JSPM's

Rajarshi Shahu College of Engineering
Tathawade, Pune-411033

(An Autonomous Institute Affiliated to Savitribai Phule Pune University, Pune)

Automotive Electronics
[EC4103B]

Department of Electronics and Telecommunication Engineering

JSPM's, Rajarshi Shahu College of Engineering,
Tathawade, Pune, Maharashtra 411033

Vision and Mission of the Institute

Vision of Institute

To satisfy the aspirations of youth force, who want to lead the nation towards prosperity through techno economic development.

Mission of Institute

To provide, nurture and maintain an environment of high academic excellence, research and entrepreneurship for all aspiring students, which will prepare them to face global challenges maintaining high ethical and moral standards.

Vision and Mission of the Department

Vision of Department

To create an educational environment to meet the challenges of modern Electronics and Telecommunication engineering industry through state of art technical knowledge and innovative approach.

Mission of Department

1. To entrust the students with fundamentals of Electronics and Telecommunication Engineering for successful carrier
2. To enable students to pursue higher education, research and promote Entrepreneurship
3. To serve the nation through techno-social development.

Programme Educational Objectives

PEO-I: The graduate shall have successful professional carrier in Electronics and Telecommunication Engineering with leadership and teamwork qualities.

PEO-II: Graduate shall utilize functional and disciplinary skills to address diversified engineering problems with social concern.

PEO-III: The graduate shall explore engineering capabilities to resolve technical problems and engage in lifelong learning and research.

Program Outcomes

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and E&TC tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. JSPM's Rajarshi Shahu College of Engineering Department of E&TC Engineering
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes (PSOs)

Upon successful completion of UG course in Information Technology, the students will attain following Program Specific Outcomes:

1. Graduate will demonstrate the ability to apply knowledge of Electronics and Telecommunication to identify, formulate and solve Engineering problems useful to society.
2. Graduate will demonstrate an ability to design, implement and analyze various functional elements of Electronics and Telecommunication domain, interpret data and work with multidisciplinary approach.
3. Graduate will demonstrate the analytical and managerial skills with a virtue of continued learning; carry out the professional and entrepreneurial responsibilities in Electronics and Telecommunication Engineering field considering environmental issues.

Lab Outcomes:

After completion of Laboratory sessions the students will be able to:-

LO1: Identify and explain the functional and performance aspects of major automotive subsystems through study and simulation..

LO2: Demonstrate selection and interfacing of sensors and actuators for automotive applications like lighting, wiper, and obstacle detection systems.

LO3: Design and simulate control systems for vehicle dynamics such as adaptive cruise control and suspension system using MATLAB Simulink.

LO4: Implement and analyze in-vehicle communication networks using CAN protocol through MATLAB Simulink modeling.

LO5: Examine and interpret the response of automotive systems under various operating and control conditions using simulation and diagnostic tools.

LO6: Develop teamwork, documentation, and problem-solving skills through integrated automotive system design and analysis tasks.

Laboratory Details

Laboratory Equipment list

Sr. No	Equipment/Software details	Cost	Manufacturer
1	Tinkercad	Freeware	AUTODESK
2	MATLAB Simulink	Institute License	MathWorks, Inc

List of Experiments

EXPERIMENT NO.1

Aim: -Demonstration of an anyone automobile engine with reference to characteristics.

Objectives: - To know different components of I. C. Engines and their function.

Introduction

An IC engine is one in which the heat transfer to the working fluid occurs within the engine itself, usually by the combustion of a fuel with the oxygen of the air. In external combustion engines, heat is transferred to the working fluid from the combustion gases via a heat exchanger. e.g. steam engines; Stirling engines.IC engines include spark ignition (SI) engines using petrol as a fuel, and compression ignition (CI) engines (usually referred to as Diesel engines) using fuel oil,
DERV, etc as a fuel. In these engines there is a sequence of processes:

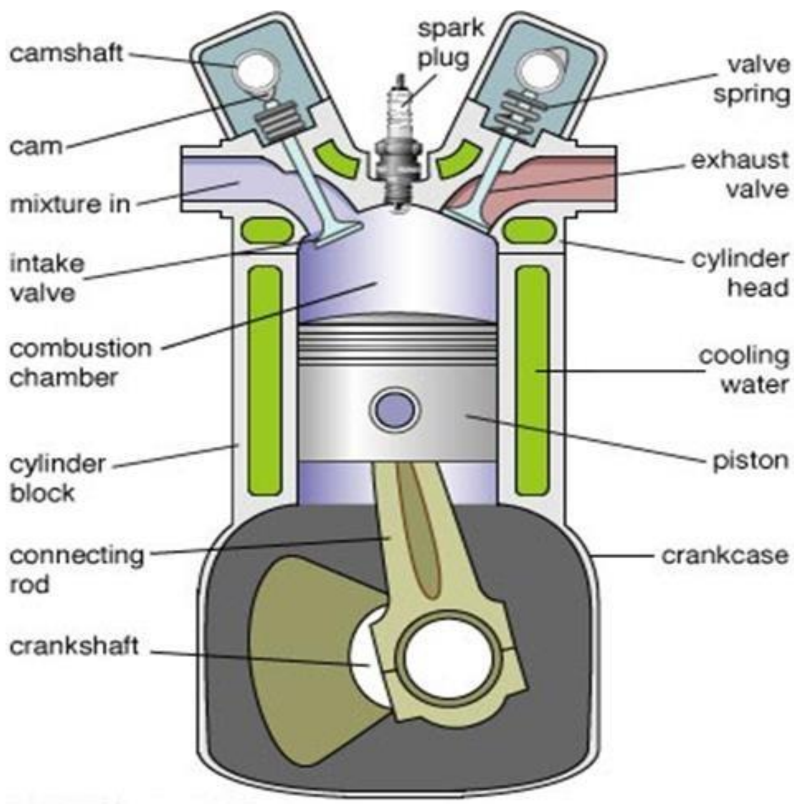
- Induction
- Compression
- Combustion
- Expansion
- Exhaust

Various Parts of IC engines are as follows:

The main components of the reciprocating internal combustion engine are shown in Figure. Engine parts are made of various materials and perform certain functions, some of which will be explained: cylinder block (g) it is integral with crankcase (m), both are made of cast iron. The piston (e) reciprocates inside the cylinder, which includes the combustion chamber.

The piston is connected to the connecting rod (h) by piston pin (f). This end of the connecting rod is known as the small end. The other end of the connecting rod called the big end is connected to the crank arm by crankpin (I). Camshaft (u) makes the cam (t) to rotate and move up and down the valve rod through the tappet (r). Mainly each cylinder has two valves; one is an admission or suction valve and the other is the exhaust valve. The ignition

system consists of a battery, an ignition coil, a distributor with cam and breaker points, and a spark plug for each cylinder. In diesel engines, there is an injection system instead of the ignition system.



