

**Details**

Ver. Rel. No.	Release Date	Prepared By	Reviewed By	To Be Approved	Remarks/Revision Details
1.0	16/02/2022	Viraj Sawant 40021162			

## Contents

### MiniProject 1- Customer Billing System

List of Figures .....	5
Modules: .....	6
Requirements.....	6
High Level Requirements .....	
Design.....	
Test Plan.....	12
Low Level Test Plan.....	
Implementation and Summary .....	13
Git Link: .....	13
Git Dashboard .....	13
Miniproject 2 –Ultrasonic Sound Sensor[Individual].....	14
Modules .....	14
Requirements.....	
High Level Requirements .....	
Low Level Requirements.....	
Design.....	
TEST PLAN: .....	
High Level Test plan:.....	
Low Level Test Plan:.....	
Implementation and Summary .....	26
Git Link: .....	26
Git Dashboard .....	26
Miniproject 3 – Virtual Costume Advisor [Team] .....	27
Modules .....	27
Requirements.....	27
High Level Requirements .....	27
Low Level Requirements.....	28
Design.....	28
Behavioural Diagram Low Level.....	28
Behavioural Diagram High Level.....	29
Structural Diagram Of High Level .....	29
Structural Diagram Of Low Level .....	30
Test Plan.....	30

High Level Test Plan .....	30
Low Level Test Plan .....	31
Implementation and Summary .....	32
Git Link: .....	32
Individual Contribution and Highlights .....	32
Summary .....	32
Miniproject 4 – Calendar Automation[Team] .....	32
Modules .....	32
Requirements.....	32
High Level Requirements .....	32
Low Level Requirements .....	33
Implementation and Summary .....	36
Git Link: .....	36
Individual Contribution and Highlights .....	36
Mini project 5 –Team BMW [Team].....	37
Module: - Applied Model Based Design Module .....	37
Requirements.....	37
<b>High Level Requirements:-</b> .....	38
Low Level Requirements:-.....	39
Running the Simulation in Seat control Mode.....	40
Miniproject 6 – Wiper Control[Team].....	44
Modules .....	45
Requirements.....	45
High Level Requirements .....	45
Low Level Requirements .....	46
Design.....	46
Test Plan.....	47
High Level Test Plan .....	47
Low Level Test Plan .....	48
Implementation and Summary .....	48
Git Link: .....	49
Individual Contribution and Highlights .....	49
Miniproject 7 – BMW Project[Team].....	50
Modules .....	50
Requirements.....	50

BMW X5 .....	
Body Control Module .....	50
Features: .....	50
Introduction .....	50
4W's and 1H .....	51
High level Requirement .....	51
Low level Requirement.....	52
Body Control Module .....	52
Power Window.....	53
Implementation and Summary .....	53
Git Link: .....	53
Miniproject 8 – EV Car[Team] .....	54
Modules .....	54
Requirements.....	55
Implementation and Summary .....	56
Individual Contribution and Highlights .....	59
Miniproject 9 – Door Locking System [Individual] .....	60
Modules .....	61
Requirements.....	61
Design.....	62
Implementation and Summary .....	63
Git Link: .....	64
Individual Contribution and Highlights .....	65

## List of Figures

Figure 1 Behavior Diagram .....	
Figure 2 Structure Diagram .....	
Figure 3 Git Dashboard.....	
Figure 4 Behavior Diagram .....	
Figure 5 Structure Diagram .....	
Figure 6 Block Diagram.....	
Figure 7 Simulation.....	
Figure 8 Behavior Diagram .....	
Figure 9 Structure Diagram .....	
Figure 10 Git Dashboard.....	
Figure 11 Structure Diagram .....	
Figure 12 Behavior Diagram .....	
Figure 13 Structure Diagram .....	

## Miniproject – 1: Customer Billing System [Individual]

### Modules:

1. C Programming
2. Git

### Requirements

Customer Billing Services System is the billing system for local wholesale billing.

gathers, sorts, and verifies customer account information,

including usage data, payment information, adjustment information, monthly and one-time service charge information, and applies taxes.

### HIGH LEVEL REQUIREMENTS : CUSTOMER BILLING SYSTEM ABOUT THE PROJECT :

HLR	Description
HLR1	Check the requirements
HLR2	Calculation
HLR3	Discount
HLR4	Billing

### FEATURES :

- Can able to add no.of. items,
- Shows Purchase date,
- total bill amount.
- We can also add customer details for future usage.

- CHARACTER ITEMS TO BE ADDED IN SUPER MARKET: OIL, ATTA, OATS, BISCUITS, SPROUTS, DRY FRITS, SUGAR and etc.
- FLOAT QUANTITY AND PRICE : MULTIPLICATION OPERATION
- TOTAL AMOUNT: ADDITION OPERATION
- BILLING DATE : getdate() Function

## LOW LEVEL REQUIREMENTS :

LLR	Description
LLR_1 HLR_2	Get Data from user
LLR_2 HLR_2	Calculate the amount

LLR	Description
LLR_1 HLR_3	Check for the Discount
LLR_2 HLR_3	Calculate the final quantity & amount.
LLR_3 HLR_3	Return the total amount

- HEADER FILES : #include<stdio.h> for printf and scanf,#include<conio.h> for clrscr() and getch(),#include<dos.h> for getdate()
- FLOAT VARIABLE : AMOUNT ,TOTAL
- CHAR : ITEMS

## TIME AND COST :

- It saves lot of time compared to the manual calculation in billing system.
- It is simple and efficient.
- There is no mis calculations during peak hours.

## **4WIH ANALYSIS :**

- What - It is an customer billing system helps supermarket employees to do billing faster.
- Where - It Can be used in any places needs fast billing system. example: Supermarket, Medicals, Coffee Shops and etc.
- when - anytime
- why - To Reduce crowd in billing Counters and to keep an eye on loss of money.
- How - after building perfect system, testing and executing in Supermarkets.

## **SWOT ANALYSIS:**

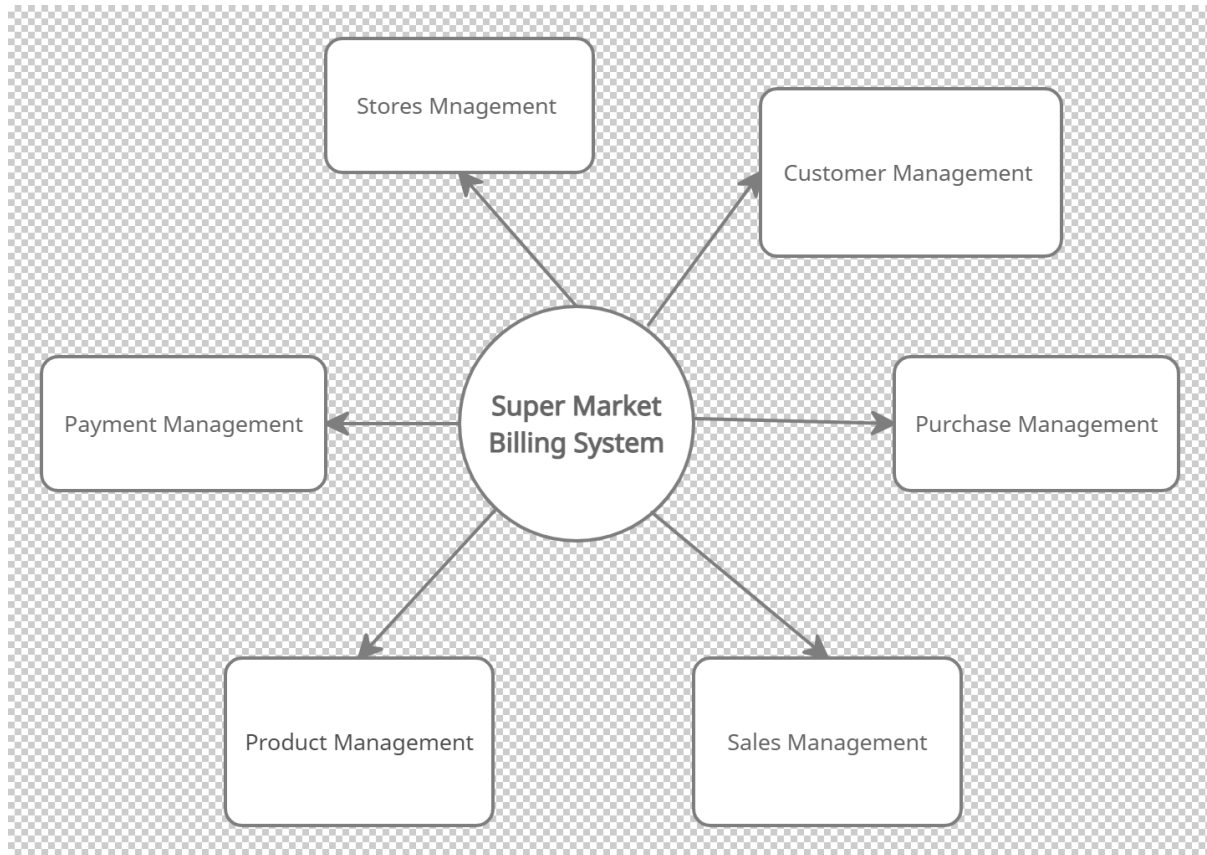
- strength - faster Operation
- Weakness - nil
- Opportunity - nil
- Threat - nil



