GENESIS-Learning Outcome & Mini-project Summary Report

Details

Ver. Rel. No.	Release Date	Prepared By	Reviewed By	To Be Approved	Remarks/Revision Details
1.0	16/02/2022	ABHISHEK BAHUGUNA 40021026			

Contents

List of Figures

Mini project – 1: Department Store Management System [Individual]

Modules: C Programming On Multiple Platforms

Requirements

Introduction

Department Store Management

The project as the name suggests is based on the departmental store management system. Departmental store consist commodities of daily human use, so they need to be managed in various ways. The program mainly focuses on manipulation of various functions by a cashier and manager on the basis of goods selected by the costumers.

Objective:-

The main objectives of this project are listed as follows:

- Proper management of selling items in departmental store.
- Can be useful for both manager and cashier.
- Good security of information for which password facilities are provided.
- Maximum providence of modification and search options.
- User friendly messages in the way for easy application of program.

SWOT Analysis

a) Strengths:

Strengths are things that your system does particularly well, or in a way that distinguishes you from your competitors

b) Weaknesses

Weaknesses, like strengths, are inherent features of your projects, so focus on your people, resources, systems, and procedures.

c) Opportunity

They usually arise from situations outside your organization, and require an eye to what might happen in the future.

d) Threats

Threats include anything that can negatively affect your business from the outside, such as supply-chain problems, shifts in market requirements, or a shortage of recruits.

4W's and 1 H's

Why:

I am developing this basic management system to perform basic operation in best easy manner and improve my coding skills.

Where:

It has to be used easily by the users.

What:

The main components of the program are to start as a manager or cashier and verification, goods entry and display, bill display, total sales display, etc.

When:

It has to be deployed 25th of November 2021.

How:

I am using C programming language for Developing this simple basic store management system.

High Level Requirements

ID	Description
HLR1	This Department Store Management System is primarily based totally on an idea to offer facts on calculating, adding, viewing goods, and different capabilities too...

ID	Description
HLR2	Provides the searching facilities based on various factors. Such as Products, Payments, Discounts, Stock.
HLR3	Department Store also manage the Inventory details online for Discounts details, Stock details, Products.

Low Level Requirements

ID	Description
LLR_1	• Create the master and slave database structure to reduce the overload of the database queries
LLR_2	• We will host the platform on online servers to make it accessible worldwide.
LLR_3	• Implement the backup mechanism for taking backup of codebase and database on regular basis on different servers.
LLR_4	• If I consider commercial field it will provide me the different Result.

Design

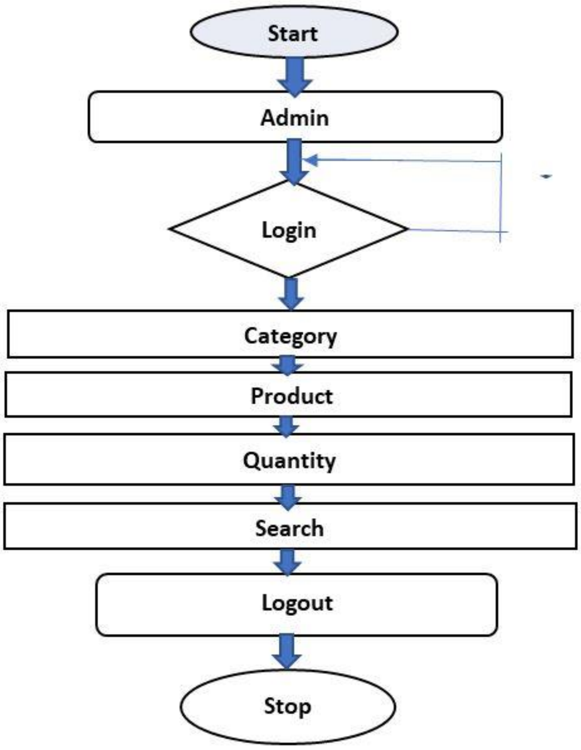


Figure 1 Behaviour Diagram

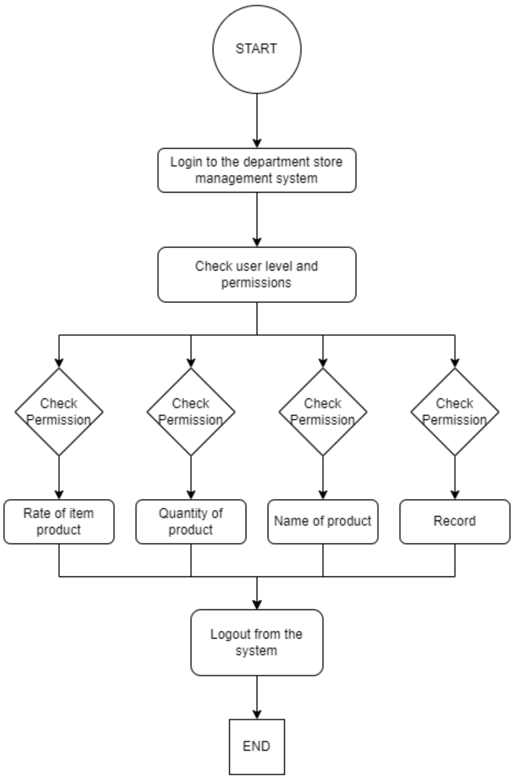
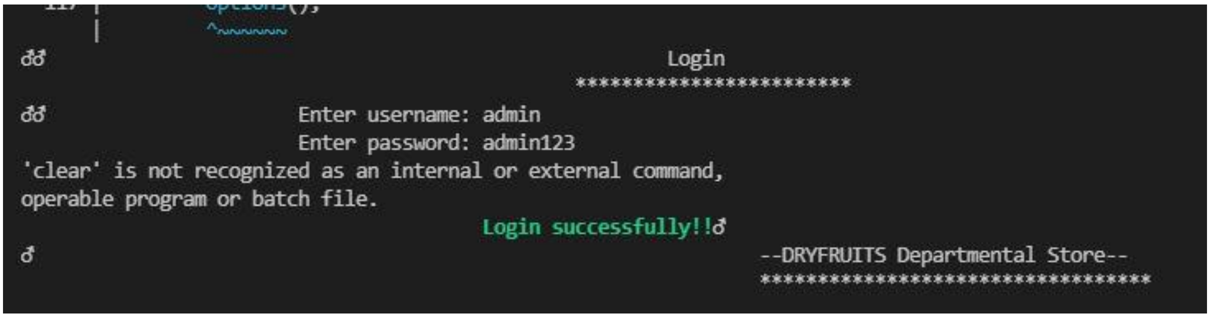


Figure 2 Flow Chart

Test Plan

Test ID	Description	Exp I/P	Exp O/P	Actual Out	Type Of Test
H_01	Enter a valid username & password	Username: user Password: pass	login Successfully	login Successfully	Requirement Based
H_02	Enter invalid username & password	Username: user Password: pass	Login Failed Enter Again Username & Password	Login Failed Enter Again Username & Password	Requirement Based
H_03	User should be able to Login with valid credentials	Username: user Password: pass	Login Failed Enter Again Username & Password	Login Successfully	Scenario Based

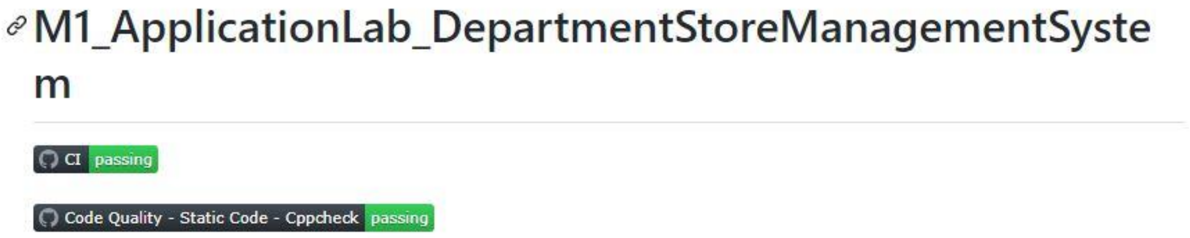
Test Case



```
  117 |         options();
      |         ^~~~~~~
ðð                                                     Login
                                            ************************

ðð                 Enter username: admin
                   Enter password: admin123
'clear' is not recognized as an internal or external command,
operable program or batch file.
                                  Login successfully!!ð
ð                                                                --DRYFRUITS Departmental Store--
                                                                 **********************************
```

Git Dashboard



M1_ApplicationLab_DepartmentStoreManagementSystem

CI passing

Code Quality - Static Code - Cppcheck passing

Figure 3 Git Dashboard

Summary

We can conclude that the project has been successful for what it aims to perform. This program under the project can be used in a departmental store for the proper management of purchase and sales of the goods along with the billing management. Hence although the project got some of its limitations the maximum effort was targeted for the gain of its aim i.e. departmental store management system.

CERTIFICATION DONE IN MODULE

- SOLO-Learn Certification
- Linux Certification
- Github Learning Certification

Mini project 2 – Ultrasonic Sound Sensor with Atmega328 Microprocessor [Individual]

Module: - Essentials of Embedded System

Topic: - ULTRASONIC SOUND SENSOR WITH ATmega328 MICROPROCESSOR

Requirements

Introduction

The project as the name suggests is based on Ultrasonic sensors. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.

Features, Hardware and Software:-

a) HARDWARE:-

1] SimulIDE:

- SimulIDE provides AVR, Arduino and PIC microcontrollers that can be accessed just like other components.
- Features like gypsum and simavr allow you to use PIC and AVR microcontrollers, respectively.

2] AVR:

- An automatic voltage regulator (AVR) is an electronic device that maintains a constant voltage level to electrical equipment on the same load.
- The AVR regulates voltage variations to deliver constant, reliable power supply.

b) SOFTWARE:-

1] ATmega328:

- ATmega328 is commonly used in many projects and autonomous systems where a simple, low-powered, low-cost micro-controller is needed
- Perhaps the most common implementation of this chip is on the popular Arduino development platform.

2] Sound:

- A sound sensor is defined as a module that detects sound waves through its intensity and converting it to electrical signals.

3] Display:

- A display device is an output device for presentation of information in visual or tactile form.

SWOT ANALYSIS:-

d) Strength:

The distance to an obstacle can be measured with the low cost ultrasonic sensor. The sensors can measure distances from 2 to 400cm with an accuracy of 3mm. This sensors module includes ultrasonic transmitter, ultrasonic receiver and control circuit.

b) Weakness:

Although we fully believe in the capability of our sensors, we understand that ultrasonic are not suited for every application. Focuses of low thickness, similar to froth and fabric, have a tendency to assimilate sound vitality; these materials may be hard to sense at long range.

c) Opportunity:

This project can be used as parking assistance systems in vehicles with high power ultrasonic transmitter. This Project Can be used as burglar alarm with suitable additional software for homes and offices.

d) Threats:

Ultrasonic sensors must view a surface (particularly a hard, level surface) unequivocally (oppositely) to get adequate sound reverberation. Additionally, solid detecting requires a base target surface range, which is indicated for every sensor sort. If connection is wrong there might be chances of short-circuit.

4W's a 1H:-

- **What:**

We have made a setup based on a microcontroller in which real time distance is sensed by an ultrasonic sensor and displays measured distance on an LCD display.

- **Where:**

It measures accurate distance using a non-contact technology - A technology that involves no physical contact between sensor and object.

-3 When: In 1959, Satomura created an ultrasonic flow meter that used Doppler technology.

-# Why: I am Developing this project for easily measure the distance between objects

- **How:**

By using Atmega328 and display an ultrasonic sensor mainly used to determine the distance of the target object.

High Level Requirements

ID	Description
HLR1	Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
HLR2	Used to measure the distance within a wide range of 2cm to 400cm
HLR3	Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water

Low Level Requirements

ID	Description
LLR_1	• Power Supply: +5V DC.
LLR_2	• Measuring Angle: 30 degree.
LLR_3	• Trigger Input Pulse width: 10uS TTL pulse.
LLR_4	• Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water.