1) **Cylinder:-** A cylinder is the central working part of a reciprocating engine or pump, the space in which a piston travels. Multiple cylinders are commonly arranged side by side in a bank, or engine block, which is typically cast from aluminum or cast iron before receiving precision machine work. Cylinders may be sleeved (lined with a harder metal) or sleeveless. A cylinder's displacement, or swept volume, can be calculated by multiplying its cross-sectional area (the square of half the bore by pi) and again by the distance the piston travels within the cylinder (the stroke). The engine displacement can be calculated by multiplying the swept volume of one cylinder by the number of cylinders.
2) **Cylinder head:-** In an internal combustion engine, the cylinder head (often informally abbreviated to just head) sits above the cylinders on top of the cylinder block. It closes in the top of the cylinder, forming the combustion chamber. This joint is sealed by a head gasket. In most engines, the head also provides space for the passages that feed air and fuel to the cylinder, and that allows the exhaust to escape. The head can also be a place to mount the valves, spark plugs, and fuel injectors.
3) **Piston:** The piston of an internal combustion engine is acted upon by the pressure of the expanding combustion gases in the combustion chamber space at the top of the cylinder. This force then acts downwards through the connecting rod and onto the crankshaft. The connecting rod is attached to the piston by a swiveling gudgeon pin. This pin is mounted within the piston: unlike the steam engine, there is no piston rod or crosshead (except big two-stroke engines).
4) **Piston ring:-** A piston ring is a split ring that fits into a groove on the outer diameter of a piston in a reciprocating engine such as an internal combustion engine or steam engine.

The three main functions of piston rings in reciprocating engines are :

1 Sealing the combustion chamber so that there is no transfer of gases from the combustion chamber to the crank.
2 Supporting heat transfer from the piston to the cylinder wall.
3 Regulating engine oil consumption.
4 The gap in the piston ring compresses to a few thousandths of an inch when inside the cylinder bore.

5 **Gudgeon pin:-** In internal combustion engines, the gudgeon pin connects the piston to the connecting rod and provides a bearing for the connecting rod to pivot upon as the piston moves. In very early engine designs (including those driven by steam and also many very large stationary or marine engines), the gudgeon pin is located in a sliding crosshead that connects to the piston via a rod. Gudgeon is a pivot or journal
6 **Connecting rod**:- In modern automotive internal combustion engines, the connecting rods are most usually made of steel for production engines, but can be made of T6- 2024 and T651-7075 aluminum alloys[citation needed] (for lightness and the ability to absorb high impact at the expense of durability) or titanium (for a combination of lightness with strength, at higher cost) for high performance engines, or of cast iron for applications such as motor scooters. They are not rigidly fixed at either end so that the angle between the connecting rod and the piston can change as the rod moves up and down and rotates around the crankshaft. Connecting rods, especially in racing engines, may be called "billet" rods, if they are machined out of a solid billet of metal, rather than being cast or forged.
7 **Crank Shaft:-** The crankshaft, sometimes casually abbreviated to crank, is the part of an engine which translates reciprocating linear piston motion into rotation.

To convert the reciprocating motion into rotation, the crankshaft has "crank throws" or "crankpins", additional bearing surfaces whose axis is offset from that of the crank, to which the "big ends" of the connecting rods from each cylinder attach. It typically connects to a flywheel, to reduce the pulsation characteristic of the four-stroke cycle, and sometimes a torsional or vibrational damper at the opposite end, to reduce the torsion vibrations often caused along the length of the crankshaft by the cylinders farthest from the output end acting on the torsional elasticity of the metal.

8 **Engine bearing**:-Bearing is a device supporting a mechanical element and providing its movement relative to another element with minimum power loss. The rotating components of internal combustion engines are equipped with sleeve type sliding

bearings. The reciprocating engines are characterized by the cycling loading of their parts including bearings. Such character of the loads is a result of alternating pressure of combustion gases in the cylinders. Rolling bearings, in which a load is transmitted by rolls (balls) to a relatively small area of the ring surface, can not withstand under the loading conditions of internal combustion engines. Only sliding bearings providing a distribution of the applied load over a relatively wide area may work in internal combustion engines.

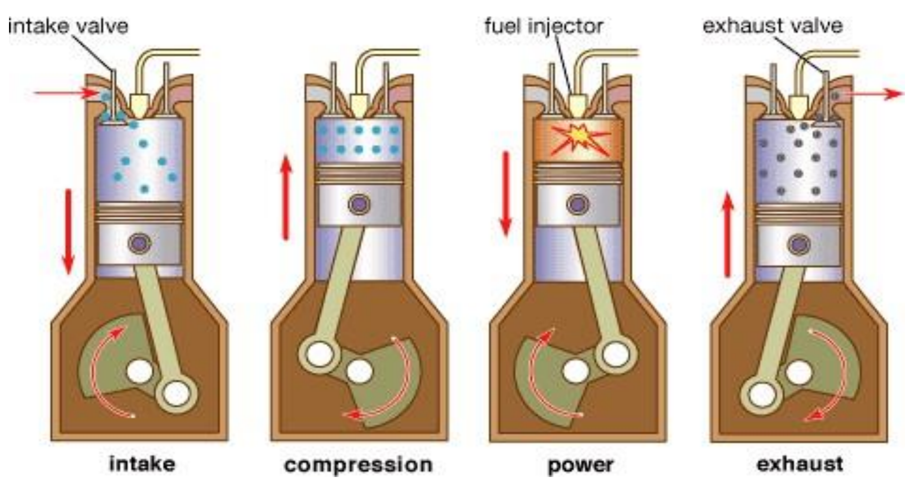
9 **Crank Case: -** In an internal combustion engine of the reciprocating type, the crankcase is the housing for the crankshaft. The enclosure forms the largest cavity in the engine and is located below the cylinder(s), which in the Maulticylinder engine are usually integrated into one or several cylinder blocks. Crankcases have often been discrete parts, but more often they are integral with the cylinder bank(s), forming an engine block. Nevertheless, the area around the crankshaft is still usually called the crankcase. Crankcases and other basic engine structural components (e.g., cylinders, cylinder blocks, cylinder heads, and integrated

combinations thereof) are typically made of cast iron or cast aluminum via sand casting.

10 **Flywheel: -** A flywheel is a rotating mechanical device that is used to store rotational energy. Flywheels have a significant moment of inertia and thus resist changes in rotational speed. The amount of energy stored in a flywheel is proportional to the square of its rotational speed. Energy is transferred to a flywheel by applying torque to it, thereby increasing its rotational speed, and hence its stored energy. Conversely, a flywheel releases stored energy by applying torque to a mechanical load, thereby decreasing its rotational speed.

