

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 | 18.02.2022 | Heamnath S |  |  |  |

Contents

[List of Figures 5](#_Toc96726242)

[Miniproject – 1: Snake game [Individual] 7](#_Toc96726243)

[Modules: 7](#_Toc96726244)

[Requirements: 7](#_Toc96726245)

[High Level Requirements 8](#_Toc96726246)

[Low Level Requirements 8](#_Toc96726247)

[Design 9](#_Toc96726248)

[Test Plan 11](#_Toc96726249)

[High Level Test Plan 11](#_Toc96726250)

[Low Level Test Plan 11](#_Toc96726251)

[Implementation and Summary 12](#_Toc96726252)

[Git Link: 12](#_Toc96726253)

[Git Inspector 12](#_Toc96726254)

[Miniproject 2 – DC Motor Controller [Individual] 13](#_Toc96726255)

[Modules 13](#_Toc96726256)

[Requirements: 13](#_Toc96726257)

[High Level Requirements 14](#_Toc96726258)

[Low Level Requirements 14](#_Toc96726259)

[Design 15](#_Toc96726260)

[Test Plan 16](#_Toc96726261)

[High Level Test Plan 16](#_Toc96726262)

[Low Level Test Plan 16](#_Toc96726263)

[Implementation and Summary 17](#_Toc96726264)

[Git Link: 17](#_Toc96726265)

[Miniproject 3 – Diary Entry System [Team] 18](#_Toc96726266)

[Modules 18](#_Toc96726267)

[Requirements 18](#_Toc96726268)

[High Level Requirements 19](#_Toc96726269)

[Low Level Requirements 19](#_Toc96726270)

[Design 20](#_Toc96726271)

[Test Plan 21](#_Toc96726272)

[High Level Test Plan 21](#_Toc96726273)

[Low Level Test Plan 22](#_Toc96726274)

[Implementation and Summary 22](#_Toc96726275)

[Git Link: 22](#_Toc96726276)

[Summary 22](#_Toc96726277)

[Miniproject 4 – Calendar Automation [Team] 23](#_Toc96726278)

[Modules 23](#_Toc96726279)

[Requirements 23](#_Toc96726280)

[High Level Requirements 23](#_Toc96726281)

[Low Level Requirements 23](#_Toc96726282)

[Test Plan 24](#_Toc96726283)

[High Level Test Plan 24](#_Toc96726284)

[Low Level Test Plan 25](#_Toc96726285)

[Implementation and Summary 25](#_Toc96726286)

[Git Link: 25](#_Toc96726287)

[Individual Contribution and Highlights 25](#_Toc96726288)

[Miniproject 5 – MahindraXUV500 Project [Team] 26](#_Toc96726289)

[Modules 26](#_Toc96726290)

[Requirements 26](#_Toc96726291)

[Design 26](#_Toc96726292)

[Miniproject 6 – Wiper Control System [Team] 28](#_Toc96726293)

[Modules 28](#_Toc96726294)

[Requirements 28](#_Toc96726295)

[Introduction 28](#_Toc96726296)

[Research 28](#_Toc96726297)

[High Level Requirements 29](#_Toc96726298)

[Low Level Requirements 30](#_Toc96726299)

[Design 31](#_Toc96726300)

[Test Plan 32](#_Toc96726301)

[High Level Test Plan 32](#_Toc96726302)

[Low Level Test Plan 34](#_Toc96726303)

[Implementation and Summary 35](#_Toc96726304)

[Git Link: 35](#_Toc96726305)

[Miniproject 7 – Range Rover Velar Project [Team] 36](#_Toc96726306)

[Modules 36](#_Toc96726307)

[Requirements 36](#_Toc96726308)

[Objectives: 36](#_Toc96726309)

[Design 37](#_Toc96726310)

[Git Link: 37](#_Toc96726311)

[Miniproject 8 – EV Car [Team] 38](#_Toc96726312)

[Requirements: 38](#_Toc96726313)

[1. Battery: 38](#_Toc96726314)

[Battery Features: 38](#_Toc96726315)

[2. Motor: 38](#_Toc96726316)

[Motor Features: 38](#_Toc96726317)

[Controller Features: 38](#_Toc96726318)

[4. Inverter: 39](#_Toc96726319)

[Inverter features: 39](#_Toc96726320)

[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. 39](#_Toc96726321)

[Implementation and Summary: 39](#_Toc96726322)

[Role in Project Team: 39](#_Toc96726323)

[Miniproject 9 – Climatization Control [Individual] 40](#_Toc96726324)

[Modules 40](#_Toc96726325)

[Requirements 40](#_Toc96726326)

[Design 41](#_Toc96726327)

[Implementation and Summary 41](#_Toc96726328)

[Git Link: 41](#_Toc96726329)

[Individual Contribution and Highlights 41](#_Toc96726330)

## 

## List of Figures

[Figure 1*:* BehaviourDiagram 9](#_Toc96726331)

[Figure 2 *:* Structural Diagram 10](#_Toc96726332)

[Figure 3: Git Inspector 12](#_Toc96726333)

[Figure 4 : DC motor Diagram 15](#_Toc96726334)

[Figure 5 : Block Diagram 15](#_Toc96726335)

[Figure 6 : Simulation 16](#_Toc96726336)

[Figure 7: Git Dashboard 17](#_Toc96726337)

[Figure 8 : git inspector 17](#_Toc96726338)

[Figure 9: Behaviour Diagram 20](#_Toc96726339)

[Figure 10:User flow Diagram 20](#_Toc96726340)

[Figure 11: Structural Diagram 21](#_Toc96726341)

[Figure 12: state flow diagram 27](#_Toc96726342)

[Figure 13: Structural Diagram 31](#_Toc96726343)

[Figure 14: Behaviour Diagram 32](#_Toc96726344)

[Figure 15: Structural Diagram 37](#_Toc96726345)

[Figure 16: VFB Diagram 41](#_Toc96726346)

# 

# Miniproject – 1: Snake game [Individual]

## Modules:

1. C Programming
2. Git

### Requirements:

Introduction:

The game called "snake" or "snake game" typically the player controlling a line or snake. the most common version of this game involves the snake or line eating items which make it longer, with the objective being to avoid running into a border or the snake itself for as long as possible. The player loses when the snake either runs into a border or its own body. The project is developed in C language.

The main objective of the project of snake game is:

1. A simple console application with very simple graphics

2. Foods are provided at the screen for the snake to eat

 3. Every time the snake eats the food, its length will be increased by one element along with the score.

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

**Why:**

Snake is a tool that can be used as an educational helping hand.

**Where:**

The Snake game can be played from anywhere.

**Who:**

It can be play by any person..

