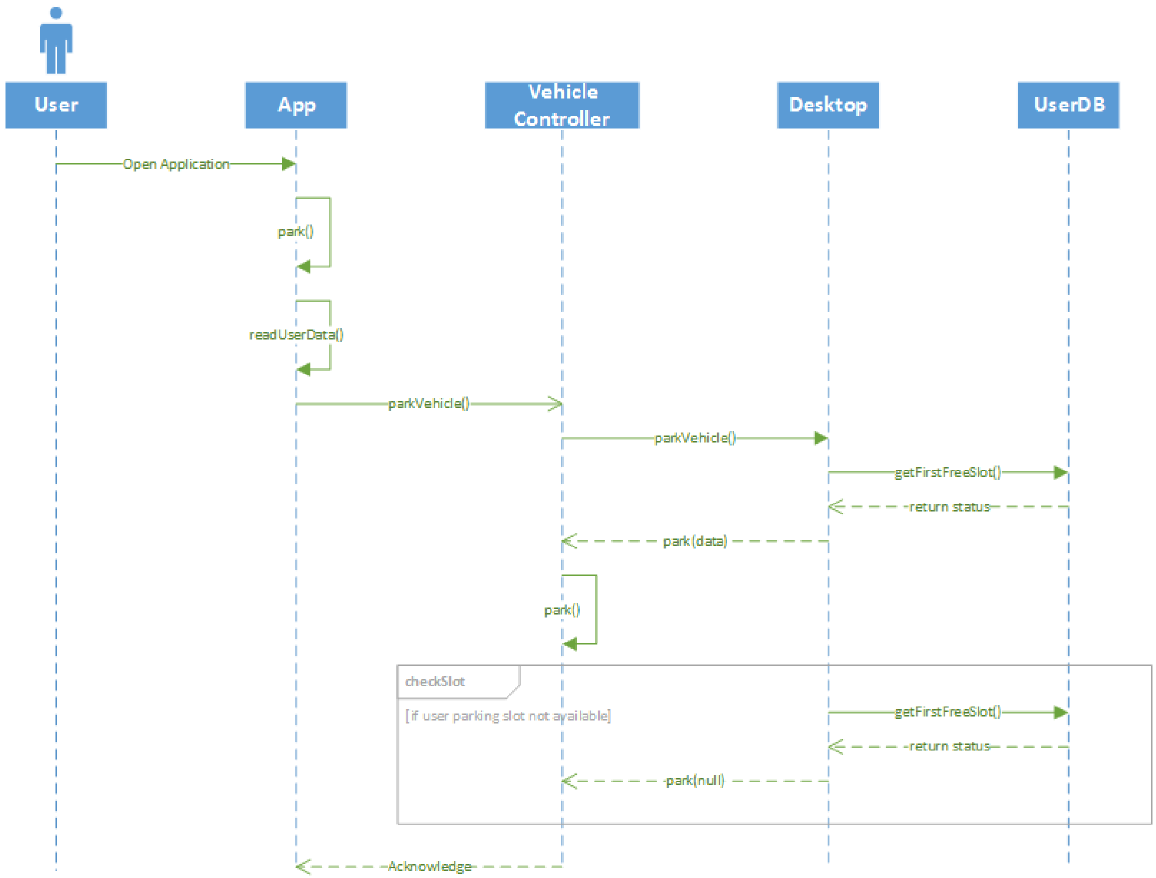

Figure 5Park Vehicle

(4) **Depart Vehicle**

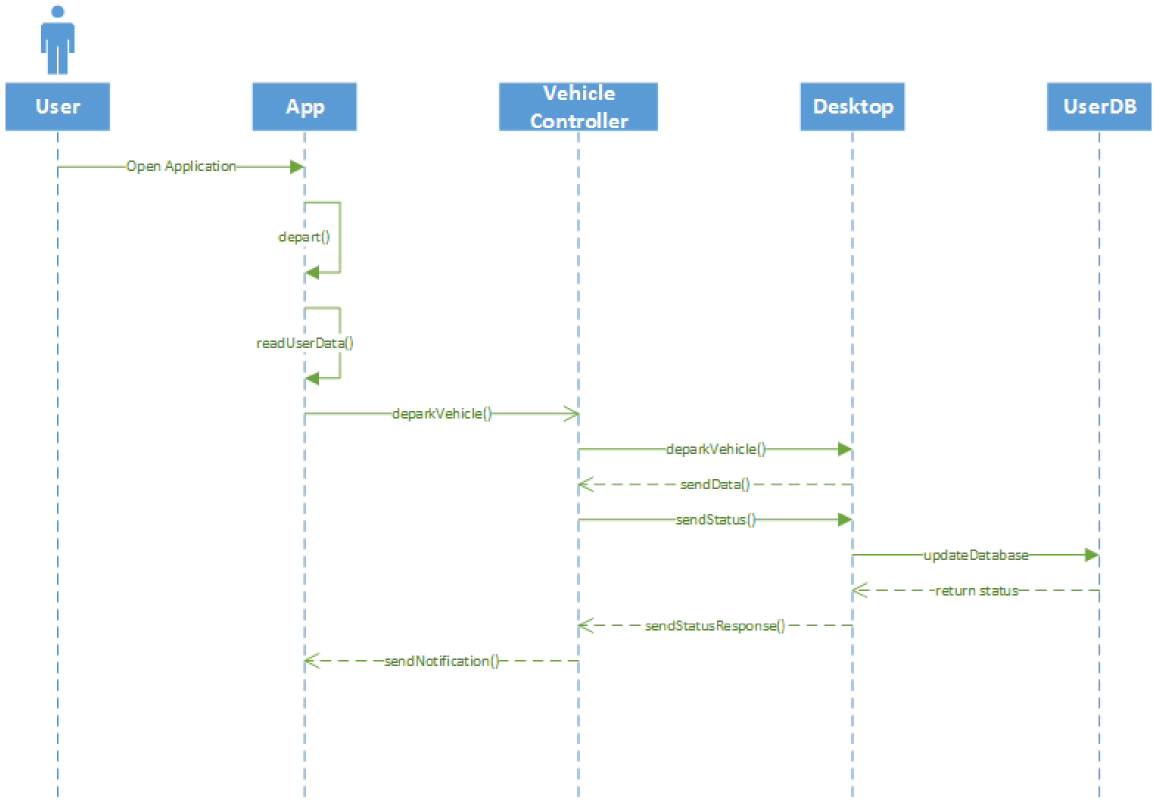

Figure 6Depart Vehicle

4. System Implementation Tools and Technologies

a. Microsoft Visual Studio 2013

[1]Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

b. C#

[2]C# (Pronounced: C Sharp) is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, procedural, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within its .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2006). C# is one of the programming languages designed for the Common Language Infrastructure.

---

[1] https://en.wikipedia.org/wiki/Microsoft_Visual_Studio
[2] http://en.wikipedia.org/wiki/C_Sharp_(programming_language)

c. Code Blocks Arduino IDE

Code Blocks Arduino IDE is used to program the onbard 8bit,16MHz microprocessor. It is a highly efficient and optimized IDE for writing code with small footprints suitable for embedded controllers.

d. Eclipse ADT Bundle

Eclipse Android Develoment tool was used to design and implement Mobile application for Andorid 4.0.

5. Software Implementation

The system consists of 3 different applicaitons. Major functions are presented here:

a. Vehicle Controller

```
#include <Arduino.h>

#include <Wire.h>

#include <Servo.h>

#include <include/L3G.h>

#include <include/LSM303.h>

#include <include/TimerThree.h>


//------------------   Utility Functions  ----------------


#define ToRad(x) ((x)*0.01745329252)      // *pi/180

#define ToDeg(x) ((x)*57.2957795131)      // *180/pi

#define epsilon ToRad(5)                  // fat guard


struct navPoint {                         // this structure contains the navigation
information required to reach a goal

  float distance;

  float direction;                        // 0 means move straight in the direction where the
car is heading right now

};
```

```
//------------------    Global Variables  ----------------

volatile double encoderTicks = 0;                    //  counts the number of encoder ticks from the drive motor
volatile double distanceToObstacle = 0;              //  just for testing, replace it afterwards

//------------------    Custom Classes  ----------------
#include <include/Vector.h>
//#include <include/AHRS.h>
#include <include/Sharp.h>
#include <include/Ultrasonic.h>
#include <include/Robot.h>
#include <include/Radar.h>
#include <include/Controller.h>
#include <include/Blended.h>
#include <include/GoToGoal.h>
#include <include/GoToAngle.h>
#include <include/AvoidObstacle.h>
#include <include/Supervisor.h>

//------------------    Global Program Objects  ----------------

Robot       car(9, 8, 45, 44, 6);                    //  driving enable, driving reference, driving forward, driving backward, steering servo
```

```
Supervisor  kernel(car, 4);                    //  supervisor object to control the complete program, 4 = Blended mode as default controller


void updateEncoderTicks();                      //  ISR to update encoder ticks

void handleTimerInterrupt();                    //  this function will be called after each timer interrupt




void setup()

{

   pinMode(13,OUTPUT);                          //  LED


   attachInterrupt(0, updateEncoderTicks, RISING); //  attaching the interrupt handler to update the encoder ticks


   Serial.begin(19200);

   Serial.println("Starting Setup...");

   long start = millis();                       //  records the starting time of the program


   car.attachServo();                           //  attach the steering servo to the pre specified pin, has to be done in Setup()


   Timer3.initialize(500000);                   //  time period in microseconds

   Timer3.attachInterrupt(handleTimerInterrupt);   //  function to call after timer interrupt

                                                //  revolve the radar

   long end = millis() - start;                 //  records the time taken by Setup
```

```
    Serial.println(end);

    Serial.println("Setup complete.");



 }


void loop()

{


    //ahrs.update();                              //  must be called once in a loop to update
DCM calculations for the AHRS

    kernel .execute();                          //  controls the program execution for the
car

}


void updateEncoderTicks(){

    encoderTicks++;                              //  increment the encoder ticks

}


void handleTimerInterrupt(){                    //  this function will be called after
each timer interrupt

                                                //  to update update encoder clicks


    digitalWrite(13,digitalRead(13)^1);          //  toggles the LED light

}
```

```
#ifndef SUPERVISOR_H

#define SUPERVISOR_H


#define maximumPoints      10                  //  maximum number of intermediate
points required to reach target


class Supervisor                              //  central class to supervise all operations

{

   private:

      Robot          *car;                    //  robot object to associate with the supervisor

      Controller     *currentController;      //  pointer to keep track of current
controller

      GoToGoal       GTG;                     //  GoToAngle controller object

      GoToAngle      GTA;                     //  GoToAngle controller object

      AvoidObstacle  AO;                      //  AvoidObstacle controller object

      Blended        AOandGTG;                //  AvoidObstacle and GoToGoal object


      float   dt;                             //  loop frequency

      float   safeDistance;                   //  distance at which to stop the vechicle and
still consider the goal as achieved

      float   loopStartTime;                  //  used to calculate dt

      float   goalAngle;                      //  used for the GTA controller


      int     currentPointIndex;              //  index to the current navigation point

      int     addedNavPoints;                 //  total number of navigation points added
in the list
```

```
        bool    enableLogOutput;              //  flag to enable / disable data logging
        bool    atTargetLocation;             //  flag to check whehter the robot is at
target location
        bool    IPAdded;                      //  flag to notify whether Intermediate point
was added or not
        bool    state;                        //  enable / disable automatic control


        String  command;                      //  command received over Serial Link


        Vector  goal;                         //  current goal vector
        Vector  IP;                           //  intermediate point
        Vector  AOGTG;
        Vector  pointsList[maximumPoints];    //  list of all points that lead to a
target


    public:
        float staticDistance;
        Supervisor(Robot &r, int controller = 1) {     //  associates the robot object
and controller with the supervisor
            safeDistance = 3;                 //  in mm
            loopStartTime = 0;


            car = &r;
            setController(controller);        //  default controller as the GoToAngle
if none specified
            currentPointIndex = 0;            //  no navigation points added right
now
```

```
        addedNavPoints = 0;                     //  no navigation points added right
now
        atTargetLocation = true;                //  true because no target was given
till this point
        state = false;                          //  automatic mode disabled
        command = "";
        IPAdded = false;
        enableLogOutput = false;
    }
    void updateOdometry(){                      //  approximates the location of the
robot

        float distance = encoderTicks * car->distanceMultiplier;
        encoderTicks = 0;                       //  reset the ticks count to zero



        staticDistance = distance;              //  for the two udater functions below


//Serial.println("St");     //debug purpose
        Vector position = car->filteredPosition;
        Vector changeInPosition;

        float theta = car->filteredHeading;
        float dynamicHeading = car->getHeading();
        float phi = ToRad(car->getSteeringAngle() / 1.875);     //30 degrees of
servo = 16 degrees on ground
        float gain = car->StoDFilterRatio;
```

```
}

    float thetaDt;

