GENESIS - Learning Outcome & Mini-project Summary Report

LTTS
GLOBAL
ENGINEERING
ACADEMY

L&T Technology Services

Details

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	To be Approved	Remarks/Revision Details
1	20/12/2020	Toti Shoba Rani			

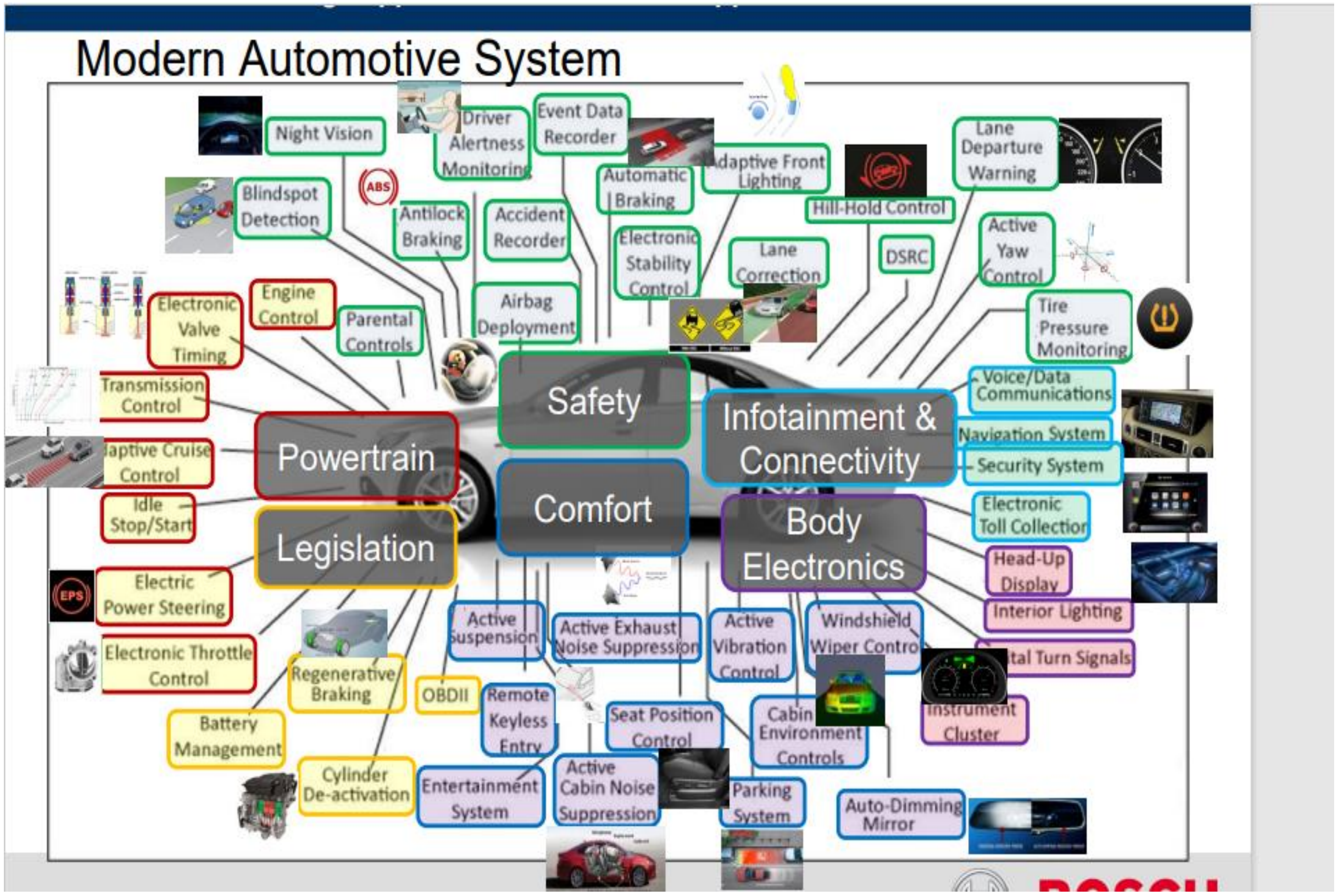
Contents

LIST OF FIGURES

LIST OF TABLES

ACTIVITY 1[TEAM]:
SUBSYSTEMS OF A CAR

Subsystems of a car contain:
1. Powertrain module
2. Chassis
3. Infotainment
4. Safety and comfort
5. Body control module