**When:**

It can be play at any time depending upon the person's preferences.

**How:**

Points based on the number of food eaten by the snake.

### High Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| HLR\_1 | A simple console application with very simple graphics | Implemented |
| HLR\_2 | Display the scores and increase the length of the snake | Implemented |

### Low Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| LLR\_1 | Create the snake and food | Implemented |
| LLR\_2 | Input from the player to move the snake | Implemented |
| LLR\_3 | finish the levels of the game | Implemented |

### Design

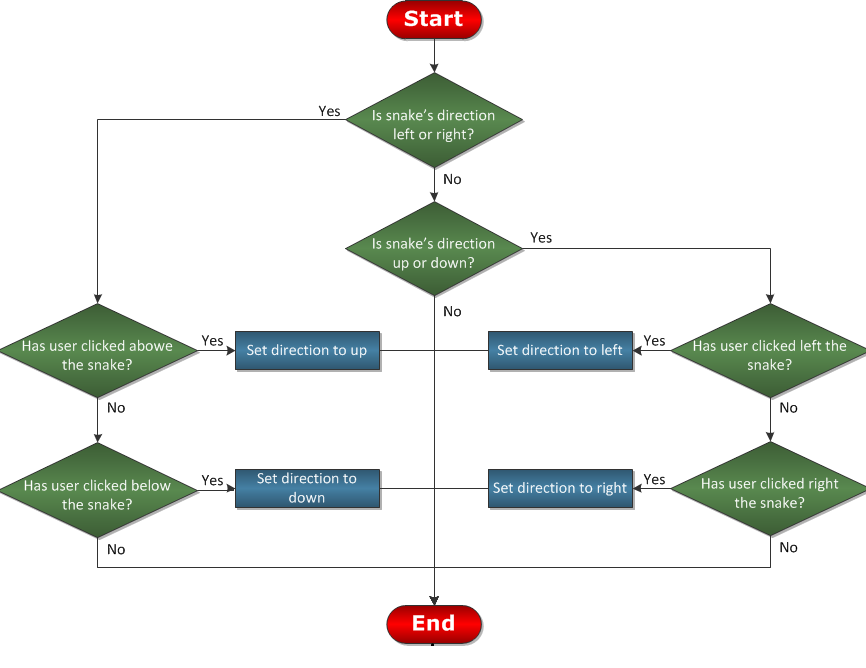


Figure 1*:* BehaviourDiagram

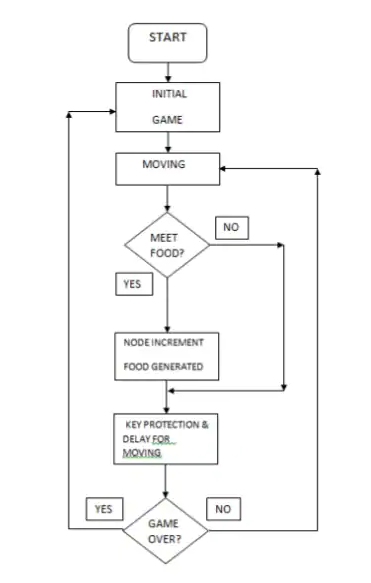


Figure 2 *:* Structural Diagram

## Test Plan

### High Level Test Plan

| **ID** | **Description** | **Expected O/P** |
| --- | --- | --- |
| HLTP\_1 | Add a snake | Implemented |
| HLTP\_2 | Display scores or points | Implemented |
| HLTP\_3 | List all information about snake game | Implemented |

### Low Level Test Plan

| **ID** | **Description** | **Expected O/P** |
| --- | --- | --- |
| LLTP\_1 | List of operations displayed | Implemented |
| LLTP\_2 | Input from the user | Implemented |
| LLTP\_3 | Exit the program | Implemented |

## Implementation and Summary

### Git Link:

Link: [https://github.com/heamnath23/M1\_Game\_Snakegame](https://github.com/heamnath23/M1_Game_Snakegame.git)

Git Inspector

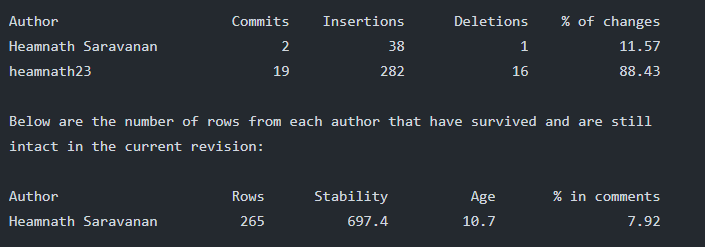


Figure 3: Git Inspector

# 

# Miniproject 2 – DC Motor Controller [Individual]

## Modules

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

### Requirements:

Introduction

The DC motor is an electrical machine with a rotating part termed as a rotor which has to be controlled. For example, consider the DC motor whose speed or direction of rotation of DC motor can be controlled using programming techniques which can be achieved by interfacing with microcontroller.

Research

To control DC motors using L293 half H-bridge integrated circuit as a motor driver which is controlled by ATmega328 microcontroller. L293 is a quadruple half-H bridge device designed used to drive DC motors and Stepper motors. It is designed to be used with a microcontroller like ATmega328 because microcontroller do not have sufficient output voltage and current to drive motors. The L293 and operating voltage of 4.5V to36V while it can output current of 600mA up to peak current of 1.2A. It can drive two DC motors and one stepper motor (or a two phase motor) at the same time and independently.

Features

Simple and high efficiency rate, DC motor controller that aims to overcome many problems industries and improves the industrial works and the system is based on ATMEGA 328.

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

**Why:**

A DC motor controller is needed which will control the speed..

**When:**

It will be very useful for solving most of the industrial works.

**Who:**

The industrial workers who used to control the speed of the motor for preferable work in the industry.

**Where:** The DC motor controller is enhanced to generate speed based on the machine on the industry at that particular instant.

**How:**

The DC motor controller is controlled by Atmega328 microcontroller. The designed motor is implemented, tested to ensure its performance and other design factors.