11 **Governer:-** A governor, or speed limiter, is a device used to measure and regulate the speed of a machine, such as an engine. A classic example is a centrifugal governor, also known as the Watt or fly-ball governor, which uses weights mounted on spring- loaded arms to determine how fast a shaft is spinning, and then uses proportional control to regulate the shaft speed.

Figure:- Working of IC engine



Conclusion:

Students should write the conclusion. Hence demonstration of IC engine is conducted**.**

Questions:

1. What is the I. C. engine? State its applications?
2. How do you classify I. C. Engines?
3. Explain different components of I. C. Engine?

EXPERIMENT NO. 2

Aim: - Implement Automotive Light System

Objectives: -.

- To design and simulate a vehicle lighting system (Parking, Fog, and Head Lights) with both manual and automatic modes.
- To implement switch debouncing in code for reliable button press detection.

Components:-

Arduino UNO
LEDs (3 – Parking, Fog, Head Light)
One LED (Auto Mode Indicator)
Resistors (330 Ω × 4)
Push Buttons × 4 (Parking, Fog, Head, Auto Mode)
Light Sensor (LDR)
Relay modules (optional)
Breadboard
Jumper wires

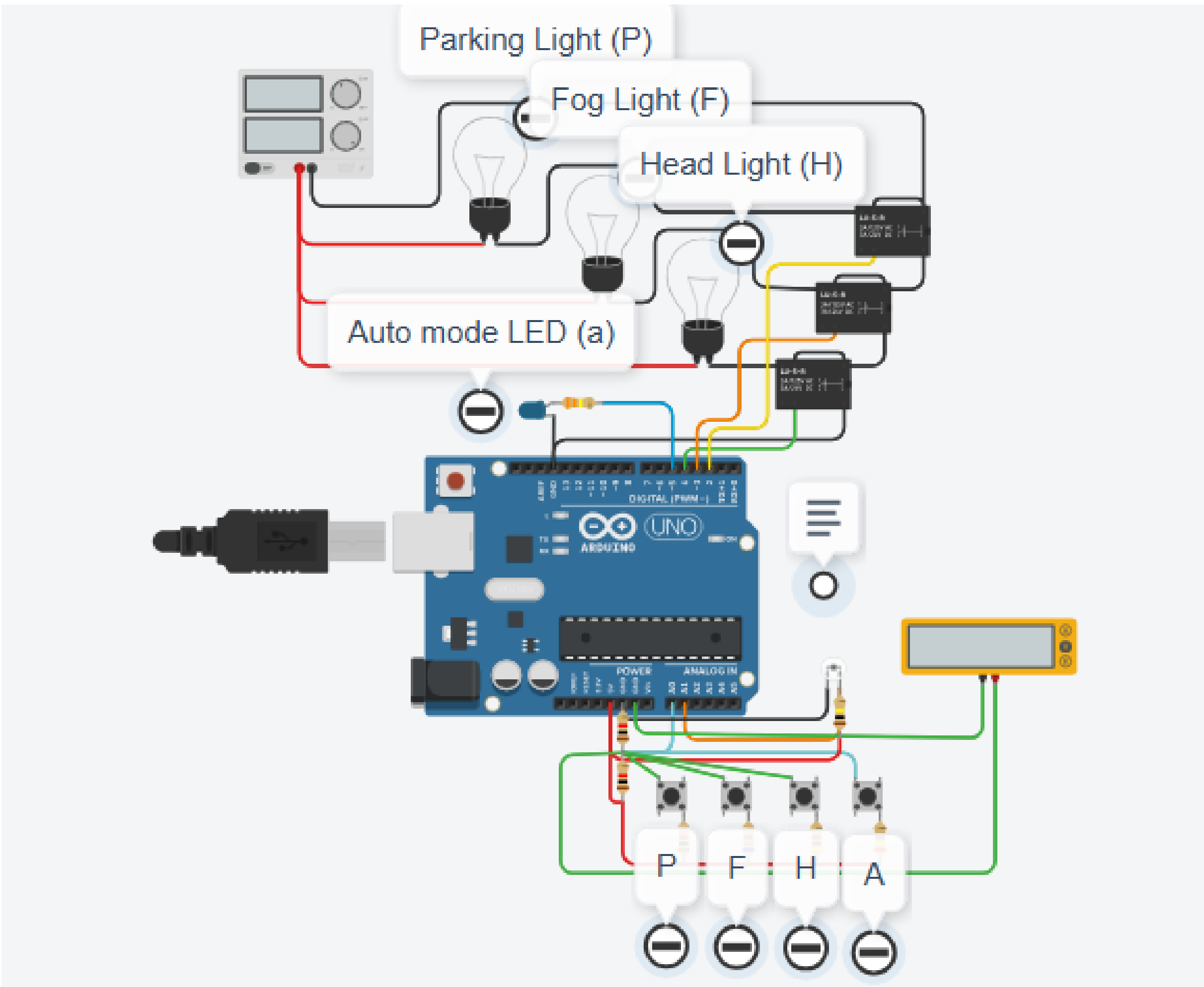
Theory:-

The automotive lighting system is essential for visibility and safety. In modern vehicles:

- Parking Lights provide visibility when the vehicle is stationary.
- Fog Lights improve visibility during foggy conditions.
- Head Lights illuminate the road ahead in low-light or nighttime driving.
- Automatic Mode uses sensors to detect ambient light and switch headlights on or off automatically.

This experiment uses Arduino UNO to control these lights via pushbuttons (manual) or a light sensor (automatic). Switch debouncing is implemented to prevent multiple triggers from a single press.

Circuit Diagram :-



Procedure :-

1 . Open Tinkercad and load the given circuit file.

2. Connect components as per the diagram.

3. Upload the given Arduino code to the UNO board.

4. Manual Mode: Test Parking, Fog, and Head Light buttons.

5. Auto Mode: Enable auto LED, adjust light sensor values to simulate day/night.

6. Use Serial Monitor to observe switch and sensor readings.

Observations :-

Action	Expected Output	Observed Output
Press Parking Light Switch (P)	Parking LED toggles	
Press Fog Light Switch (F)	Fog LED toggles	
Press Head Light Switch (H)	Head Light toggles (Manual Mode only)	
Press Auto Switch (A)	Auto Mode LED toggles, Head Light auto	
Light Sensor bright → Head OFF	Head Light OFF in Auto Mode	
Light Sensor dark → Head ON	Head Light ON in Auto Mode	

Result:-

- The Auto Vehicle Light System was successfully simulated.
- Manual and automatic operations worked as intended.
- Light sensor responded accurately to ambient light changes in auto mode.