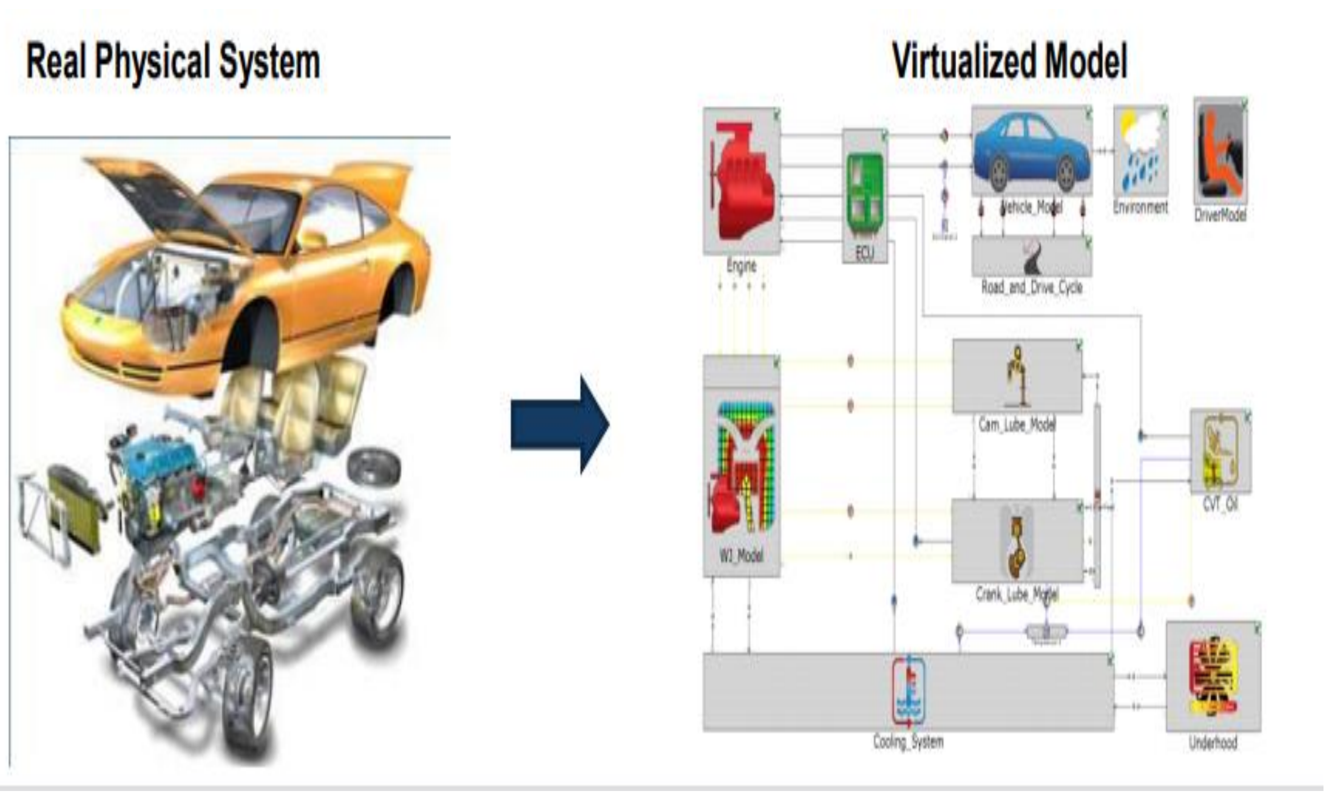


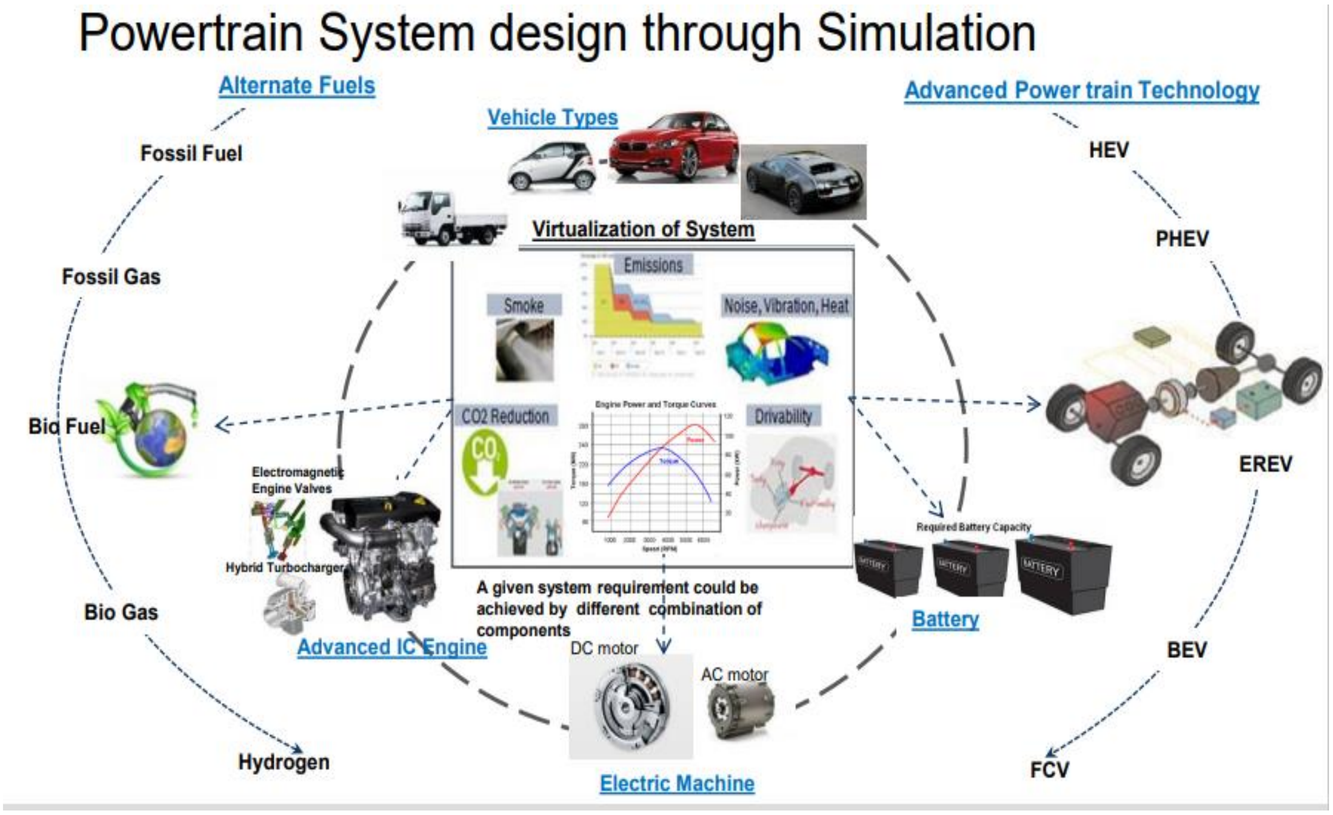
1. Powertrain Control Module:
- PCM controls and coordinates the different subsystems of the car. PCM is also known as an engine control module.
- It consists of two subsystems i.e., Engine Control Units (ECU) and Transmission Control Units (TCU).

- The function of the Engine control Module or unit is to ensure the optimum Engine performance by controlling a set of actuators on the internal combustion engine of cars.
- The TCU or Transmission control module takes inputs from the engine sensors or TCU to change the gears for the optimum engine performance and fuel efficiency of cars.
- The PCM gets inputs from the sensors which are spread around the car and which gives information related to Engine management and performance.
- The responsibility of the PCM is to alert the driver whenever there is an issue with a subsystem or auto car part.

MBD approach for powertrain systems:

- System analysis and design for optimum performance
- Control function development
- Early V&V of control function
- Auto-coding
- Integration with standard and V&V





Cause for the failure of PCM:
It fails due to natural causes such as corrosion or failed due to an electrical issue.

Symptoms of a bad powertrain control module:

1. Illuminated check engine light: Engine light only turns on if the ECM is analyzing the data
2. Intermittent cylinder misfires: The information relayed by the crankshaft position sensor is used to control fuel injection, the ignition system and general engine timing which are all essential for the cylinders to fire properly.
3. Fails to start: ECU is responsible to know whether the vehicle starts or not.