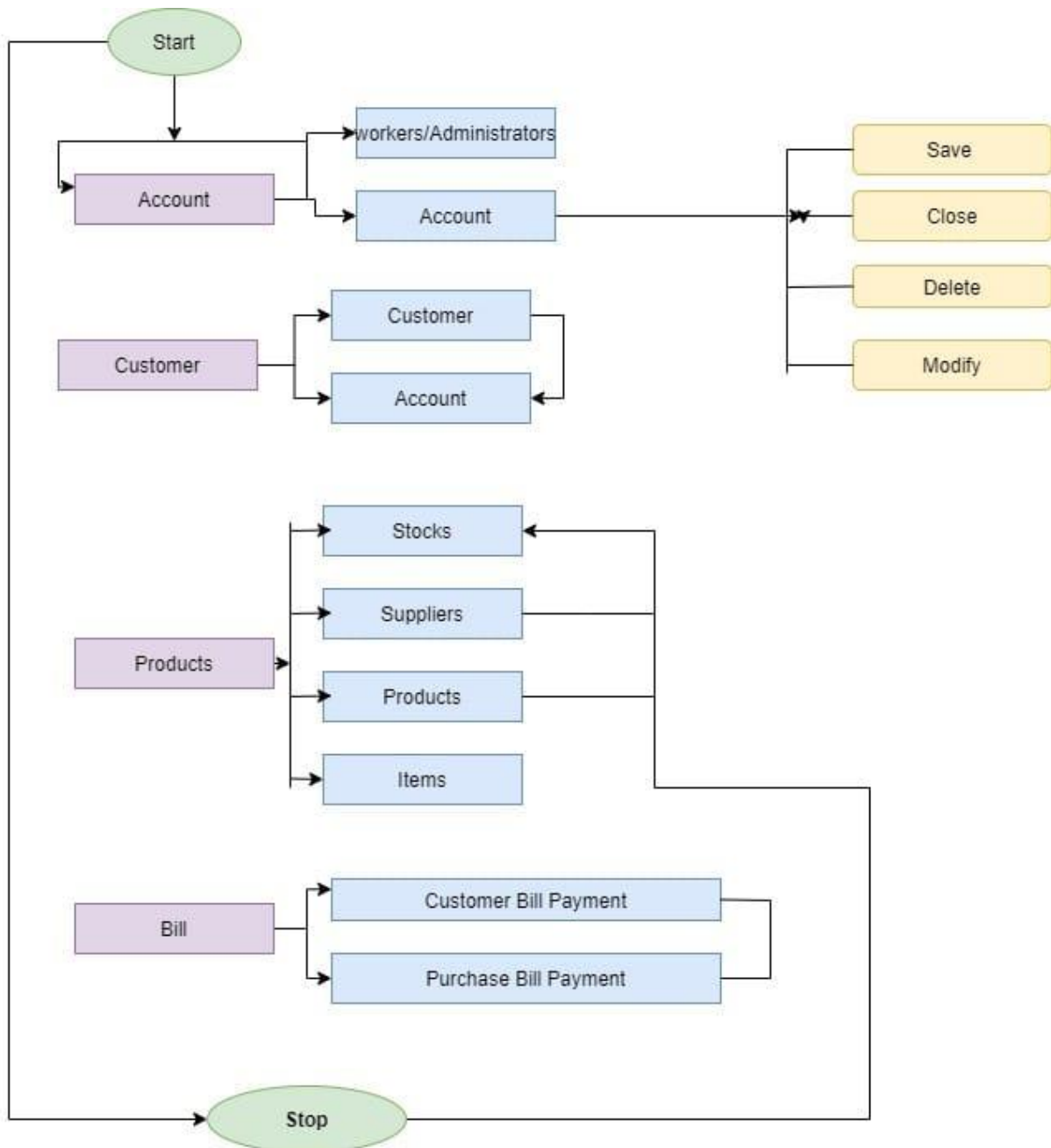
## Behavioural Diagram 1



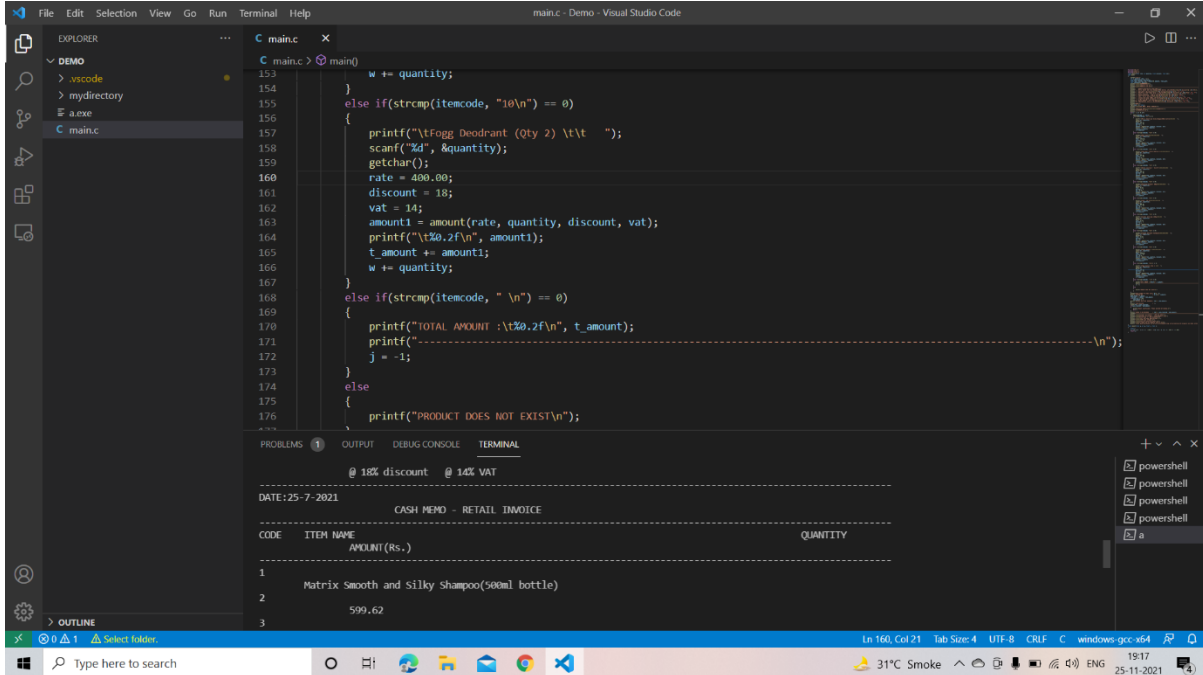
## Behavioural Diagram 2



## Structural Diagram



## Test Plan



```

153      w += quantity;
154    }
155    else if(strcmp(itemcode, "10\n") == 0)
156    {
157      printf("\tfogg Deodrant (Qty 2) \t\t ");
158      scanf("%d", &quantity);
159      getchar();
160      rate = 400.00;
161      discount = 18;
162      vat = 14;
163      amount1 = amount(rate, quantity, discount, vat);
164      printf("\t%0.2f\n", amount1);
165      t_amount += amount1;
166      w += quantity;
167    }
168    else if(strcmp(itemcode, "\n") == 0)
169    {
170      printf("TOTAL AMOUNT : \t%0.2f\n", t_amount);
171      printf("-----\n");
172      j = -1;
173    }
174    else
175    {
176      printf("PRODUCT DOES NOT EXIST\n");
177    }

```

@ 18% discount @ 14% VAT  
 DATE:25-7-2021  
 CASH MEMO - RETAIL INVOICE  
 CODE ITEM NAME AMOUNT(Rs.) QUANTITY  
 1 Matrix Smooth and Silky Shampoo(500ml bottle)  
 2 599.62  
 3

## Test Plan & Output

TestID	Description	Exp Ip	Exp op	Type of test
01	Calculate Total bill with Discount	539	539	Requirement Based
02	Generate Invoice	Success	Success	Requirement Based
03	Keep record of the Invoice	Success	Success	Requirement Based

## Implementation and Summary

### Git Link:

Link: [https://github.com/Virajsawant619/M1\\_Appl\\_customer\\_Billing\\_system](https://github.com/Virajsawant619/M1_Appl_customer_Billing_system)

### Git Dashboard

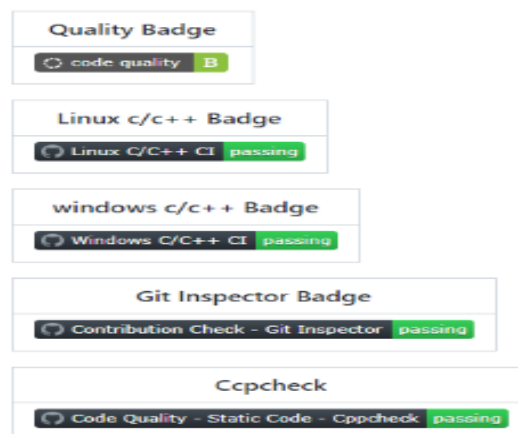


Figure 1 Git Dashboard

## Miniproject 2 – Ultrasonic Sound Sensor [Individual]

### Modules

1. C Programming
2. Embedded System
3. SimulIDE
4. Git

### Requirements:-

### Introduction

#### ULTRASONIC SOUND SENSOR WITH ATmega328 MICROPROCESSOR

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.

### Research:-

The requirements for the program to run or for the code to be in effect are basic and a great solution for the detection of clear objects.

### 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 gpsim 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.

## Defining Our System: -

Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensors, like many others, use a single transducer to send a pulse and to receive the echo

## 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 ultrasonics 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 an 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.

**• When:**

In 1959, Satomura created an ultrasonic flowmeter that used doppler technology.