Design

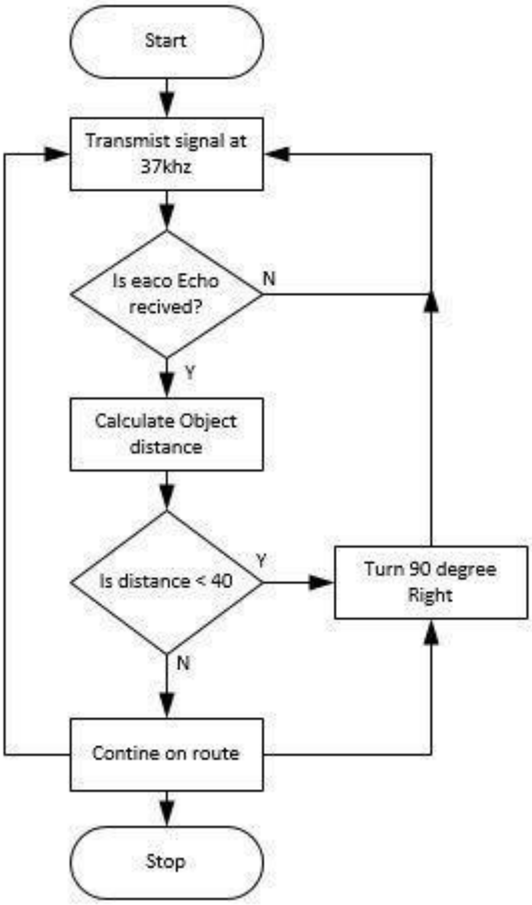


Figure 4 Behaviour Diagram

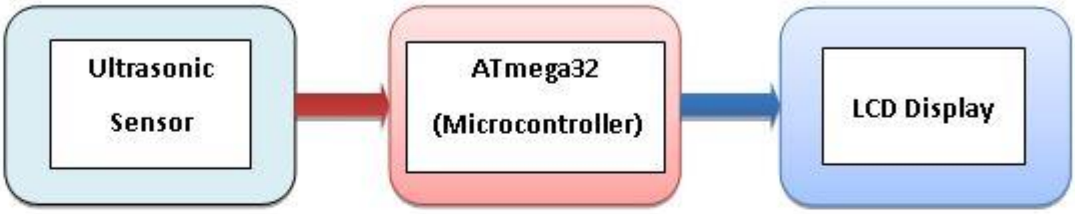
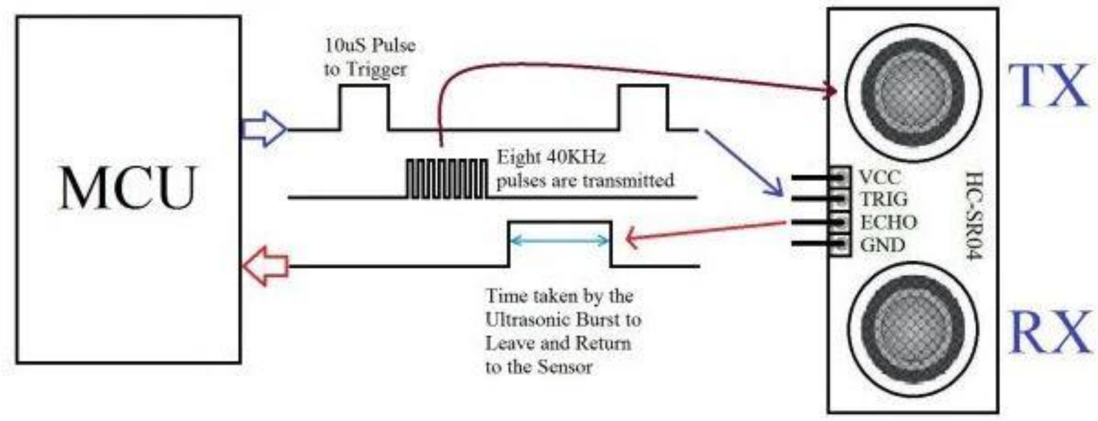


Figure 5 Block Diagram


MCU
10uS Pulse to Trigger
Eight 40KHz pulses are transmitted
Time taken by the Ultrasonic Burst to Leave and Return to the Sensor
VCC
TRIG
ECHO
GND
HC-SR04
TX
RX

Working of HC-SR04 Ultrasonic Sensor

Figure 7 Structural Diagram

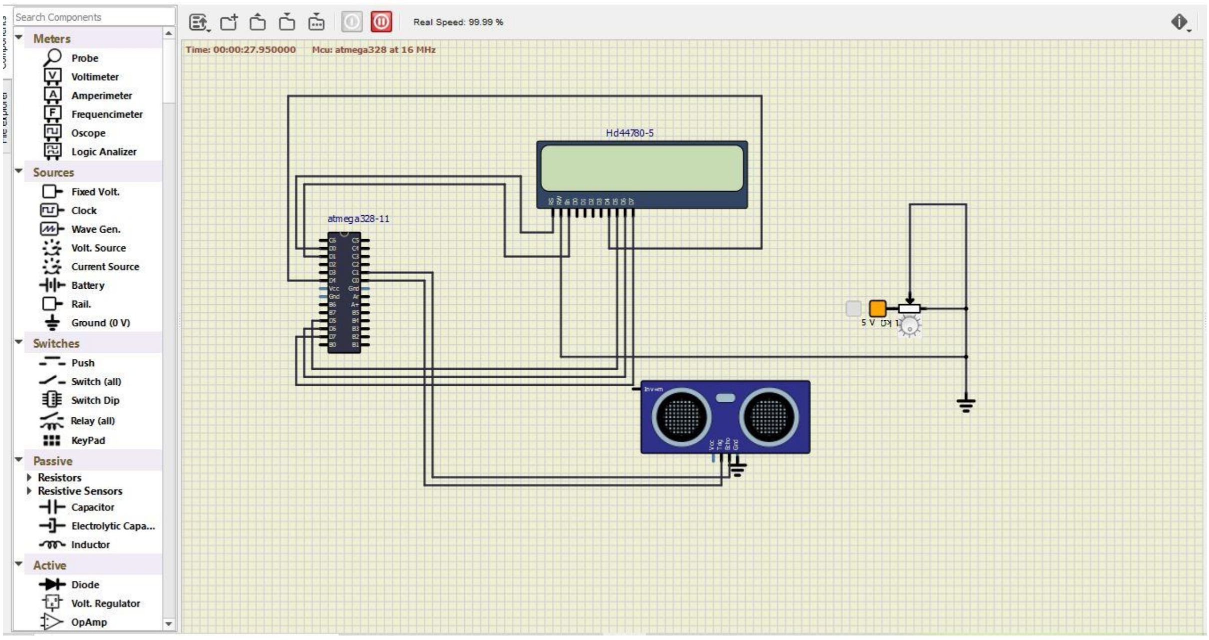


Figure 8 Simulation

Test Plan

Obstacle Detection:

How: Our implementation for this step requires multiple steps:

Step 1: Find a distance value between each pair of sensors. To test the distance

Value, we may use the numbers we see for the height and length, as well as the Pythagorean Theorem. ####Step 2: Check the angle found between each pair of sensors using the distance value initially found. ####Step 3: Using these values, determine what each angle should approximately be to detect different types of obstacles. ####Step 4: Detect the obstacles.

Output: As we had steps for each test, we will again make steps for the expected outputs:

Step 1: Compare the outputted (through serial) value for the hypotenuse to the

Pythagorean calculated value. We expect them to be the same.

Step 2: Using the same technique as step 1 except calculating the angle, we should

See the same value for this calculation as well.

Step 3: The values and outputs for the "obstacle detected" will be constantly

Checked and rechecked to make sure the angles determine the correct obstacle.

Step4: Adding Audio to the Ultrasonic Sensors.

Testing cases

Average Speed(m/s)	**0.8**	**1.5**	**2.0**
Mean RMS error (cm)*	19.4	12.7	10.2
SD**	11.2	14.3	13.4
Sensing error (%)	5.0	1.6	1.0

RMS error: Root mean square error between actual and sensing distance.

SD**: Standard deviation of the RMS errors.

Summary

The objective of the project was to design and implement an ultrasonic distance meter. The device described here can detect the target and calculate the distance of the target. The ultrasonic distance meter is a low cost, low a simple device for distance measurement. The device calculates the distance with suitable accuracy and resolution. It is a handy system for non-contact measurement of distance. The device has its application in many fields. It can be used in car backing system, automation and robotics, detecting the depth of the snow, water level of the tank, production line. This device will also have its application in civil and mechanical field for precise and small measurements. For calculating the distance using this device, the target whose distance is to be measured should always be perpendicular to the plane of propagation of the ultrasonic waves. Hence the orientation of the target is a limitation of this system. The ultrasonic detection range also depends on the size and position of the target. The bigger is the target, stronger will be the reflected signal and more accurate will be the distance calculated. Hence the ultrasonic distance meter is an extremely useful device.

Git Dashboard

Figure 6 Git Dashboard

Mini project 3 – PARKING MANAGEMENT SYSTEM

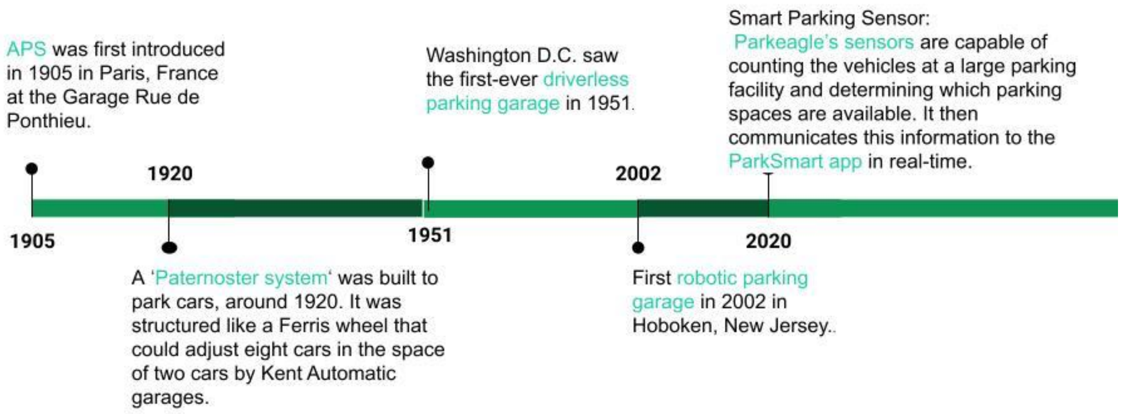
Modules: - Applied SDLC and Software Testing

Requirements

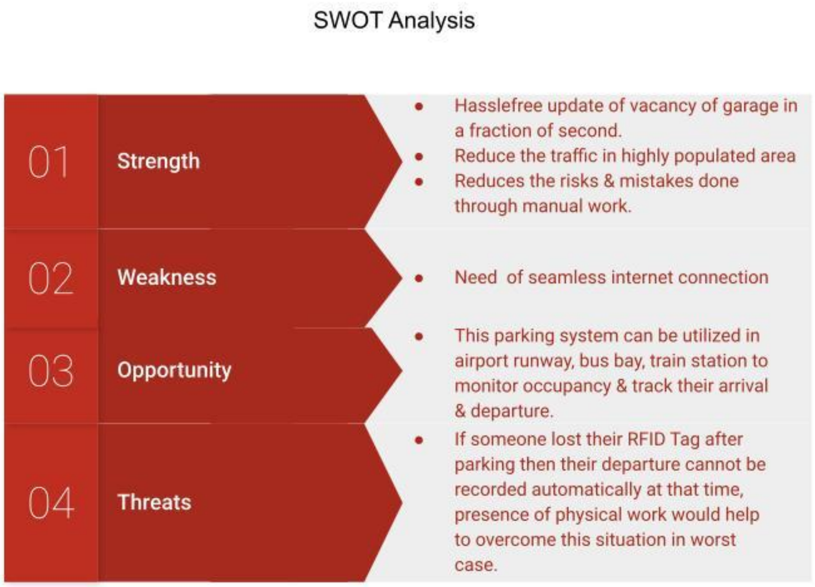
Introduction:

This project is about Parking Management System. This system provides vacant parking slots up to date in the vicinity and reduces the traffic issues due to illegal parking along the roads. Car Parking Reservation System is based on a concept to generate and maintain parking details with their total charge. Before stepping into the main system a user has to pass through a login system to get access, then only he/she can use all the features of the system which includes maintaining arrival car, view all cars, parking charges, and another one is Departure of the car. Parking control system has been generated in such a way that it is filled with many secure devices such as, parking control gates, toll gates, time and attendance machine, car counting system etc. These features are hereby very necessary nowadays to secure your car and also to evaluate the fee structure for every vehicles entry and exit.

HISTORY OF PARKING MANAGEMENT SYSTEM


APS was first introduced in 1905 in Paris, France at the Garage Rue de Ponthieu.
1905
1920
A 'Paternoster system' was built to park cars, around 1920. It was structured like a Ferris wheel that could adjust eight cars in the space of two cars by Kent Automatic garages.
Washington D.C. saw the first-ever driverless parking garage in 1951.
1951
2002
First robotic parking garage in 2002 in Hoboken, New Jersey..
Smart Parking Sensor: Parkeagle's sensors are capable of counting the vehicles at a large parking facility and determining which parking spaces are available. It then communicates this information to the ParkSmart app in real-time.
2020

SWOT analysis:



SWOT Analysis

01	Strength	• Hasslefree update of vacancy of garage in a fraction of second. • Reduce the traffic in highly populated area • Reduces the risks & mistakes done through manual work.
02	Weakness	• Need of seamless internet connection
03	Opportunity	• This parking system can be utilized in airport runway, bus bay, train station to monitor occupancy & track their arrival & departure.
04	Threats	• If someone lost their RFID Tag after parking then their departure cannot be recorded automatically at that time, presence of physical work would help to overcome this situation in worst case.

Cost estimation:

Estimated cost = hours needed to build an app * cost per hour to build an app.

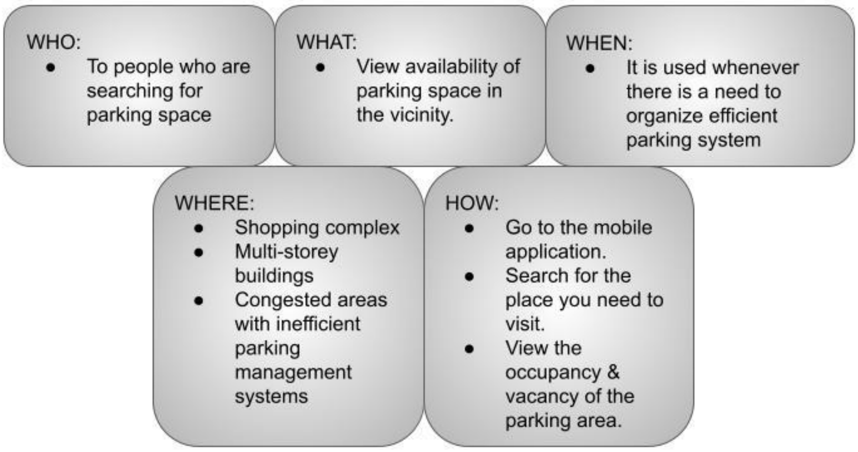
Features

- Logging details about the vehicle.
- View the available space in the garage.
- Display arrival and departure of the vehicle.