Link: https://www.youtube.com/watch?v=KUI4MsyZaRc

2. Chassis:

It consists of a support engine, transmission system, brake system, suspension system, steering system, cooling system, wheels etc.

An example of chassis is a vehicle frame, an underpart of a motor vehicle on which the body is mounted.

Functions of Chassis:

- To carry a load of passengers or goods carried in the body.
- To support the body, engine, gearbox.

- To withstand the forces during sudden braking or acceleration.
- To withstand stresses.
- To withstand centrifugal force.

Classification of chassis:

1. Conventional control:
    - Engine is mounted in front of the driver's cabin.
    - This type of arrangement avoids full utilization of space.
2. Semi-forward control:
    - Engine is mounted that half of it is in driver's cabin, whereas another half is in front outside the driver’s cabin.
    - Tata SE series of vehicles are examples of this type of chassis.
3. Full-Forward control chassis:
    - In which the engine is mounted completely inside the driver's cabin.
    - Maximum utilization of space is achieved in this type of arrangement.
      Example: Tata E series of vehicles.

3. Infotainment

In -vehicle infotainment head unit is a touch screen based, mounted on the vehicle dashboard to deliver entertainment and information to the passengers. An infotainment system depending on how advanced it is, plays you songs (both audio and video), lets you take calls, reads messages, and lets you navigate to your destination and a whole lot of stuff. In advanced cars with a lot of sensors, an IVI can display tire pressure, kilometers to tank empty, proximity to near-by vehicles and much more.

The infotainment unit has three different form factors, Embedded, Tethered, and Integrated.

- Embedded systems are those where the intelligence (software and applications) and the connectivity (modem) are built into the infotainment unit. Embedded systems are expensive and are mainly found on mid- to high-end cars.
- Tethered systems have the intelligence built in but require an external device for connectivity. Tethered systems are cheaper than embedded systems but expensive than integrated systems.
- Integrated systems are simpler and are totally dependent on the user’s smartphone for intelligence and connectivity. Users connect their smartphones with the infotainment unit and their screen is mirrored on to the infotainment unit’s display and all the features and functions of the phone can be accessed through the infotainment unit.

Integrated systems are inexpensive and user friendly. Additionally, the customer need not purchase separate data plans as required for embedded systems, keeping post-purchase expenses at a minimum.

Link: https://youtu.be/G3k4sVUVssU

4. Safety and control:
Brake subsystem:

The principle used by every brake is friction. Opposing any force is something friction does but we increase it using brakes. Friction converts kinetic energy into heat energy thus effectively bringing the car to a stop.

- Mechanical drum brakes
- Disc brakes.

Link: https://youtu.be/82qBBJ8iwcc

5. Body Control Module:

To achieve the mentioned objectives, OEMs equip modern cars with features such as Anti-Lock Braking System (ABS), power-controlled steering, power windows, turn indicators, Android Infotainment System and more.

These features are controlled and managed by Electronic Control Units (ECUs) that work independently.

Hardware & Software Components of Body Control Module Architecture:

A typical Body Control Module ECU consists of a microprocessor to control the various functions of vehicle’s body electronics (power window, wiper, side-view mirrors and more). Additionally, there are ports provided on the BCM platform for communication with different ECUs, instrument clusters, sensors, actuators, etc. Various other components can also be integrated to the BCM unit depending on the specific requirements and automotive use-cases. The devices or hardware that connects to a Body Control Module ECU can be categorized into input and output devices:

Input Devices: Devices that feed data to the body control module and include sensors (potentiometers, variable resistors, magnetic pickup, etc.)

Output Devices: Devices that are tasked with generating a response to the signal received from the input devices (relays and solenoids are typical examples of output devices in the context of BCM)
How Does a BCM Control Unit Work?

- The body control module receives data from the input devices and controls the output devices based on this data.
- For instance, when the user presses the power window switch, the car’s battery sends power to the BCM unit, to communicate with the ignition module. This, in turn, sends a signal to the load that will rotate the motor and control the window.
- Likewise, there are innumerable body control functions that need to be performed smoothly and reliably. Controlling these parts would also be possible without a BCM unit; but that would amount to additional wiring inside the vehicle.

- A body control module eliminates the need for this additional wiring. It also manages the flow of power so that the electrical module of the car does not get over-burdened when multiple functions are carried out at the same time.

Link: https://www.youtube.com/watch?v=jBp kAkUChi 8

ACTIVITY 2[TEAM]

2.1 Electronic Control Unit:

An engine control unit (ECU), also commonly called an engine control module (ECM) or powertrain control module (PCM), is a type of electronic control unit that controls a series of actuators on an internal combustion engine to ensure optimal engine performance. It does this by reading values from a multitude of sensors within the engine bay, interpreting the data using multidimensional performance maps (called lookup tables), and adjusting the engine actuators.

If the ECU has control over the fuel lines, then it is referred to as an electronic engine management system (EEMS). The fuel injection system has the major role of controlling the engine's fuel supply. The whole mechanism of the EEMS is controlled by a stack of sensors and actuators.

Control of air–fuel ratio:

Most modern engines use some type of fuel injection to deliver fuel to the cylinders. The ECU determines the amount of fuel to inject based on a number of sensor readings. Oxygen sensors tell the ECU whether the engine is running rich (too much fuel or too little oxygen) or running lean (too much oxygen or too little fuel) as compared to ideal conditions (known as stoichiometric).

Air–fuel mixture control of carburetors with computers is designed with a similar principle, but a mixture control solenoid or stepper motor is incorporated in the float bowl of the carburetor.