    float backwards = theta + phi + PI;
    backwards    = atan2(sin(backwards),cos(backwards));

    float original = theta + phi;
    original    = atan2(sin(original),cos(original));
//Serial.println("b4if");     //debug purpose

    if (car->getState() == 'b'){         //  if the car is moving backwards, subtract
the change in position

        changeInPosition.x = -staticDistance * cos(backwards);
        changeInPosition.y = -staticDistance * sin(backwards);
        thetaDt = staticDistance / car->wheelBase * sin(-phi);
        position = position - changeInPosition;
    }
    else{

        changeInPosition.x = (  (( gain ) * distance * cos(original))  + (( 1.0 -
gain ) * distance * cos(dynamicHeading)) );
        changeInPosition.y = (  (( gain ) * distance * sin(original))  + (( 1.0 -
gain ) * distance * sin(dynamicHeading)) );
        thetaDt = staticDistance / car->wheelBase * sin(phi);
        position = position + changeInPosition;
    }
```

```
//Serial.println("afif");     //debug purpose
            float filteredHeading,staticHeading;



            staticHeading = theta + thetaDt;
            staticHeading = atan2(sin(staticHeading),cos(staticHeading));

            filteredHeading =  ((( gain ) * staticHeading)  + (( 1.0 - gain ) *
dynamicHeading));
            filteredHeading = atan2(sin(filteredHeading),cos(filteredHeading));



            car->filteredHeading  = filteredHeading;    //  updated heading  of the car
            car->filteredPosition = position;           //  updated position of the car


        }
        void updateDynamicOdometry(){                   //  approximates the location of
the robot
                                                        //  should be called in every iteration
                                                        //  the location of the robot is updated based
on the difference to the previous encoder
                                                        //  ticks. This is only an approximation.
            Vector position = car->getPosition();
            Vector changeInPosition;
            float theta = car->getHeading();
```

```
        changeInPosition.x = staticDistance * cos(theta);

        changeInPosition.y = staticDistance * sin(theta);


        if (car->getState() == 'b')        //  if the car is moving backwards, subtract
the change in position

           position = position- changeInPosition;


        else

           position = position+ changeInPosition;


        car->setPosition(position);  //  updated position of the car

    }

    void updateStaticOdometry(){


        Vector staticPosition = car->staticPosition;

        Vector changeInPosition;

        float theta = car->staticHeading;

        float phi = ToRad(car->getSteeringAngle() / 1.875);     //30 degrees of
servo = 16 degrees on ground


        float backwards = theta + phi + PI;

        backwards    = atan2(sin(backwards),cos(backwards));


        float original = theta + phi;

        original    = atan2(sin(original),cos(original));
```

```
        changeInPosition.x = staticDistance * cos(original);
        changeInPosition.y = staticDistance * sin(original);


        float thetaDt = staticDistance / car->wheelBase * sin(phi);


        if (car->getState() == 'b'){        //  if the car is moving backwards, subtract the change in position
            changeInPosition.x = -staticDistance * cos(backwards);
            changeInPosition.y = -staticDistance * sin(backwards);
            thetaDt = staticDistance / car->wheelBase * sin(-phi);
            staticPosition = staticPosition - changeInPosition;
        }
        else{
            staticPosition = staticPosition + changeInPosition;
        }


        car->staticHeading += thetaDt;
        car->staticHeading = atan2(sin(car->staticHeading),cos(car->staticHeading));



        car->staticPosition=staticPosition;  //  updated position of the car
    }


    void execute(){                             //  selects and executes the current controller
```

```
}

//--------------    this code block is used to keep track of the time step   -------------

if (loopStartTime == 0)
   dt = 1/50;                    //  50 Hz
else
   dt = millis() - loopStartTime;  //  actual time step
loopStartTime = millis();


executeCommand();       //  executes any command(s) received over the serial port
car->update();          //  updates the heading of the car from the AHRS
updateOdometry();       //  updates the location of the car
updateStaticOdometry();
updateDynamicOdometry();


if ( state == false ){                  //  the kernel is in manual state
                                        //  do nothing, act on commands only
}
else if ( atTargetLocation ){           //  at the target
   car->stop();
   car->setSteeringAngle(0);
}
```

```
else{
   Vector distanceToGoal;
   distanceToGoal = goal - car->filteredPosition  ; //Encoder odometery

   float  distance = distanceToGoal.getMagnitude();

   if ( distance < safeDistance ){        //  quite close to the goal so
consider it as achieved
      achievedCurrentGoal();             //  mark the current goal as
achieved and move the current goal pointer ahead
   }
   else{                                  //  execute the controller to reach the goal

      float headingCorrection;

      headingCorrection = currentController->execute( *car, goal, dt );
      float error = currentController->getError();        //  get the error
without applying PID parameters

      char state = car->getState();
      if (state == 'f'){
         if ( abs(error) > (PI/2 + epsilon) && currentController-
>detectObstacle() == 0 ){   //  activate reverse gear
            car->setSteeringAngle(-headingCorrection);
            car->moveBackward();
         }
         else{
```

```
                car->setSteeringAngle(headingCorrection);
                car->moveForward();
            }
        }
        else if (state == 'b'){
            if ( abs(error) < (PI/6) ){
                car->setSteeringAngle(headingCorrection);
                car->moveForward();
            }
            else{
                car->setSteeringAngle(-headingCorrection);
                car->moveBackward();
            }
        }
    }
}
if (enableLogOutput)
    printGraph();

}

void updateCurrentGoal(){
    if (currentPointIndex >= 0 && currentPointIndex < maximumPoints){
        goal.x = pointsList[currentPointIndex].x;
        goal.y = pointsList[currentPointIndex].y;
```

```
    }
    else
        atTargetLocation = true;

}
void achievedCurrentGoal(){                    //  mark the current goal as
achieved and move to the next goal



    Serial.print("Total Goals : ");
    Serial.println(addedNavPoints);
    Serial.print("Goal acheived : #");
    Serial.println(currentPointIndex);

    currentPointIndex++;
    if (currentPointIndex >= addedNavPoints)
        atTargetLocation = true;               //  no more goal points left to reach
    else
        updateCurrentGoal();                   //  load next goal

}
void printPointList(){