Conclusion:-

The Auto Vehicle Light System was successfully implemented and simulated using Arduino UNO in Tinkercad. The system allowed both manual and automatic control of parking, fog, and head lights. Switch debouncing ensured accurate button detection, while the light sensor enabled automatic headlight operation based on ambient light levels. This experiment demonstrated the integration of hardware and software to mimic real-world automotive lighting systems, enhancing understanding of embedded control in vehicle electronics.

Questions for Oral:-

1. What is the purpose of debouncing in a switch?
2. Why do we use an LDR in an automotive lighting system?
3. How does the Arduino differentiate between multiple buttons using one analog pin?
4. Why are relays used in real vehicles for lighting circuits?
5. What modifications are required to implement this system in real vehicle

EXPERIMENT NO. 3

Aim: - To design and implement an **Obstacle Detection System**

Objectives: -.

To design and implement an **Arduino-based Obstacle Detection System** that measures the distance to an object using an ultrasonic sensor and provides visual (LED) and audio (buzzer) alerts based on the proximity of the object.

Components:-

S. No.	Component Name	Specification / Part No.	Quantity
1	Arduino Uno	ATmega328P	1
2	Ultrasonic Sensor	HC-SR04	1
3	LED	Red, Yellow, Green	1 each
4	Buzzer	5V DC	1
5	Breadboard	-	1
6	Jumper Wires	M-M, M-F	As req.
7	USB Cable	Type-A to Type-B	1
8	Resistors	220 Ω	3

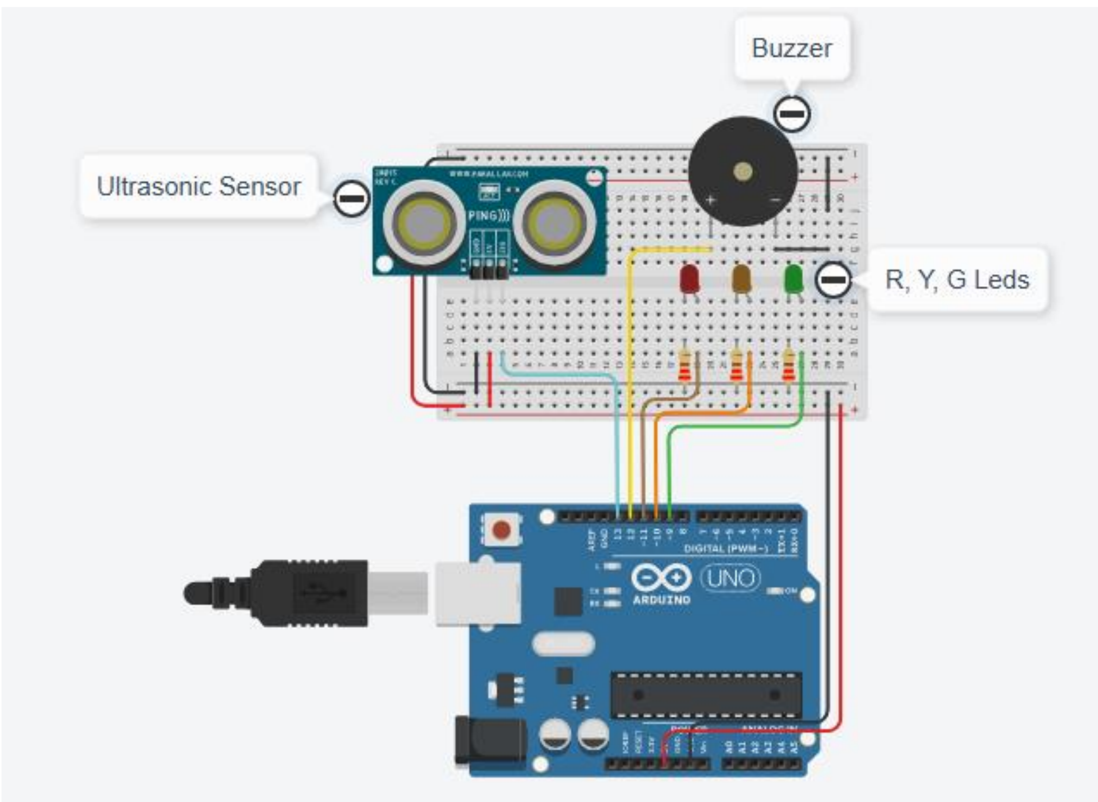
Theory:-

The **Obstacle Detection System** is based on the working principle of an **ultrasonic sensor**, which measures distance using sound waves. The sensor emits an ultrasonic pulse and measures the time taken for the echo to return after reflecting from an obstacle. The Arduino calculates the distance and triggers corresponding LEDs and buzzer:

- **Green LED ON** – Object far (>150 cm)
- **Yellow LED ON** – Object at moderate range (60–150 cm)
- **Red LED + Buzzer** – Object close (<60 cm)

The buzzer beeps faster as the obstacle gets closer, simulating a car parking sensor.

Circuit Diagram :-



Connections & Procedure :-

Arduino Pin	Component Connection
Pin 9	Green LED
Pin 10	Yellow LED
Pin 11	Red LED
Pin 12	Buzzer
Pin 13	Ultrasonic Sensor (Trig/Echo)
GND	All components' GND
5V	Ultrasonic Sensor & LEDs

1. Start Arduino and initialize pins for LEDs, buzzer, and ultrasonic sensor.
2. Send a 10 µs HIGH pulse from the Trig pin to the ultrasonic sensor.
3. Measure the time taken for the echo to return on the Echo pin.
4. Calculate distance using:

$$Distance = \frac{Time \times Speed\ of\ Sound}{2}$$

Here:

Speed of sound = 343 m/s → 0.0343 cm/µs, hence factor ≈ 0.01715.

5. Compare distance with set thresholds:
    - If distance > 150 cm → Green LED ON
    - If 60–150 cm → Yellow LED ON
    - If < 60 cm → Red LED ON and buzzer beeps with delay based on distance.
6. Repeat continuously.

Observations :-

Sr. No.	Measured Distance (cm)	LED Status	Buzzer Status	Remarks
1	> 150			
2	100			
3	50			

Result:-

- When the distance was **greater than 150 cm**, only the green LED turned ON.
- At distances between **60 cm and 150 cm**, the yellow LED turned ON.
- At distances **less than 60 cm**, the red LED turned ON and the buzzer produced beeps with a rate proportional to proximity.
- The system accurately detected object distances and provided appropriate visual and audible alerts.

Conclusion:-

The experiment successfully demonstrated an **Obstacle Detection System** using Arduino, ultrasonic sensor, LEDs, and buzzer. The system was able to detect obstacles at varying distances and alert using visual and audio cues.