### High Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| HLR1 | When the industrial works to be vary with the speed of motor. | Implemented |
| HLR2 | In the traction systems, the DC motor controller used for driving heavy loads. | Implemented |

### Low Level Requirements

| **ID** | **Description** | **Status** |
| --- | --- | --- |
| LLR1 | The DC motor controller runs effectively on linux but it will also run equally well on other. | Implemented |
| LLR2 | The Motor runs based on the speed we control. | Implemented |

## Design

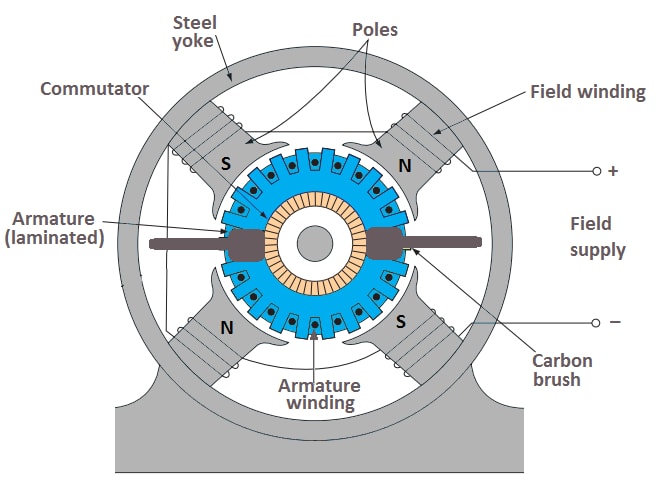


Figure 4 : DC motor Diagram

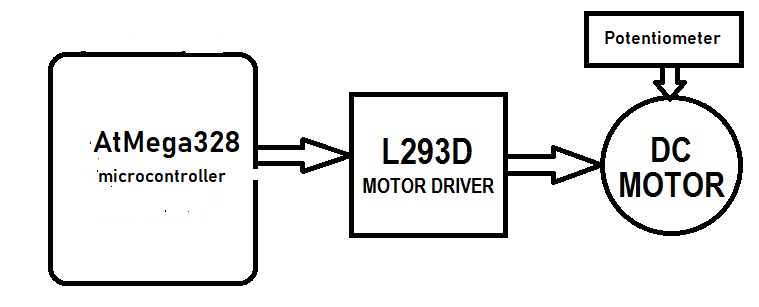


Figure 5 : Block Diagram

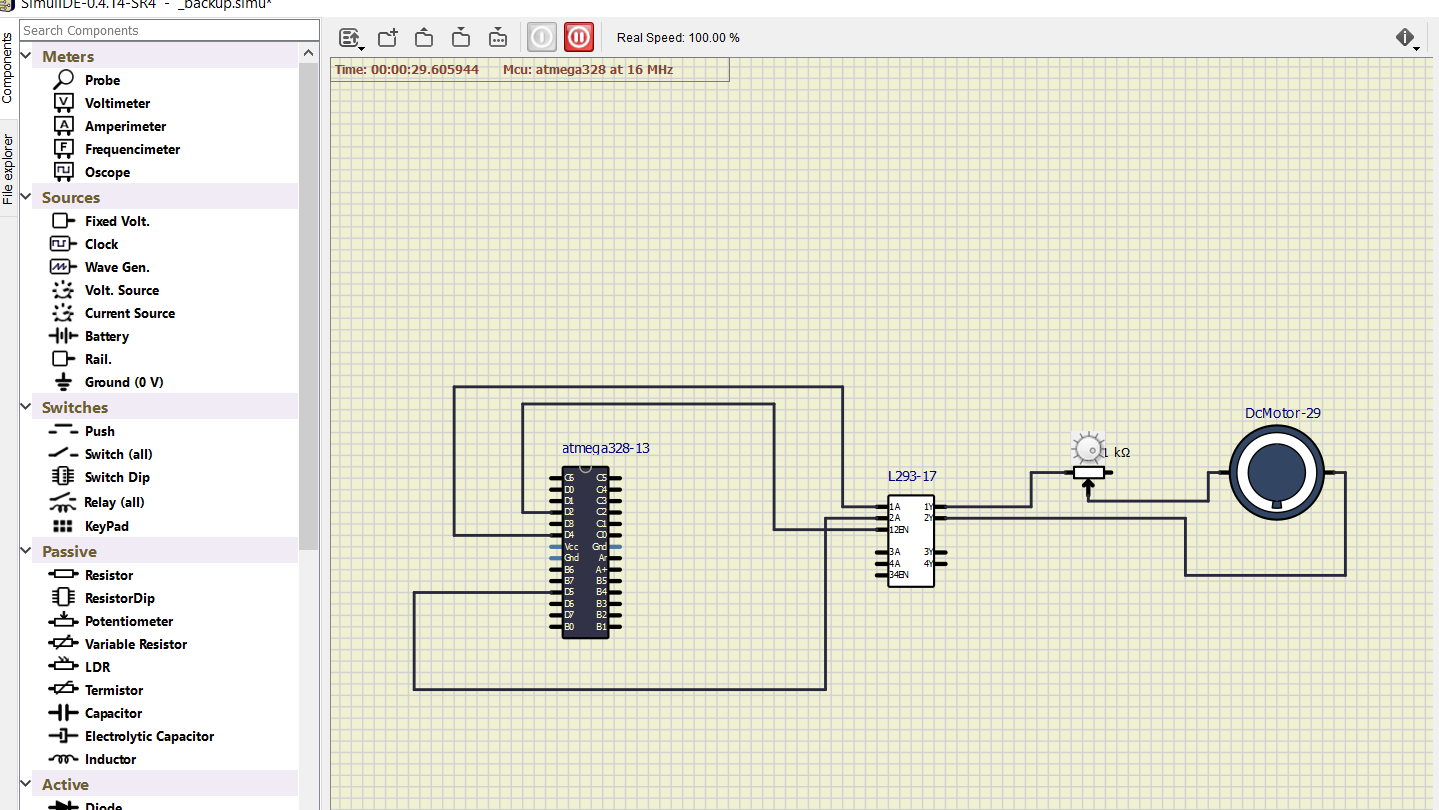


Figure 6 : Simulation

## Test Plan

### High Level Test Plan

| **ID** | **Description** | **Expected O/P** | **Actual O/P** | **Type of Test** |
| --- | --- | --- | --- | --- |
| H\_01 | DC motor runs | PASSED | SUCCESS | Requirement |
| H\_01 | Speed of the Motor | PASSED | SUCCESS | Scenario |
| H\_03 | control the speed | PASSED | SUCCESS | Boundary |

### Low Level Test Plan

| **ID** | **Description** | **Expected O/P** | **Actual O/P** | **Type of Test** |
| --- | --- | --- | --- | --- |
| L\_01 | DC motor speed variation | PASSED | SUCESS | Requirement |
| L\_02 | View the rotation of the motor | PASSED | SUCESS | Scenario |
| L\_03 | motor runs at industrial condition | PASSED | SUCCESS | Boundary |

## Implementation and Summary

### Git Link:

Link: [https://github.com/heamnath23/M1-Embedded\_DC-motor-controller](https://github.com/heamnath23/M1-Embedded_DC-motor-controller.git)