    for(int i =0; i<addedNavPoints; i++){
        Serial.print("X : ");
```

```
        Serial.print( pointsList[i].x);
        Serial.print("  Y : ");
        Serial.println( pointsList[i].y);
      }
      Serial.println(currentPointIndex);
      Serial.print(goal.x);
      Serial.print(":");
      Serial.println(goal.y);

    }
    void addGoal( navPoint pt ){                    //  add a goal location to the goal list
      if(addedNavPoints < maximumPoints - 1 ){

        Vector carPosition;


        carPosition = car->filteredPosition;

        float distance  = pt.distance;
        float goalAngle = pt.direction;

        goalAngle = car->filteredHeading + goalAngle;
        goalAngle = atan2(sin(goalAngle), cos(goalAngle));
```

```
            goal.x = carPosition.x + distance * cos(goalAngle);
            goal.y = carPosition.y + distance * sin(goalAngle);

            pointsList[addedNavPoints].x = goal.x;
            pointsList[addedNavPoints].y = goal.y;

            addedNavPoints++;
            atTargetLocation = false;          //  have to reach this point before setting this flag true
            updateCurrentGoal();               //  update the goal variable with current goal
        }
    }
    void addGoalPoint( Vector pt ){            //  add a goal location to the goal list [x,y] form
        if(addedNavPoints < maximumPoints - 1 ){

            pointsList[addedNavPoints].x = pt.x;
            pointsList[addedNavPoints].y = pt.y;

            addedNavPoints++;
            atTargetLocation = false;          //  have to reach this point before setting this flag true
            updateCurrentGoal();               //  update the goal variable with current goal
        }
    }
```

```
void setController(int type){                    //  sets the controller for the robot

    if (type == 1){                              //  sets GoToAngle controller as the current controller

        currentController = &GTA;

    }

    else if (type == 2){                         //  sets GoToGoal controller as the current controller

        currentController = &GTG;

    }

    else if (type == 3){                         //  sets AvoidObstacle as the current controller

        currentController = &AO;

    }

    else if (type == 4){                         //  sets AvoidObstacle and GoToGoal as the current controller

        currentController = &AOandGTG;


    }

}

void printGraph(){

    Vector position = car->getPosition();

    Vector staticPosition = car->staticPosition;

    Vector filteredPosition = car->filteredPosition;


    float x=position.x;

    float y=position.y;
```

```
    Serial.print(filteredPosition.x);
    Serial.print(",");
    Serial.print(filteredPosition.y);
    Serial.print(",");
    Serial.print(car->getState(),DEC);
    Serial.print(",");
    Serial.print(goal.x);
    Serial.print(",");
    Serial.print(goal.y);
    Serial.print(",");
    Serial.print(ToDeg(car->filteredHeading));


    Serial.println();
}
void setGoalAngle( float angle ){
    goalAngle = angle;
}
void setState( bool status ){
    state = status;
}
bool getState(){
    return state;
```

```
}

void toggleLogOutput(){

    enableLogOutput = !enableLogOutput;

}

bool commandAvailable(){                    //  read a full line from serial input
as a single command

    if (Serial.available()){

        String cmd = "";

        delay(10);                          //  give time to receive message

        while(Serial.available()){

            char inByte = Serial.read();

            cmd += inByte;


        }

        command = cmd;

        Serial.print(" Rx Command = : ");

        Serial.println(command);

        return true;

    }

    return false;

}

void executeCommand(){                      //  checks for any commands at
the serial input and executes it

    if (!commandAvailable()){

         return;

    }
```

```
int startIndex = command.indexOf('*');
String sDistance,sDirection,sX,sY;
int angle,commandEnd,distance,direction,ptX,ptY;
if (startIndex >= 0){
  char cmd = command[startIndex + 1];

  switch(cmd){
  case 'f':
      car->moveForward();
      break;
  case 'b':
      car->moveBackward();
      break;
  case 's':
      car->stop();
      break;
  case 'a':          //  set steering angle
      command = command.substring(startIndex + 2);
      angle = command.toInt();
      car->setSteeringAngle( angle );
      Serial.print("Setting Steering angle: " );
      Serial.println(angle);

      break;
  case 'c':          //  set central steering position
```

```
        command = command.substring(startIndex + 2);
        angle = command.toInt();
        car->setCenterPosition( angle );
        Serial.print("Setting central position : " );
        Serial.println(angle);

        break;
    case 'v':           //  set ALPHA bleding value
        command = command.substring(startIndex + 2);
        char val[5];
        command.toCharArray(val,5);
        AOandGTG.setAlpha(atof(val));
        Serial.print("Alpha Blender : " );
        Serial.println(atof(val));
        break;
    case 'k':           //  set Static to Dynamic Filter Ratio
        command = command.substring(startIndex + 2);
        char val2[5];
        command.toCharArray(val2,5);
        car->StoDFilterRatio = (atof(val2));
        Serial.print("Static to Dynamic Filter Ratio : " );
        Serial.println(atof(val2));
        break;
    case 'g':             //   enable automatic mode
        setState(true);
```

```
        car->setState('f');
        Serial.println("Automatic mode Enabled");
        break;
    case 'm':
        setState(false);    //   disable automatic mode
        Serial.println("Automatic mode Disabled");
        break;
    case 'l':
        toggleLogOutput();  //   toggle log output
        break;
    case 'r':
        car->toggleRadarOutput();  //   toggle radar output
        break;
    case 'p':                  //  print all points
        printPointList();
        break;



    case 'n':                  //  add navigation point
        commandEnd    = command.lastIndexOf(',');
        sDistance  = command.substring(startIndex+3 ,commandEnd);
        sDirection = command.substring(commandEnd+1);
        distance = sDistance.toInt();
        direction = sDirection.toInt();
```

```
        Serial.print("Added navigation point : ");
        Serial.println(distance);
        Serial.println(direction);

        navPoint pt;
        pt.direction = ToRad(direction);
        pt.distance  = distance;

        addGoal(pt);
        break;
    case 'd':              //  add x,y point
        commandEnd    = command.lastIndexOf(',');
        sX = command.substring(startIndex+3 ,commandEnd);
        sY = command.substring(commandEnd+1);
        ptX = sX.toInt();
        ptY = sY.toInt();

        Serial.print("Added point : ");
        Serial.println(ptX);
        Serial.println(ptY);

        Vector point(ptX,ptY);

        addGoalPoint(point);
```

```
                break;
            }
        }
    }
};

#endif // SUPERVISOR_H
```