Types of ECU:

- ECM – engine control module
- EBCM – electronic brake control module
- PCM – powertrain control module
- VCM -vehicle control module
- BCM-body control module

Applications:

- Anti-lock braking system
- Electronics brake distribution force
- Power windows
- Climate control

- Park assists
- Collision warning
- Heads-up display
- Cruise control
- Airbag control system

2.2 Application notes on Body Control Module

2.2.1 INFINEON:

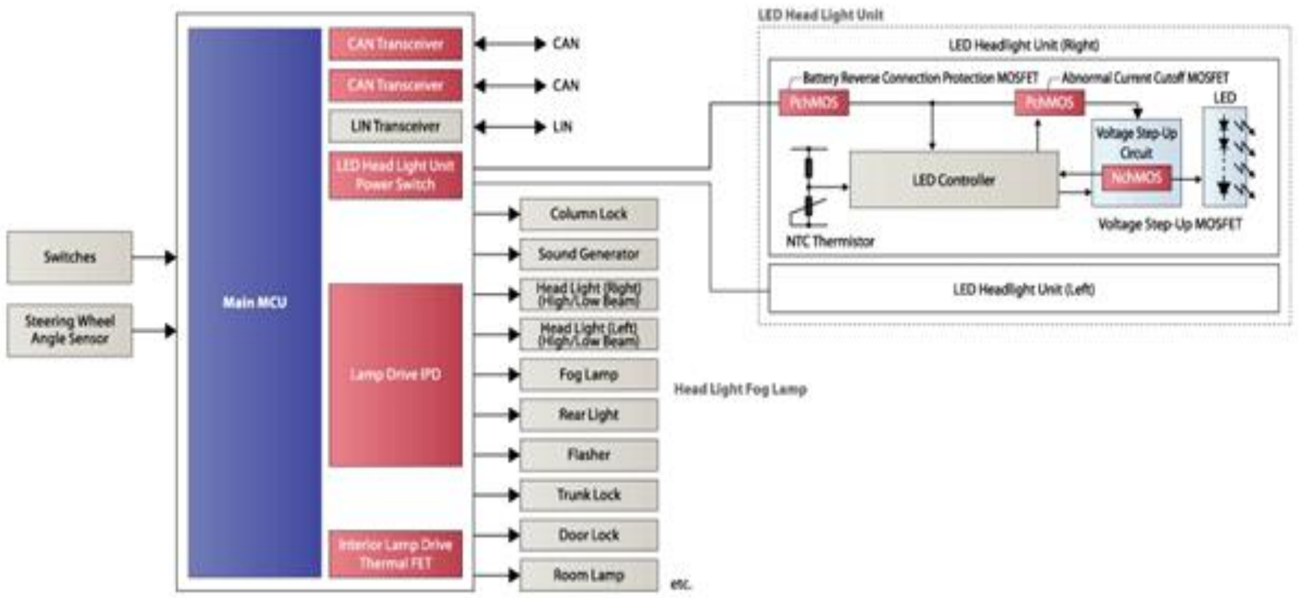
- Supply of the module with Infineon Supply IC&#39; s out of the OPTI REG-Linear, OPTIREG-Switcher, and OPTIREG-SB (System Basis Chip with Integrated drop voltage regulator or DCDC) families.
- Actuation of the vehicle function loads with Power ICs of the PROFET, Power PROFET, SPOC, SPIDER, HITFET, LITIX, Nova lithic, Multi MOSFET Driver and MOSFET Families.
- Sensing with pressure sensors of the XENSIV Family (and monitoring in the BCM) or integrated into the Power ICs. For car access, Infineon offers Transceiver and Receiver in Remote Keyless Entry (RKE) and Passive Keyless Entry (PKE).
- Control of the whole Vehicle Body with Traveo II and AURIX Families of Microcontroller.
- Communication with other car functions via the vehicle bus system by LIN or CAN Transceiver Families or integrated with Supply into the System Basis Chip Family.
- The number of body functions has increased, but also their variants of these functions required for different vehicle trim levels. For example; one module may support seat movement to bidirectional lift, slide, recline and adjust lumbar and headrests, as well as heating and cooling but this module may only include some of these for a low-end vehicle. This creates a need for flexibility and scalability of product families to create a BCM platform that can accommodate easy variants and changes throughout the design to years after SOP.
- Infineon’s Power Switch families PROPHET, SPOC, and SPIDER have the most scalable portfolio of protected switches in the market to enable this flexibility in hardware and software.

2.2.2 RENESAS:

A body control module (BCM) integrates several functions, including various lights and door locks. It has a gateway function for communication with a CAN or LIN. Such a unit requires components with CAN/LIN network support, a selection of package sizes and memory capacities to match a variety of system scales, low EMI to prevent radio noise interference with vehicle wiring systems, and low current consumption to reduce battery drain when in standby status.

Renesas offers kit solutions that include microcontrollers (MCUs), analog devices and power devices for optimized body control. We also supply power devices and LED headlight controllers with overcurrent detection and diagnostic functions, enabling low current consumption in LED headlight units, which have become increasingly popular in recent years.

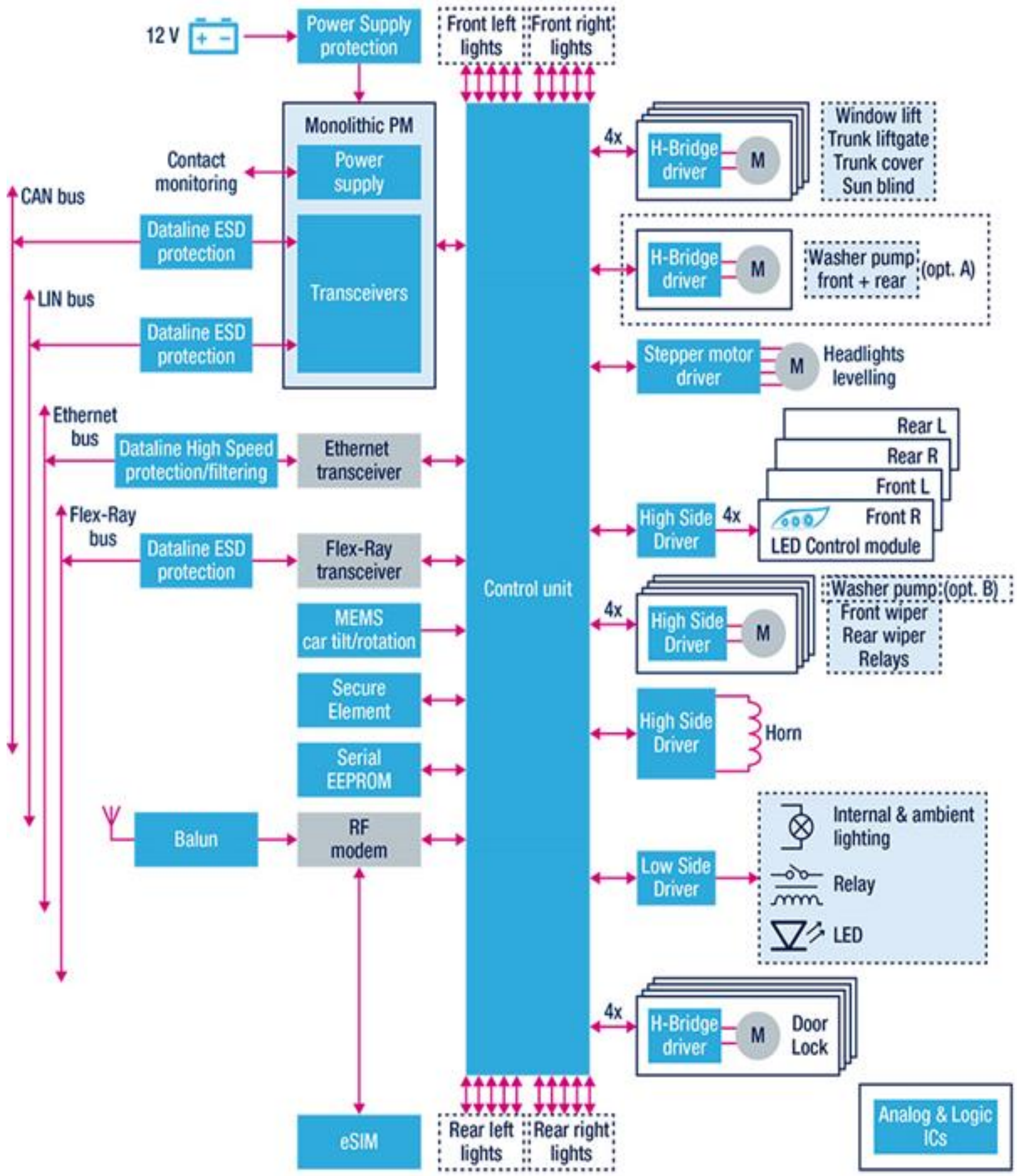
Electronic control units (ECUs) can now be found everywhere in automobiles, from power windows and airbags to lamps, mirrors, doors, and seats. Due to this trend, the burden on software development for ECUs has also increased. Renesas offers a wide variety of products for automotive bodies, including MCUs, SoCs, sensors and power management devices.