Features to be added in future

- Integrate GPS routing system with this mobile application to guide the users to vacant position.
- Ability to reserve their space in the garage from mobile application.
- Integrate RFID tag operation with the mobile application for seamless payment facility and update the vehicle occupation within blink of an eye. (fully automated parking system).

4 W's & 1 H



WHO:

- To people who are searching for parking space

WHAT:

- View availability of parking space in the vicinity.

WHEN:

- It is used whenever there is a need to organize efficient parking system

WHERE:

- Shopping complex
- Multi-storey buildings
- Congested areas with inefficient parking management systems

HOW:

- Go to the mobile application.
- Search for the place you need to visit.
- View the occupancy & vacancy of the parking area.

High Level Requirements

ID	Description	Status
HLR01	Log vehicle details	Implemented
HLR02	Update Parking status in app	Implemented
HLR03	Integrate payment portal using RFID Tag	Future
HLR04	Guide to vacant space with Routing sensor	Future
HLR05	Update arriving status	Implemented
HLR06	Update departure status	Implemented
HLR07	Verify the data details	Implemented
HLR08	Register for the Service details	Implemented

ID	Description	Status
HLR09	View the details Selected parking area	Implemented
HLR10	Generate the Session ID each reservation	Implemented
HLR11	Modify the parking status	Implemented

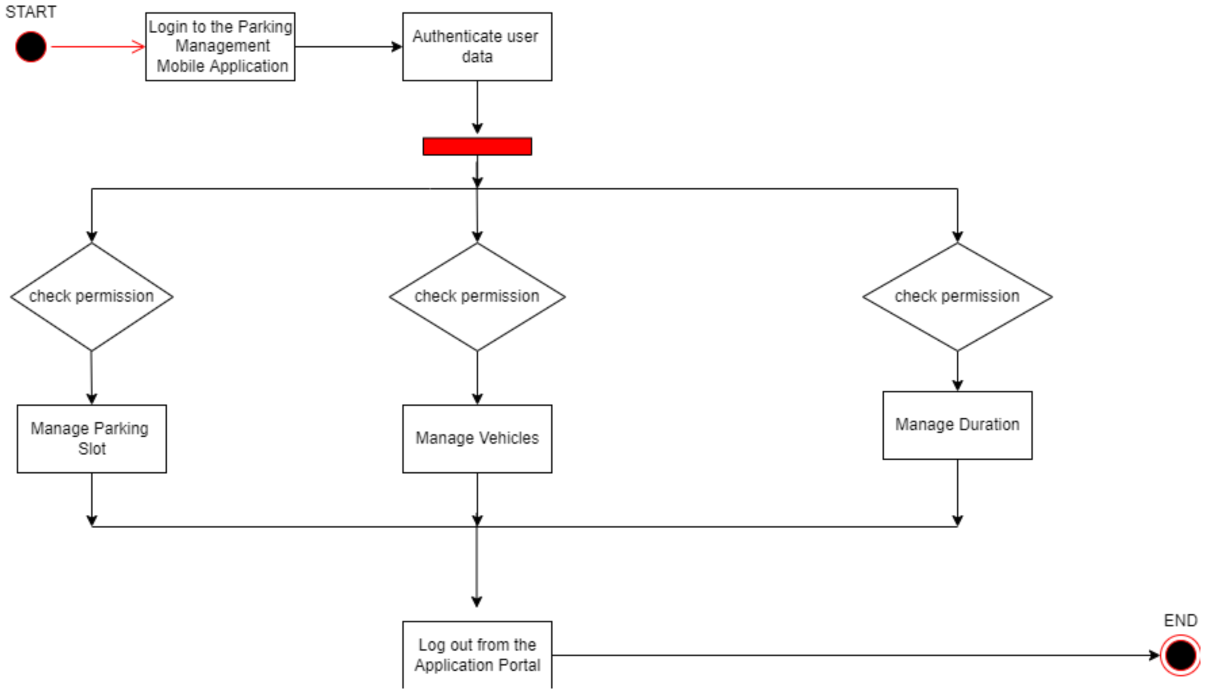
Low Level Requirements

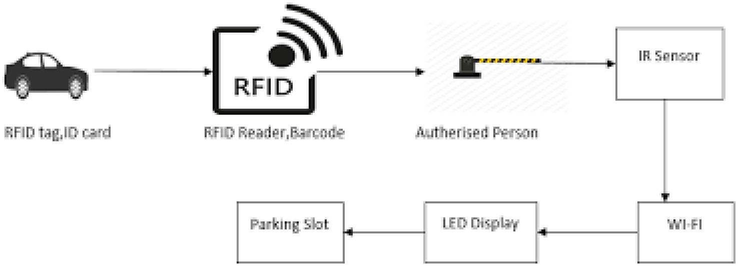
ID	Description	Status
LLR01	Open the app	Implemented
LLR02	View the parking status	Implemented
LLR03	Connect to GPS	Future
LLR04	Enter the vehicle number	Implemented
LLR05	Reserve your space	Implemented
LLR06	Enter the vehicle type (car or scooty)	Implemented
LLR07	Update arriving status	Implemented
LLR08	Update departure status	Implemented
LLR09	Calculate the bill	Implemented
LLR10	Exit the application	Implemented

Design

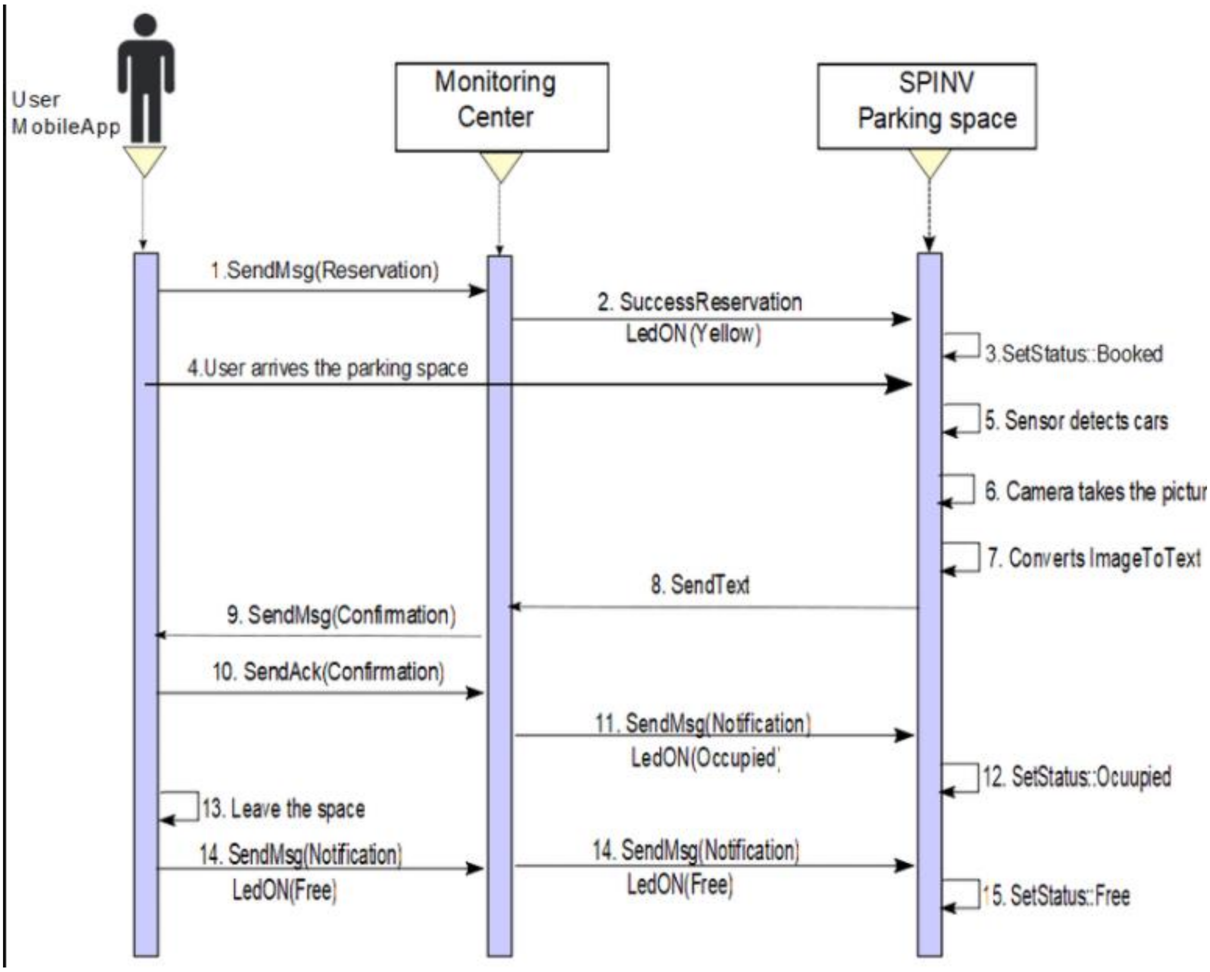
BEHAVIORAL DIAGRAM:

Activity Diagram

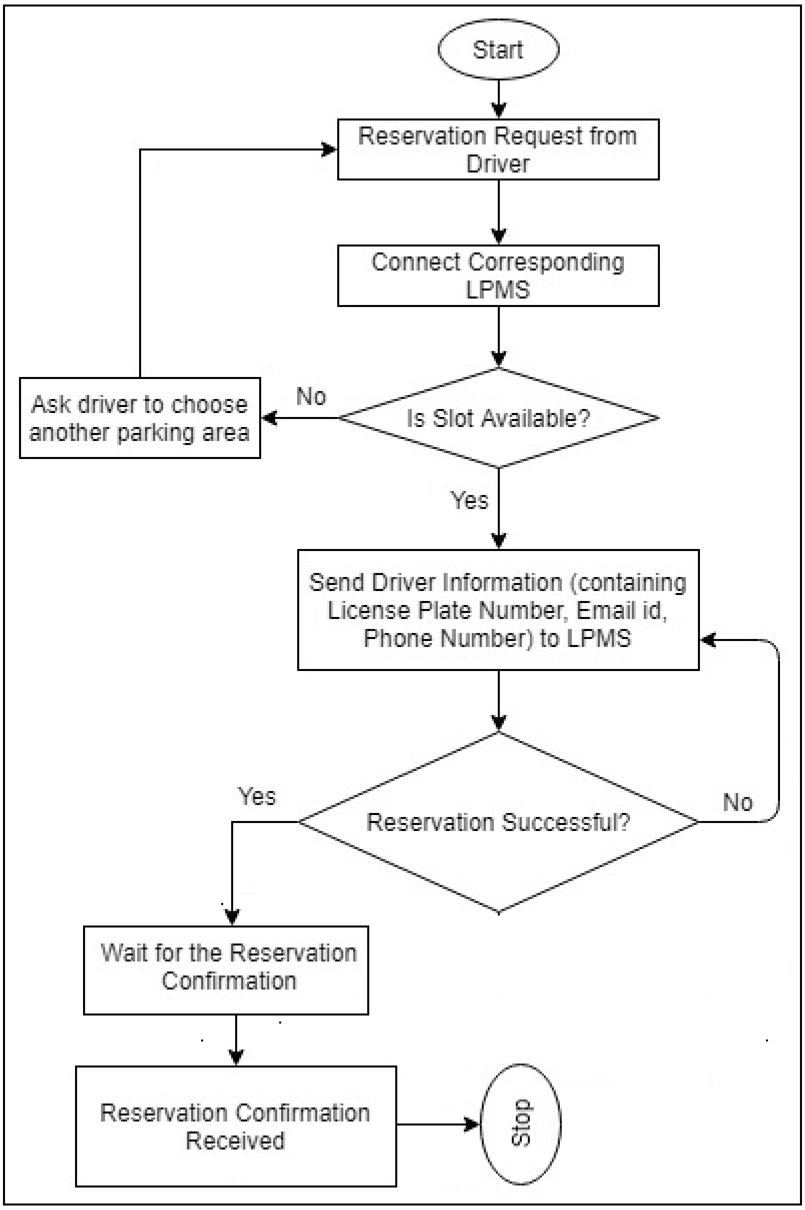




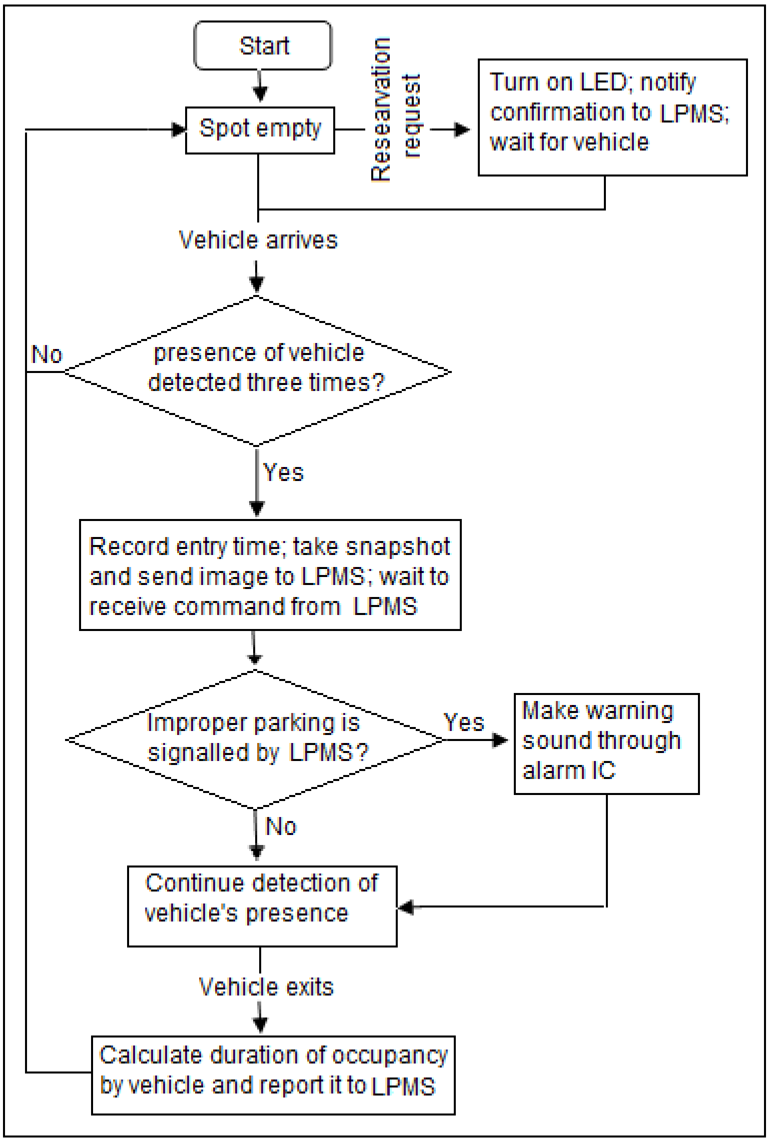
Sequence Diagram:


User
MobileApp
Monitoring Center
SPINV Parking space
1.SendMsg(Reservation)
2. SuccessReservation LedON(Yellow)
3.SetStatus::Booked
4.User arrives the parking space
5. Sensor detects cars
6. Camera takes the pictur
7. Converts ImageToText
8. SendText
9. SendMsg(Confirmation)
10. SendAck(Confirmation)
11. SendMsg(Notification) LedON(Occupied)
12. SetStatus::Ocuupied
13. Leave the space
14. SendMsg(Notification) LedON(Free)
14. SendMsg(Notification) LedON(Free)
15. SetStatus::Free

Flow chart 1:

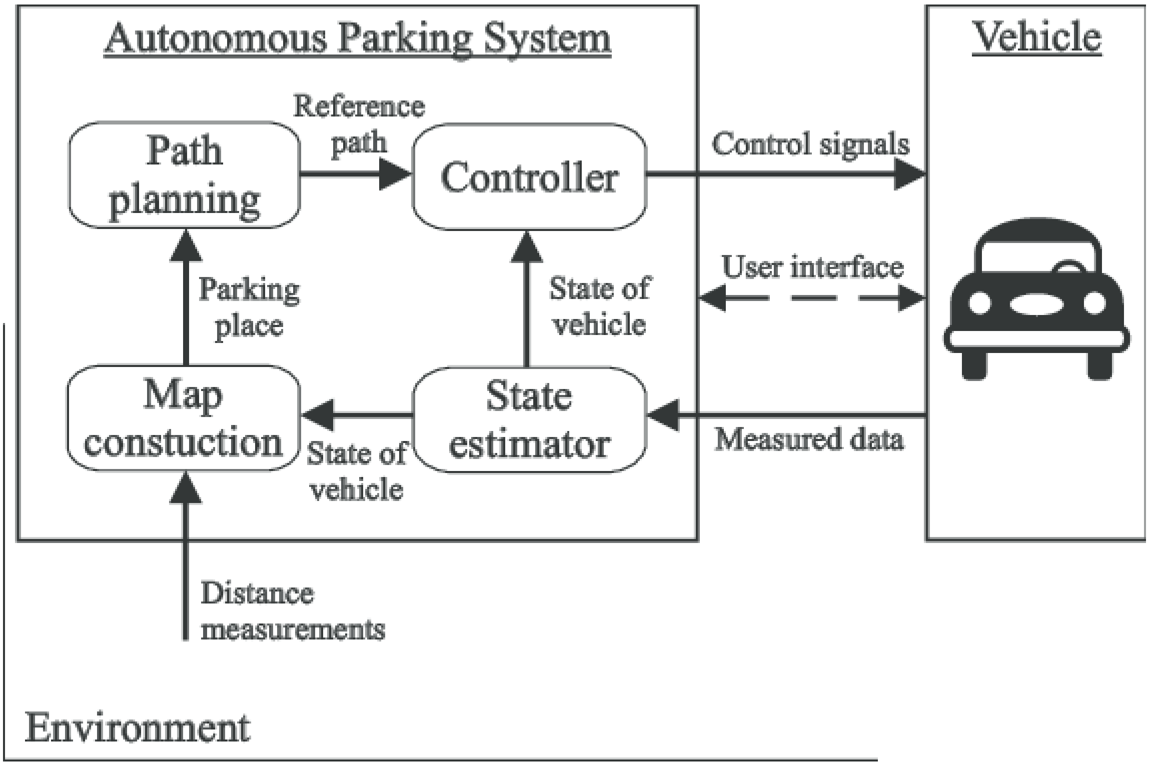

Start
Reservation Request from Driver
Connect Corresponding LPMS
Is Slot Available?
No
Ask driver to choose another parking area
Yes
Send Driver Information (containing License Plate Number, Email id, Phone Number) to LPMS
Reservation Successful?
Yes
No
Wait for the Reservation Confirmation
Reservation Confirmation Received
Stop

Flow chart: 2

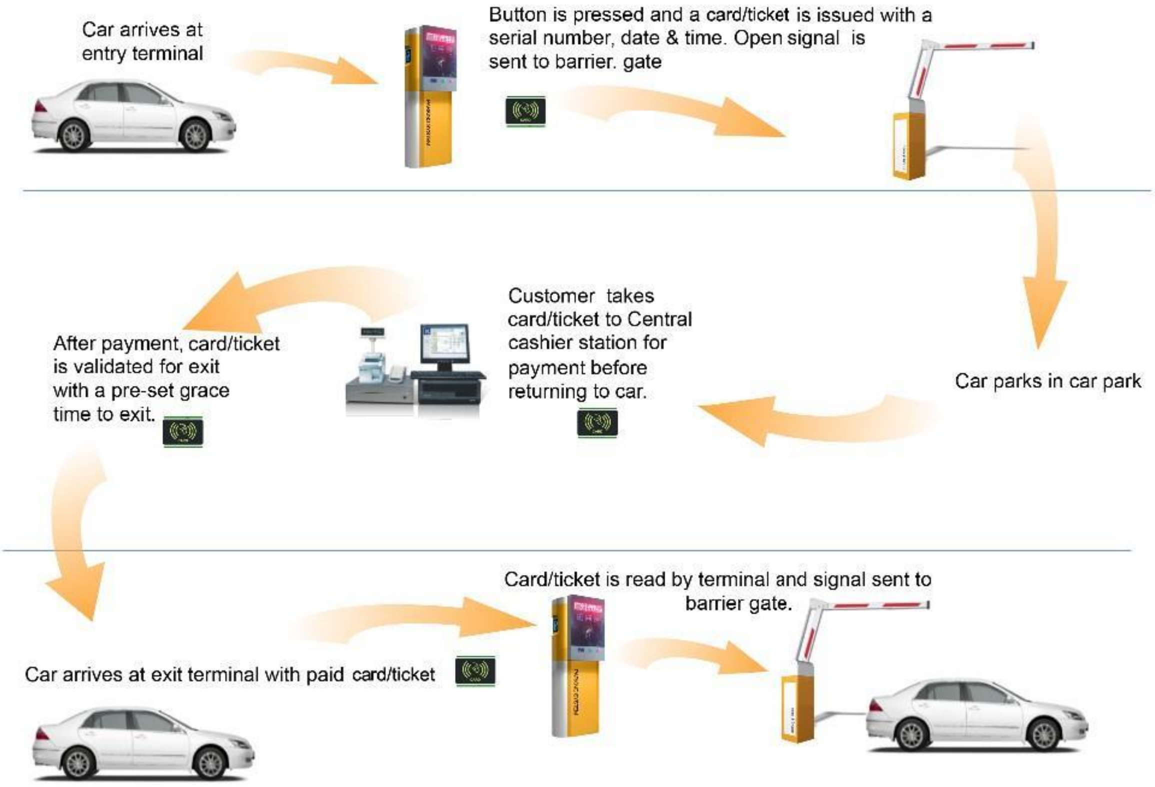

Start
Spot empty
Researvation request
Turn on LED; notify confirmation to LPMS; wait for vehicle
Vehicle arrives
presence of vehicle detected three times?
No
Yes
Record entry time; take snapshot and send image to LPMS; wait to receive command from LPMS
Improper parking is signalled by LPMS?
Yes
Make warning sound through alarm IC
No
Continue detection of vehicle's presence
Vehicle exits
Calculate duration of occupancy by vehicle and report it to LPMS

Structural Diagram

1) Component diagram

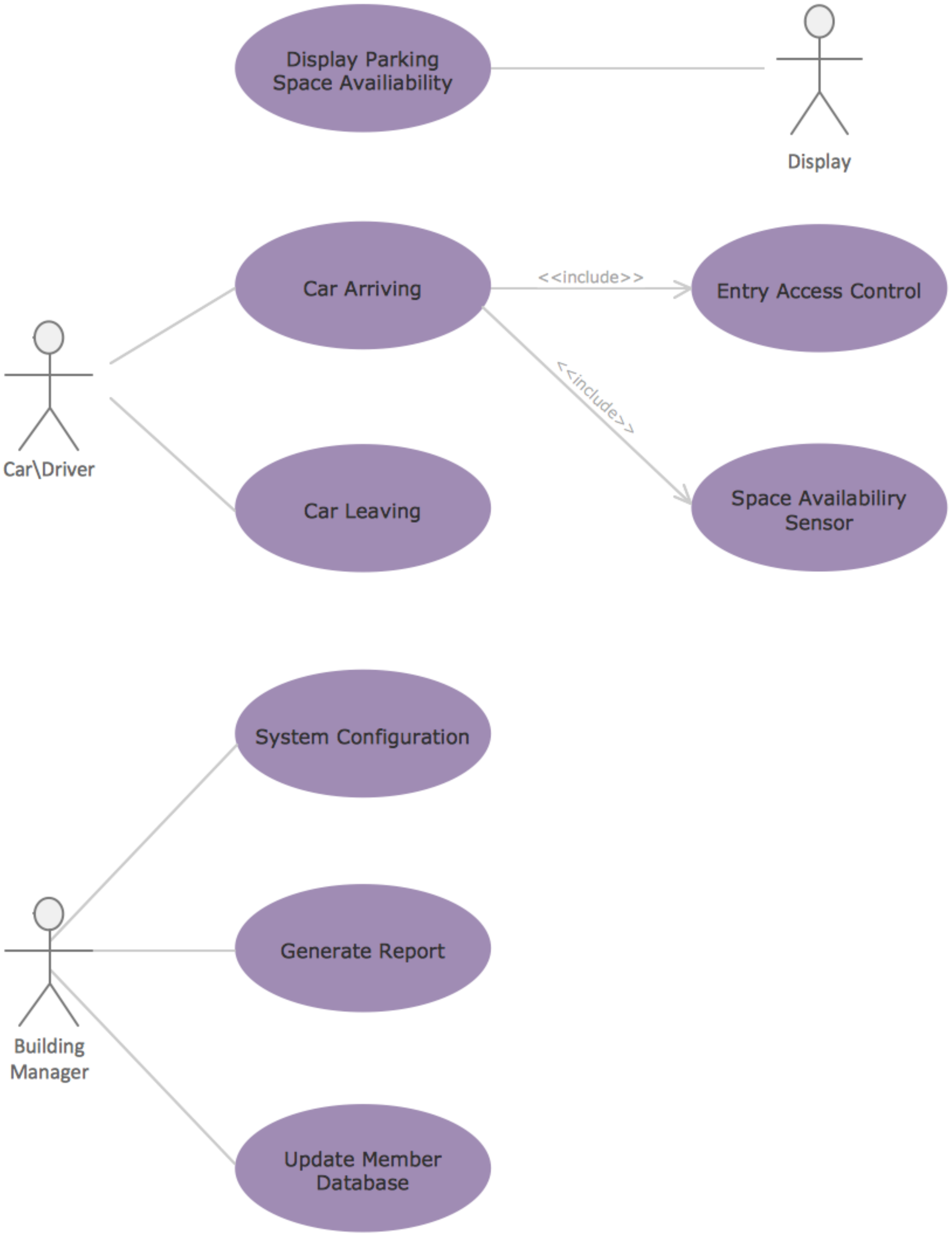


2) Deployment Diagram:


Car arrives at entry terminal
Button is pressed and a card/ticket is issued with a serial number, date & time. Open signal is sent to barrier. gate
Car parks in car park
Customer takes card/ticket to Central cashier station for payment before returning to car.
After payment, card/ticket is validated for exit with a pre-set grace time to exit.
Card/ticket is read by terminal and signal sent to barrier gate.
Car arrives at exit terminal with paid card/ticket

CENTRAL PAYMENT POINT

3) UML Diagram:



Test Plan:

High Level Test Plan

ID	Description	Expected O/P	Actual O/P	Type of Test
H_01	Obtain vehicle details	PASSED	SUCCESS	Requirement
H_02	Verify details	PASSED	SUCCESS	Scenario
H_03	Update Parking status in app	PASSED	SUCCESS	Boundary
H_04	Update arriving status	PASSED	SUCCESS	Boundary
H_05	Update departure status	PASSED	SUCCESS	Boundary
H_06	Verify the data details	PASSED	SUCCESS	Scenario
H_07	Register for the Service details	PASSED	SUCCESS	Boundary
H_08	View the details Selected parking area	PASSED	SUCCESS	Requirement
H_09	Generate the Session ID each reservation	PASSED	SUCCESS	Boundary
H_10	Modify the parking status	PASSED	SUCCESS	Boundary