6. Project Analysis and Evaluation

a. Testing

To ensure quality of the product, testing is conducted. Accuracy and efficiency of tasks performed by our system had to be tested to analyze the system and verify and validate it. Software testing techniques and results obtained are discussed in the coming sections.

b. Testing Levels

Separate modules were developed to provide different functionalities of the system. All of these modules were tested at different levels during development and after integration. Different levels of testing and results have been described here:

(1) Unit Testing

Each module was designed, developed and tested individually. Each functionality was also tested separately. Detailed procedure of each test alongwith the expected and recieved results are presented below:

Test Case ID	1
Unit to Test	Mobile - Refresh status
Assumptions	1. GSM Network is avaiable 2. Stub Vehicle Controller is running
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	1. Drivers clicks send request button 2. Properly generated header is generated and sent to Vehicle Controller 3. Stub Vehicle Controller Replies the dummy status to Mobile Application 4. Mobile Application updates parking area graphical user interface
Expected Result	Graphical user interface is updated
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	2
Unit to Test	Mobile – Show Map
Assumptions	1. Internet connectivity is available to Mobile Application

	2. GPS coordinates successfully received from Vehicle Controller
Test Data	1. Dummy GPS coordinates
Steps to be Executed	1. Drivers clicks Show Map Button 2. New Android OS avtivity is started showing Google Maps 3. Location marker is placed at vehicl’s location
Expected Result	Marker is placed at exact location on Google Maps
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	3
Unit to Test	Mobile – Emergency Stop
Assumptions	1. GSM Network is avaiable 2. Stub Vehicle Controller is running
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	1. Drivers clicks emergency stop button

	2. Properly generated header is generated and sent to stub Vehicle Controller 3. Stub Vehicle Controller immediately halts the car
Expected Result	Vehicle is stopped
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	4
Unit to Test	Mobile – Register vehicle
Assumptions	1. GSM Network is avaiable 2. Stub Vehicle Controller is running
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	1. Drivers clicks register vehicle button 2. Properly generated header is generated and sent to stub Vehicle Controller containing user name, car registration and security pin

	3. Stub Vehicle Controller returns success message
Expected Result	Success message is shown on screen
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	5
Unit to Test	Mobile – De-register vehicle
Assumptions	1. GSM Network is avaiable 2. Stub Vehicle Controller is running
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	1. Drivers clicks de-register vehicle button 2. Properly generated header is generated and sent to stub Vehicle Controller 3. Stub Vehicle Controller returns success message
Expected Result	Success message is shown on screen
Actual Result	As Expected

Pass/Fail	Pass

Test Case ID	6
Unit to Test	Mobile – Park vehicle
Assumptions	1. GSM Network is avaiable 2. Stub Vehicle Controller is running
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	4. Drivers clicks any free slot button 5. Properly generated header is generated and sent to stub Vehicle Controller containing slot number for parking 6. Stub Vehicle Controller returns success message
Expected Result	Success message is shown on screen
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	7
Unit to Test	Mobile – Depart vehicle