2.2.3 ATMEL:

Body Control Module (BCM): It is connected directly to the engine control module and is used as an authentication unit to enable engine start. It issues challenges and evaluates responses from the key to enable or disable access. 2. Base Station (BS): This module acts as a gateway between the key fob and the BCM. It communicates with the key via its physical interface. 3. Transponder (TP): The transponder unit receives data from the base station including commands and payload data and responds accordingly. Its internally stored secret key (128 bits) is used to encrypt challenges before it replies with a response prior to authentication.

The authentication protocol is based on challenge response topology and it can be implemented either as a unidirectional protocol where only a key is authenticated by the vehicle or as a bidirectional protocol where a key and a vehicle are both authenticated. Varying data frame sizes may be used to send and receive payload data. Plain text challenge data received by the fob is encrypted by the AES-128 hardware-based crypto module, after which the cipher response is sent to the base station. The energy to receive and transmit the protocol data is supplied by the magnetic field which is directly coupled from the base station coil to the transponder coil during the data exchange. The transponder can also be powered by an internal battery. This feature is used for fob configuration and during key fob debug and development.


12 V
Power Supply protection
Front left lights
Front right lights
Monolithic PM
Power supply
Transceivers
Contact monitoring
CAN bus
Dataline ESD protection
LIN bus
Dataline ESD protection
Ethernet bus
Dataline High Speed protection/filtering
Ethernet transceiver
Flex-Ray bus
Dataline ESD protection
Flex-Ray transceiver
MEMS car tilt/rotation
Secure Element
Serial EEPROM
Balun
RF modem
eSIM
Control unit
Rear left lights
Rear right lights
4x
H-Bridge driver
M
Window lift
Trunk liftgate
Trunk cover
Sun blind
H-Bridge driver
M
Washer pump front + rear
(opt. A)
Stepper motor driver
M
Headlights levelling
Rear L
Rear R
Front L
Front R
LED Control module
High Side Driver
4x
4x
High Side Driver
M
Washer pump (opt. B)
Front wiper
Rear wiper
Relays
High Side Driver
Horn
Low Side Driver
Internal & ambient lighting
Relay
LED
4x
H-Bridge driver
M
Door Lock
Analog & Logic ICs

2.3 Comparative analysis of sensors and actuators in BCM (Car industry):

2.3.1 Sensors:

2.3.1.1 Engine Speed Sensor

The main purpose of this sensor is to monitor the crankshaft' rotating speed. So that fuel injection & the engine timing can be controlled.

2.3.1.2 Spark Knock Sensor

The spark knock sensor is used to ensure whether the fuel is burning smoothly, otherwise it will cause an unexpected ignition.

2.3.1.3 Fuel Temperature Sensor

The fuel sensor is used to check the temperature of the fuel continually whether the fuel utilization is optimum or not. If the fuel of the engine is cold, then it will take much time to burn due to its high density.

2.3.1.4 Throttle Position Sensor

The throttle position sensor in automobiles mainly uses feedback carburetion & electronic fuel injection (EFI). It informs the computer regarding the throttle opening rate as well as the position of the relative throttle.

2.3.1.5 Mass Air Flow Sensor

This sensor is used in the engine of the car. This sensor can be controlled by a computer and can calculate the air density in the engine.

2.3.1.6 Coolant Sensor

The coolant sensor is the most significant sensor used in automobiles. Because the computer depends on the sensor inputs to control all the functions. For instance, turn ON/OFF the EFE system (Early Fuel Evaporation), retard, spark advance, the flow of EGR, and canister purge.

2.3.2 Actuators

2.3.2.1 Generator Current Control

When turning on headlamps or heating wire in idling, engine rpm will instantly fall down and then recover due to increased generator load. At that time increasing electrical load will generate rapid engine rpm change, resulting in vibration and poor comfortableness. Generator current control system depends on the engine ECU.

2.3.2.2 Cooling Fan Control

In order to maximize cooling efficiency and minimize cooling fan motor drive current, radiator fan and condenser fan speeds are controlled using three speed modes such as low, medium, and high speed, on the basis of coolant temperature, car speed, air conditioning switch signal, and condition compressor operation signal.

2.3.2.3 Spark Plugs

Spark plugs transmit electrical energy that turns fuel into working energy.

2.3.2.4 Fuel Pump

Fuel pump is powered from the vehicle battery and connected to the engine ECU, to give the engine the fuel at the right pressure suitable for its work.

2.3.2.5 Injectors

Injectors are an injection nozzle with solenoid that is controlled by ECM.