**• Why:**

I am Developing this project for easily measure the distance between objects



## • How:

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

## Detail requirements :-

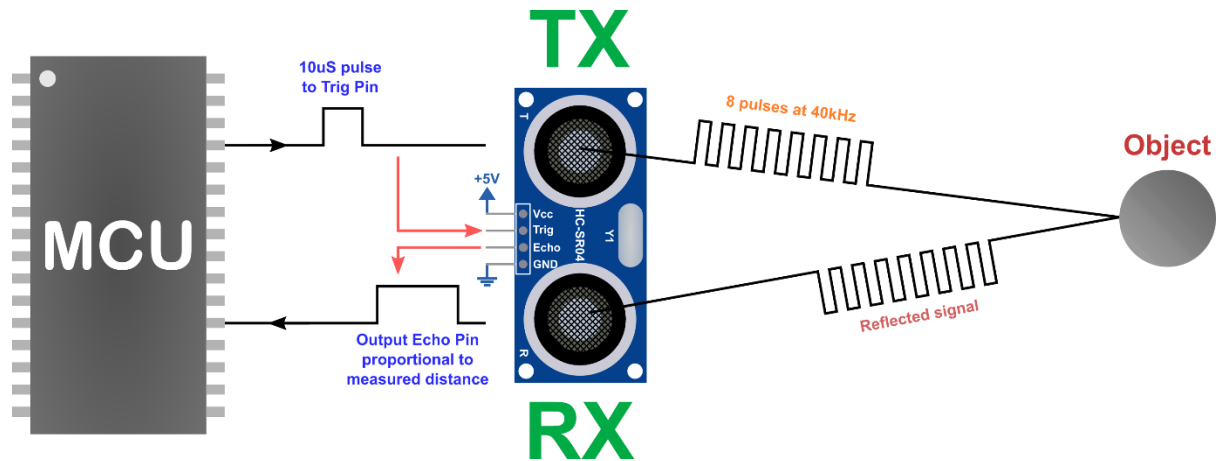
### High Level Requirements

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

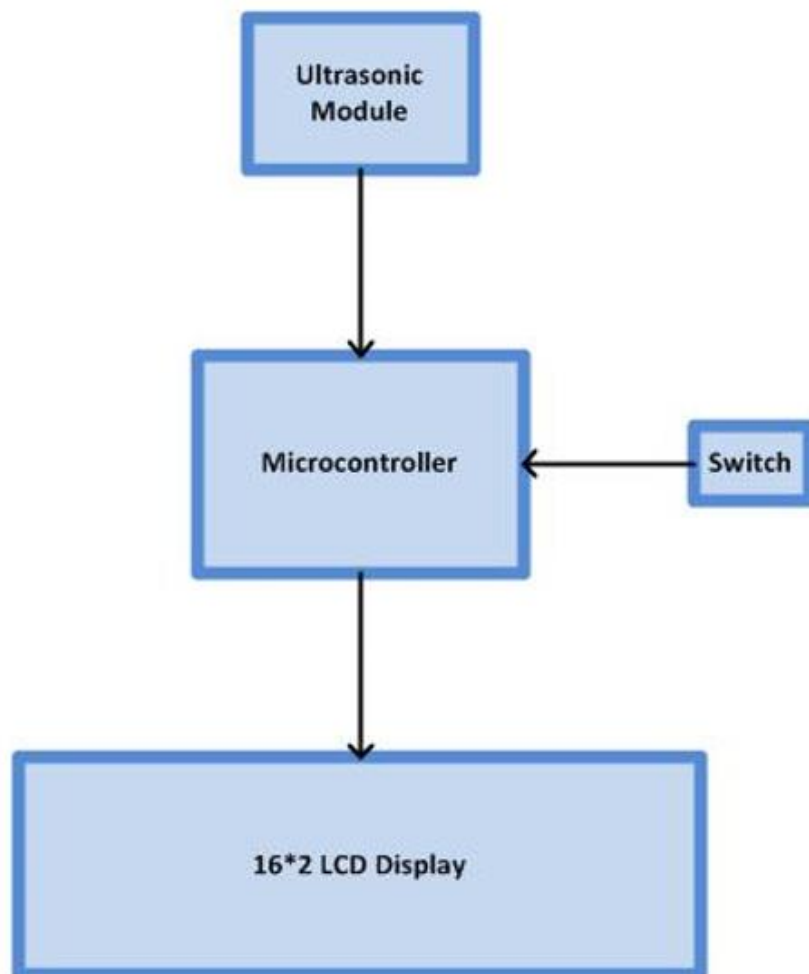
### Low Level Requirements

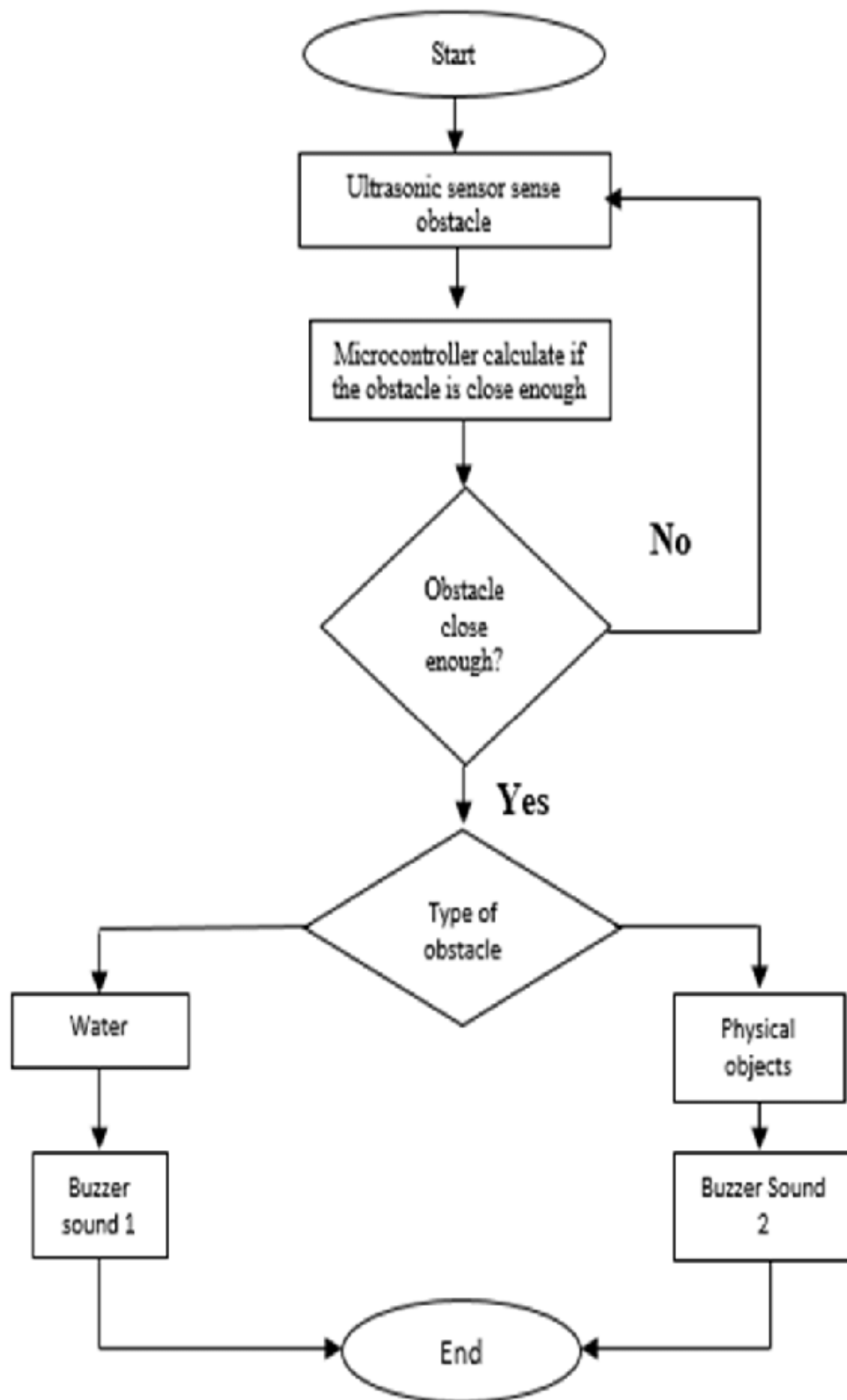
ID	Description
	• Power Supply :+5V DC.
LLR_1_HLR1	• Measuring Angle: 30 degree.
	• Trigger Input Pulse width: 10uS TTL pulse.
LLR_1_HRL1	• Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water.

## Structural Diagram 1



## Behavioural Diagram





## Bill Of Materials:

### a) SOFTWARE :-

#### 1] SimulIDE:

- SimulIDE provides AVR, Arduino and PIC microcontrollers that can be accessed just like other components.
- Features like gpsim and simavr allow you to use PIC and AVR microcontrollers, respectively.
- Editor/Debugger is still in its first stages of development, with basic functionalities, but it is possible to write, compile and perform basic debugging with breakpoints, watch registers and global variables.

#### 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.
- Features:
  - i] SENSING INPUT. Voltage.
  - ii] POWER INPUT (PMG) Voltage.
  - iii] REGULATION. +/- 1% (see note 1)

### b) HARDWARE :-

#### 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.
- ATMEGA328P is an 8-bit microcontroller based on AVR RISC architecture.

#### 2] Sound (HC-SR04 sensor):

- A sound sensor is defined as a module that detects sound waves through its intensity and converting it to electrical signals.
- The HC-SR04 sensor works best between 2cm – 400 cm (1" - 13ft) within a 30 degree cone, and is accurate to the nearest 0.3cm.
- Features:
  - i] Input Voltage: 5V
  - ii] Current Draw: 20mA (Max)
  - iii] Digital Output: 5V
  - iv] Digital Output: 0V (Low)
  - v] Working Temperature: -15°C to 70°C

- vi] Sensing Angle: 30° Cone
- vii] Angle of Effect: 15° Cone
- viii] Ultrasonic Frequency: 40kHz
- ix] Range: 2cm - 400cm

### 3] Display (16x2 lcd):

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

- Features:

- i] The operating voltage of this LCD is 4.7V-5.3V.
- ii] It includes two rows where each row can produce 16-characters.
- iii] The utilization of current is 1mA with no backlight.
- iv] Every character can be built with a 5x8 pixel box.

### 4] Potentiometer:

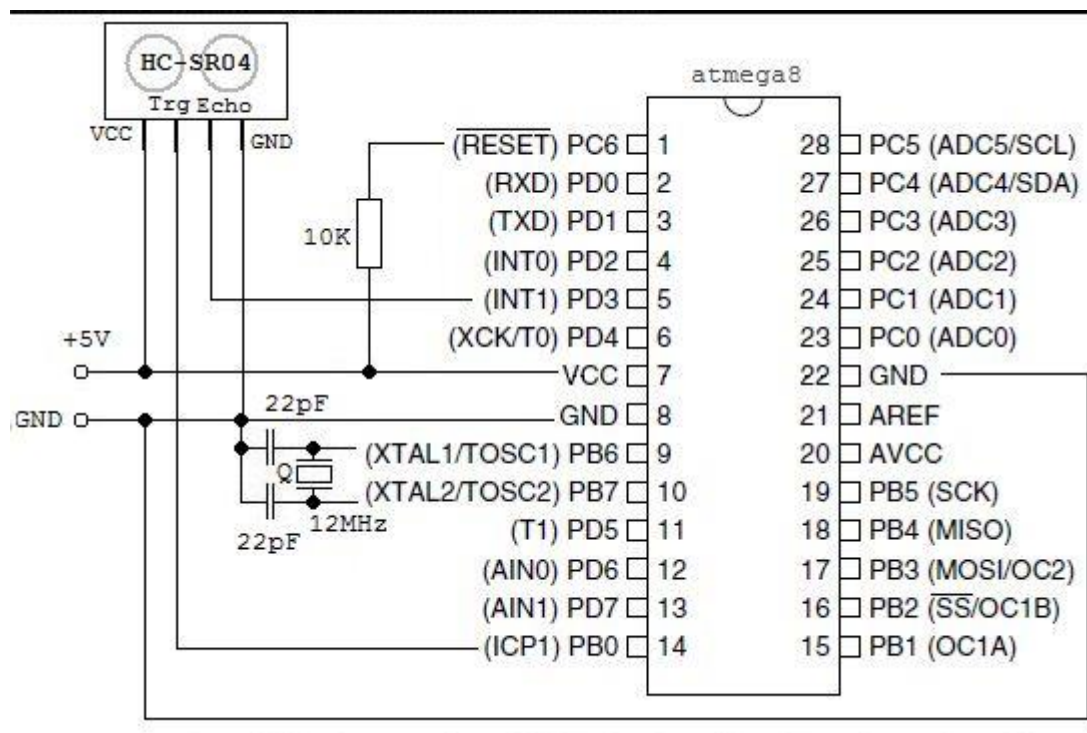
- the input impedance to be an order of magnitude (10 times) higher than the output (source) impedance.

- Potential gradient is calculated as  $K = V/L$ , where V is the voltage across the potentiometer wire and the L is the length of the wire in the potentiometer.

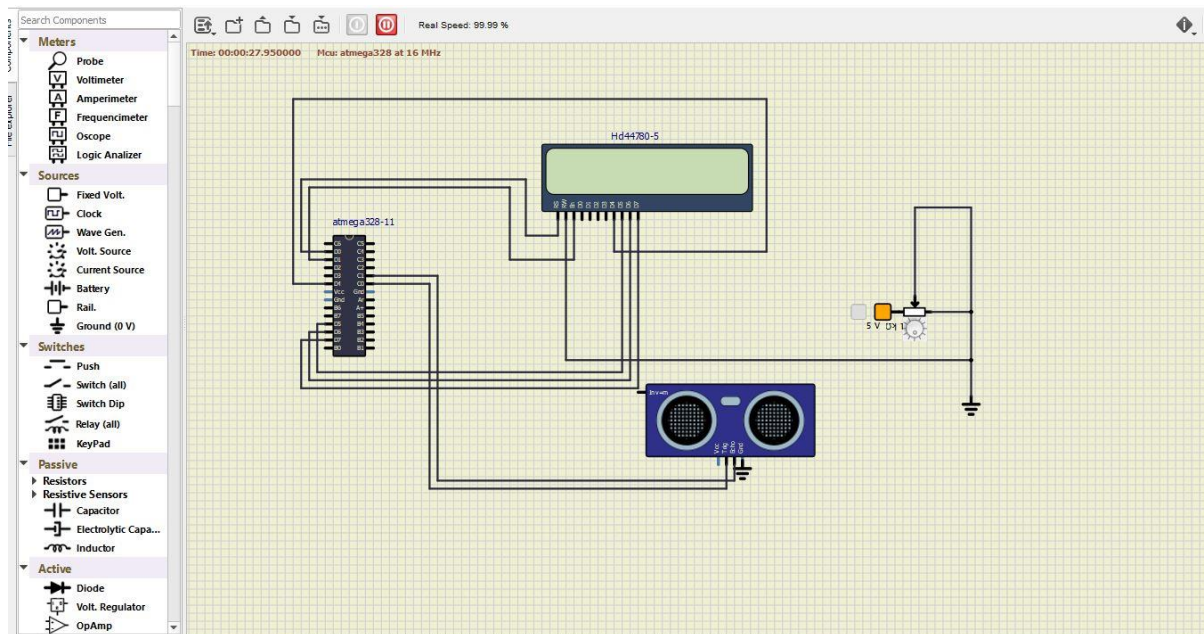
- Features:

- i] Resistance : 1K Ohm
- ii] Potentiometer Type Pot Potentiometer
- ii] Material Carbon Film and Metal
- iv] Interface 3 Pin
- v] Shaft Length 15mm
- vi] Dimensions 1 x 1 x 1cms
- vii] Weight :15 grams

## Circuit Diagram



## Simulation on Simul IDE



## 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)*	18.9	12.5	10.3
SD**	12.1	14.6	13.3
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.*

## Communication Protocols

### Wired

*In this project we use UART communication protocol.*

### UART - TX & RX (2 devices)

- 2 Wire
- Individual clocks used by the both parties
- Standard speed (9600, 115200) - Baud rate
- Parity



## Factors to decide on which communication protocol to select

### \* Frequency :

We need 20KHz frequency for this project because of this UART is suitable for this model.

### \* Data throughput

### \* Number of pins available :

There are two pins are available.