Assumptions	1. GSM Network is avaiable 2. Stub Vehicle Controller is running
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	7. Drivers clicks own parked slot button 8. Properly generated header is generated and sent to stub Vehicle Controller containing slot number for parking 9. Stub Vehicle Controller returns success message
Expected Result	Success message is shown on screen
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	8
Unit to Test	Mobile – Depart vehicle
Assumptions	1. GSM Network is avaiable 2. Stub Vehicle Controller is running
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	1. Drivers clicks parked slot button, other than own slot

	2. Properly generated header is generated and sent to stub Vehicle Controller containing slot number for parking 3. Stub Vehicle Controller returns failure message
Expected Result	Failure message is shown on screen
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	9
Unit to Test	Vehicle Controller – Refresh Request
Assumptions	1. GSM Network is avaiable 2. WiFi network is connected 3. Stub Desktop Application is working 4. Stub Mobile Application is working
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	1. Parking request is received from stub Mobile Application

	2. Request is forwared to stub Desktop Application 3. Response received from stub Desktop Application 4. Response forwarded to stub Mobile Application
Expected Result	Message sent and received successfully
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	10
Unit to Test	Vehicle Controller – Halt request
Assumptions	5. GSM Network is avaiable 6. WiFi network is connected 7. Stub Desktop Application is working 8. Stub Mobile Application is working
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	5. Halt request is received from stub Mobile Application 6. Vehicle is stopped immediately

Expected Result	Vehicle stopped
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	11
Unit to Test	Vehicle Controller – Park request
Assumptions	9. GSM Network is avaiable 10. WiFi network is connected 11. Stub Desktop Application is working 12. Stub Mobile Application is working
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	7. Parking request is received from stub Mobile Application 8. Request is forwared to stub Desktop Application 9. Response received from stub Desktop Application 10. Response forwarded to stub Mobile Application
Expected Result	Vehicle starts parking procedure

Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	12
Unit to Test	Vehicle Controller – Depart request
Assumptions	13. GSM Network is avaiable 14. WiFi network is connected 15. Stub Desktop Application is working 16. Stub Mobile Application is working
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	11. Depart request is received from stub Mobile Application 12. Request is forwared to stub Desktop Application 13. Response received from stub Desktop Application 14. Response forwarded to stub Mobile Application
Expected Result	Vehicle starts departing procedure
Actual Result	As Expected

Pass/Fail	Pass

Test Case ID	13
Unit to Test	Vehicle Controller – Register request
Assumptions	17. GSM Network is avaiable 18. WiFi network is connected 19. Stub Desktop Application is working 20. Stub Mobile Application is working
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	15. Register request is received from stub Mobile Application 16. Request is forwared to stub Desktop Application 17. Response received from stub Desktop Application 18. Response forwarded to stub Mobile Application
Expected Result	User is registered
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	14
Unit to Test	Vehicle Controller – De-register request
Assumptions	21. GSM Network is avaiable 22. WiFi network is connected 23. Stub Desktop Application is working 24. Stub Mobile Application is working
Test Data	1. GSM Messaging protocol datagram
Steps to be Executed	19. De-register request is received from stub Mobile Application 20. Request is forwared to stub Desktop Application 21. Response received from stub Desktop Application 22. Response forwarded to stub Mobile Application
Expected Result	User is de-registered
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	15
Unit to Test	Vehicle Controller – Check modules
Assumptions	25. Battery is charged 26. Modules are turned on 27. Serial communication has been established with car
Test Data	1. Check module command has been given
Steps to be Executed	1. Vehicle Controller queries the following modules a. Global Positioning System b. Ultra sonic sensor c. Infrared sensor d. Hall encoder e. Steering servo f. Ethernet controller g. GSM Module h. Innertial measurement unit
Expected Result	All modules working flawlessly and no error is reported
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	16
Unit to Test	Vehicle Controller – Go to Goal
Assumptions	28. Battery is charged 29. Modules are turned on 30. Serial communication has been established with car
Test Data	1. Goal has been given through serial port
Steps to be Executed	2. Vehicle Controller calculates the direction vector to the goal 3. Go to goal behaviour calculates steering servo angle and speed to reach the goal 4. Vehicles starts moving towards the goal
Expected Result	Vehicle stops at the goal location
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	17

Unit to Test	Vehicle Controller – Go to Angle
Assumptions	31. Battery is charged 32. Modules are turned on 33. Serial communication has been established with car
Test Data	1. Angle has been given through serial port
Steps to be Executed	5. Go to angle behaviour calculates steering servo angle for the given input 6. Vehicles starts moving in the direction of given angle
Expected Result	Vehicle keeps moving in the given direction
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	18
Unit to Test	Vehicle Controller – Avoid obstacle
Assumptions	34. Battery is charged 35. Modules are turned on

	36. Serial communication has been established with car
Test Data	1. Move forward command is given through serial port
Steps to be Executed	7. Vehicle starts moving foward 8. Vehicle avoid obstacles en route
Expected Result	Vehicle doesn’t slam into objects
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	19
Unit to Test	Desktop Application – Refresh request received
Assumptions	37. Stub Vehicle Controller is functioning 38. WiFi connection has been established
Test Data	1. Properly formatted TCP / Ip datagram is received over wifi

Steps to be Executed	9. Desktop Application queries the updated status from database 10. Desktop Application returns the updated status to Stub Vehicle Controller
Expected Result	Message sent successfully
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	20
Unit to Test	Desktop Application – Parking request received
Assumptions	39. Stub Vehicle Controller is functioning 40. WiFi connection has been established
Test Data	2. Properly formatted TCP / Ip datagram is received over wifi
Steps to be Executed	11. Approved message is sent to the Vehicle Controller