Questions for Oral:-

1. What is the principle of operation of an ultrasonic sensor?
2. Explain the working of the `tone()` and `noTone()` functions.
3. Why do we multiply the time by 0.01715 in the Arduino code?
4. What changes in the code would make the buzzer sound continuously for close obstacles?
5. Can ultrasonic sensors detect transparent glass? Why or why not?

EXPERIMENT NO. 4

Aim: - To design and implement an **automatic and manual wiper control system**

Objectives: -.

- To understand the working of a **wiper control system** in automobiles.
- To implement **manual and automatic wiper modes** using Arduino.
- To study how wiper speed can be varied according to rain intensity using sensor input.

Components:-

S. No.	Component	Specification / Part No.	Quantity
1	Arduino Uno	ATmega328P	1
2	DC Motor (Wiper Motor)	5V–12V	1
3	Potentiometer	10kΩ (Rain sensor)	1
4	Switches	Push button	3
5	LEDs	Red, Green	2
6	Breadboard	-	1
7	Jumper Wires	-	As req.
8	Power Supply/USB Cable	5V	1

Theory:-

A wiper system in vehicles ensures clear visibility for drivers during rain by removing water from the windshield. In modern vehicles, **automatic rain-sensing wiper systems** are used.

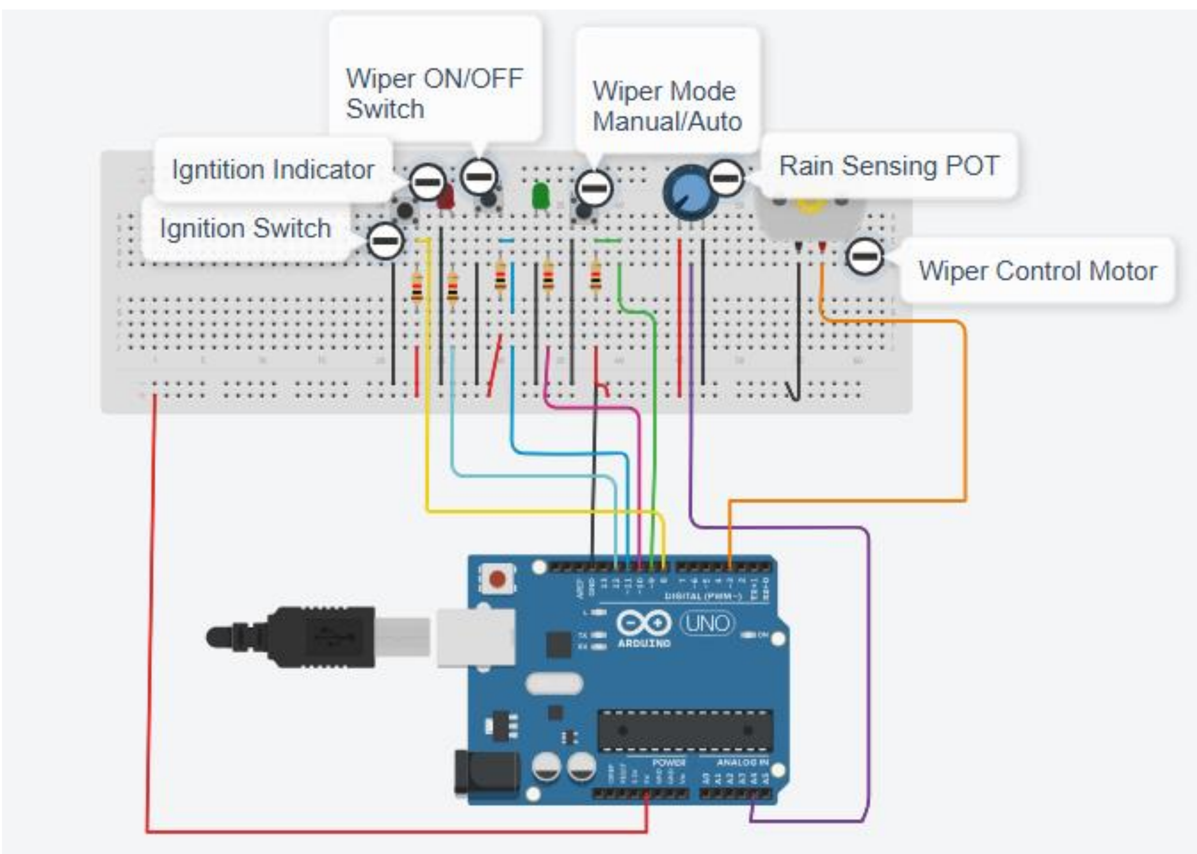
- **Manual Mode**: Driver controls the wiper speed using a switch.
- **Automatic Mode**: Wiper speed is adjusted automatically depending on rain intensity, detected by a rain sensor.

In this experiment, a **potentiometer is used as a rain sensor**, providing an analog input to the Arduino. The Arduino reads the sensor value and adjusts the **PWM duty cycle** of the DC motor to control wiper speed.

System operation:

- Ignition ON → Wiper system active.
- Ignition OFF → Wiper system inactive (no operation).
- Wiper switch ON → Motor runs according to mode selected.
- Mode Switch:
    - Auto → Speed controlled by rain sensor.
    - Manual → Motor runs at maximum speed.

Circuit Diagram :-



Connections & Procedure :-

1. Connect the **ignition switch, wiper switch, and mode select switch** to Arduino digital input pins.
2. Connect **RED Led** to indicate ignition status and **Green Led** to indicate wiper status.
3. Connect the **potentiometer** to an analog input pin (A4) of Arduino to act as rain sensor.
4. Connect the **DC motor** to PWM pin (D3) for speed control.
5. Write and upload the Arduino code to implement:
6. Ignition ON/OFF control.
7. Wiper ON/OFF based on switch.
8. Auto/Manual mode selection.
9. PWM-based motor speed control.
10. Observe the system behaviour for different switch positions and potentiometer values.

Observations :-

Sr. No.	Ignition Switch	Wiper Switch	Mode Select	Rain Sensor Reading	Motor Speed (PWM)	System Status
1	OFF	Any	Any	Any	0	System OFF
2	ON	OFF	Any	Any	0	Wiper OFF
3	ON	ON	Manual	Any	100% (255)	Wiper at full speed
4	ON	ON	Auto	Low	25% (64)	Wiper at low speed
5	ON	ON	Auto	Medium	50% (128)	Wiper at medium speed
6	ON	ON	Auto	High	75–100% (192–255)	Wiper at high/full speed

Result:-

- In manual mode, the motor ran at maximum speed.
- In automatic mode, the motor speed varied according to the potentiometer (rain sensor) input.
- The system functioned correctly for all combinations of ignition, wiper, and mode switches.