Comparsion of Actuators

Parameters	**L298**	**L293x**
Voltage range	+5 to +46V	4.5 V to 36 V
Storage and Junction Temperature	–40 to 150	65 150
protection	Internal ESD protection	Over temperature
Maximum Peak motor current:	1.2A	3A
Application	• Automatic door control systems. • CNC machines	• Stepper Motor Drivers • DC Motor Drivers

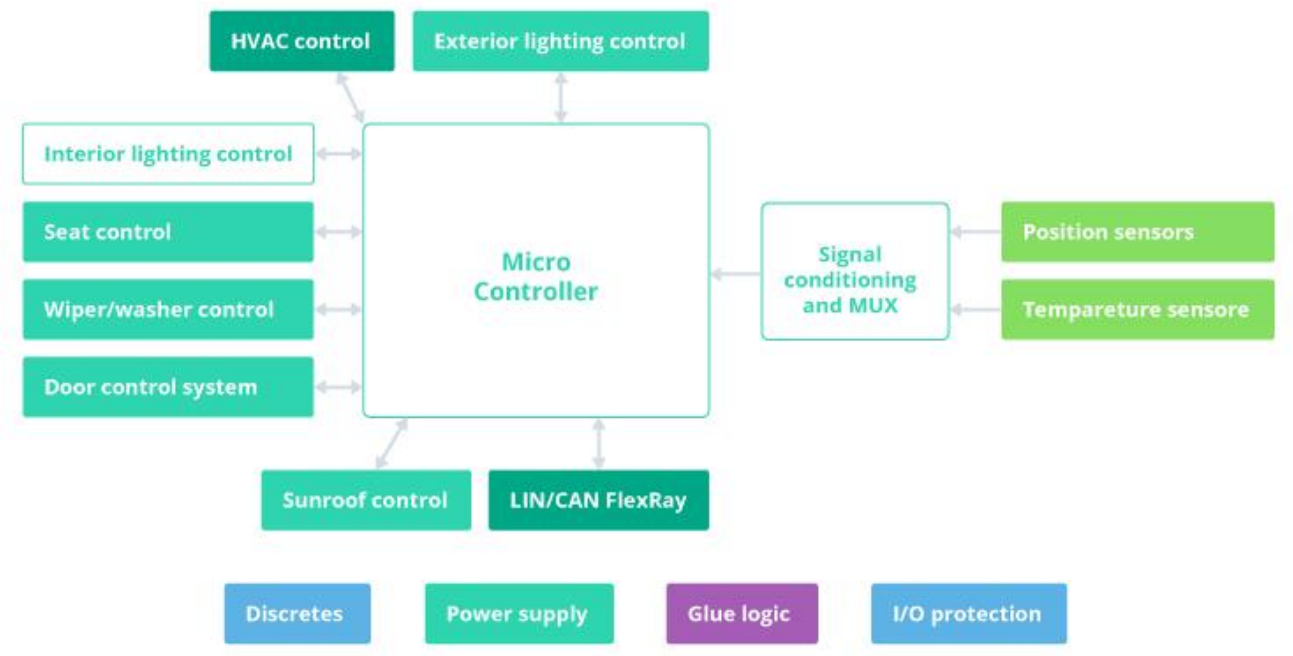
Comparsion of Sensors

Temperature sensor	**Infineon**	**Renesas**
Temperature Accuracy	+/- 4 degree Celsius	+/- 2 degree Celsius
Supply Voltage	1.2V – 3.6 V	2.3V – 5.5V
Operational Range	-40 deg Cel to +85 deg Cel	-30 deg Cel to 60 deg Cel
Applications	Local Weather Station and Thermostat	Outdoor Navigation

2.4 Features of BCM:

- Ensuring safety, testing, and control of crucial electrical loads, including lights, immobilizers, air conditioning systems, locking systems, and windscreen wipers

- Maintaining communication between integrated control units via the vehicle bus system (CAN, LIN, or Ethernet)
- Working as an integration gateway
- Providing a user-friendly interface for complex data management



General representation of BCM

ACTIVITY 3[TEAM]
How is MBD applicable in the transportation domain (Car industry)?
MBD as the following subsystems

1. Function Requirement
2. Function Development
3. Software Development
4. Software Integration
5. Function Integration
6. Function Calibration

1. Function Requirement

The process starts with the analysis of the function requirements. This is a very important step because it defines what is expected from the control software in terms of functionality. There is an entire engineering area dedicated to requirements, which is called requirements engineering. Some companies have dedicated requirements engineers working on a daily basis defining requirements. Most of the time, the system engineer is responsible with requirement definition.

The function requirements describe what the software should do from the functional point of view. When defining requirements, the author will use dedicated keywords, which have a precise meaning. The keywords are defined as follows

Keyword	**Meaning**
MUST	This word, or the terms “REQUIRED” or “SHALL”, mean that the definition is an absolute requirement of the specification.
MUST NOT	This phrase, or the phrase “SHALL NOT”, means that the definition is an absolute prohibition of the specification.
SHOULD	This word, or the adjective “RECOMMENDED”, mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
SHOULD NOT	This phrase, or the phrase “NOT RECOMMENDED” mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full Implications should be understood and the case carefully weighed before implementing any behavior described with this label.
MAY	This word, or the adjective “OPTIONAL”, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

Example: In case of an engine management system (EMS) software development, there should be a protection function which doesn’t allow the engine to exceed the maximum possible value (e.g., 6500 rpm). The requirement will be written as:
“The engine control software MUST limit the engine speed at 6500 rpm.”

A function requirement, most of the time, describes what should happen and doesn't necessarily give details regarding the exact implementation. If you notice, the above requirement specifies what the engine controls should do, but not how it should be achieved. It's often to the function developer to decide on the implementation details.

When defining requirements, the requirements/systems engineer must work closely with the function developer to make sure that the requirements are clearly defined and implementable. Most of the cases the requirement come in a form of a document (*.pdf) or managed in a dedicated requirements database like DOORS.

2. Function Development

MBSE based on SysML helps capture and process information from multiple departments within a single model accessible to all stake holders. The user can define dependencies within each diagram, changes in one diagram of the model are automatically reflected in other model diagram that need to be updated based on changes. This feature allows organization to maintain consistency, accuracy and updated information easily understandable in the MBSE model. Moreover, different checks can be performed to ensure the completeness of the model. MBSE also enables in defining hierarchy of model from top level system to subsystem and the detailed component level.