	12. Desktop Application updates the parking slot status
Expected Result	Approved message is sent and database is updates successfully
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	21
Unit to Test	Desktop Application – Depart request received
Assumptions	41. Stub Vehicle Controller is functioning 42. WiFi connection has been established
Test Data	3. Properly formatted TCP / Ip datagram is received over wifi
Steps to be Executed	13. Approved message is sent to the Vehicle Controller 14. Desktop Application updates the parking slot status

Expected Result	Approved message is sent and database is updates successfully
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	22
Unit to Test	Desktop Application – Register request received
Assumptions	43. Stub Vehicle Controller is functioning 44. WiFi connection has been established
Test Data	4. Properly formatted TCP / Ip datagram is received over wifi
Steps to be Executed	15. Success message is sent to the Vehicle Controller 16. Desktop Application updates the user databse
Expected Result	success message is sent and database is updated successfully

Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	23
Unit to Test	Desktop Application – De-register request received
Assumptions	45. Stub Vehicle Controller is functioning 46. WiFi connection has been established
Test Data	5. Properly formatted TCP / Ip datagram is received over wifi
Steps to be Executed	17. Success message is sent to the Vehicle Controller 18. Desktop Application updates the user databse
Expected Result	success message is sent and database is updated successfully
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	24
Unit to Test	Desktop Application – Pause Button
Assumptions	47. Stub Vehicle Controller is functioning 48. WiFi connection has been established
Test Data	6. Pause button is clicked
Steps to be Executed	19. Pause message is sent to Stub Vehicle Controller 20. All ongoing operations are paused
Expected Result	Pause message is sent to Vehicle Controller
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	25
Unit to Test	Desktop Application – Disable Button
Assumptions	49. Stub Vehicle Controller is functioning 50. WiFi connection has been established

Test Data	7. Disable button is clicked
Steps to be Executed	21. Stop message is sent to Stub Vehicle Controller 22. Driverless Intelligent Parking System is disablled
Expected Result	Stop message is sent to Vehicle Controller
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	26
Unit to Test	Desktop Application – Enable Button
Assumptions	51. Stub Vehicle Controller is functioning 52. WiFi connection has been established
Test Data	8. Enable button is clicked
Steps to be Executed	23. Enable message is sent to Stub Vehicle Controller

	24. Driverless Intelligent Parking System is enabled
Expected Result	Stop message is sent to Vehicle Controller
Actual Result	As Expected
Pass/Fail	Pass

Test Case ID	27
Unit to Test	Desktop Application – TCP / IP Communication
Assumptions	53. Stub Vehicle Controller is functioning 54. WiFi connection has been established
Test Data	9. “Test” message is sent to Vehicle Controller
Steps to be Executed	25. Vehicle Controller should echo sent message
Expected Result	Message successfully echoed
Actual Result	As Expected

Pass/Fail	Pass

(2) **Integration Testing**

(a) All the stub controllers created for unit testing were replaced with actual applications and modules.

(b) All the expected results were confirmed with real testing and the results were successful.

(c) In the first phase, mobile application was interfaced successfully with the Vehicle controller

(d) In the next step, vehicle controller's functionality was successfully tested with desktop application

(3) **System Testing**

(a) System testing was performed at the end of development. All the functional requirements were verified and whole system was analyzed for performance and other attributes (failures, response delays, connection losses etc).

c. Results

The results of the tests were in the acceptable range. Detailed data is provided below.

Parking Requests

Number of Requests	Response Time (ms)	Time Limits (ms)	Difference (ms)	Inaccuracy (m)
10	1987	5000	> 3000	0.1

De-Parting Requests

Number of Requests	Response Time (ms)	Time Limits (ms)	Difference (ms)	Inaccuracy (m)
10	1750	5000	> 3000	0.08

d. Analysis

The results were very encouraging and reported errors were well within acceptable range. DIPS is an idea under development therefore there is a room for further improvements and updates to the system. As a whole, the system is fully functional and reliable.

7. Conclusion and Future Work

The goal of this project was to give a proof of concept which is revolutionary in nature. Technology is advancing at a very rapid pace and robots are taking over many of the laborious tasks which were once performed by human beings. Modern cars are now controlled by computer i.e. Electronic Control Unit(ECU) and provide great relief to drivers by taking over few of the control parameters for example “Cruise Control” where the onboard controller maintains the car at a specific speed.

Semi automatic parking has been a hallmark of Toyota Prius model for quite a few years now. It takes over the steering control from the driver as well as the accelerator. It allows for parallel parking in congested areas. DIPS on the other hand is a completely automatic process of parking cars. Google has been involved in development of Autonomous Driverless Cars for past few years. They have successfully demonstrated working of the cars on various occasions.

There is a strong need for sponsored funding for such research oriented projects as the cost involved in working with real cars is very high. DIPS suffered from lack of sufficient budget therefore it was developed on small scale cars. In future, such sponsorships will provide a launchpad for development of DIPS on real cars which require a high level of accuracy and reliability. We believe that Military College of Signals will carry this project further in the coming years and it will be refined and implemented on real cars.

Appendix A: Glossary

DIPS

Driverless Intelligent Parking System

I2C

I^2C (Inter-Integrated Circuit, referred to as I-squared-C, I-two-C, or IIC) is a multimaster serial single-ended computer bus invented by Philips used for attaching low-speed peripherals to a motherboard, embedded system, cellphone, or other electronic device. Not to be confused with the term Two Wire Interface which only describes a compatible hardware interface

UART

A Universal Asynchronous Receiver/Transmitter, abbreviated UART , is a piece of computer hardware that translates data between parallel and serial forms. UARTs are commonly used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485. The universal designation indicates that the data format and transmission speeds are configurable. The electric signaling levels and methods (such as differential signaling etc.) are handled by a driver circuit external to the UART.