Conclusion:-

This experiment demonstrates the working of an **automatic rain-sensing wiper system** using Arduino. It provides a practical understanding of **switch interfacing, PWM motor control, and sensor-based automation** in automotive electronic.

Questions for Oral:-

1. What is the purpose of the ignition switch in this system?
2. How is the speed of the wiper motor controlled in auto mode?
3. What is the function of PWM in this experiment?
4. Differentiate between **manual mode** and **automatic mode** in wiper control.
5. Give two real-life advantages of automatic wiper systems in vehicles.

EXPERIMENT NO. 5

Aim: - To design and implement an **Automotive Alarm System**

Objectives: -.

- To study the working of an automotive security alarm system.
- To implement **door lock/unlock operations** using motors.
- To generate an **alarm (sound + flashing LEDs)** when unauthorized entry is detected.
- To understand the role of embedded systems in automotive security.

Components:-

S.No.	Component	Specification / Part No.	Quantity
1	Arduino Uno	ATmega328P	1
2	PIR Sensor	HC-SR501	1
3	Push Buttons	SPST	2
4	Slide Switches	Analog inputs (A0, A1)	2
5	LEDs	Red, Green, Yellow, Blue	6
6	Buzzer	5V DC	1
7	DC Motor	Door Lock/Unlock	2
8	Resistors	220Ω – 1kΩ	As req.
9	Breadboard & Jumper Wires	-	As req.

Theory:-

The **Automotive Alarm System** enhances vehicle security by detecting unauthorized entry or suspicious movement.

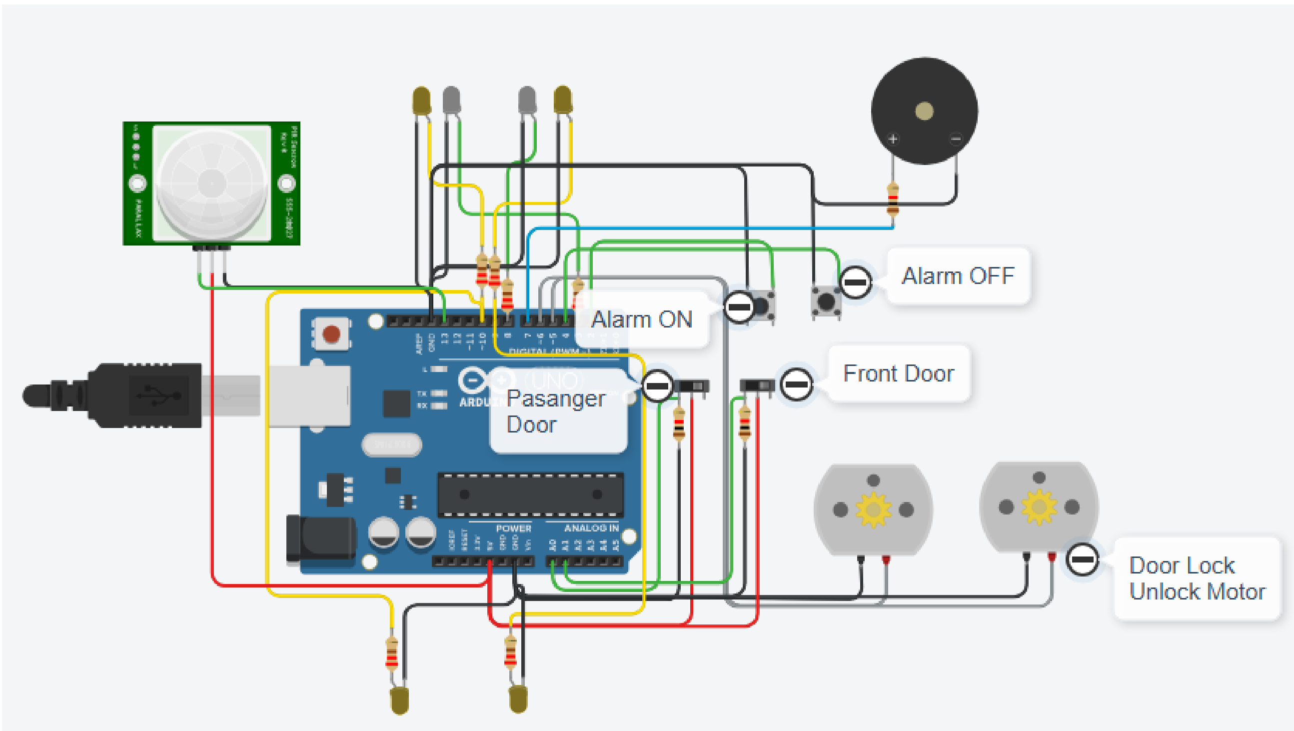
- **Inputs**:
  - **Door sensors** detect whether front and passenger doors are open or closed.
  - **PIR sensor** detects human movement near the car.
  - **Push Buttons** are used for **Alarm ON (Arm)** and **Alarm OFF (Disarm)**.
- **Outputs**:
  - **LEDs** simulate car lights flashing during alarm.
  - **Buzzer** generates a warning sound.
  - **DC Motors** simulate door lock/unlock mechanism.

Working Principle:

- When **Alarm ON** button is pressed → system is armed.
- If any door is opened or movement is detected → **Alarm triggers** (LEDs blink + buzzer sounds).
- When **Alarm OFF** button is pressed → system is disarmed and doors are locked/unlocked.
- If doors are opened while the alarm is OFF, headlights (LEDs) glow for a preset time.

This replicates real automotive anti-theft systems.

Circuit Diagram :-



Connections & Procedure :-

1. Connect **door sensors** to analog inputs A0 & A1.
2. Connect **PIR motion sensor** to digital pin 13.
3. Connect **push buttons** for Alarm ON (pin 2) and Alarm OFF (pin 4).
4. Connect **LEDs** to pins 3, 8, 9, and 10 to simulate headlights and indicators.
5. Connect **buzzer** to pin 7 for alarm sound.
6. Connect **DC motors** for door lock/unlock mechanism to pins 5 and 6.
7. Upload the Arduino code to implement the logic for:
    - Alarm ON/OFF,
    - Door lock/unlock,
    - Motion detection,
    - LED + buzzer alarm activation.
8. Observe the behaviour by pressing buttons, opening doors (simulated), and triggering motion sensor.

Observations :-

Sr. No.	Alarm Status	Door Status	PIR Sensor	Motor Action	LEDs	Buzzer	System Response
1	OFF	Closed	No Motion	Locked	OFF	OFF	System inactive
2	OFF	Open	No Motion	Locked	Headlights ON	OFF	Doors open → headlights ON
3	ON	Closed	No Motion	Locked	OFF	OFF	Alarm armed, no intrusion
4	ON	Open	No Motion	Locked	All LEDs Blinking	Beeping	Alarm triggered due to door open
5	ON	Closed	Motion	Locked	All LEDs Blinking	Beeping	Alarm triggered due to motion detection

Result:-

- The system could **lock/unlock doors** with motors.
- **LEDs and buzzer** successfully simulated alarm signals.
- The alarm was triggered correctly when doors opened or motion was detected while the system was armed.