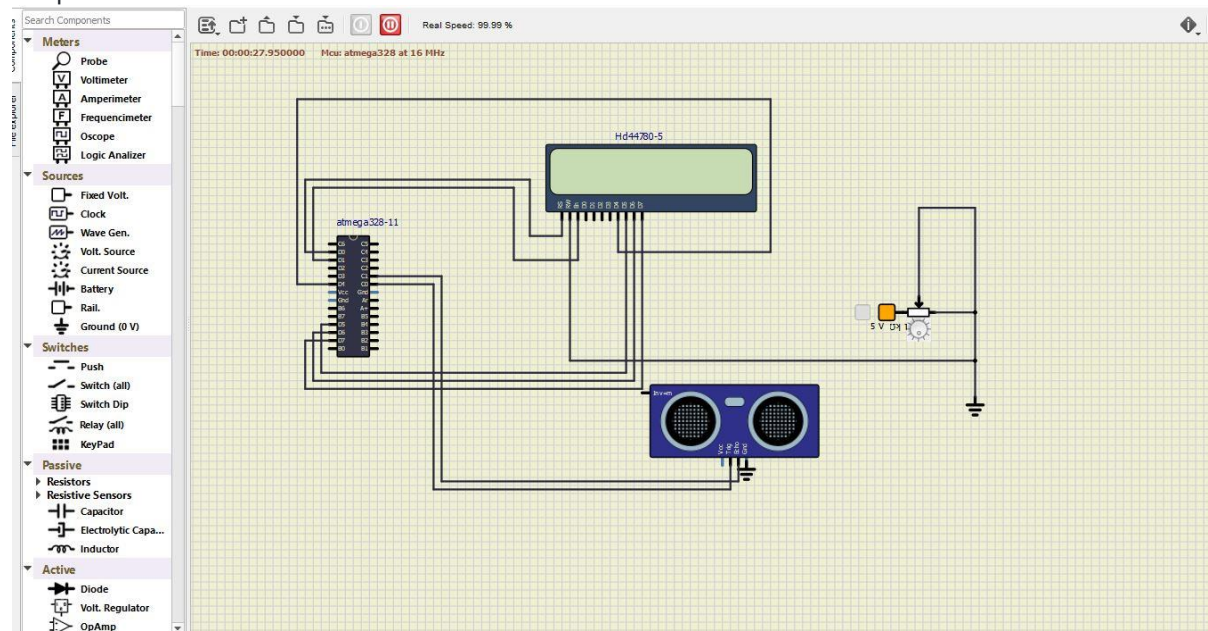
[1] TX (Pin no 3)

[2] RX (Pin no 2).

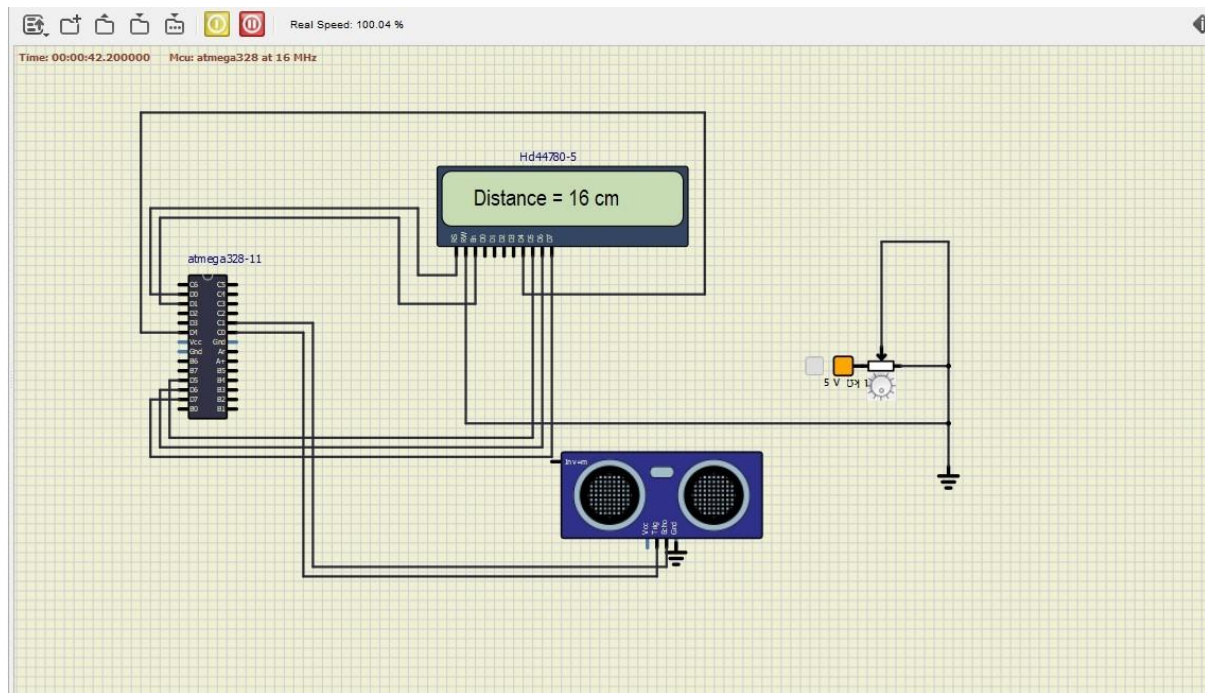
TX and RX pins we need to display our result .

## Output

Output:1



Output:2



## Conclusion

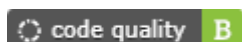
The objective of the project was to design and implement an ultrasonic distance meter. The device described here can detect the target and calculate the distance of the target. The ultrasonic distance meter is a low cost, low a simple device for distance measurement. The device calculates the distance 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

## Implementation and Summary

### Git Link:

Link: [https://github.com/Virajsawant619/M2-Embedded\\_UltrasonicsoundSensor](https://github.com/Virajsawant619/M2-Embedded_UltrasonicsoundSensor)

### Git Dashboard



## Miniproject 3 – Virtual Costume Advisor [Team]

### Modules

1. SDLC
2. Git

### Requirements

#### 4W's and 1 H's

##### Who:

People who want to look good by getting targeted outfit ideas for their body shape.

##### What:

Calculates the body shape and occasion they are addressing then suggests them the best suitable outfit for their body.

##### When:

Anytime they want to get themselves dressed well for particular occasions.

##### Where:

In the Application/system which has this program.

##### How:

By entering the measurements of the individuals bust size, waist size, high hip size, hip size.

### High Level Requirements

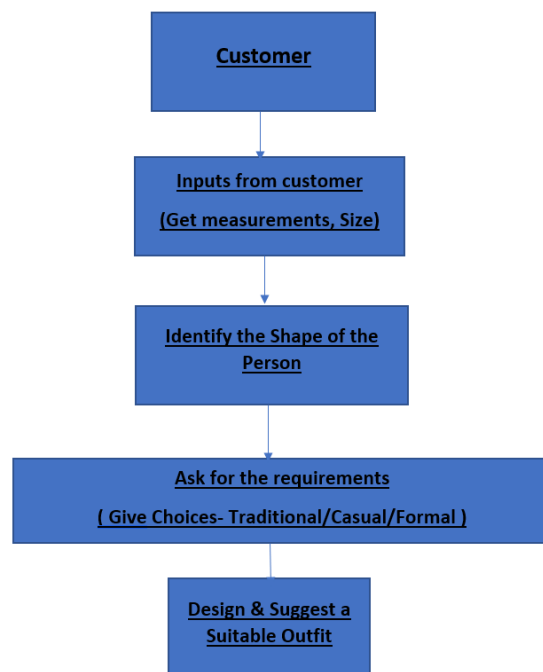
ID	Description	Status
HLR_1	Getting the measurements from the user	Application
HLR_2	Calculating the body type	Vs code
HLR_3	Getting the choice of outfit type from the user	Application
HLR_4	Getting the choice of listed costume from the user	Application

## Low Level Requirements

ID	Description	Platform
LLR_1	The measurements should be properly taken and entered correctly by the user	Application
LLR_2	Coding formula to calculate body type should be accurate	Vs Code
LLR_3	The Choice of outfit type should be properly Chosen and entered correctly by the user	Application
LLR_4	The Choice of costume should be properly taken and given correctly by the user	Application

## Design

### Behavioural Diagram Low Level



## Behavioural Diagram High Level

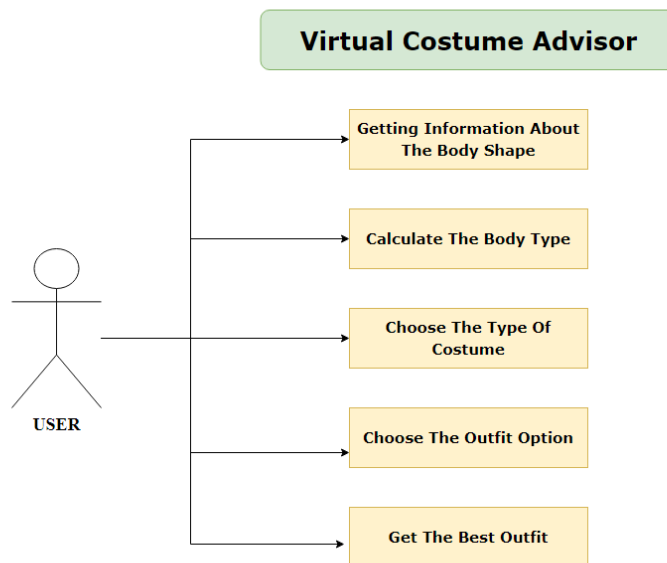
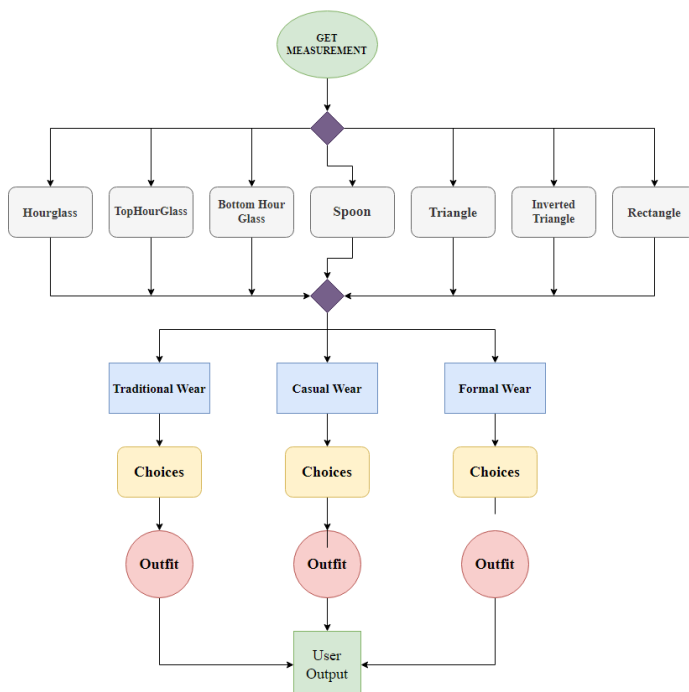


Figure 2 Behavior Diagram

## Structural Diagram Of High Level



## Structural Diagram Of Low Level

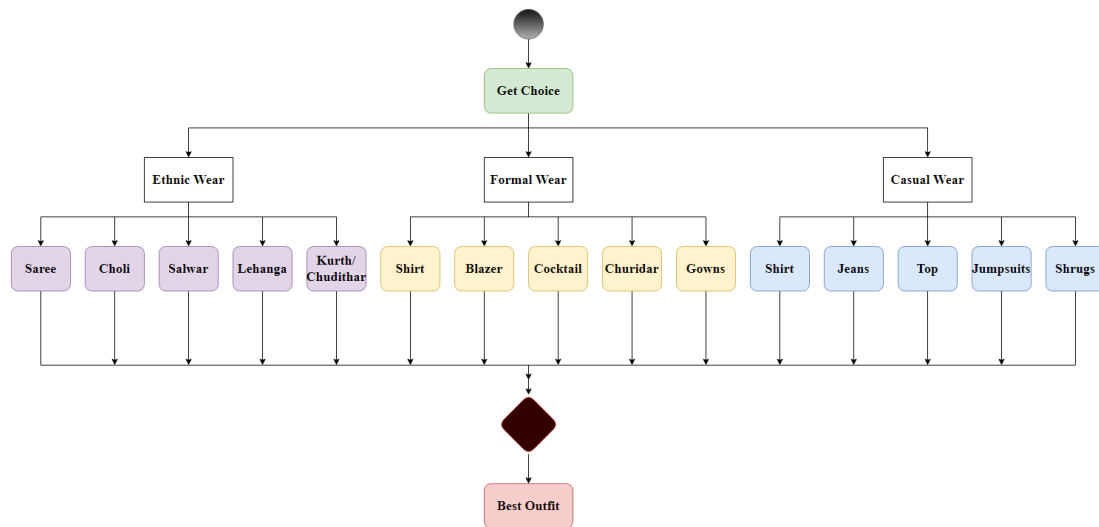


Figure 3 Structure Diagram

## Test Plan

## High Level Test Plan

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
HLTP_1	Create NFT	Click	SUCCESS	SUCCESS	Requirement Based
HLTP_2	Sell NFT	Click	SUCCESS	SUCCESS	Requirement Based
HLTP_3	Bid NFT	Click	SUCCESS	SUCCESS	Requirement Based
HLTP_4	Buy NFT	Click	SUCCESS	SUCCESS	Requirement Based
HLTP_5	Contact	Click	SUCCESS	SUCCESS	Requirement Based