**Git Dashboard:**

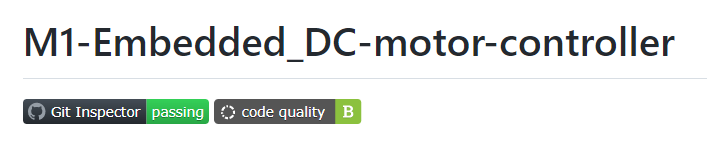
****

Figure 7: Git Dashboard

**Git Inspector:**

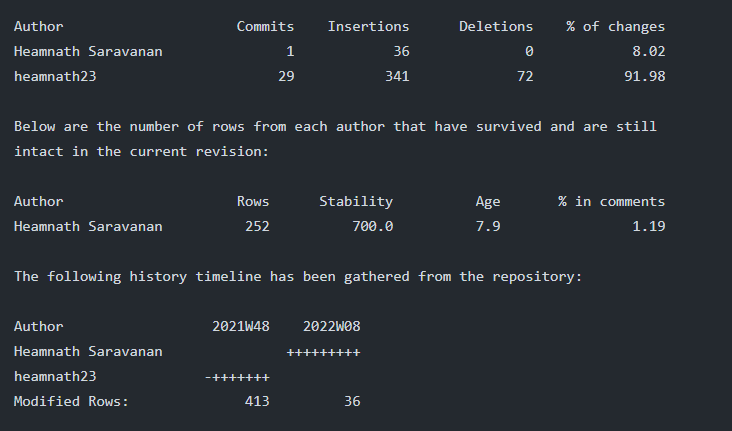
****

Figure 8 : git inspector

# Miniproject 3 – Diary Entry System [Team]

## Modules

1. SDLC
2. Git

### Requirements

Introduction

This Diary Entry System is developed in C programming language. It helps the users in the daily record of their necessary conferences, functions etc. The term diary is a few time referred to as personal diaries, usually supposed to stay non-public or to possess a restricted circulation amongst friends or relative it’s similar to the records that we have a tendency to detain our personal Diary. In this Diary system user can add, edit, delete and examine the records. Here users will add the records like Date, Time, and Place and also the person you're attending to meet. In this project, user can keep their personal record like they do in a diary. You can keep records of the important things you do in your daily life, like meetings and various other tasks. This project on Personal Diary entry system in C is compiled in Code::Blocks IDE using GCC compiler. It is complete and totally error-free.

This is a password protected Diary system to increase the security standards for the same.

The purpose of the system is to help you keep and manage the time for your incoming events.

Here you can add the records such as Date, Time, Place and View the Records.

Research

\* This is a protected diary system to increase the security standards for the same.

\* The purpose of the system is to help you keep and manage the time for your incoming events.

Cost and Features and Timeline

1) Old System: - There has been no password covered Diary system.

2) New System: - Password protected Diary system is built.

**4W's and 1’H:-**

**Why:**

Makes a difference the client to effectively include their imperative meetings, presentation records, additionally can be altered.

**Where:**

It can be used by Travel Specialists or doctor’s infect anybody can utilize it to keep their records safe.

**When:**

At whatever point the client needs to keep his individual records secure at a place.

**Who:**

Anybody can use it.

**How:**

It will keep your all individual records securely at one place.

### High Level Requirements

|  |  |
| --- | --- |
| **ID** | **Description** |
| HLR1 | Add the inputs to add records, edit, view and delete. |
| HLR2 | Users can add the record in the system. |
| HLR3 | Can view that record for further. |
| HLR4 | User can Edit the added record and can make some changes in it. |
| HLR5 | Can delete the record permanently if not needed. |
| HLR6 | User can edit the Password for security purpose. |

### Low Level Requirements

|  |  |
| --- | --- |
| **ID** | **Description** |
| LLR\_1 | Login page of Diary Entry system. |
| LLR\_2 | The system will ask password to view and Edit the records. |
| LLR\_3 | Edit data. |
| LLR\_4 | Enter username and password. |
| LLR\_5 | Newly added details should be recorded |