Conclusion:-

The experiment demonstrates the working of an **automotive alarm system** integrating sensors, actuators, and Arduino. The system enhances vehicle security by providing both **visual and audible alerts** against unauthorized entry.

Questions for Oral:-

1) What is the function of the PIR sensor in this system?
2) How does the Arduino differentiate between alarm ON and OFF states?
3) Why are door lock/unlock motors used in automotive alarm systems?
4) What role do LEDs and buzzer play in the alarm system?
5) Suggest two improvements for making this alarm system more secure.

EXPERIMENT NO. 6

Aim: - **To design, simulate, and analyze an Adaptive Cruise Control (ACC) system using MATLAB Simulink**

Objectives: -.

- To understand the principle of cruise control and adaptive cruise control in vehicles.
- To model the vehicle dynamics using differential equations in Simulink.
- To implement a PID controller for speed regulation.
- To simulate the effect of road disturbances and target speed variations.

Software Requirements: -

- MATLAB & Simulink software
- Blocks used:
  - Constant source (reference speed)
  - Sum blocks (error calculation)
  - PID controller block
  - Gain blocks (k1/m and k2/m)
  - Integrator block (vehicle velocity)
  - Scenario signal (disturbance / road slope effect)
  - Scope block (output visualization)

Theory:-

Cruise control is an electronic system that automatically controls the throttle of a car to maintain a constant speed set by the driver.

- **Conventional Cruise Control** maintains a constant speed regardless of external factors.
- **Adaptive Cruise Control (ACC)** is an advanced version, where the system adapts to road conditions, slope, or varying demands by adjusting the throttle.

The vehicle dynamics are governed by:

$$\frac{dv}{dt} = \frac{k_1}{m}(input) - \frac{k_2}{m}v$$

where,

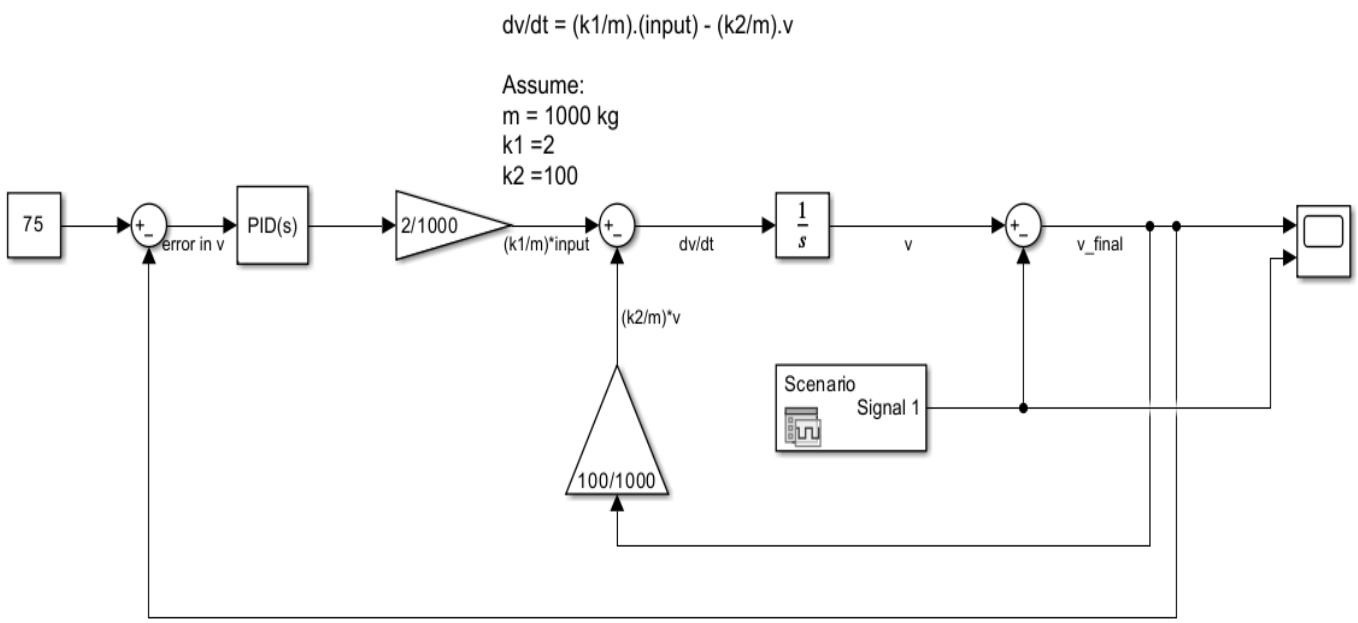
- m = vehicle mass (1000 kg)
- k_1 = throttle gain (2)
- k_2 = damping / drag coefficient (100)

The PID controller minimizes the error between reference speed and actual speed, thereby adjusting the throttle input dynamically.

Working Principle:

1. The reference speed (set speed = 75 km/h) is compared with the actual vehicle speed to obtain an error signal.
2. The **PID controller** processes this error and generates the required throttle control input.
3. The vehicle model uses throttle input to compute acceleration (dv/dt) and velocity (v).
4. A disturbance input (road scenario block) affects the velocity.
5. The feedback loop continuously adjusts throttle to maintain the desired speed.
6. The final output is visualized in the scope, showing how velocity adapts with disturbances.

Circuit Diagram :-



Expt: Adaptive Cruise Control System

Connections & Procedure:-

1. **Reference Speed Input**
   - A Constant block is used to set the desired speed (e.g., 75 km/h).
2. **Error Calculation**
   - The reference speed is compared with the actual speed using a Sum block to get error in velocity (verror).
3. **PID Controller**
   - The error signal is passed through the PID Controller block which generates throttle control input.
4. **Vehicle Dynamics Model**
   - The throttle signal is multiplied with k1/m to calculate acceleration input.

- Vehicle damping term is modeled using gain block k2/m and actual velocity.
- Both inputs are combined in a Sum block to calculate dv/dt.

5. **Integrator Block**
   - The acceleration (dv/dt) is integrated to get velocity (v).
6. **Disturbance Input**
   - A Scenario Signal block is added as an external disturbance (road slope or load change) and connected to the velocity.
7. **Final Output**
   - Velocity after adding disturbance is shown as v_final
   - A Scope block is connected to display speed response with respect to time.