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
HLTP_6	Sign In	Click	SUCCESS	SUCCESS	Requirement Based
Test ID	Description	Exp I/P	Exp O/P	Type of Test	Test ID

### Low Level Test Plan

ID	Description	Expected I/P	Expected O/P	Actual O/P	Type Of Test
LLTP_1	Connect Wallet	Click	SUCCESS	SUCCESS	Requirement Based
LLTP_2	Activity	Click	SUCCESS	SUCCESS	Requirement Based
LLTP_3	Forgot password	Click	SUCCESS	SUCCESS	Requirement Based
LLTP_4	To check whether none of the fields should be empty	Empty value in the input module	Prompt message mandatory field missing	SUCCESS	Requirement Based
LLTP_5	E-mail ID should be in the perfect format i.e. <a href="mailto:group2@gmail.com">group2@gmail.com</a>	<a href="mailto:group2@gmail.com">group2@gmail.com</a>	Prompt message invalid E-mail ID	SUCCESS	Requirement Based

## Implementation and Summary

### Git Link:

Link: [https://github.com/GENESIS2021Q1/Applied\\_SDLC-Dec\\_Team\\_50](https://github.com/GENESIS2021Q1/Applied_SDLC-Dec_Team_50)

## Individual Contribution and Highlights

### Summary

1. requirements contents, research on the topic,
2. added the test plan, HLR and LLR of project
3. Help to do code

## Miniproject 4 – Calendar Automation [Team]

### Modules

1. Python
2. Git

### 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



ID	Feature	MATLAB v0 Status	Python v0 Status
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

### 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

ID	Feature	High Level ID	MATLAB v0 Status	Python v0 Status
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
LR10	Master Calendar: Course code correction	HR02	Implemented	Not Available
LR11	Master Calendar: Course title correction	HR02	Not Available	Not Available
LR12	Master Calendar: display the dates that were not analysed	HR02	Implemented	Not Available
LR12	Faculty Calendar: display Month wise	HR03	Implemented	Implemented
LR13	Faculty Calendar: display Initiative wise	HR03	Implemented	Not Available
LR14	Faculty Calendar: Appending	HR03	Implemented	Not Available
LR15	Faculty Calendar: Differentiate Initiatives (Color Codes/Numbers)	HR03	Implemented	Implemented
LR16	Faculty name correction/validation in faculty calendar	HR03	Not Available	Not Available
LR17	Faculty Calendar: Highlight conflicts (Red highlight/pop-up/Concatenated Numbers)	HR03	Not Available	Not Available

ID	Feature	High Level ID	MATLAB v0 Status	Python v0 Status
LR18	Faculty Load Sheet: display Month wise	HR04	Implemented	Not Available
LR19	Faculty Load Sheet: display Initiative wise	HR04	Implemented	Not Available
LR20	Faculty name correction/validation in Faculty Load Sheet	HR04	Not Available	Not Available
LR21	Faculty Load Sheet: Display Available Slots Faculty wise	HR04	Implemented	Not Available
LR22	Faculty Load Sheet: Warn User if Available Slots goes Negative	HR04	Not Available	Not Available
LR23	Faculty load insight : OVERLOAD, UNDERLOAD, OPTIMUM	HR04	Not Available	Not Available
LR24	Faculty Load Sheet: Appending	HR04	Implemented	Not Available
LR25	Let User know that the Output has been Successfully Updated	HR02, HR03, HR04	Implemented	Implemented
LR26	Validate correct number of days in month	HR02, HR03	Not Available	Not Available
LR27	Let the User/Faculty book their Slots Themselves	HR05	Not Available	Not Available
LR28	Accessible by everyone with login Credentials	HR06	Not Available	Not Available

ID	Feature	High Level ID	MATLAB v0 Status	Python v0 Status
LR29	Bar graph indicating the number of slots planned for each faculty	HR07	Not Available	Not Available
LR30	Bar Graph indicating the sessions per initiative	HR07	Not Available	Not Available
LR31	Pie chart indicating the % of sessions being taken according to initiatives	HR07	Not Available	Not Available

## Implementation and Summary

### Git Link:

Link: [https://github.com/Ramki17/Calendar\\_Automation-Genesis21\\_Team49](https://github.com/Ramki17/Calendar_Automation-Genesis21_Team49)

## Individual Contribution and Highlights

1. Improved implementation of Python Programming
2. Source code management using GitHub

## **Mini project 5 –Team BMW [Team]**

**Module: - Applied Model Based Design Module**

**Individual Topic: - Car Suspension System**

### **Requirements**

## **INTRODUCTION**

Simply put, it is part of a car that opposes the amount of power a car gets from driving on the road ensuring that the cabinet stays steady. It could be small stones on the road to big pits, suspension deals and all. This is a common misconception that the function of suspension is to provide a pillow only when a bump or crack appears on the road. It does more than that. In fact, it makes driving easier.

## **OVERVIEW**

The car suspension system refers to a group of mechanical components which connects the wheels to the frame or body. Great engineering effort engaged in the construction of suspension systems as a result of the relentless efforts of this improve vehicle mobility and grip and safety and comfort for passengers. Kwe days of horse and buggy, the suspension system simply consisted of a beam (axle) extended across the width of the car. Previously, wheels were available attached to the edge of the axle and the ax rotated in the center to give

to direct. Early cars used a single-axle axle but instead circled to the center, fixed to the car using springs to provide a reduction in the burden of shock due to road failure. The wheels were like that mounted horizontally at the edge of the axis to provide direction. First springs

ID	Description
<b>HLR1</b>	Satisfactorily absorb large and small road impacts to protect the vehicle occupants from shocks
<b>HLR2</b>	Maintain an even keel for the vehicle body when travelling over rough ground or when cornering, to minimize pitch and roll
<b>HLR3</b>	Reduce the impact stress on the various mechanisms of the vehicle.

it is made up of thin layers of thin metal strips that are joined together elliptical shape and was called leaf springs. In the later installation, the fountains of leaves they were replaced by coil springs. In cars behind the engine, front beam the axle has been replaced by self-propelled mounted wheels. The wheels were like that they are supported by short upper and lower wing arms that hold them straight road as does the design of the previous axle axle.

## REQUIREMENTS

### High Level Requirements: -

### Low Level Requirements: -

ID	Description
<b>LLR1</b>	Reduce the unsprung mass (that is the part of the total mass, which is not suspended to a minimum to allow the wheels to follow the contour of the road surface more closely
<b>LLR2</b>	Safeguard the occupants from road shocks
<b>LLR3</b>	Provide good road holding while driving, cornering and braking.

## Analysis and Physics

The front and rear suspension are modelled as spring/damper systems. A more detailed model would include a tire model, and damper nonlinearities such as velocity-dependent damping (with greater damping during rebound than compression). The vehicle body has pitch and bounce degrees of freedom. They are represented in the model by four states: vertical displacement, vertical velocity, pitch angular displacement, and pitch angular velocity. A full model with six degrees of freedom can be implemented using vector algebra blocks to perform axis transformations and force/displacement/velocity calculations. **Equation** describes the influence of the front suspension on the bounce (i.e. vertical degree of freedom):

$$F_f = 2K_f(L_f\theta - (z + h)) + 2C_f(L_f\dot{\theta} - \dot{z})$$

where:

$F_f, F_r$  = upward force on body from front/rear suspension

$K_f, K_r$  = front and rear suspension spring constant

$C_f, C_r$  = front and rear suspension damping rate

$L_f, L_r$  = horizontal distance from gravity center to front/rear suspension

$\theta, \dot{\theta}$  = pitch (rotational) angle and its rate of change

$z, \dot{z}$  = bounce (vertical) distance and its rate of change

$h$  = road height

**Equations 2** describe pitch moments due to the suspension.

$$M_f = -L_f F_f$$

$$F_r = -2K_r(L_r\theta + (z + h)) - 2C_r(L_r\dot{\theta} + \dot{z})$$

$$M_r = L_r F_r$$

where:

$M_f, M_r =$  Pitch moment due to front/rear suspension

**Equations 3** resolves the forces and moments result in body motion, according to Newton's Second Law:

$$m_b \ddot{z} = F_f + F_r - m_b g$$

$$I_{yy} \ddot{\theta} = M_f + M_r + M_y$$

where:

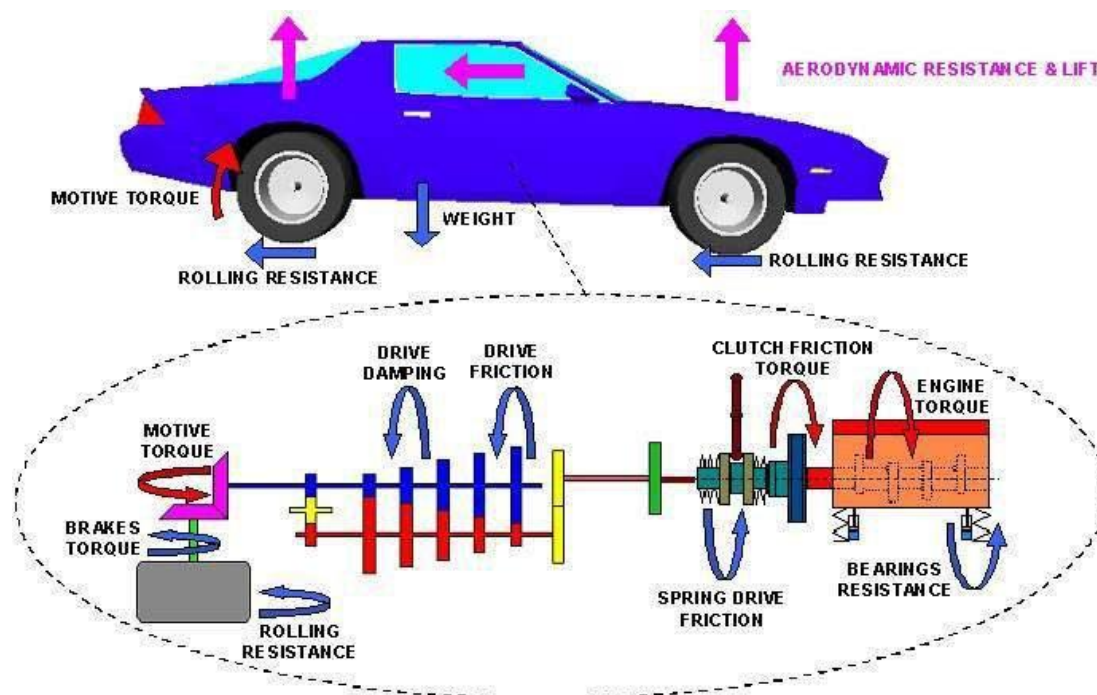
$m_b =$  body mass

$M_y =$  pitch moment induced by vehicle acceleration

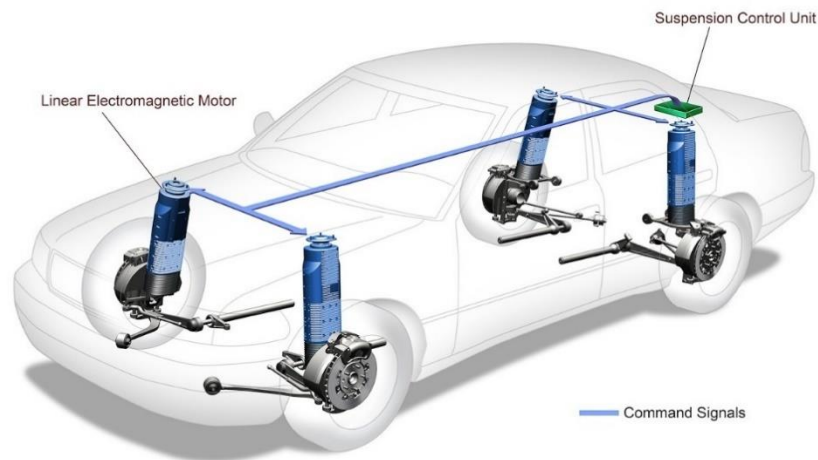
$I_{yy} =$  body moment of inertia about gravity center