## Design

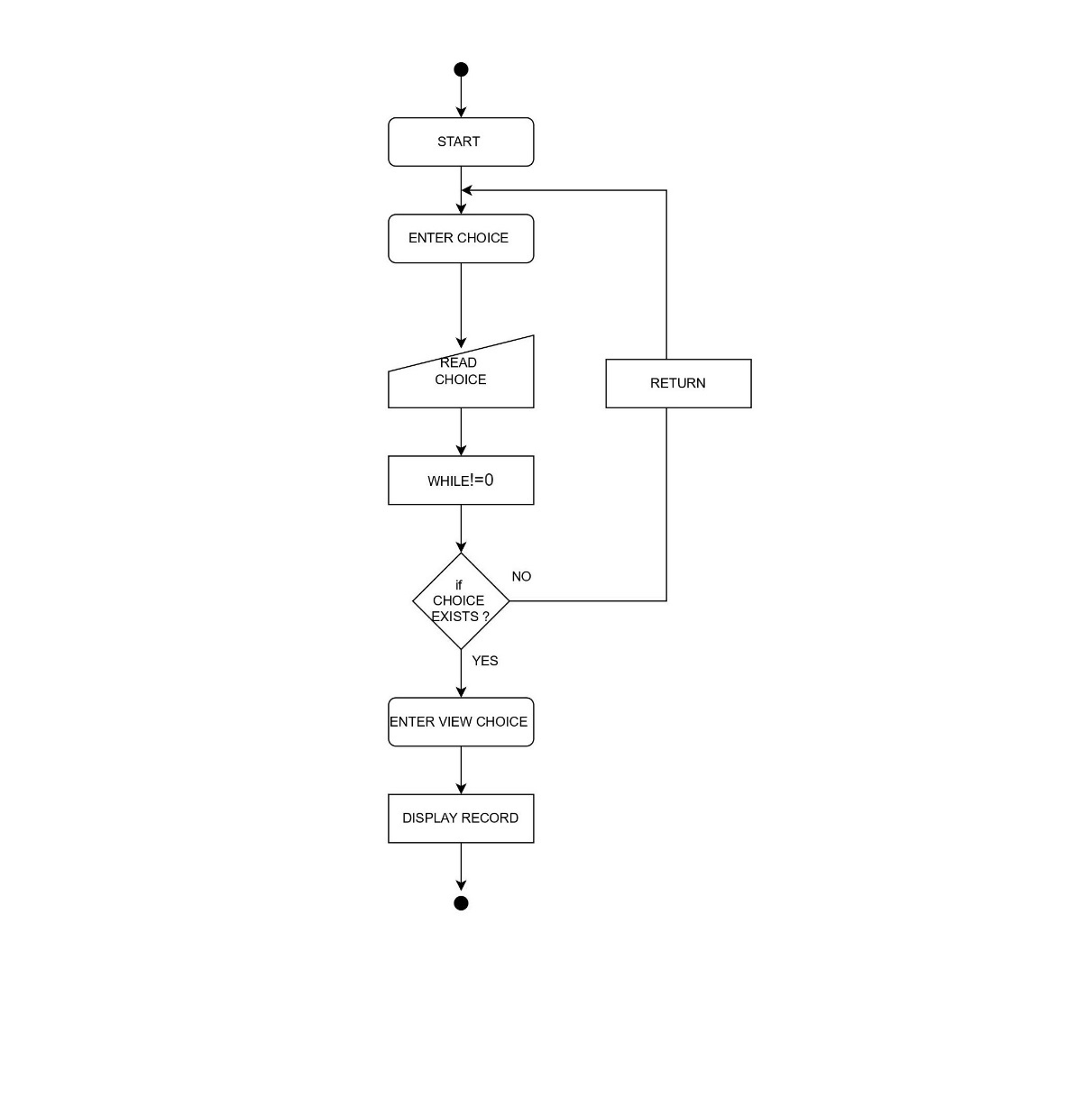


Figure 9: Behaviour Diagram

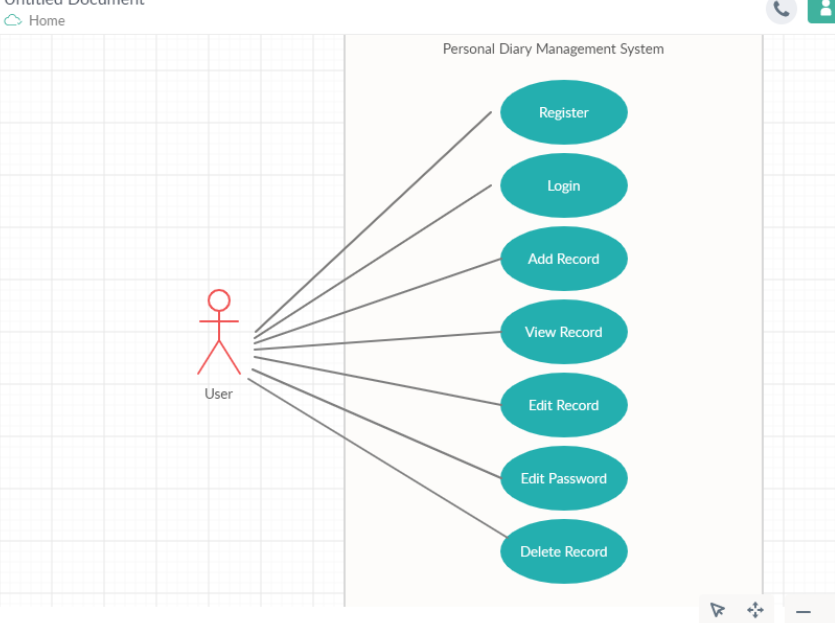


Figure 10:User flow Diagram

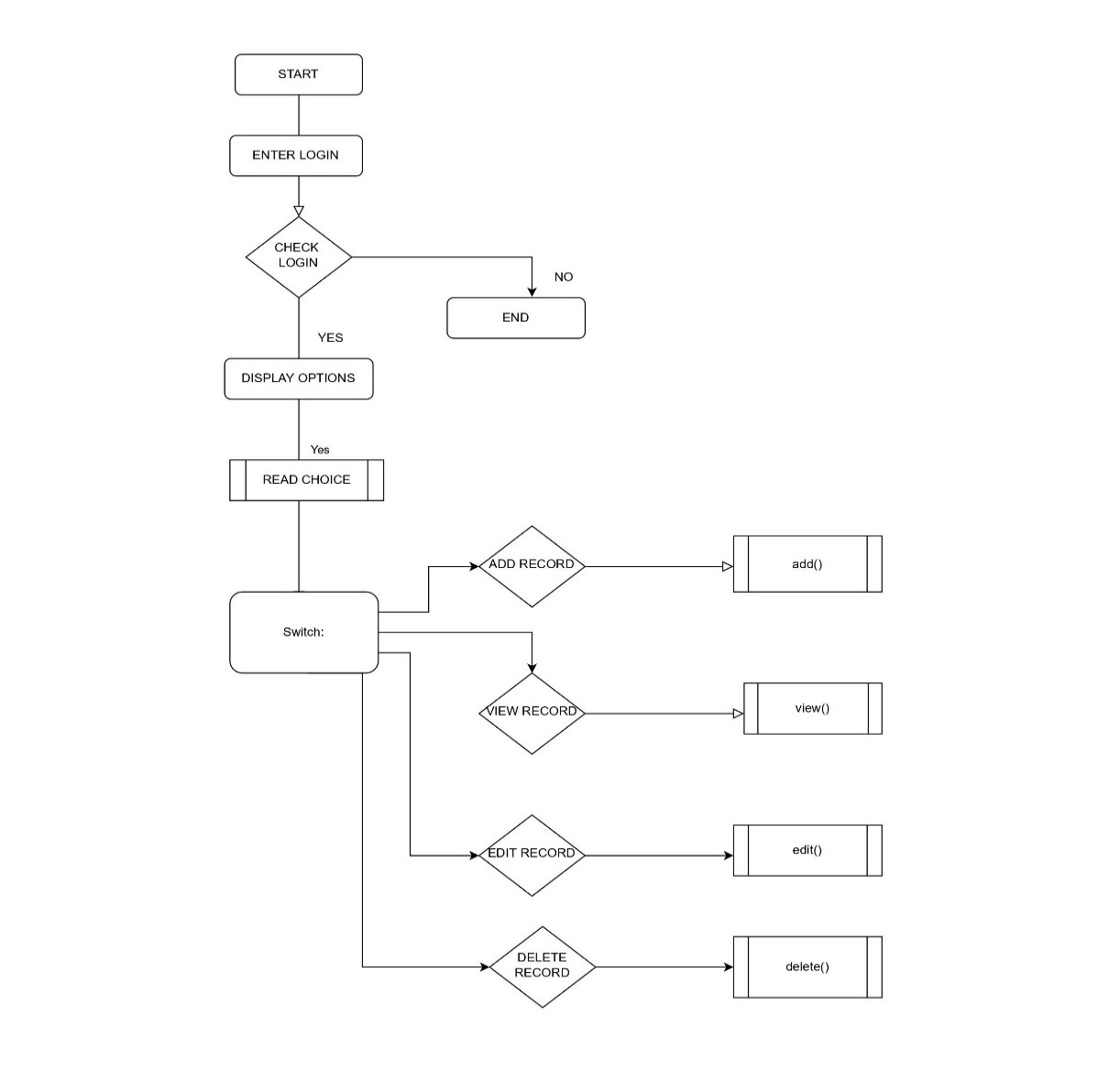


Figure 11: Structural Diagram

## Test Plan

### High Level Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Description** | **Expected Output** | **Actual Output** | **Pass/fail (Result)** |
| H\_01 | Check if the record is viewed or not | SUCCESS | SUCCESS | PASS |
| H\_02 | Check if the record information is added or not | SUCCESSS | SUCCESS | PASS |
| H\_03 | Check if the record is edited | SUCCESS | SUCCESS | PASS |
| H\_04 | Check if the password is edited or modified | SUCCESSS | SUCCESS | PASS |
| H\_05 | Check if the record is deleted or not | SUCCESS | SUCCESS | PASS |