Low Level Test Plan

ID	Description	Expected O/P	Actual O/P	Type of Test
L_01	Open the app	PASSED	SUCCESS	Requirement
L_02	Enter the vehicle number	PASSED	SUCCESS	Requirement
L_03	View the parking details	PASSED	SUCCESS	Scenario
L_04	Reserve your space	PASSED	SUCCESS	Boundary
L_05	Enter the vehicle type (car or scooty)	PASSED	SUCCESS	Scenario
L_06	Update arriving status	PASSED	SUCCESS	Boundary
L_07	Update departure status	PASSED	SUCCESS	Boundary
L_08	Calculate the bill	PASSED	SUCCESS	Requirement
L_09	Exit the application	PASSED	SUCCESS	Scenario

Git Link:

Link: GENESIS2021Q1/Applied_SDLC-Dec_Team_46: Details (github.com)

Mini project 4 – Calendar Automation [Team]

Modules:- OOPS with Python

Requirements

High Level Requirements

ID	Feature	MATLAB v0 Status	Python v0 Status
HR01	GUI	Implemented	Implemented
HR02	Master calendar	Implemented	Implemented
HR03	Faculty calendar	Implemented	Implemented
HR04	Faculty load sheet	Implemented	Implemented
HR05	Showing Available Open Slots based on faculty and modules	Not Available	Not Available
HR06	Output file generated across different computers (windows + Linux)	Not Available	Implemented
HR07	Visualizing data to create Meaningful Insights	Not Available	Not Available
HR08	Calculate Individual Faculty Load	Implemented	Implemented

Low Level Requirements

ID	Feature	High Level ID	MATLAB v0 Status	Python v0 Status
LR01	GUI should allow user to login using credentials	HR01	Not Available	Not Available
LR02	Input Files Based on Different Initiatives and Timelines	HR01	Implemented	Not Available
LR03	GUI should get Base Calendar as Input	HR01	Implemented	Implemented
LR04	GUI should get Month and Initiative as Input	HR01	Implemented	Implemented
LR05	GUI should be able to show Conflicts/Warnings	HR01	Implemented	Not Implemented
LR06	Master Calendar: display Month wise	HR02	Implemented	Implemented
LR07	Master Calendar: display Initiative wise	HR02	Implemented	Not Available
LR08	Master Calendar: Differentiate Initiatives (Color Codes/Numbers)	HR02	Implemented	Implemented
LR09	Master Calendar: Appending	HR02	Implemented	Not Available

ID	Feature	High Level ID	MATLAB v0 Status	Python v0 Status
LR10	Master Calendar: Course code correction	HR02	Implemented	Not Available

Link for template standard input template :

https://docs.google.com/spreadsheets/d/1EWYp_1iyK2wLMfKGJOiTJAk5WexZusCP/edit?usp=sharing&ouid=113003694561146884677&rtpof=true&sd=true

- Using the template above, training schedule can be added monthwise and initiatives wise
- The name of the input excel sheet MUST be named as "Test_vector"(as shown in template)
- Along with the Test_vector sheet, "Key" sheet MUST be present under the columns assigned as in the template
- The "Key" sheet must contain all times the 6 fixed initiatives with their respective codes and total list of course code and course title in order to refer for corrections while writing to output files
- Appending additional slots for existing courses is possible by adding just the additional slots in the input file for the same course

Requirements for updating Master calendar using Master calendar as input

Link for template

2 Slots format - M/A
: https://docs.google.com/spreadsheets/d/1jtKnXV12VE1fH20CGDo4B3uNWRTAhQCWz-hHUDWUe3I/edit?usp=sharing

4 Slots format - M1/M2/A1/A2
: https://docs.google.com/spreadsheets/d/1jVheSPZkOtfNKRNoc_858nwk2UaHCe0gExTNZfZ8vxA/edit?usp=sharing

- Any of the two templates can be used for updating Master calendar month wise on to the drive

- The blocked slots must have the corresponding initiative code in the cell according to the key as shown in the sample data in the template
- The name of the sheet must be the name of the month to be updated
- The "Key" sheet must be present with the fixed list of initiatives and initiative code

App deployment

- The app is deployed on heroku servers.
- To add/modify new features, you will be required to install HEROKU CLI link
- After installation, open terminal in working directory and enter the following commands:
  - "heroku git:clone -a gea calendar"
  - login using heroku credentials
- After pulling and making changes, enter the following commands to push app and deploy on server
  - Git add.
  - git commit -m "commit message"
  - git push heroku master

Additional features for V1 to do

- Update key sheet by appending new initiatives/courses list
- Check for duplicate course entries in input file
- Using built in libraries to identify number of days in month, current year and highlight weekend and holidays
- Function to remove a course schedule
- Read multiple months data in one sheet as input file (currently takes data one by one month)
- Calculate individual faculty load

Git Link:

Pradnya579/GENESIS2021-OOP-Python_Team_46 (github.com)

Mini project 5 –Team Scorpio [Team]

Module: - Applied Model Based Design Module
Individual Topic: - Anti-Lock Braking System

Requirements

INTRODUCTION

Anti-lock brake systems (ABS) prevent brakes from locking during braking. Under normal braking conditions the driver controls the brakes. However, during severe braking or on slippery roadways, when the driver causes the wheels to approach lockup, the antilock system takes over. ABS modulates the brake line pressure independent of the pedal force, to bring the wheel speed back to the slip level range that is necessary for optimal braking performance. An antilock system consists of wheel speed sensors, a hydraulic modulator, and an electronic control unit. The ABS has a feedback control system that modulates the brake pressure in response to wheel deceleration and wheel angular velocity to prevent the controlled wheel from locking. The system shuts down when the vehicle speed is below a pre-set threshold.

OVERVIEW

- ABS was first invented and applied in the aircraft industry and then was introduced to automobile industry in the early 1970's. However, it had not been used popularly until the middle of the 1980's due to technical difficulties and high cost.
- ABS functions in place of the traditional brake system at times of wheel lock-up. A quick test sequence checks all the components of the system.
- If ever the test sequence fails, the normal brake system is in control.
- Although the normal brake system can give instant and efficient braking, it can cause the wheels to be lock up, therefore, the driver cannot steer and would lose control of the car.
- If any of the wheels happen to be skidding, the driver must recognize wheel-skid and manually 'pump the brakes' to avoid a skid. The advantage of ABS lies in its ability to allow the driver retain steering control in order to keep the car moving in the direction that the wheels are turned towards, rather than skidding in the direction of the car's forward momentum.
- ABS has the classic design of an embedded system.

1. controller Sensor
2. Wheel speed.
3. Actuators (valve and ABS reservoir) at each wheel.

Analysis and Physics

The wheel rotates with an initial angular speed that corresponds to the vehicle speed before the brakes are applied. We used separate integrators to compute wheel angular speed and vehicle speed. We use two speeds to calculate slip, which is determined by Equation 1. Note that we introduce vehicle speed expressed as an angular velocity (see below).

$$\omega_v = \frac{V}{R} \text{ (equals the wheel angular speed if there is no slip)}$$

Equation 1

$$\omega_v = \frac{V_v}{R_r}$$

$$slip = 1 - \frac{\omega_w}{\omega_v}$$

$$\omega_v = \text{ vehicle speed divided by wheel radius}$$

$$V_v = \text{ vehicle linear velocity}$$

$$R_r = \text{ wheel radius}$$

$$\omega_w = \text{ wheel angular velocity}$$

From these expressions, we see that slip is zero when wheel speed and vehicle speed are equal, and slip equals one when the wheel is locked. A desirable slip value is 0.2, which means that the number of wheel revolutions equals 0.8 times the number of revolutions under non-braking conditions with the same vehicle velocity. This maximizes the adhesion between the tire and road and minimizes the stopping distance with the available friction.

REQUIREMENTS

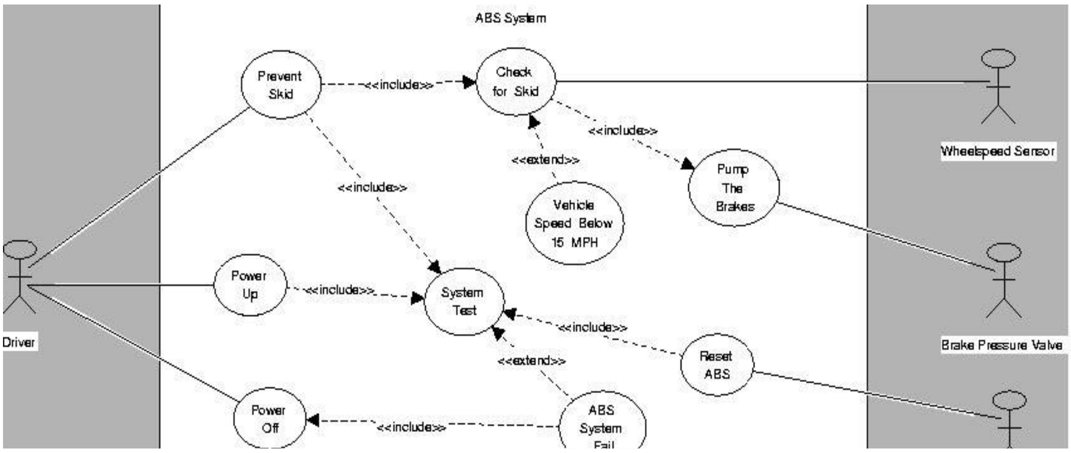
High Level Requirements:-

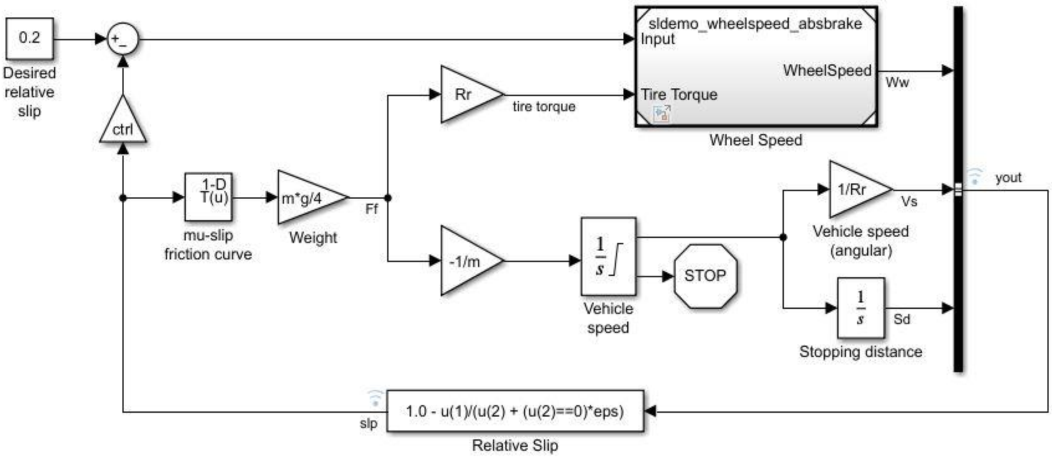
ID	Description
HLR1	Receive on/off signals from the brakes, and use them for engaging and disengaging the ABS system
HLR2	Receive rotational speed data from four wheel speed sensors
HLR3	The same signal that is used to turn on the brake lights is read by the ABS system in order to determine whether or not the brake has been pressed engaged. The ABS will then only be able to become engaged if this signal shows the brakes are being used.
HLR5	The ABS will receive information from a wheel speed sensor for each wheel. Each wheel speed sensor will send the speed of the wheel it is monitoring in meters/second.
HLR6	Run a system diagnostic test sequence at ignition and determine if any errors are present in the system.