## Model

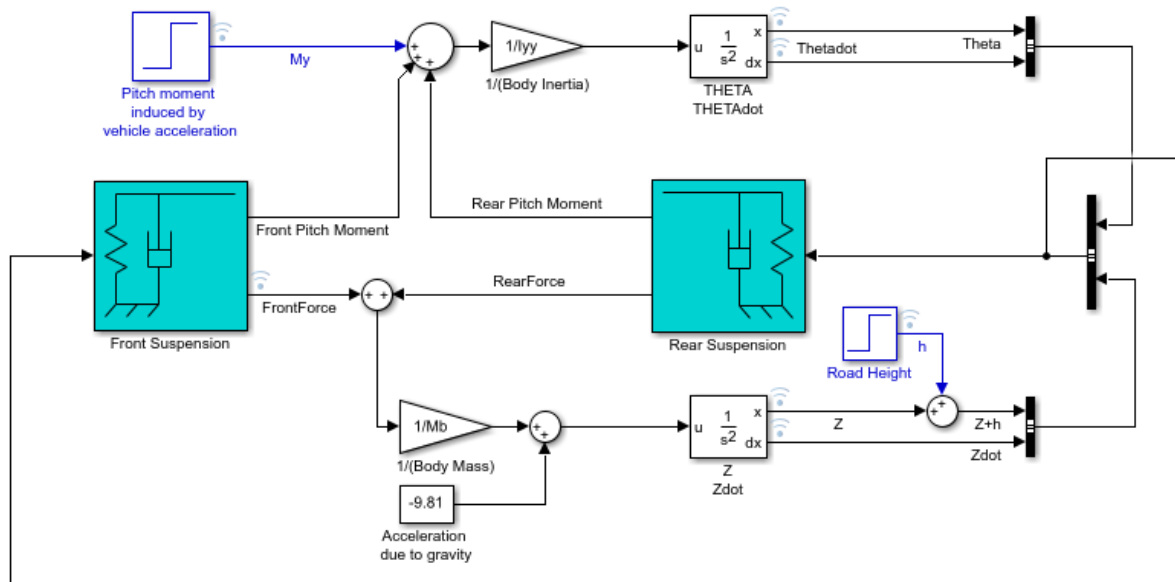
# DESIGN

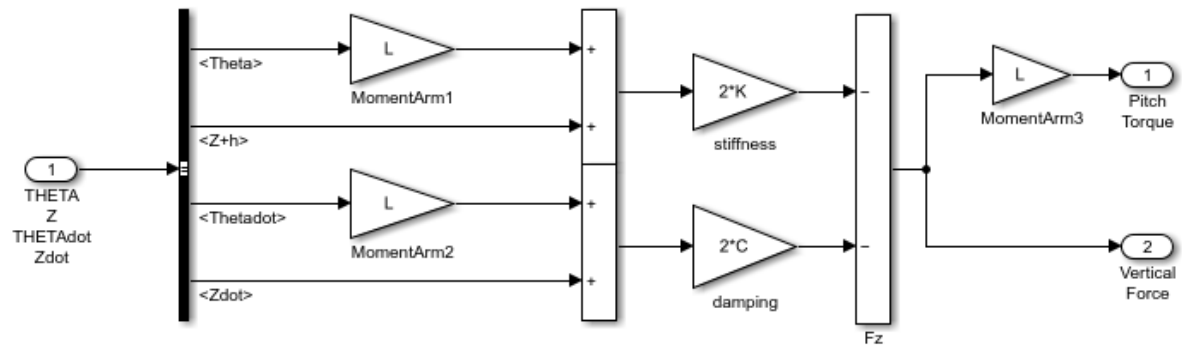






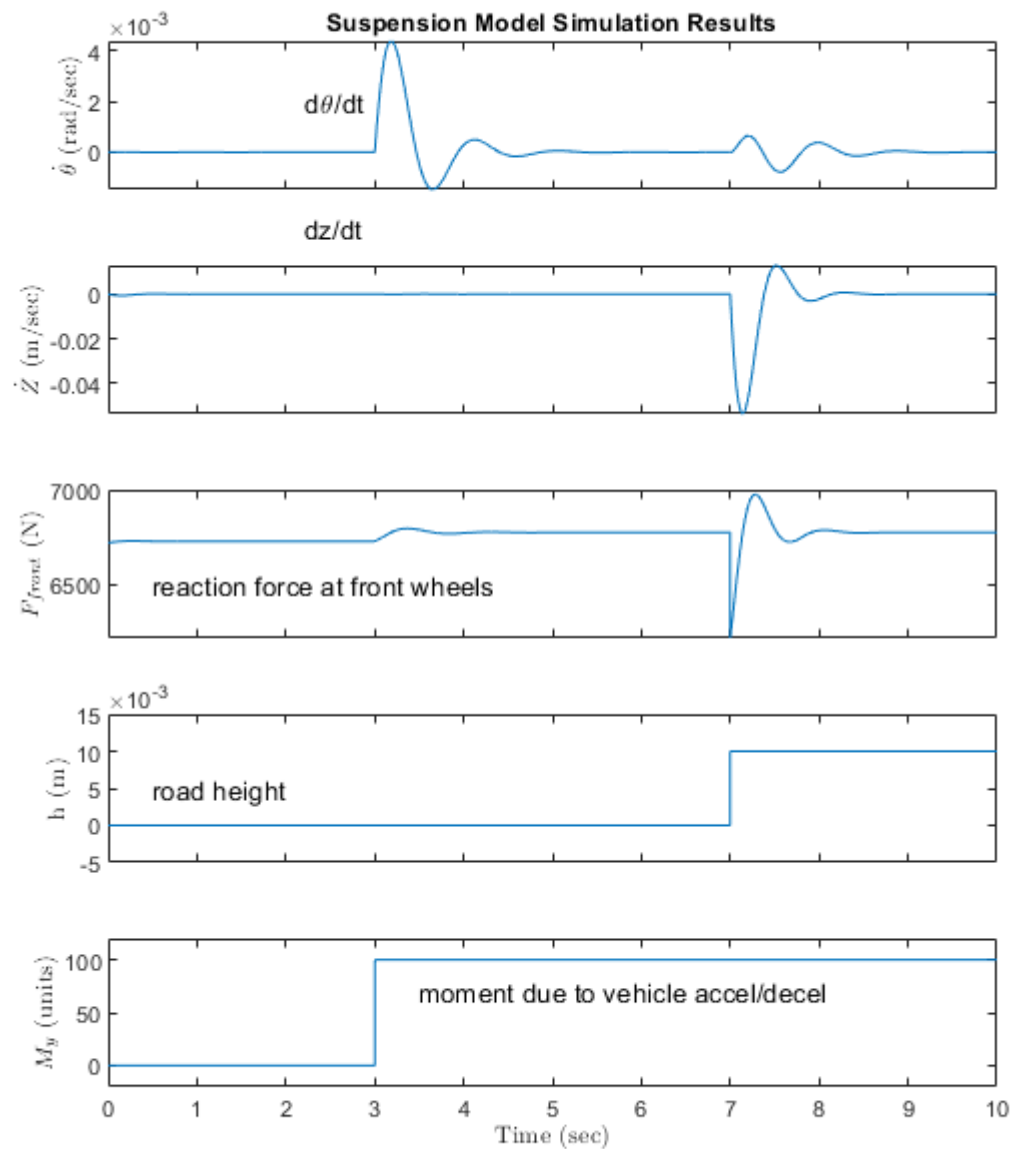
## MODELING





## OUTPUT

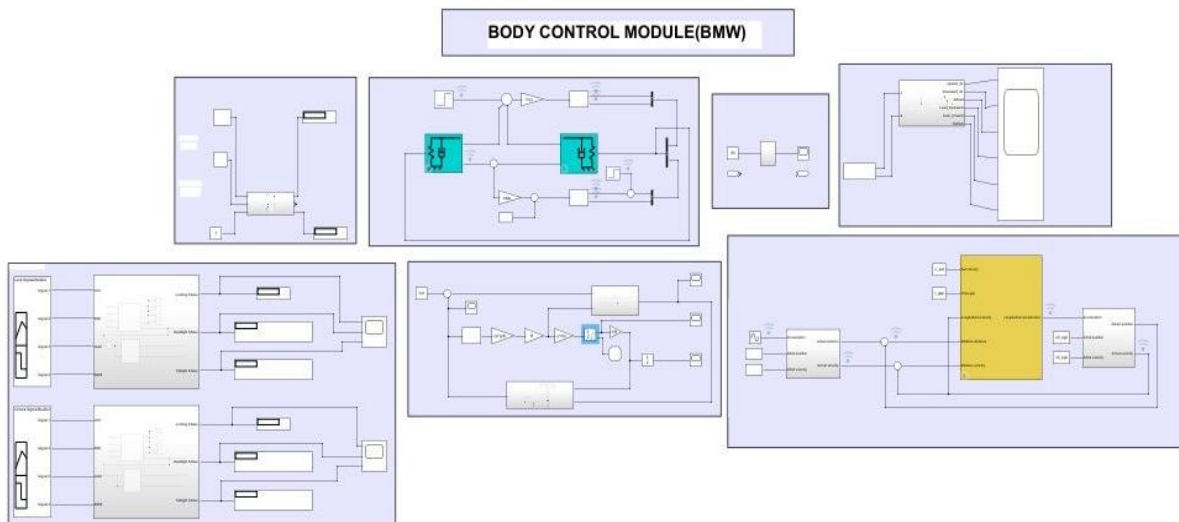
## Running the Simulation



## Conclusion

This model allows you to simulate the effects of changing the suspension damping and stiffness, thereby investigating the tradeoff between comfort and performance. In general, racing cars have very stiff springs with a high damping factor, whereas passenger vehicles have softer springs and a more oscillatory response..

## Team Activity



## Miniproject 6 – Wiper Control [Team]

**Modules**

1. C Programming
2. STM32

**4W's and 1H****Who:**

A wiper control system for an car wiper. It is mainly used in cars.

**Where:**

It is implemented through STM32F4 microcontroller in embedded system.

**When:**

Whenever the water hit a devoted sensor that found on windscreen, it'll send a flag to move on the wiper motor. Once water isn't identified by sensor, the wiper will naturally halt. This will offer assistance the driver to donate more concentration and decrease the car mishap probability.

**What:**

Vehicles are presently accessible with driver-programmable brilliantly windscreen wipers that identify the presence and sum of rain employing a rain sensor.

**How:**

Windshield wipers are controlled by the stalk on the proper side of your controlling wheel. Essentially moving the stalk down will turn your windshield wipers on. Moving the stalk down will turn your you wipers on.