### Low Level Test Plan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test ID** | **Description** | **Exp IN** | **Exp OUT** | **Actual Out** |
| L\_01 | Check if record information is properly added | Data and information | SUCCESS | SUCCESS |
| L\_02 | if the data is collected from diary during when the user needed | Data | SUCCESS | SUCCESS |
| L\_03 | If the record data is deleted | Diary data | SUCCESS | SUCCESS |

## Implementation and Summary

### Git Link:

Link: <https://github.com/GENESIS2021Q1/Applied_SDLC-Dec_Team_49>

### Summary

1. Designed Homepage
2. Search Option
3. Header
4. Footer
5. Integrating All Pages Together

# 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 |
| 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 (Colour Codes/Numbers) | HR02 | Implemented | Implemented |
| LR09 | Master Calendar: Appending | HR02 | Implemented | Not Available |
| LR10 | Master Calendar: Course code correction | HR02 | Implemented | Not Available |

## 

## Test Plan

### High Level Test Plan

| **ID** | **Description** | **Expected I/P** | **Expected O/P** | **Actual O/P** | **Type Of Test** |
| --- | --- | --- | --- | --- | --- |
| HLTP\_01 | Schedule events | User Input | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_02 | Review Scheduled events | User Input | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_03 | Reminding events | User Input | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_04 | Showing available timing Slots based on faculty and modules | User Input | SUCCESS | SUCCESS | Requirement Based |
| HLTP\_05 | Output file generation | User Input | SUCCESS | SUCCESS | Requirement Based |

### Low Level Test Plan

| **ID** | **HLTP ID** | **Description** | **Expected I/P** | **Actual O/P** |
| --- | --- | --- | --- | --- |
| LLTP\_01 | HLTP\_01 | User can update their events | SUCCESS | SUCCESS |
| LLTP\_02 | HLTP\_01 | User can review previous schedules | SUCCESS | SUCCESS |
| LLTP\_03 | HLTP\_02 | User will get the remainder of the event | SUCCESS | SUCCESS |
| LLTP\_04 | HLTP\_04 | Multiple event allocated per day will be displayed | SUCCESS | SUCCESS |

## Implementation and Summary

### Git Link:

Link: <https://github.com/tlnsnani/OopsWithPython_Calendar_Automation_Team-48>

**Summary**

 Calendar API to read and modify events on Calendar. Once one creates an event with a special name format, this program will update its colour automatically to reflect its level of priority. It is meant to facilitate time management when many assignments of different lengths and levels of completion are due for different dates.

### Individual Contribution and Highlights

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

Role in Project Team

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

# Miniproject 5 – MahindraXUV500 Project [Team]

## Modules

1. MATLAB
2. Git

## Requirements

We have implemented following features

1. Sunroof control

2. HVAC

3. Wiper control

## Design

This project was implemented in state flow diagram using MATLAB.

**HVAC:**

Heating, ventilation, and air conditioning (HVAC)is the use of various technologies to control the temperature, humidity, and purity of the air in an enclosed space. Its goal is to provide thermal comfort and acceptable indoor air quality. HVAC system design is a sub discipline of mechanical engineering, based on the principles of thermodynamics, fluid mechanics, and heat transfer.

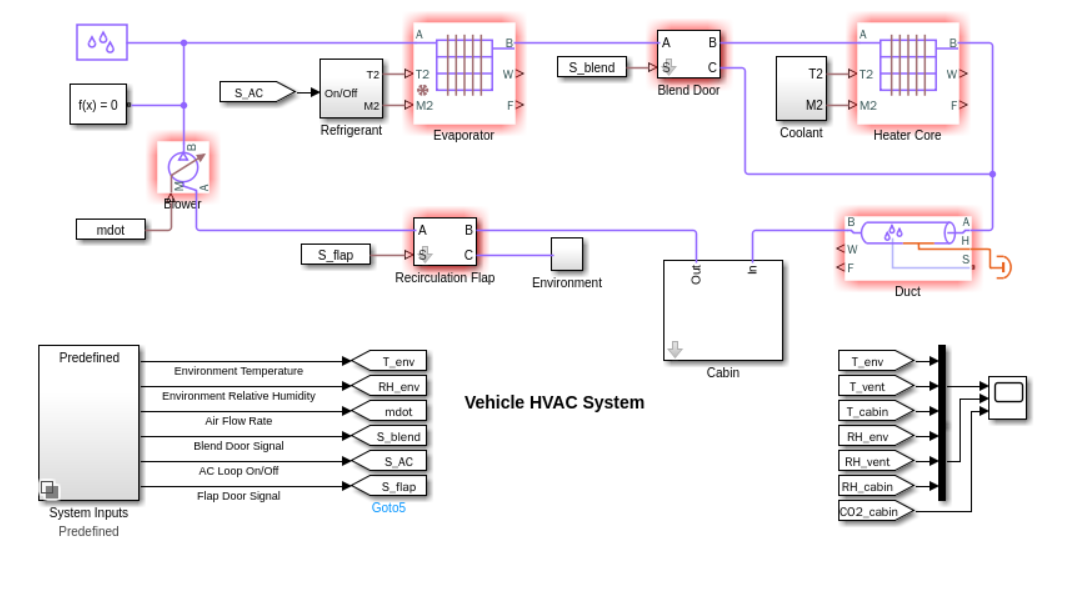


Figure 12: state flow diagram

# Miniproject 6 – Wiper Control System [Team]

## Modules

1. C Programming
2. STM32

## Requirements

## Introduction

A Windscreen wiper, windshield wiper or wiper blade is a device used to remove rain, washer fluid, water and debris from a vehicle's front window and it is important because they permit driving during rainy and snowy conditions. Note that the last remarkable development was that of electric wipers. Intermittent wipers which are convenient when it is drizzling or just after it has stopped raining have recently been developed. In addition, step less intermittent wipers, the interval of which can be freely selected by the driver, and speed following wipers, the interval to the vehicle speed, are becoming popular. In addition, some wiper systems which use rain sensors to control the interval in response to rainfall and vehicle speed variations are beginning to appear. This is because there is an increasing demand for windshield to meet the drivers comfort.