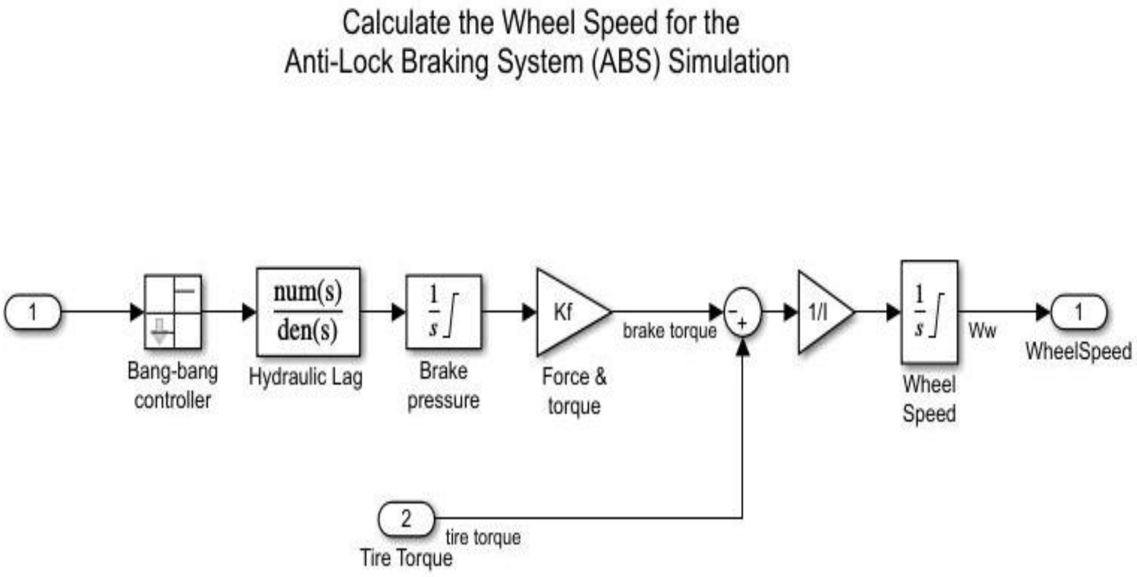
Low Level Requirements:-

ID	Description
LLR1	The wheel rotates with an initial angular speed that corresponds to the vehicle speed before the brakes are applied.
LLR2	The system test will engage when the car is turned on.
LLR3	Calculate rotational deceleration from the wheel speed data, and determine if wheel lock-up is imminent.
LLR4	The same signal that is used to turn on the brake lights is read by the ABS system in order to determine whether or not the brake has been pressed engaged. The ABS will then only be able to become engaged if this signal shows the brakes are being used.
LLR5	Terminate system execution if any failure occurs form either test. The termination shall not affect normal braking behaviour of the vehicle

DESIGN


ABS System
Prevent Skid
<<include>>
Check for Skid
<<include>>
<<extend>>
Pump The Brakes
<<include>>
Vehicle Speed Below 15 MPH
Wheelspeed Sensor
Driver
Power Up
<<include>>
System Test
<<include>>
Reset ABS
<<extend>>
Brake Pressure Valve
Power Off
<<include>>
ABS System Fail

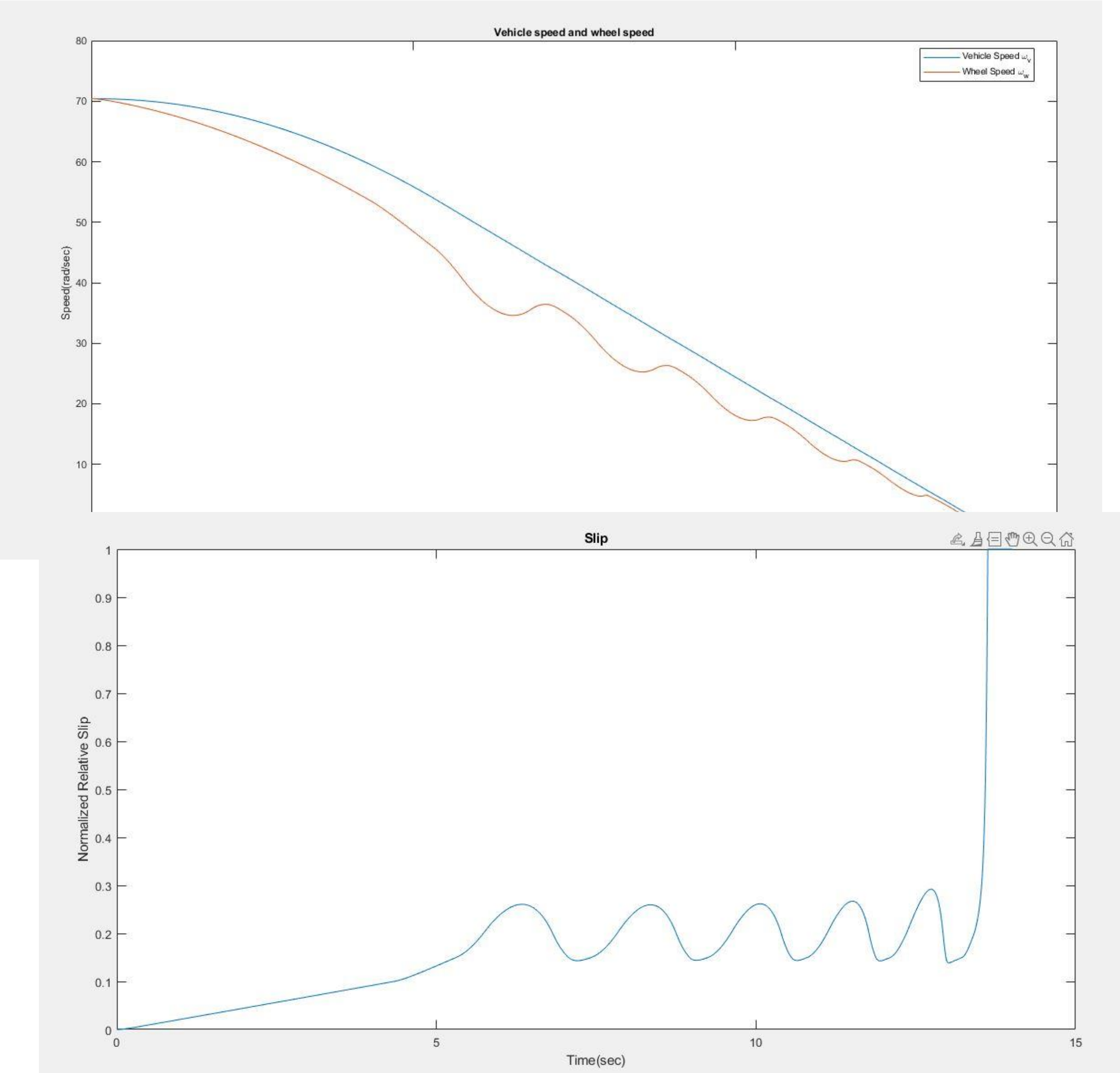

0.2
Desired relative slip
+
-
ctrl
sldemo_wheelspeed_absbrake
Input
WheelSpeed
Tire Torque
Wheel Speed
Ww
Rr
tire torque
1-D T(u)
mu-slip friction curve
m*g/4
Weight
Ff
-1/m
1/s
Vehicle speed
STOP
1/Rr
Vs
Vehicle speed (angular)
1/s
Sd
Stopping distance
yout
slp
1.0 - u(1)/(u(2) + (u(2)==0)*eps)
Relative Slip


Calculate the Wheel Speed for the
Anti-Lock Braking System (ABS) Simulation
1
Bang-bang controller
num(s)/den(s)
Hydraulic Lag
1/s
Brake pressure
Kf
Force & torque
brake torque
-
+
1/I
1/s
Wheel Speed
Ww
1
WheelSpeed
2
Tire Torque
tire torque

OUTPUT

Running the Simulation in ABS Mode

- **<u>Vehicle Speed and Wheel Speed</u>**



Conclusion

This model shows how you can use Simulink to simulate a braking system under the action of an ABS controller. The controller in this example is idealized, but you can use any proposed control algorithm in its place to evaluate the system's performance. You can also use the Simulink® Coder™ with Simulink as a valuable tool for rapid prototyping of the proposed algorithm. C code is generated and compiled for the controller hardware to test the concept in a vehicle. This significantly reduces the time needed to prove new ideas by enabling actual testing early in the development cycle.

For a hardware-in-the-loop braking system simulation, you can remove the 'bang-bang' controller and run the equations of motion on real-time hardware to emulate the wheel and vehicle dynamics. You can do this by generating real-time C code for this model using the Simulink Coder. You can then test an actual ABS controller by interfacing it to the real-time hardware, which runs the generated code. In this scenario, the real-time model would send the wheel speed to the controller, and the controller would send brake action to the model.

Mini project 6 – Wiper Control [Team]

Modules: - Mastering Microcontrollers with Embedded Driver Development Module

WIPER CONTROL SYSTEM

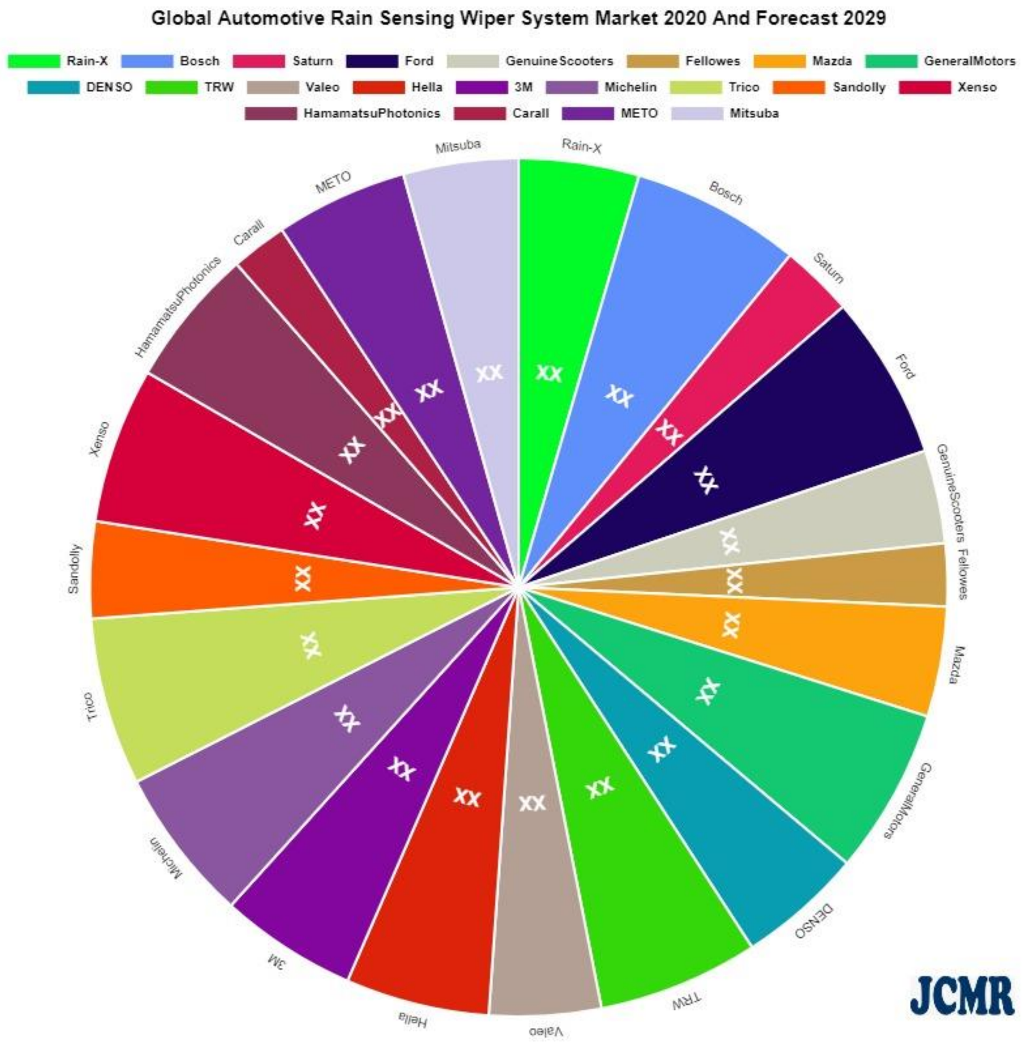
Introduction

Automotive wipers form an essential part for any vehicle. They perform to remove water, ice, snow, and dust from a windshield of a vehicle. An automotive wiper is either powered by an electric motor or pneumatic power. Almost all motor vehicle including cars, trucks, buses, train locomotives and watercraft with a cabin are equipped with one or more such wipers. The automotive wiper market is multiplying as there is an exponentially increased production of automobiles globally.

Features

- To achieve high safety
- To reduce man power
- To increase the efficiency of the vehicle
- To reduce the work load
- To reduce the vehicle accident
- To reduce the fatigue of workers
- To high responsibility
- Less Maintenance cost

State of Art



SWOT Analysis

Strength

- It is possible to operate Manually/automatically by proving On/Off switch
- Improve Visibility of car in rain.Can drive easily in any climatic situation.

Weakness

- This system applied in the case of water falling on the class only.
- Addition cost is required to install this system to four wheeler.

Opportunities

- To increase automation in vehicle driving system
- To dispense with troublesome wiper operation needed when rainfall condition change or driving condition change, including the car speed and entry to or exit from tunnels.
- To operate the wiper with response to changing rainfall or driving conditions, thus keeping the driver's windshield clear.

Threats

- Dust particles and non-conductive particles accumulated on the surface of sensors cannot be detected by conductive sensors.

4W 1H:

Who:

- A wiper speed control system for an automotive wiper controls the operational speed of a wiper in accordance with rain conditions.

What:

- Vehicles are now available with driver-programmable intelligent (automatic) windscreen wipers that detect the presence and amount of rain using a rain sensor.

Where:

- It is located underneath the dashboard, above the brake and accelerator pedal, and is responsible for the complete operation of the windshield wiper system.

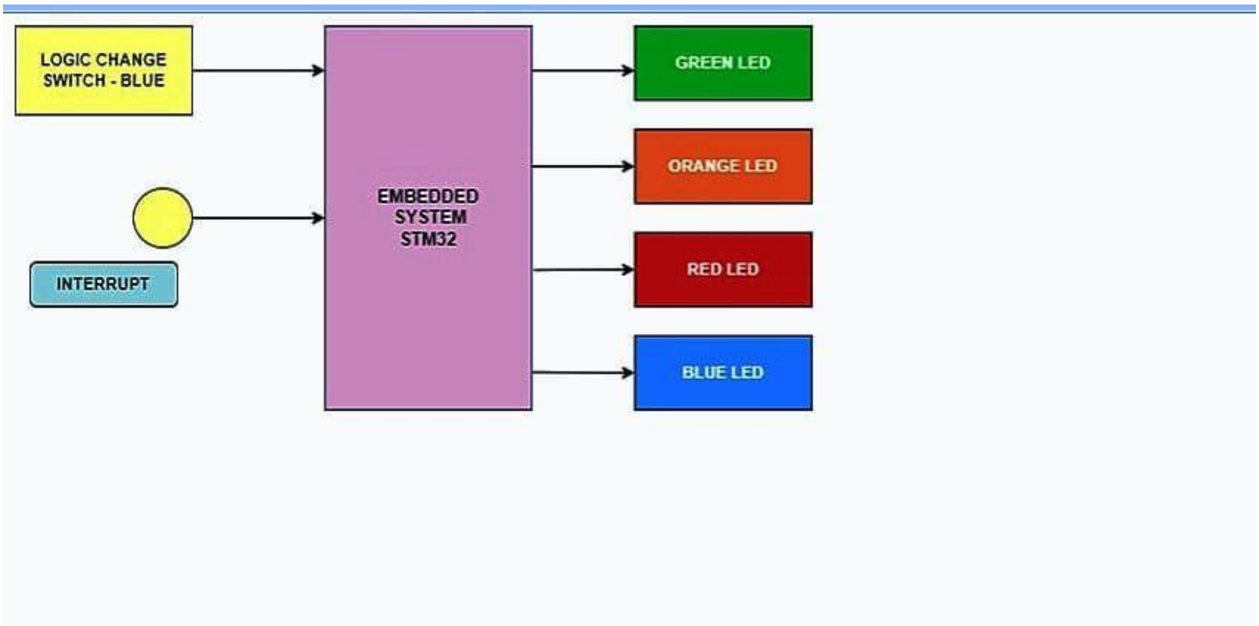
When:

- Whenever the water hit a dedicated sensor that located on windscreen, it will send a signal to move on the wiper motor. Once water is not detected by sensor, the wiper will automatically stop. This will help the driver to give more concentration and reduce the car accident probability.