**High Level Requirements**

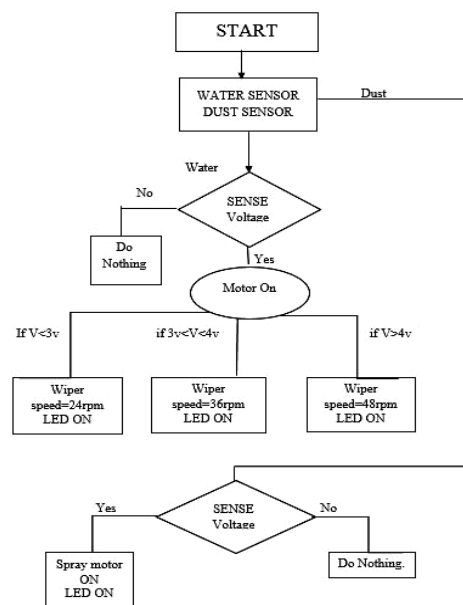
id	Description	Status
HLR1	A windscreen wiper is a device used to do away with rain, snow, ice and particles from a windscreen or windshield.	Implemented

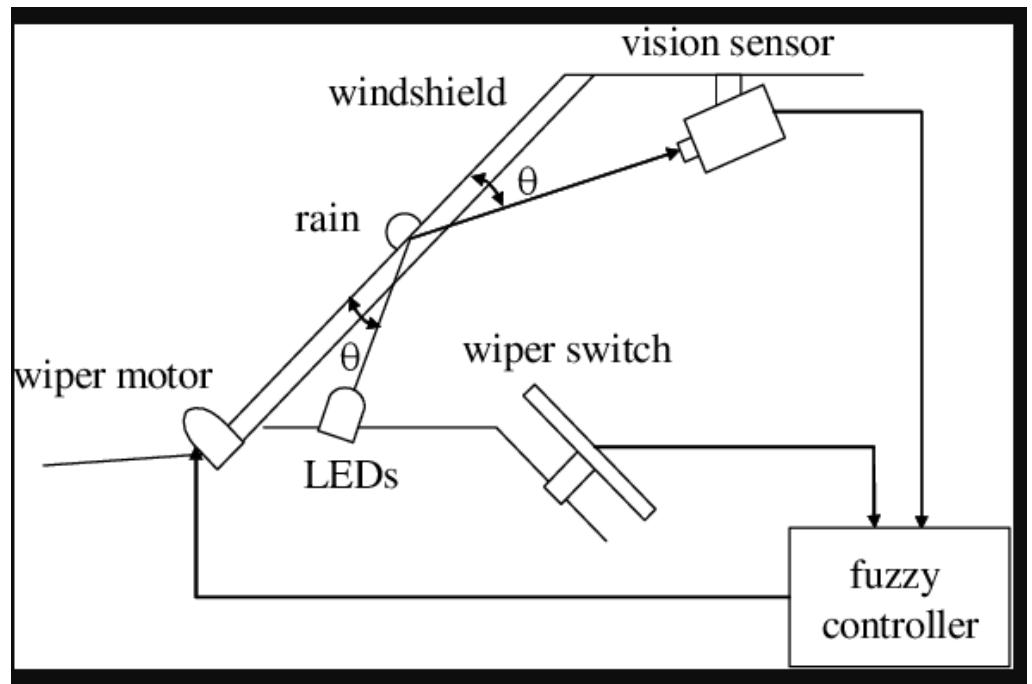
id	Description	Status
HLR2	The quality and reliability wiper systems meet the highest technical requirements and are the basis for vehicles with state-of-the-art capabilities.	Implemented
HLR3	Our assignment brings ahead this device to automate the wiper gadget having no need for manual intervention.	Implemented

### Low Level Requirements

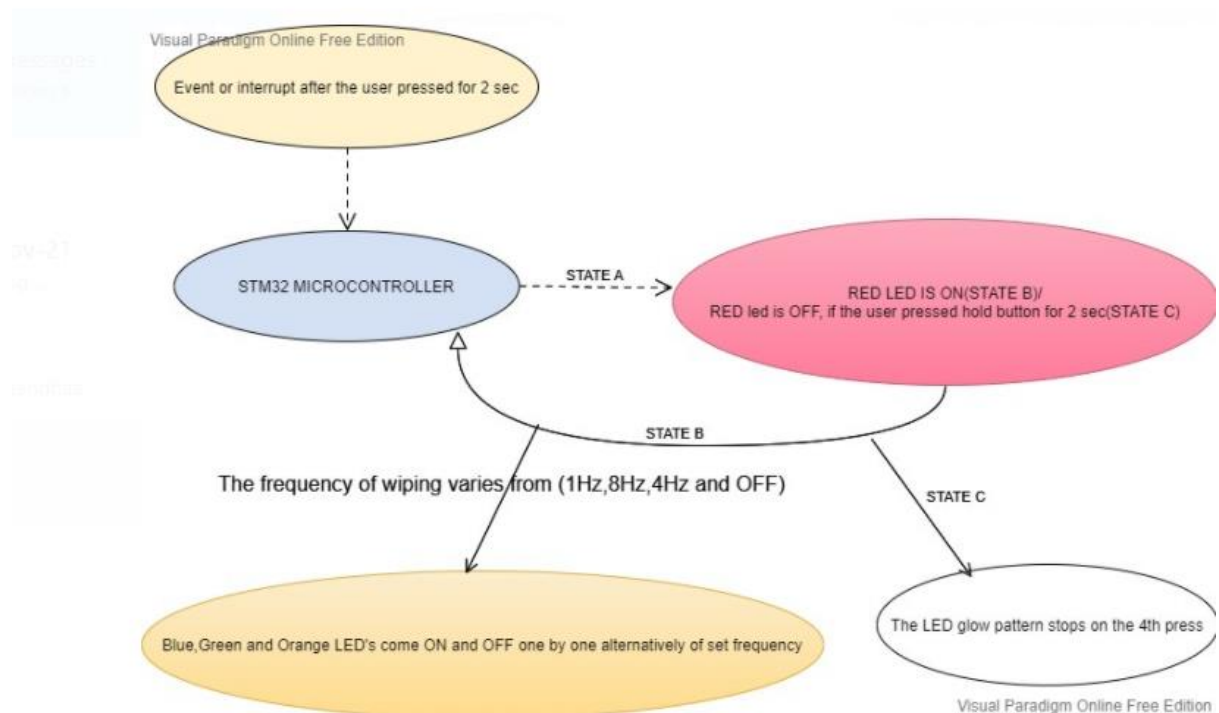
id	Description	Status
LLR1	Lower degree parsing , below the hood, Requirement elegance does maximum of the heavy lifting. magnificence necessities.	Implemented
LLR2	These structures discover droplets of rain at the windshield and robotically activate and adjust the wiper machine	Implemented
LLR3	These structures discover droplets of rain at the windshield and robotically activate and adjust the wiper machine.	Implemented

### Design: Flowchart Diagram





## Usecase Diagram



**TEST PLAN:****High level test plan**

HLR_ID	Description	Expected i/p	Expected o/p	Actual o/p	Test type
HLR_1	Check Every is working or not	Call the function	All functions working correctly	All functions working properly	Requirement
HLR_2	Checking individual functionality working	Call individual function	Individual function is called	Individual function is called	Scenario
HLR_3	Check For not available function	Choosing different values	No response from the program	No response from the program	Boundary
HLR_4	Rain Sensing Devices	When Rain fall	Automatic Wiper On	not Implemented	Future

**Low level test plan**

LLR_ID	Description	Expected i/p	Expected o/p	Actual o/p	Test type
LLR_1	Testing system is ON or Off by pressing button for 2 seconds	Press button for 2 Seconds	RED Led will glow	RED Led will glow	Requirement



LLR_ID	Description	Expected i/p	Expected o/p	Actual o/p	Test type
LLR_2	Testing the modes of the Wiper control system	Pressing the button one time	Blue Led will glow	Blue Led will glow	Requirement
LLR_3	Testing the modes of wiper control system	Pressing the button for 1 second	Blue will Off and Green will be glow	Blue will Off and Green will be glow	Requirement

## Implementation and Summary

**Git Link:** <https://github.com/GENESIS-2022/MasteringMCU-Team66>

## Individual Contribution and Highlights

1. Wiper System using C Programming
2. Source code management using GitHub

### Role in Project Team

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

## Miniproject 7 – BMW Project [Team]

### Modules

1. Automotive Systems
2. Git

### Requirements

#### BMW X5

The BMW X5 is a mid-size luxury crossover SUV produced by BMW. The X5 made its debut in 1999 as the E53 model. It was BMW's first SUV. At launch, it featured all-wheel drive and was available with either a manual or automatic gearbox.

### Body Control Module

#### Features:

- Door Lock System
- Interior Light Control
- Power Mirror
- Power Window

#### Individual Topic :- Door Lock System

### Introduction

The most commonly used system for locking and unlocking the door is a lock and a physical key. The entire process is a mechanical one. ... The RFID card reader unit is installed near the door. When the card is brought near the reader, it identifies the radio frequency of the card and thus verifies the key.

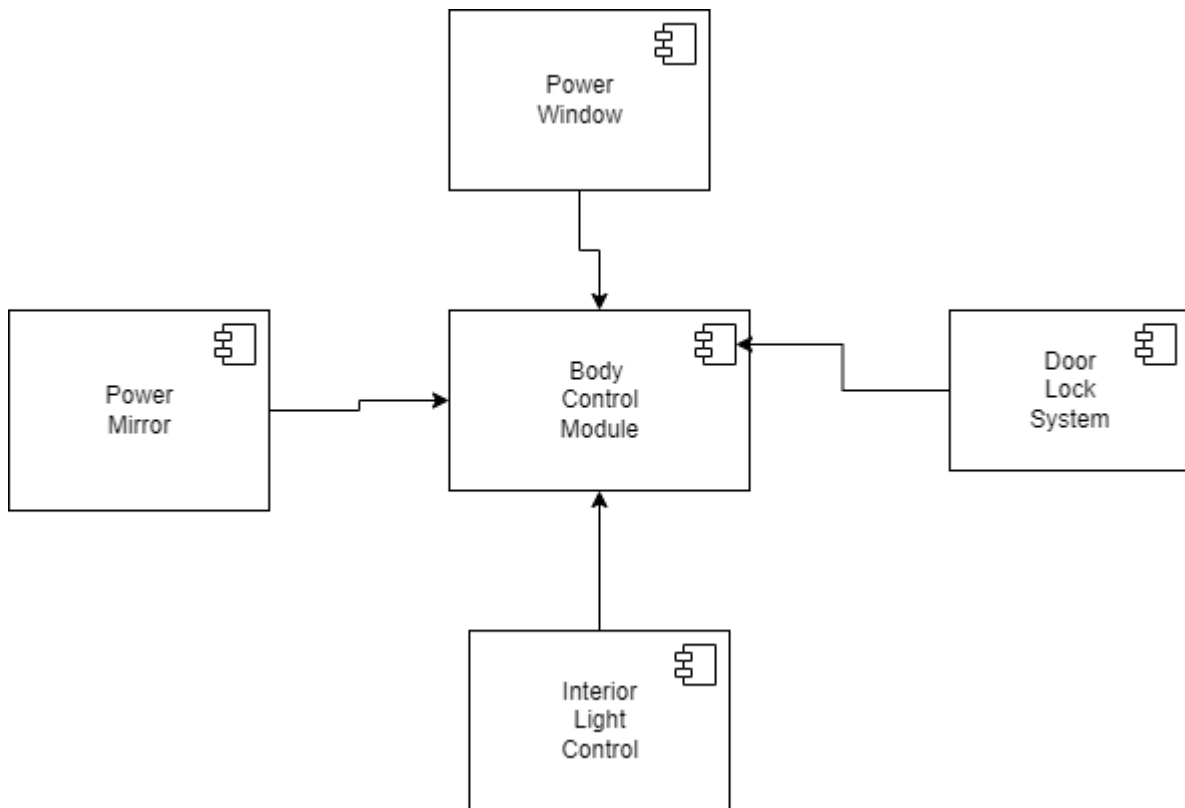
### DOOR LOCKING SYSTEM:

High level Requirement	Description
HLR1	The lock button on the key should lock all the doors of the car.