TOOLS BASED ON SPECIFIC JOB:

1. Brainstorming-Requirements Brainstorming is used in requirement gathering to get as many ideas as possible from group of people.
    - Document Analysis
    - Focus Group
    - Interface analysis
    - Interview
    - Observation
    - Prototyping
    - Requirement Workshops
2. IBM Rhapsody -System Medalling and sub-system modelling IBM Rhapsody for Systems Engineers is an integrated, model-driven systems engineering environment for complex projects. It uses Systems Modelling Language (SysML) and Unified Modelling Language (UML) to enable rapid requirements analysis and visual, model-driven design. ✓ Cameo Systems Modeler- System Modelling No Magic's Cameo Systems Modeler is a Model-Based Systems Engineering (MBSE) solution in one easy-to-use package, enabling single users or an entire engineering team to create, collaborate, and manage systems requirements and designs.
3. Capella Open Source MBSE tool-Function Simulation tool Capella is an Open Source MBSE tool implementing the Arcadia method. It is a comprehensive, extensible and field-proven MBSE solution focusing on the design of systems architectures. Capella is strongly driven by the current practices and concerns of system engineering practitioners and can be easily mastered by those familiar with SysML language.

3. Software Development

Software developer performs the actual software development. Almost all electronic control units (ECUs) within a modern vehicle are programmed in C language. The input for the software developer is either the model developed by the function developer or a document with detailed description of the function.

In addition to the algorithm which needs to be coded, the software developer needs to know what data types needed for each of the software variables. The software developer is also responsible in delivering an optimized C code, in:

- Embedded Coder (needs Simulink to be pre-installed) from Mathworks
- Target Link (needs Simulink to be pre-installed) from dSpace
- ASCET from ETAS

With automatic code generation capabilities, the developer can perform Software in the Loop (SiL) test. This technique allows to compile and run the production intent C code on the development laptop/computer. By doing SiL tests, the function developer can check if the software is providing the required functionality (compared to the model). Also, it is possible the see the rounding errors in case of fixed-point implementation of the C code. The output of the software development phase is the corresponding C code files (*.c and *.h) for the required functionality.

4. Software Integration

- The software integration is the process of combining together all software modules required for particular projects.
- Integration means compiling and linking all the files (*.c, *.h) for a specific application and turning them into a machine code file (*.hex and *. a21).
- At this stage, the testing of the required functionality is done at system level (compute ECU software).
- The purpose of the integration test is to verify the interaction between the software modules and to check the impact of the functional changes on the legacy code (through non-regression tests).
- Most of the time, integration tests are performed using hardware in the loop (Hil) techniques.
- A hill environment consists of using a simulator which has the role of replacing the real vehicle.
- The simulator will have all the electrical connections (battery supply, sensors and actuators) required by the electronic control module (ECU). Also, it will simulate the dynamic behavior of the vehicle in order to be able to test closed loop control scenarios.
- Hil environments can be at the component level (engine, transmission, etc.) or at the vehicle level.
- A vehicle level usually connects in a network several simulators and ECUs. The advantage of the vehicle level is the possibility to test a disturbed function.
- The tests are verification tests. Verification confirms that the software properly reflects the specified requirements. In the other words, verification ensures that “you built it right”.

5. Function Integration

The function integration is usually performed by the function developer or by a test engineer. The purpose is to test the developed function at the vehicle level (production model or a prototype). The purpose of the vehicle test is to validate the correct implementation of the requirements and the integration with the other control modules (e.g., Transmission Control Module, Anti-lock Braking System control module, etc.).

The function integration is performed with a computer connected at the target ECU. The target electronic control unit can be the Powertrain Control Module (PCM), Transmission Control Module (TCM), Body Control Module (BCM), etc.

The laptop must be equipped with proper tools for accessing software variables (INCA, Canape) and network bus messages (Catalyser). These tools connect to the target ECU using different communication standards (ETK, CAN, etc.)

Vehicle testing is performed after a defined test scenario. The test engineer will drive the vehicle in certain operating condition suitable for the activation of the function subject to test. For example, in order to test the engine speed limit, the engine needs to be accelerated until it reaches the maximum speed. For this particular example it is recommended that this function is tested on a simulation environment (HiL) before vehicle testing.

The function integration tests are validation vehicle testing is performed after a defined test scenario. The test engineer will drive the vehicle in certain operating condition suitable for the activation of the function subject to test. For example, in order to test the engine speed limit, the engine needs to be accelerated until it reaches the maximum speed. For this particular example it is recommended that this function is tested on a simulation environment (HiL) before vehicle testing.

The function integration tests are validation tests. Validation confirms that the product, as provided, will fulfil its intended use. In other words, validation ensures that ”you built the right thing.”

6. Function Calibration

- Function calibration is the last step of the model-based design
- A number of models for car following have been proposed for homogeneous traffic and some of have been modified are adapted to represent mixed traffic conditions.
- The last step in software development is the function calibration. This task is performed by a calibration/tuning engineer.
- Most of the software control function are generic, they are suitable for different vehicle applications (variants). The role of the calibration engineer is to set the right parameters for the software functions. For our example, the calibration is the value of the maximum speed limit. Depending on the engine type, it can have different values (e.g., 6500 rpm for gasoline engines, 4500 rpm for diesel engines).
- The calibration engineer will tune the software parameters so that it achieves best performances in terms of drivability, performance, emissions, etc.

- The function calibration activities, most of the time, are performed on a vehicle with the same setup as for a function integration.
- The main advantages and disadvantages of the V-cycle for software development process are listed in the table below:

Advantages:

- It is relatively simple and easy to use
- Each phase has clear roles and specific deliverables
- Works very well and efficient if the requirements are clear and understood by the developer
- As there are many calibration kits available on the market, software calibration is very well suited to general monitor use. Since software calibration can be used to create profiles for the monitors users currently use, it&#39; s also an effective means of ongoing color management.

Disadvantages:

- It is rigid, doesn’t allow shortcuts in case of emergency situations
- There are no early prototypes of the software since it’s developed at a later stage
- There is no clear process on how to handle problems found during testing phases
- Since monitor adjustment is conducted by the user, problems with precision can occur, with minor variations occurring each time calibration is conducted.

Hardware calibration:
Advantages:

- As this method controls the monitor hardware directly it offers high precision and good gradation characteristics. This process is also easy to do, particularly as some monitors can be set up to calibrate automatically.