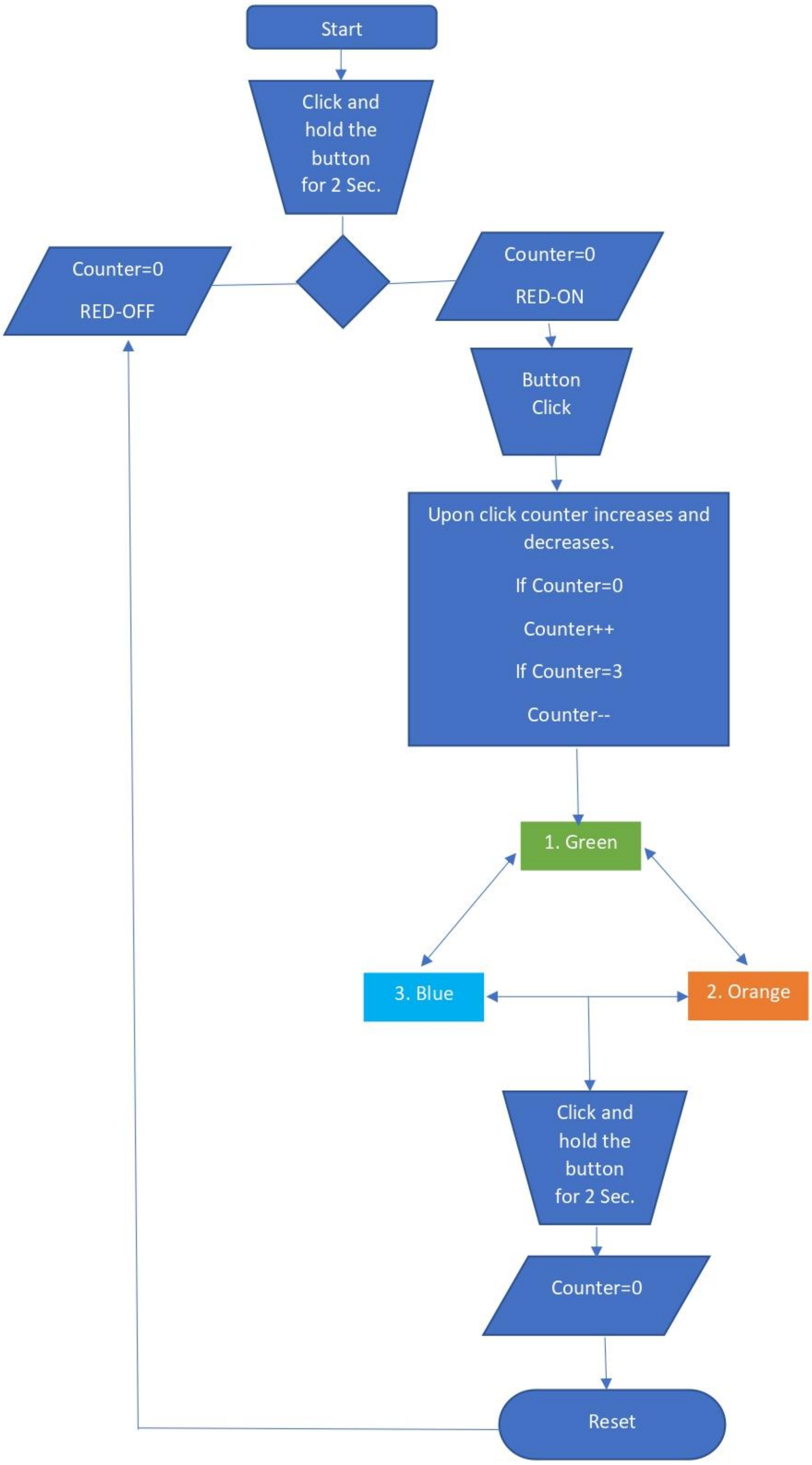
How:

- Windshield wipers are controlled by the stalk on the right side of your steering wheel. Simply moving the stalk down will turn your windshield wipers on. Moving the stalk down will turn your you wipers on.

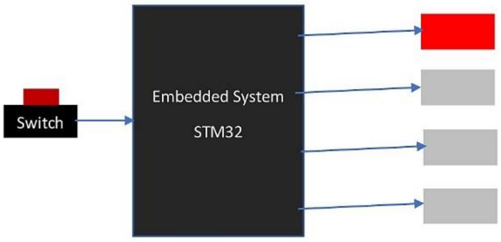
BLOCK DIAGRAM



FLOW CHART


Start
Click and hold the button for 2 Sec.
Counter=0
RED-OFF
Counter=0
RED-ON
Button Click
Upon click counter increases and decreases.
If Counter=0
Counter++
If Counter=3
Counter--
1. Green
3. Blue
2. Orange
Click and hold the button for 2 Sec.
Counter=0
Reset

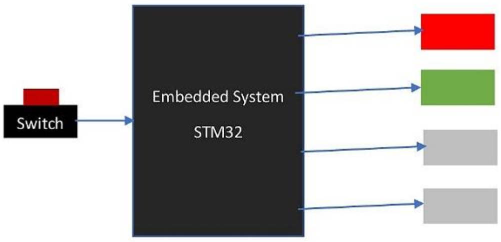
CASE STUDY



User holds button for 2 seconds.

Turns – Red LED ON.

Stage - 0



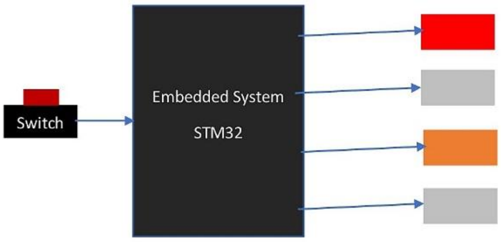
User click's button 1 time(s).

Turns – Green LED ON

Wiper is in Level – 1.

Frequency - 1 Hz

Stage - 1



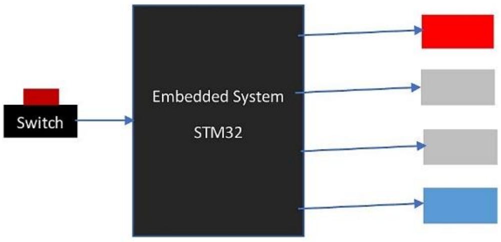
User click's button 1 time(s).

Turns – Orange LED ON

Wiper is in Level – 2.

Frequency - 4 Hz.

Stage - 2



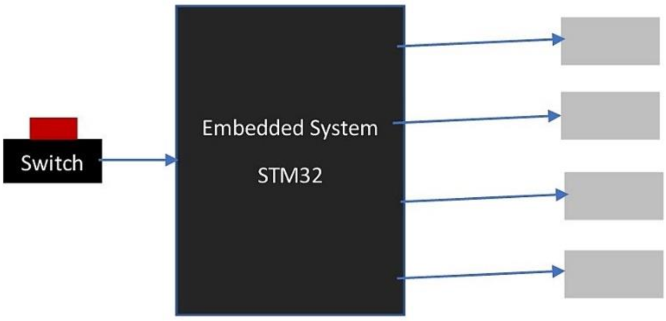
User click's button 1 time(s).

Turns – Blue LED ON

Wiper is in Level – 3.

Frequency – 8Hz.

Stage - 3



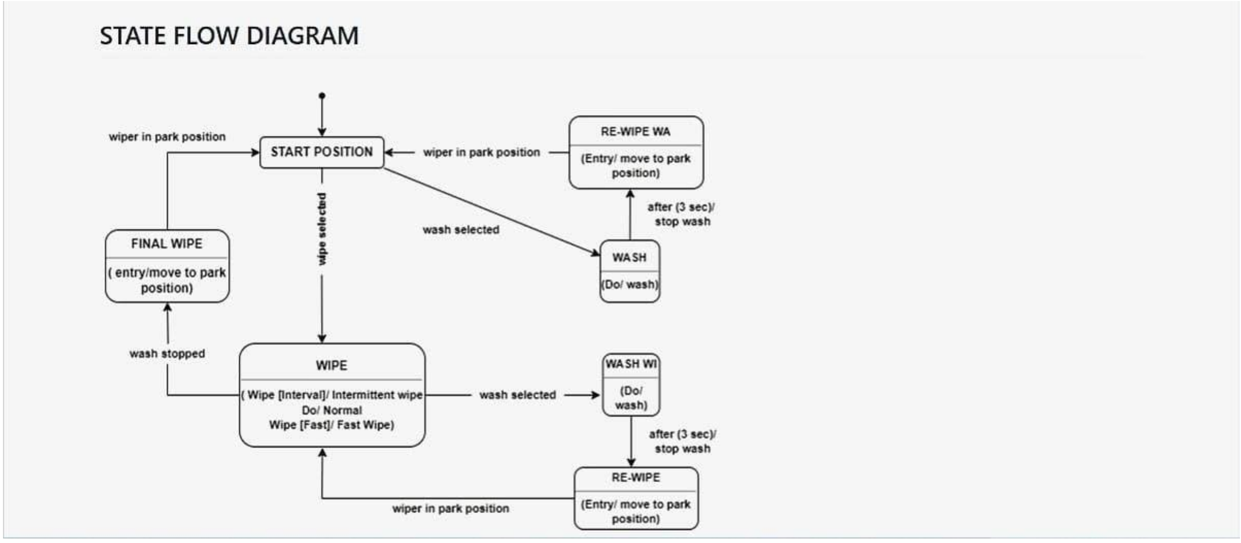
User holds button for 2 seconds.

Turns – All LED OFF.