TTL

Transistor–transistor logic (TTL) is a class of digital circuits built from bipolar junction transistors (BJT) and resistors. It is called transistor–transistor logic because both the logic gating function (e.g., AND) and the amplifying function are performed by transistors (contrast with RTL and DTL).

ICSP

Micro-controllers are typically soldered directly to a printed circuit board and usually do not have the circuitry or space for a large external programming cable to another computer. A separate piece of hardware, called a programmer is required to connect to an I/O port of a PC on one side and to the PIC on the other side. The type of programmer, how it connects to the PC, and the various advantages and disadvantages of each are not within the scope of this document. However, a short list of the features for each major programming type is given here.

1. **Parallel port** - large bulky cable, most computers have only one port and it may be inconvenient to swap the programming cable with an attached printer. Most laptops newer than 2010 do not support this port. Parallel port programming is very fast.
2. **Serial port** (COM port) - At one time the most popular method. Serial ports usually lack adequate circuit programming supply voltage. Most computers and laptops newer than 2010 lack support for this port.
3. **Socket** (in or out of circuit) - the CPU must be either removed from circuit board, or a clamp must be attached to the chip making access an issue.
4. **USB cable** - Small and light weight, has support for voltage source and most computers have extra ports available. The distance between the circuit to be programmed and the computer is limited by the length of USB cable - it must usually be less than 180 cm. This can make programming devices deep in machinery or cabinets a problem.

ICSP programmers have many advantages, with size, computer port availability, and power source being major features. Due to variations in the interconnect scheme and the target circuit surrounding a micro-controller, there is no programmer that works with ***all*** possible target circuits or interconnects.

DBW

Drive-by-wire, DBW, by-wire, or x-by-wire technology in the automotive industry replaces the traditional mechanical control systems with electronic control systems using electromechanical actuators and human-machine interfaces such as pedal and steering feel emulators. Hence, the traditional components such as the steering column, intermediate shafts, pumps, hoses, belts, coolers and vacuum servos and master cylinders are eliminated from the vehicle. Examples include electronic throttle control and brake-by-wire.

GPS

The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The system provides critical capabilities to military, civil and commercial users around the world. It is maintained by the United States government and is freely accessible to anyone with a GPS receiver.

GSM

GSM (Global System for Mobile Communications, originally Groupe Spécial Mobile), is a standard set developed by the European Telecommunications Standards Institute (ETSI) to describe protocols for second generation (2G) digital cellular networks used by mobile phones. It became the de facto global standard for mobile communications with over 80% market share.

IMU

An inertial measurement unit, or IMU, is an electronic device that measures and reports on a craft's velocity, orientation, and gravitational forces, using a combination of accelerometers and gyroscopes, sometimes also magnetometers. IMUs are typically used to maneuver aircraft, including unmanned aerial vehicles (UAVs), among many others, and spacecraft, including satellites and landers. Recent developments allow for the production of IMU-enabled GPS devices. An IMU allows a GPS to work when GPS-signals are unavailable, such as in tunnels, inside buildings, or when electronic interference is present. A wireless IMU is known as a WIMU.

TEA

In cryptography, the Tiny Encryption Algorithm (TEA) is a block cipher notable for its simplicity of description and implementation, typically a few lines of code. It was designed by David Wheeler and Roger Needham of the Cambridge Computer Laboratory; it was first presented at the Fast Software Encryption workshop in Leuven in 1994, and first published in the proceedings of that workshop.

Appendix B: References

- S.-Y. Cheung, S. Coleri Ergen and P. Varaiya. Traffic surveillance with wireless magnetic sensors. In 12th ITS World Congress, Nov. 2005.
- "Four Wheels On Jacks Park Car", Popular Science, September 1934.
- http://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/self-parking-car.htm
- Paromtchik, Igor; Laugier, Christian (1996). "Autonomous Parallel Parking of a Nonholonomic Vehicle", Proceedings of the IEEE Intelligent Vehicles Symposium, Tokyo, Japan, September 1996, pp. 13-18.
- Paromtchik, Igor; Laugier, Christian (1998). "Automatic Parallel Parking and Returning to Traffic", Video Proceedings of the IEEE International Conference on Robotics and Automation, Belgium, May 1998.
- Paromtchik, Igor (2004). "Steering and Velocity Commands for Parking Assistance", Proceedings of the 10th IASTED Conference on Robotics and Applications, USA, August 2004, pp. 178-183.
- http://web.archive.org/web/20070505060924/http://www.modbee.com/life/wheels/story/8015603p-8880060c.html
- http://www.theautochannel.com/news/2006/07/12/014519.html?title=BMW
- http://www.valeo.com/en/home/the-group/business-groups/comfort-and-driving-assistance-systems.html?0=
- http://www.mobility.siemens.com/mobility/global/en/urban-mobility/road-solutions/parking-space-management/pages/parking-space-management.aspx
- http://delcantechnologies.com/technologies/intelligent-parking/
- http://edition.cnn.com/2003/TECH/ptech/09/01/toyota.prius.reut/index.html

- http://news.bbc.co.uk/2/hi/technology/3198619.stm
- http://delcantechnologies.com/technologies/intelligent-parking/
- http://www.mobility.siemens.com/mobility/global/en/urban-mobility/road-solutions/parking-space-management/pages/parking-space-management.aspx
- http://www.ece.nus.edu.sg/research/achieve_list.html
- J. P. Lynch, K. Loh. A Summary Review of Wireless Sensors and Sensor Networks for Structural Health Monitoring. Shock and Vibration Digest, Sage Publications, 38(2):91-128, 2005.