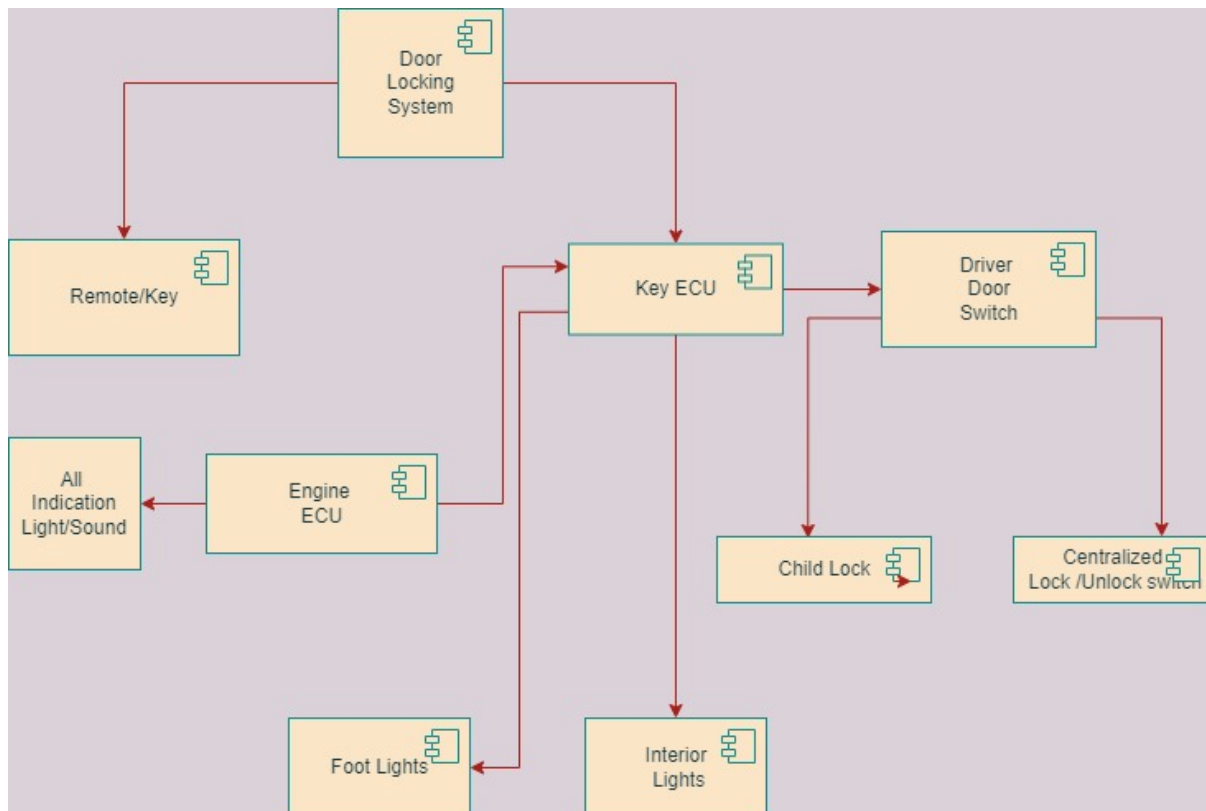
High level Requirement	Description
HLR2	The unlock button on the key should unlock all the doors of the car.
HLR3	After locking exterior lights of the car should get pop up for 3 seconds
HLR4	Exterior lights should get pop up after unlocking the car.
HLR5	When unauthorized person tries to get into your vehicle by breaking the door their will be a beeping sound.
HLR6	If the car is in motion and any door is open the system will display an indication.
Low level Requirement	Description
LLR1	Status of the door should have to be visible on the display
LLR2	Locking and the exterior light pop up should be in synchronised manner
LLR3	Unlocking and the exterior light pop up should be in synchronised manner
LLR4	If any door is opened it should indicate on the display

## Design

### Body Control Module



## Door Locking System



## Implementation and Summary

### Git Link:

Link: [https://github.com/Nayan349/Automotive\\_BMW\\_Project](https://github.com/Nayan349/Automotive_BMW_Project)

## Mini project 8 – EV Car[Team]

### Modules

1. Matlab
2. Matlab Script

### Requirements

Requirements

#### 1.Battery:

1.Battery type used in this EV is Lithium-ion Polymer.

2.Lithium is also the lightest of all metals. However, lithium-ion (Li-ion) batteries contain no lithium metal, they contain ions. For those wondering what an ion is, an ion is a an atom or molecule with an electric charge caused by the loss or gain of one or more electrons.

#### Battery Features:

3.Lithium-ion batteries are one of the most popular forms of energy storage in the world, accounting for 85.6% of deployed energy storage systems.

#### 2.Motor:

1.BLDC motors are used, which have traction characteristics like high starting torque, high efficiency around 95 98%,etc.

2.BLDC motors are suitable for high power density design approach. The BLDC motors are the most preferred motors for the electric vehicle application due to its traction characteristics.

### **Motor Features:**

3. Electronically commutated
4. High energy, rare earth magnets used for rotor field
5. Requires speed control with 6-lead connection (3 power, 2 Hall Effects, 1 drain)
6. Rated speed 2500 RPM; minimum 150-200 RPM
7. Linear speed torque curves
8. Built-in tach pulse for economical speed readout
9. Encoder options for servo performance.

### **3.Controller:**

1. Programmable BLDC motor controller provides efficient smooth and quiet control for electric vehicles which operate 36V/48V battery system.

### **Controller Features:**

1. E-Braking will release the motor from the controller as long as brake is applied.
2. Over current protection to protect controller during faulty conditions or short circuit.
3. Low voltage detection ensures Battery life.
4. Pedal assist mode controls the motor speed based on the speed of peddling.
5. Accelerator fault protection to prevent runaway.

6.Provision for 120°/ 60° Selection.

7.Speedometer output.

8.Brake inputs with high active and low active provision.

9.Speed limit control provision.

#### **4.Inverter :**

1.The traction inverter converts energy from the vehicle's battery in order to drive the motors in the drivetrain. This key component has a direct impact on road performance, driving range and reliability of the vehicle also as a consequence of their weight and size.

2.Subject to all the possible stress found in a road vehicle from heat and vibrations, these converters must be able to handle high power and currents along with the associated Electro Magnetic Compatibility (EMC) challenges as well as provide fail-safe operation to ensure dependability and safety for the driver and passengers.

3.To help developers increase the automotive inverter's power efficiency and reduce size and weight, ST has a wide offer of discrete semiconductors including AEC-Q101 qualified IGBTs and both silicon and silicon-carbide (SiC) MOSFETs and diodes, AEC-Q100 qualified galvanically isolated IGBT and MOSFET gate drivers and SPC5 32-bit automotive microcontrollers for designing scalable, cost-effective and energy-efficient EV traction inverter solutions.

#### **Inverter features :**

4.Traction inverters are typically capable of transferring power in the 20 to 100 kW range, with switching voltages in the 200 V to 800 V range and currents in the hundreds of amperes.



## Analysis

Specification	MG ZS EV	Hyundai Kona Electric
Model Price	21.08 lakh Rs	23.79 lakh Rs
Gross combination weight	1609 kg	2020 kg
Range	419 km/Charge	452 Km/Charge
Battery type	lithium-ion	Lithium-ion Polymer
Max torque	350Nm	395 Nm
Power	140.8bhp	134.14bhp
Charging time	18-19 hrs	7 h 30 min
Electric range	340 km	305 km
Wheel base	2585 mm	2600 mm

## Research

### Model 1 MG ZS EV

#### Specifications

- Model Price- 21.08 lakh Rs
- Gross combination weight -1609 kg
- Range - Certified range of 419 km
- Battery type used -lithium-ion
- Battery - 44.5 kWh 394 V lithium-ion
- Max torque - 350Nm@5000rpm

- Power – 140.8bhp
- Charging time - 18-19 hrs for fully charge
- Drive line- ZS EV is front wheel drive and can accelerate from 0 to 62 miles per hour in 8.2 seconds
- Motor- Permanent Magnet Synchronous Electric Motor (EV)
- Engine -72kWh battery and single e-motor ,154bhp,206lb ft
- Transmission -5-speed manual 4-speed automatic 6-speed automatic 6-speed DCT CVT e-CVT (EV)
- Electric range -211 mi (340 km) (claimed) 163 mi (262 km) (WLTP)
- Rear suspension -Torsion beam
- Front suspension -MacPherson Strut
- Wheel base – 2585 mm

## Hyundai Kona Electric

### Specifications

- Model price – 23.79 lakh Rs
- Gross combination weight – 2020 kg
- Range - 452 Km/Charge
- Battery type used – Lithium-ion Polymer
- Battery - 39.2 kWh 327 V lithium polymer
- Max torque - 395 Nm (40.27 Kgm)
- Power - 134.14bhp
- Charging time - 7 h 30 min for full charge
- Drive line – trim has front-wheel drive and the same powertrain
- Motor - Permanent Magnet Synchronous Motor (PMSM)
- Engine – a single-speed transmission and a 201-horsepower 150-kW electric motor.
- Transmission - a single-speed transmission and Automatic Single Speed Reduction Gear
- Electric range - Maximum range 305 km for the 39.2 kWh battery version

- Rear suspension - McPherson Strut
- Front suspension - multi-Link
- Wheel base – 2600 mm

### Model 3 EV car

- Model weight – 1700-1800 kg
- Range – 450-490 Kg/Charge
- Battery type used – Lithium-ion Polymer
- Battery -44.5 kWh
- Max torque - 395 Nm
- Charging time - 7 h 30 min for full charge
- Motor- Permanent Magnet Synchronous Electric Motor (EV)
- Rear suspension -Torsion beam
- Front suspension -MacPherson Strut
- Wheel base – 2585 mm

### Implementation and Summary

Submission: Submitted in GEALearn

### Individual Contribution and Highlights

1. Done in Matlab Script

#### Role in Project Team

1. Done Matlab scripting for EV Bike
2. Researcher: Done case study for EV Bike

## Miniproject 9 – Door Locking System [Individual]

### Modules

1. Autosar
2. Git

### Requirements

#### High level Requirement

Requirement Id	Description
HLR_1	Remote control locks are used Instead of using keys to control car locks & components
HLR_2	Door locks & components inside the vehicle authorize access to owners only
HLR_3	When the velocity is 0 and the battery is on their will be an indication without any sound
HLR_4	When the car is in motion the door lock should be firmly closed
HLR_5	The door open indication will be off only when all seat belts are locked .
HLR_6	If the car is in motion and any door is open the system will display an indication

## Low level Requirement

Requirement Id	Description
LLR_1	When unauthorized person tries to get into your vehicle by breaking the door their will be a beeping sound
LLR_1	Beeping sound will stop when the doors are closed.
LLR_2	When the doors are open there will be individual light indication.
LLR_3	When the boot is open there will be indication on the dashboard.

## Design

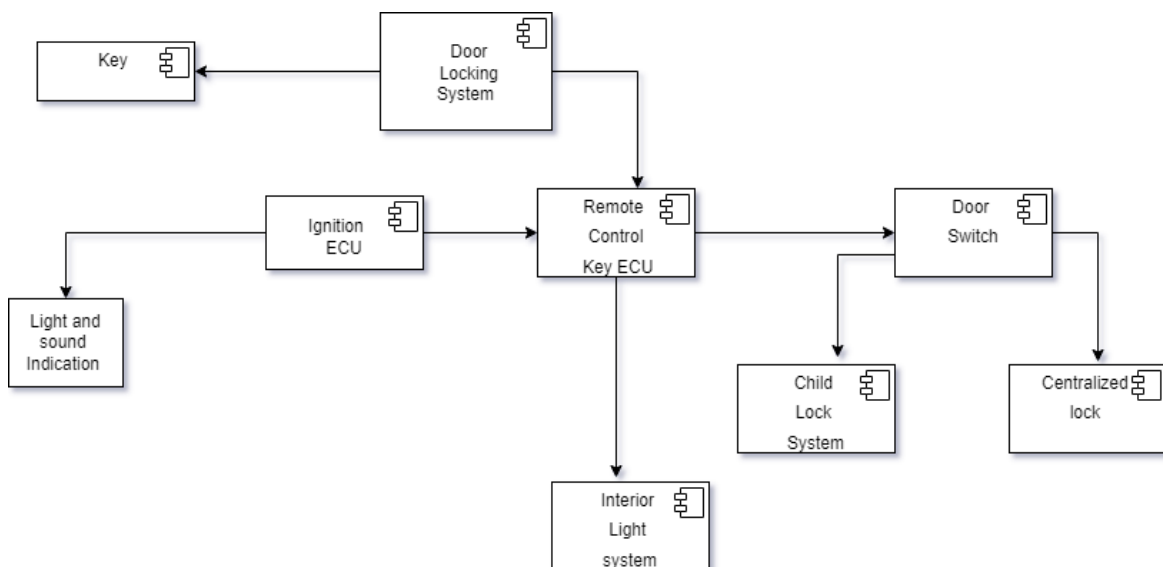


Figure 30. VFB Diagram

## **Implementation and Summary**

### **Git Link:**

Link: [https://github.com/Nayan349/Automotive\\_BMW\\_Project](https://github.com/Nayan349/Automotive_BMW_Project)

## **Individual Contribution and Highlights**

1. Door Locking system
2. Source code management using GitHub
3. AtomicSwComponent
4. SWCInternalBehavior
5. SWCImplementation