## Research

Research and development for raindrop detection started a long time ago, but became more popular around 1970. A windscreen wiper or windshield wiper is a device used to remove rain and debris from a windscreen. Almost all motor vehicle, including trains, aircraft and watercraft, are equipped with such wipers, which are usually an essential requirement. A wiper generally consists of an arm, pivoting at one end and with a long rubber blade attached to the other. The blade is swung back and forth over the glass, pushing water from its surface. The speed is normally adjustable, with several continuous speeds and often one or more "intermittent"settings.Most automobiles use two synchronized radial type arms. It takes a lot of force to accelerate the wiper blades back and forth across the windshield so quickly. In order to generate this type of force, a worm gear is used on the output of a small electric motor.

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

**Who:**

Users, Employees and Students will get the help from this project.

**Why:**

User can use the program efficiently and get the required results.

We are making this project to improve the existing Wiper Control System.

**When:**

When it rains we use wipers.

**Where:**

To wipe the rain drops of the cars, trucks we use wipers.

**How:**

It is flexible and user friendly.

### High Level Requirements

| **ID** | **Description** | **Status (Implemented/Not)** |
| --- | --- | --- |
| HR01 | System should be recognize the pushbutton | Implemented |
| HR02 | System should be able to access pushbutton press | Implemented |
| HR03 | System should be able to access how many time pushbutton press | Implemented |
| HR04 | System should be recognize the status of pushbutton | Implemented |
| HR05 | System should be recognize the status of LEDs | Implemented |

### Low Level Requirements

| **ID** | **Description** | **Status (Implemented/Not)** |
| --- | --- | --- |
| LR01 | LEDs must be ON/OFF after the pushbutton press | Implemented |
| LR02 | LEDs ON for set frequencies (timer ) | Implemented |
| LR03 | If the interrupt occur LEDs must ON | Implemented |
| LR04 | System has to shows the LEDs status | Implemented |

## Design

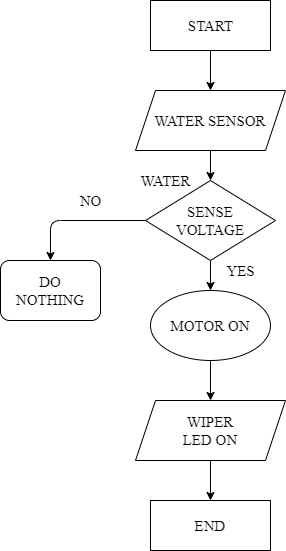


Figure 13: Structural Diagram

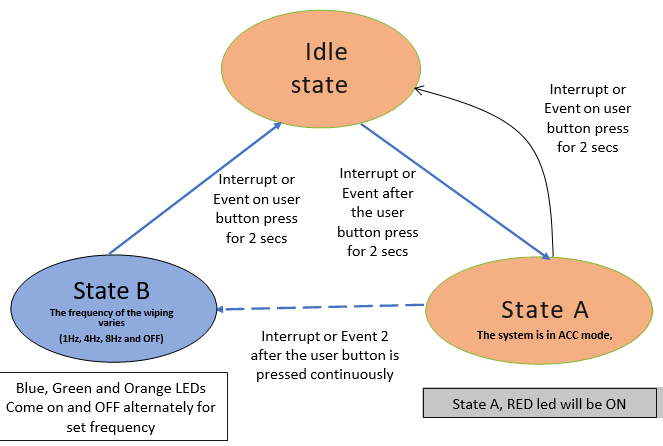


Figure 14: Behaviour Diagram

## Test Plan

### High Level Test Plan

| **TEST PLAN ID** | **Description** | **exp I/P** | **Exp O/P** | **Status** |
| --- | --- | --- | --- | --- |
| HLR01 | Check Red LED on & wiper work | Press and held 2sec | Red LED On & Wiper works | pass |
| HLR02 | Press button 1 more time | press button | Blue LED ON | Pass |
| HLR03 | Check the frequency of blue LED | 1HZ | Wiper receive 1HZ | Pass |
| HLR04 | Press the button again 1 more time | press button | Green LED ON | Pass |
| HLR05 | Check the frequency of orange LED | 4HZ | wiper receive 4HZ | Pass |
| HLR06 | Press the button 1 more time again | Press Button | green LED ON | Pass |
| HLR07 | Check the frequency of Orange LED | 8HZ | Wiper receive 8HZ | Pass |
| HLR08 | Check the frequency of LED & check wiper work | 5,6,7,8 HZ | Wiper receive 5HZ for blue,6HZ for green & 7HZ for orange | Pass |
| HLR09 | Check all LED OFF and wiper stops work | press and held 5secs | All LEDs are OFF & wiper stops | Pass |

### Low Level Test Plan

| **LLR-ID** | **Description** | **HLR-ID** | **exp I/P** | **Exp O/P** | **Status** |
| --- | --- | --- | --- | --- | --- |
| LLR01 | Run the system | HLR01 | Check the LEDs & wiper | LEDs off & wiper off | Pass |
| LLR02 | Press the button for 2sec | HLR01 | Press and held 2sec | Red LED will be ON | Pass |
| LLR03 | Press the button for 1sec | HLR01 | Press and held 1sec | Red LED will be OFF | Pass |
| LLR04 | Press the button for 3sec | HLR01 | Press and held more than 3sec | Red LED will be OFF | Pass |
| LLR05 | Press the button for 2sec | HLR01 | Press and held 2sec | Red LED will be OFF | Fail |
| LLR06 | After red LED off wiper is also ON | HLR01 | Wiper ON | Wiper will start working | fail |
| LLR07 | After red LED ON wiper is also ON | HLR01 | Wiper OFF | Wiper will stop working | Fail |
| LLR08 | Press the button again & again check the LED's | HLR10 | Press button again & again | Blue,Green,Orange LED's ON | Pass |
| LLR09 | Press the button again & again check the LED's | HLR10 | Press button again & again | Blue,Green,Orange LED's ON | Fail |

## Implementation and Summary

### Git Link:

Link: <https://github.com/GENESIS-2022/MasteringMCU-Team19>

Individual Contribution and Highlight

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 Test cases and testing the integrated code

# Miniproject 7 – Range Rover Velar Project [Team]

## Modules

1. Automotive Systems
2. Git

### Requirements

### Objectives:

The aim of automotive climate control system is to control the temperature and humidity of the atmospheric air and circulate the same in the automobile. The main purpose of the car air conditioner is to make the cabin ambiance comfortable for the occupants.