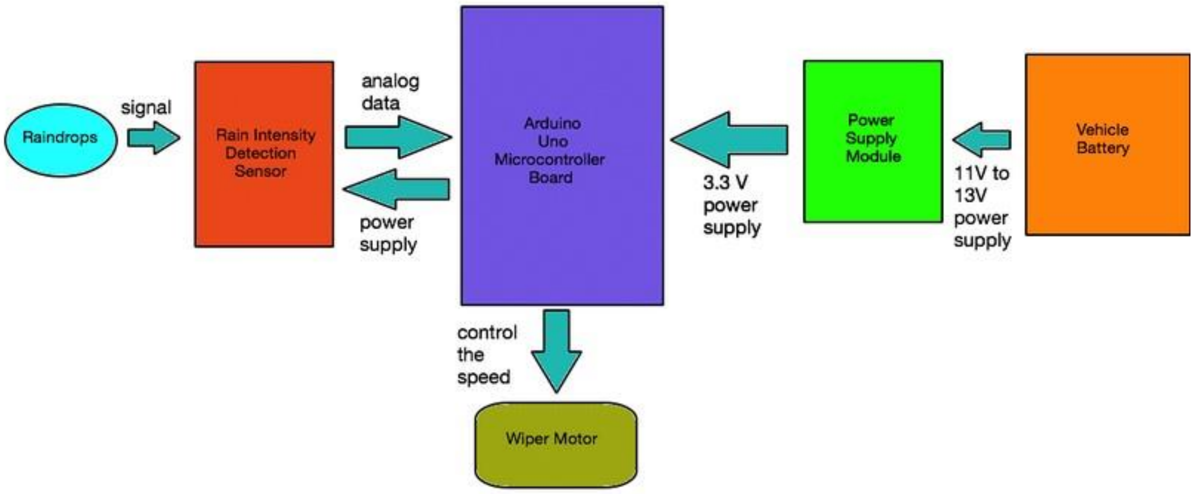
Stage - 0

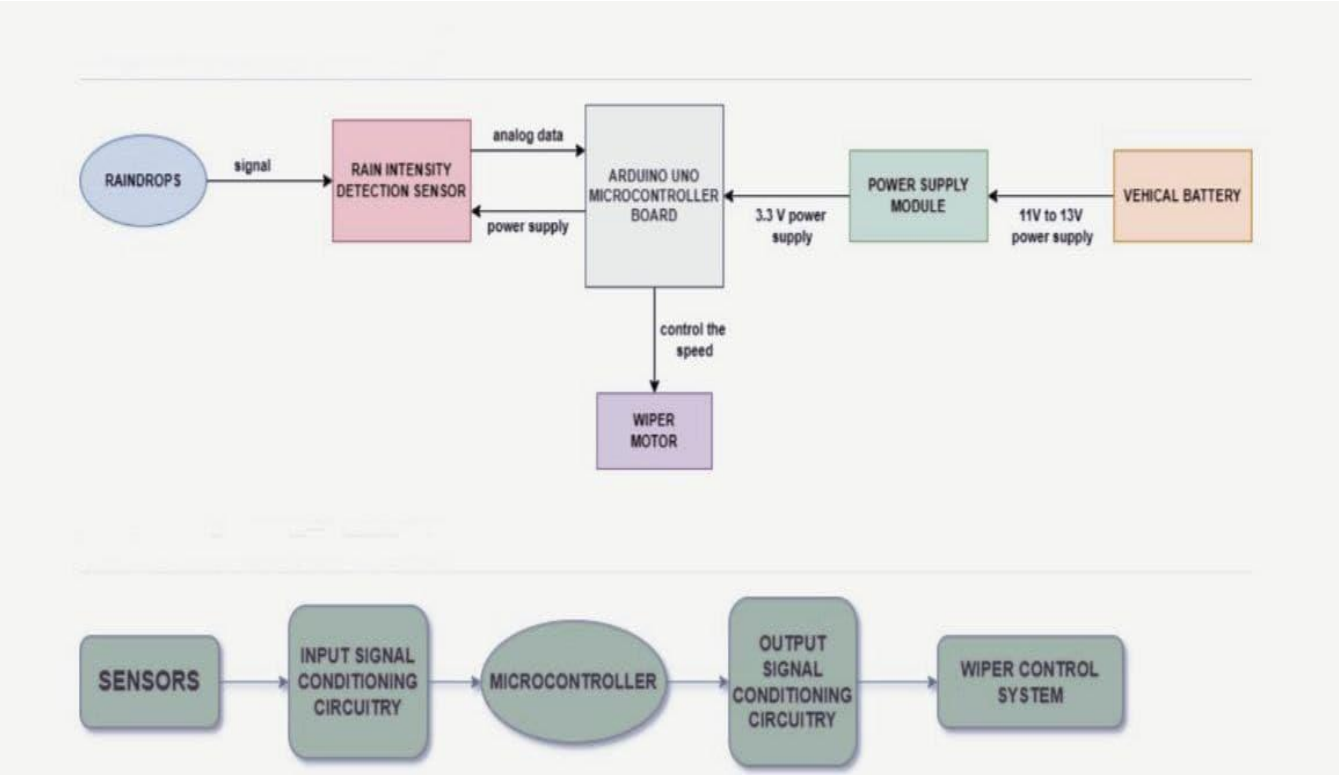
According to the requirement user can change the wiper levels, this process is shown in forward direction, and the three LEDs other than red LED can be changed from level to level.

STATE FLOW



SYSTEM DESIGNS





High Level Requirements

ID	Description
HLR1	These systems detect droplets of rain on the windshield and automatically turn on and adjust the wiper system in accordance to the level of precipitation.

ID	Description
HLR2	A windscreen wiper or windshield wiper is a device used to remove rain, snow, ice and debris from a windscreen or windshield.
HLR3	Quality and reliability wiper systems meet the highest technical requirements and are the basis for vehicles with sophisticated features.
HLR5	Almost all motor vehicle, including trains, aircraft and watercraft, are equipped with such wipers, which are usually an essential requirement.
HLR6	Our project brings forward this system to automate the wiper system having no need for manual intervention.

Low Level Requirements

ID	Description
LLR1	A new mechatronic reversing system can now be used to clean the windshield with two wiper arms, whereby one wiper arm is powered directly and the other indirectly using a connection link.
LLR2	Wiper motor is automatically ON during the time of rainfall.
LLR3	Existing system manually used control stalk to activate wiper and the process of pulling up wiper is difficult to be handled.
LLR4	Lower level parsing¶. Under the hood, the Requirement class does most of the heavy lifting... Class requirements.
LLR5	These systems detect droplets of rain on the windshield and automatically turn on and adjust the wiper system.

Implementation and Summary

Git Link:

Link: GENESIS-2022/MasteringMCU-Team70: Details (github.com)

Individual Contribution and Highlights

1. Wiper System using C Programming
2. Source code management using GitHub
3. REQUIREMENTS+Implementation(start-up’s , STM32F407XX.H)

Role in Project Team

1. Programmer: Done Programming for Wiper System
2. Integrator: Integrated all the codes
3. Tester: Writing Test cases and testing the integrated code

Mini project 7 – Tesla Project [Team]

Modules

1. Automotive Systems
2. Git

Requirements

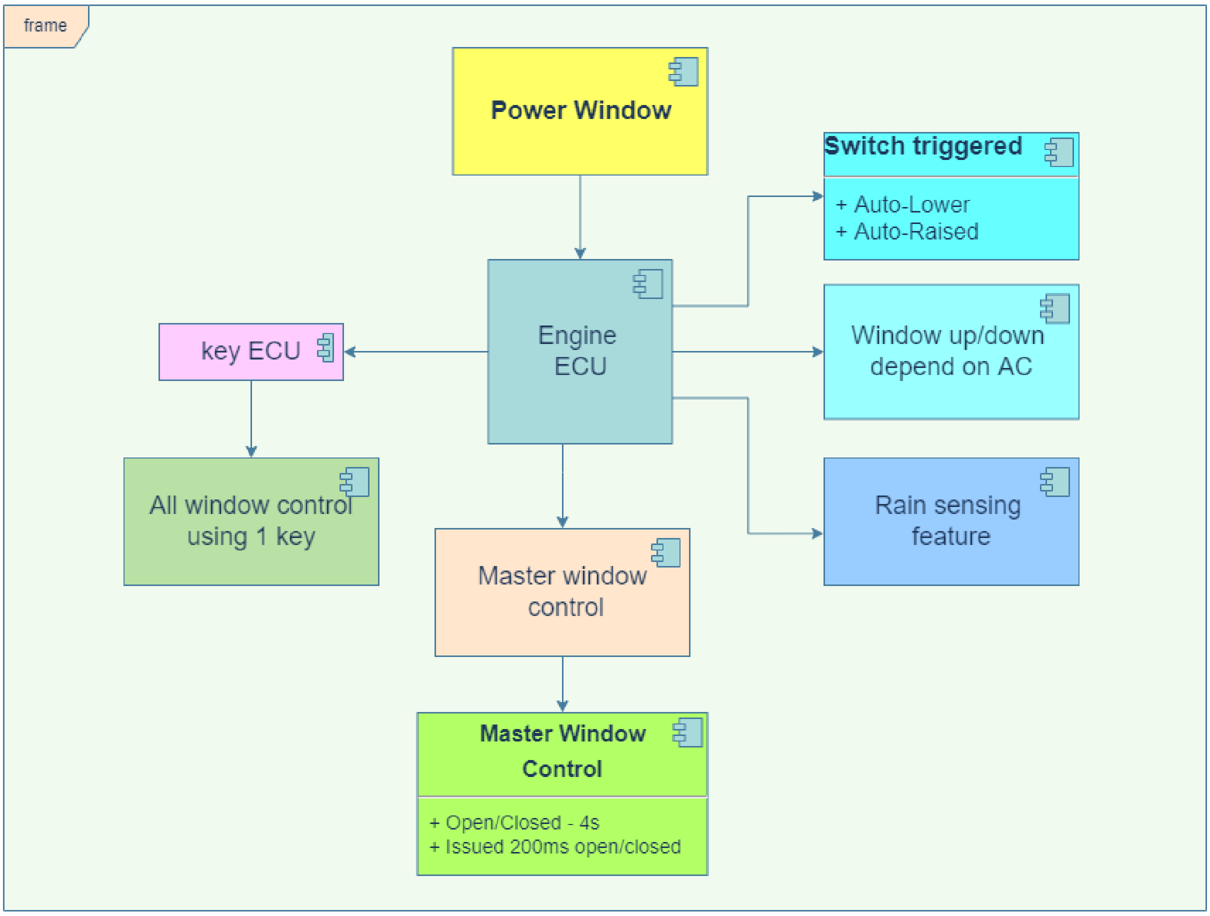
In this Tesla project we have taken following features and I have contributed to Power windows

1. Power Window
2. Power Mirrors
3. Door Locking Control
4. Lighting System
5. Wiper Control

POWER MIRROR REQUIREMENTS:

High level Requirement	Description
HLR1	When the engine is on, mirror has to be automatically open and close when engine is off.
HLR2	Depend upon the user signal adjust the mirror direction.
HLR3	All the condition should pass, one condition should pass at a time.
HLR4	Set the mirror direction as per the user requirement.

Low level Requirement	Description
LLR1	Put the input values.
LLR2	The system test will engage when the car is turned on.
LLR3	Compare the conditions.



Implementation and Summary

Git Link:

[Okayvvk/automotive_team_tesla (github.com)](https://github.com/Okayvvk/automotive_team_tesla)

Individual Contribution and Highlights

1. Power Mirrors System Case Study

Role in Project Team

1. Designer: Done Designing for Project
2. Researcher: Done case study for Body Control Module of Tesla.

Mini project 8 – EV Car [Team]

Module: - Applied Control Systems and Vehicle Dynamics

Modules

1. MATLAB
2. MATLAB Script

Requirements

Battery Performance:

1. Both cars use the same Lithium-ion battery type as it is the industry standard right now.
2. Hyundai Kona has a massive 452 km lead in terms of range which is more than double of what the Tata Nexon.
3. Battery charging times are longer in the Hyundai Kona due to its larger 39.1 kWh battery compared to the 37.4 kWh.
4. Hyundai Kona offer fast charging.

Braking Performance:

1. Hyundai Kona has ESP, not in Tata Nexon.
2. Hyundai Kona has TCS, not in Tata Nexon.

Suspension Performance:

1. Hyundai Kona has Independent MacPherson strut with coil spring and Tata nexon has McPherson Strut Type front suspension.
2. Hyundai Kona has Twist beam with dual path Strut and Tata nexon has multi-Link rear suspension.

Implementation and Summary

Submission: Submitted in GEA Learn

Individual Contribution and Highlights

1. Done in MATLAB Script

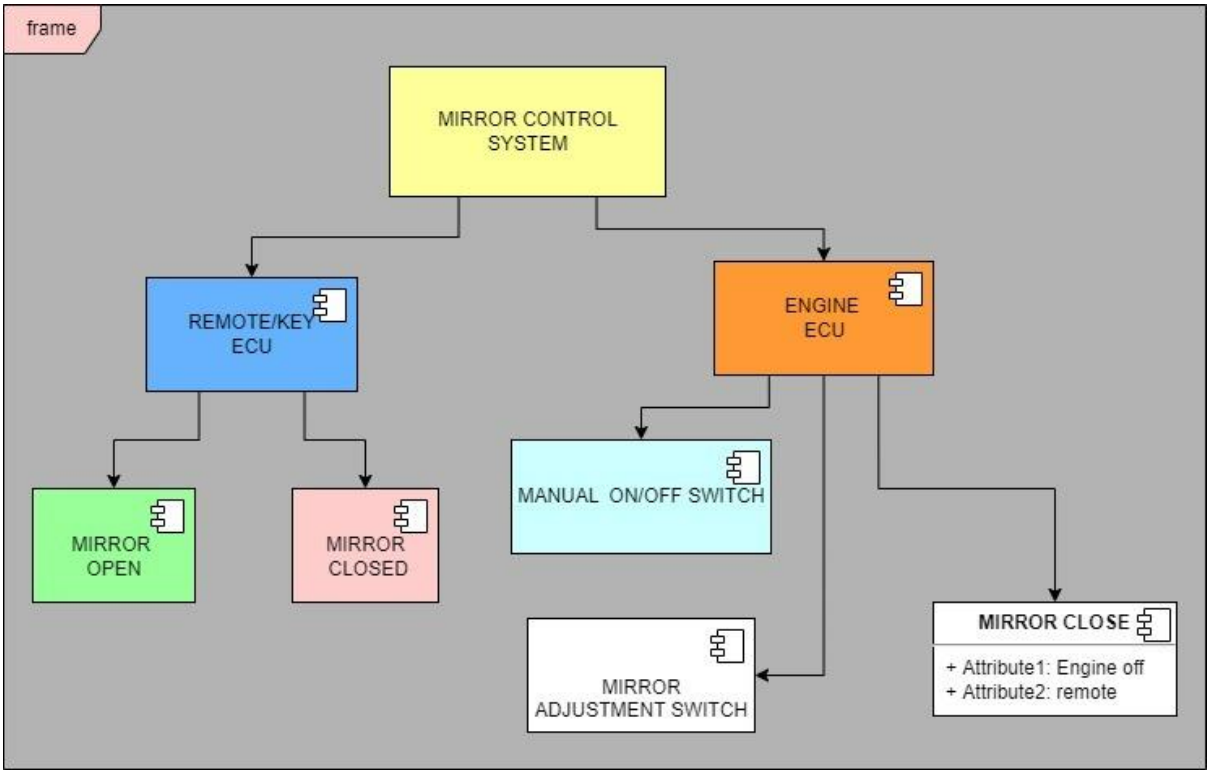
Mini project 9 – Power Mirrors [Individual]

Module: - Classic Autosar Basic to Intermediate

POWER MIRROR REQUIREMENTS:

High level Requirement	Description
HLR1	When the engine is on, mirror has to be automatically open and close when engine is off.
HLR2	Depend upon the user signal adjust the mirror direction.
HLR3	All the condition should pass, one condition should pass at a time.
HLR4	Set the mirror direction as per the user requirement.

Low level Requirement	Description
LLR1	Put the input values.
LLR2	The system test will engage when the car is turned on.
LLR3	Compare the conditions.



Implementation and Summary

Git Link:

Link: Okayvvk/automotive_team_tesla (github.com)

Individual Contribution and Highlights

1. Power Windows Control Case Study
2. Source code management using GitHub