Disadvantages:

- Since calibration monitors are specially designed, they cost more than ordinary monitors.

Link: https://x-engineer.org/graduate-engineering/modeling-simulation/model-based-design/essential-aspects-of-the-v-cycle-software-development-process/

ACTIVITY 4[TEAM]

Development of algorithm using a datasheet developed from Andro Sensor App

Location Mapping :

```
%% Import data from spreadsheet
% Script for importing data from the following spreadsheet:
%
%    Workbook: C:\Users\training\Desktop\matlight.xlsx
%    Worksheet: Sensor_record_20201217_162934_A
%
% Auto-generated by MATLAB on 18-Dec-2020 12:21:12

%% Set up the Import Options and import the data
opts = spreadsheetImportOptions("NumVariables", 31);

% Specify sheet and range
opts.Sheet = "Sensor_record_20201217_162934_A";
opts.DataRange = "A2:AE24";

% Specify column names and types
opts.VariableNames = ["ACCELEROMETERXms", "ACCELEROMETERYms", "ACCELEROMETERZms",
"GRAVITYXms", "GRAVITYYms", "GRAVITYZms", "LINEARACCELERATIONXms",
"LINEARACCELERATIONYms", "LINEARACCELERATIONZms", "GYROSCOPEXrads",
"GYROSCOPEYrads", "GYROSCOPEZrads", "LIGHTlux", "MAGNETICFIELDXT",
"MAGNETICFIELDYT", "MAGNETICFIELDZT", "ORIENTATIONZazimuth", "ORIENTATIONXpitch",
"ORIENTATIONYroll", "PROXIMITYi", "SOUNDLEVELdB", "LOCATIONLatitude",
"LOCATIONLongitude", "LOCATIONAltitudeM", "LOCATIONAltitudegoogleM",
"LOCATIONSpeedKmh", "LOCATIONAccuracyM", "LOCATIONORIENTATION", "SatellitesInRange",
"TimeSinceStartInMs", "YYYYMODDHHMISS_SSS"];
opts.VariableTypes = ["double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "categorical", "string", "double", "string", "categorical",
"double", "string"];

% Specify variable properties
opts = setvaropts(opts, ["LOCATIONSpeedKmh", "LOCATIONORIENTATION",
"YYYYMODDHHMISS_SSS"], "WhitespaceRule", "preserve");
opts = setvaropts(opts, ["LOCATIONAltitudegoogleM", "LOCATIONSpeedKmh",
"LOCATIONORIENTATION", "SatellitesInRange", "YYYYMODDHHMISS_SSS"], "EmptyFieldRule",
"auto");

% Import the data
tbl = readtable("C:\Users\training\Desktop\matlight.xlsx", opts, "UseExcel", false);

%% Convert to output type
ACCELEROMETERXms = tbl.ACCELEROMETERXms;
ACCELEROMETERYms = tbl.ACCELEROMETERYms;
ACCELEROMETERZms = tbl.ACCELEROMETERZms;
GRAVITYXms = tbl.GRAVITYXms;
GRAVITYYms = tbl.GRAVITYYms;
```

```
GRAVITYZms = tbl.GRAVITYZms;
LINEARACCELERATIONXms = tbl.LINEARACCELERATIONXms;
LINEARACCELERATIONYms = tbl.LINEARACCELERATIONYms;
LINEARACCELERATIONZms = tbl.LINEARACCELERATIONZms;
GYROSCOPEXrads = tbl.GYROSCOPEXrads;
GYROSCOPEYrads = tbl.GYROSCOPEYrads;
GYROSCOPEZrads = tbl.GYROSCOPEZrads;
LIGHTlux = tbl.LIGHTlux;
MAGNETICFIELDXT = tbl.MAGNETICFIELDXT;
MAGNETICFIELDYT = tbl.MAGNETICFIELDYT;
MAGNETICFIELDZT = tbl.MAGNETICFIELDZT;
ORIENTATIONZazimuth = tbl.ORIENTATIONZazimuth;
ORIENTATIONXpitch = tbl.ORIENTATIONXpitch;
ORIENTATIONYroll = tbl.ORIENTATIONYroll;
PROXIMITYi = tbl.PROXIMITYi;
SOUNDLEVELdB = tbl.SOUNDLEVELdB;
LOCATIONLatitude = tbl.LOCATIONLatitude;
LOCATIONLongitude = tbl.LOCATIONLongitude;
LOCATIONAltitudeM = tbl.LOCATIONAltitudeM;
LOCATIONAltitudegoogleM = tbl.LOCATIONAltitudegoogleM;
LOCATIONSpeedKmh = tbl.LOCATIONSpeedKmh;
LOCATIONAccuracyM = tbl.LOCATIONAccuracyM;
LOCATIONORIENTATION = tbl.LOCATIONORIENTATION;
SatellitesInRange = tbl.SatellitesInRange;
TimeSinceStartInMs = tbl.TimeSinceStartInMs;
YYYYMODDHHMISS_SSS = tbl.YYYYMODDHHMISS_SSS;

%% Clear temporary variables
clear opts tbl
LOCATIONLatitude=13.04
LOCATIONLongitude=77.61
geoplot(LOCATIONLatitude,LOCATIONLongitude,'--or');
```

HTML Link generated from MATLAB: File:///C:/Users/Sushma/Desktop/html/sensors_code.html

ACTIVITY 5[TEAM]
Automation script Generation:
Automation script can be generated using

1. Auto build
2. Auto fetch and replace
3. Auto find

HTML Link generated from MATLAB:
File:///C:/Users/Sushma/Desktop/html/autobuild_block_line.html

Summary:
1. My contributions:
We are assigned with totally six team tasks/activities.

In activity 2, I Wrote Body Control Module and Comparative analysis of Sensors and Actuators and also comparsion of Actuators, comparsion of Sensors.

In activity 3, I wrote all the functions which included in the MBD application in transportation domain.

In activity 4, my task is to write the algorithm to find the Location using data sheet generated from the andro sensor app using MATLAB.

In activity 5, my task is automating the script based in the concept using auto find using MATLAB and Simulink.

In activity 6, my task is to develop a subsystem of External Lights of car using Simulink.

2.Challenges faced and how were they overcommed:

As we are new to the MATLAB and Simulink ,I faced many problems to develop a subsystem of a car and also to do automation scripting. Based on the inputs given by GEA team helped us a lot to overcome these problems. Even colleagues helped me a lot to learn new things and how to implement it.