Working:

The air-conditioning system in a car works by manipulating refrigerant between a liquid and a gaseous state. As the refrigerant changes states, it absorbs heat and humidity from the vehicle and allows the system to give off cool, dry air. To change the refrigerant between a liquid and a gaseous state, the air-conditioning system works to control pressure and temperature. Operation mode can be selected by pressing the Mode button.The major components of Air conditioning system are

Compressor-It is also known as the heart of the AC sytem.The AC cycle starts with the compressor compressing the low-pressure gaseous refrigerant and the refrigerant leaves the compressor as a high-pressure gaseous refrigerant. The compressor is the central component of the AC system.

Condenser - It lowers the temperature of the compressed liquid refrigerant. As hot compressed gasses are introduced into the top of the condenser, they are cooled off. As the gas cools, it condenses and exists the bottom of the condenser as a high-pressure liquid.

Receiver Dryer-The receiver dryer is used in between evaporator and compressor to convert that remaining liquid into vapors before sending it to compressor for compression.

Expansion Valve-It is a device used to expand the high pressure, low temperature liquid refrigerant sent by the condenser.

Evaporator Coil-Its primary duty is to remove heat from the inside of your vehicle. It is used to convert the liquid refrigerant sent by the expansion valve into vapor, which in turn provides cooling through the fan inside a passenger's cabin.

## Design

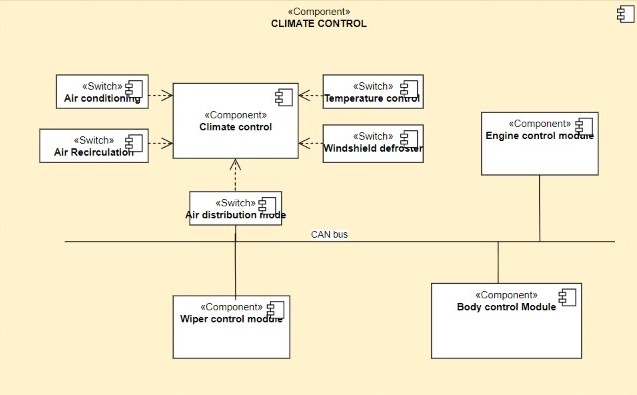


Figure 15: Structural Diagram

### Git Link:

Link: <https://github.com/kumaravel5/Range_Rover_Velar>

Individual Contribution and Highlights

* 1. Climate control Case Study
  2. Source code management using GitHub

Role in Project Team

1. Designer: Done Designing for Project
2. Researcher: Done case study for Climate control

# Miniproject 8 – EV Car [Team]

**Module: Applied Control Systems and Vehicle Dynamics**

# Requirements:

## 1. Battery:

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

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 an atom or molecule with an electric charge caused by the loss or gain of one or more electrons.

### Battery Features:

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:

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

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:

Electronically commutated

High-energy, rare-earth magnets used for rotor field

Requires speed control with 6-lead connection (3 power, 2 Hall Effects, 1 drain) Rated speed 2500 RPM; minimum 150-200 RPM. Linear speed torque curves. Built-in tack pulse for economical speed readout.

Encoder options for servo performance.

### Controller Features:

E- Braking will release the motor from the controller if brake is applied.

Over current protection to protect controller during faulty conditions or short circuit. Low voltage detection ensures Battery life.

Pedal assist mode controls the motor speed based on the speed of peddling. Accelerator fault protection to prevent runway.

Provision for 120°/ 60° Selection. Speedometer output.

Brake inputs with high active and low active provision. Speed limit control provision.

## 4. Inverter:

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

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.

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:

## 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.

## Implementation and Summary:

**Submission**: Submitted in GEA Learn.

## Role in Project Team:

1. Done MATLAB scripting for EV Car
2. Researcher: Done case study for EV Car

# Miniproject 9 – Climatization Control [Individual]

## Modules

1. Autosar
2. Git

### Requirements

| **REQUIREMENT ID** | **Description** |
| --- | --- |
| SYS\_1 | If driving in a heat wave, then AC should be ON and adjust the temperature control dial as per the user requirement. |
| SYS\_2 | If summer or hot condition then the air-recirculation ON to ensure air-con gets as cold as possible as quickly as possible. |
| SYS\_3 | If air distribution damper motor commanded position not reachable, check and remove obstruction from air distribution damper motor. |
| SYS\_4 | If the windshield defogger switch is not switched OFF manually, it will turn OFF automatically after a pre-defined time, based on ambient temperature. |

## Design

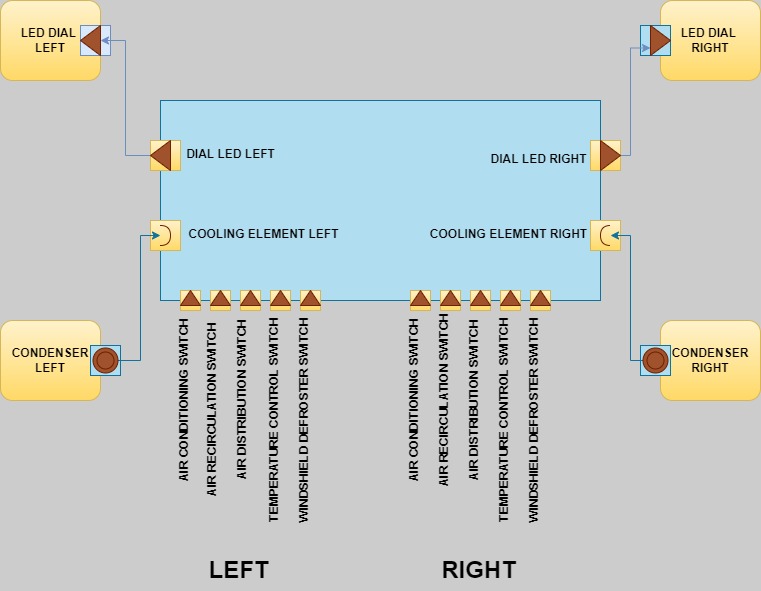


Figure 16: VFB Diagram

## Implementation and Summary

### Git Link:

Link: [https://github.com/kumaravel5/Range\_Rover\_Velar](https://github.com/kumaravel5/Range_Rover_Velar.git)

### Individual Contribution and Highlights

1. Climatization Control Case Study
2. Source code management using GitHub
3. AtomicSwComponent
4. SWCInternalBehavior
5. SWCImplementation.