Observations :-

- The yellow curve (vehicle speed, v_final) initially overshoots but stabilizes around the reference speed (75 km/h).
- The blue curve (disturbance signal) represents road conditions affecting the vehicle.
- The PID controller compensates for disturbances, ensuring minimal deviation in final velocity.

Result:-

The Adaptive Cruise Control system successfully maintains the desired speed of the vehicle despite disturbances. The PID controller ensures stability with reduced steady-state error and quick settling time.

Conclusion:-

- Adaptive Cruise Control (ACC) enhances conventional cruise control by adapting to disturbances.
- PID controller plays a crucial role in maintaining speed stability.
- Simulation results prove the effectiveness of control design for automotive applications.

Task for Students

- Modify PID parameters (Kp, Ki, Kd) and study the effect on system response.
- Simulate the system with varying vehicle mass (m) and drag coefficient (k2).
- Replace disturbance input with random noise and analyse system performance.
- Compare performance with and without PID controller.

Questions for Oral:-

1. Define cruise control and adaptive cruise control. How are they different?
2. Write the vehicle dynamics equation used in this experiment.
3. Explain the role of each parameter: mass (m), k1, and k2.
4. What is the significance of the PID controller in this system?
5. How do Kp, Ki, and Kd values affect overshoot, settling time, and steady-state error?
6. What would happen if feedback is removed from the system?

EXPERIMENT NO. 7

Aim: - Design and simulate a quarter-car suspension model in MATLAB/Simulink. Compare vehicle body displacement x(t) (output) with road profile z(t) (input) and study the effect of suspension stiffness and damping on ride response.

Objectives: -.

- Model the quarter-car as a 1-DOF sprung mass system and derive the equation of motion.
- Build the Simulink diagram implementing the model using integrators and algebraic blocks.
- Simulate with given parameters (m, b, c) and a sinusoidal road input; plot x(t) and z(t).
- Observe amplitude/phase difference, transient/steady response, and effect of parameter changes (b, c).
- Learn how to tune model blocks, solver and logging for accurate simulation.

Software Requirements: -

- MATLAB (R2018b or later)
  - Simulink
  - Signal Processing Toolbox (optional)
  - PC with 4 GB RAM or more

Theory:-

Quarter-car (sprung mass) model dynamics:
$m * \ddot{x} + b * \dot{x} + c * x = b * \dot{z} + c * z$

Transfer function (Laplace domain):
$X(s) / Z(s) = (b*s + c) / (m*s^2 + b*s + c)$

Natural frequency and damping ratio:
$\omega n = \sqrt{(c/m)}$
$\zeta = b / (2\sqrt{(m*c)})$

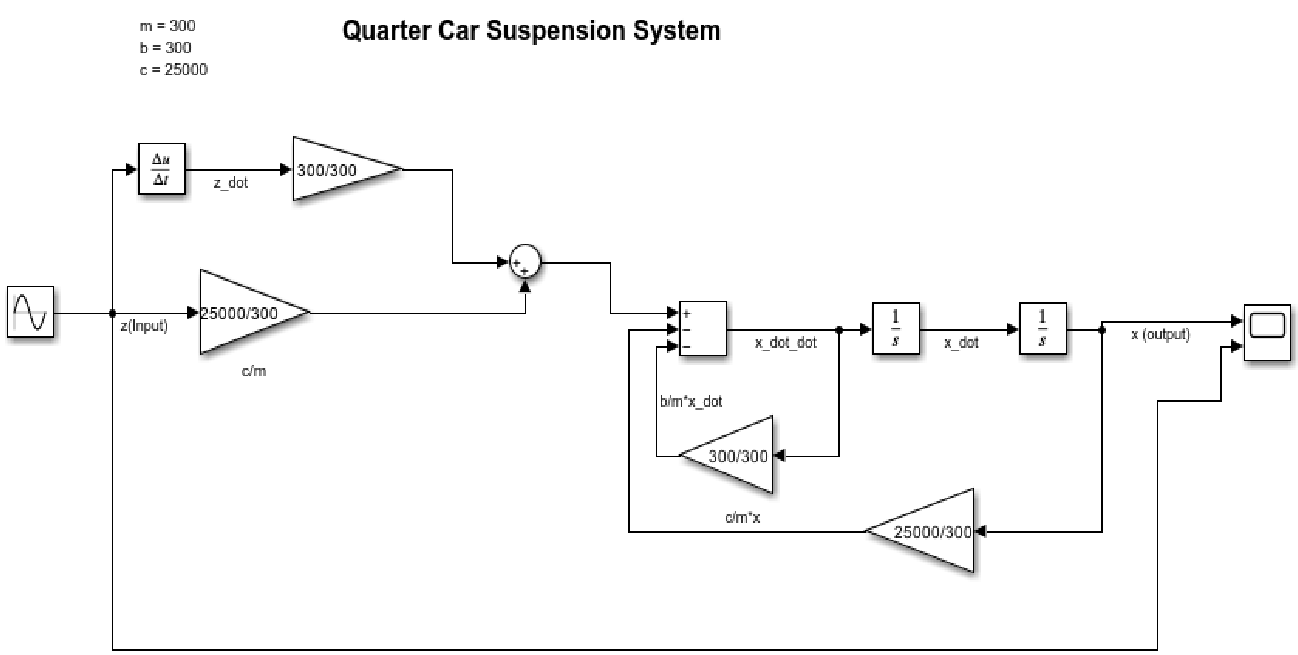
Given parameters:

m = 300 kg, b = 300 N·s/m, c = 25000 N/m

Working Principle:

- The system consists of a sprung mass supported by a spring and damper. The road displacement z(t) excites the system, producing a relative motion x(t).
- The spring exerts a restoring force proportional to displacement (c * (z - x)).
- The damper exerts a force proportional to velocity (b * (ż - ẋ)).
- The mass responds according to the net force, giving the second-order differential equation above.
- In Simulink, these physical relationships are implemented using gain, sum, and integrator blocks.

Circuit Diagram :-



Connections & Procedure:-

Blocks Used:
- Sine Wave (z input)
- Derivative ($\dot{z}$)
- Gain blocks for (b/m) and (c/m)
- Sum blocks for force balance
- Two Integrators (for $\dot{x}$ and x)
- Scope for visualization

Gains:
b/m = 300/300 = 1
c/m = 25000/300 ≈ 83.33

Functional Flow:

1. Input z(t) → Derivative → multiplied by b/m → $\dot{z}$ term.
2. z(t) multiplied by c/m → spring force term.
3. Feedback x and $\dot{x}$ (multiplied by negative b/m and c/m) → combined in summing block.
4. The resulting signal is $\ddot{x}$ (acceleration).
5. Integrate twice to obtain velocity ($\dot{x}$) and displacement (x).
6. Display z(t) and x(t) on Scope.

Procedure

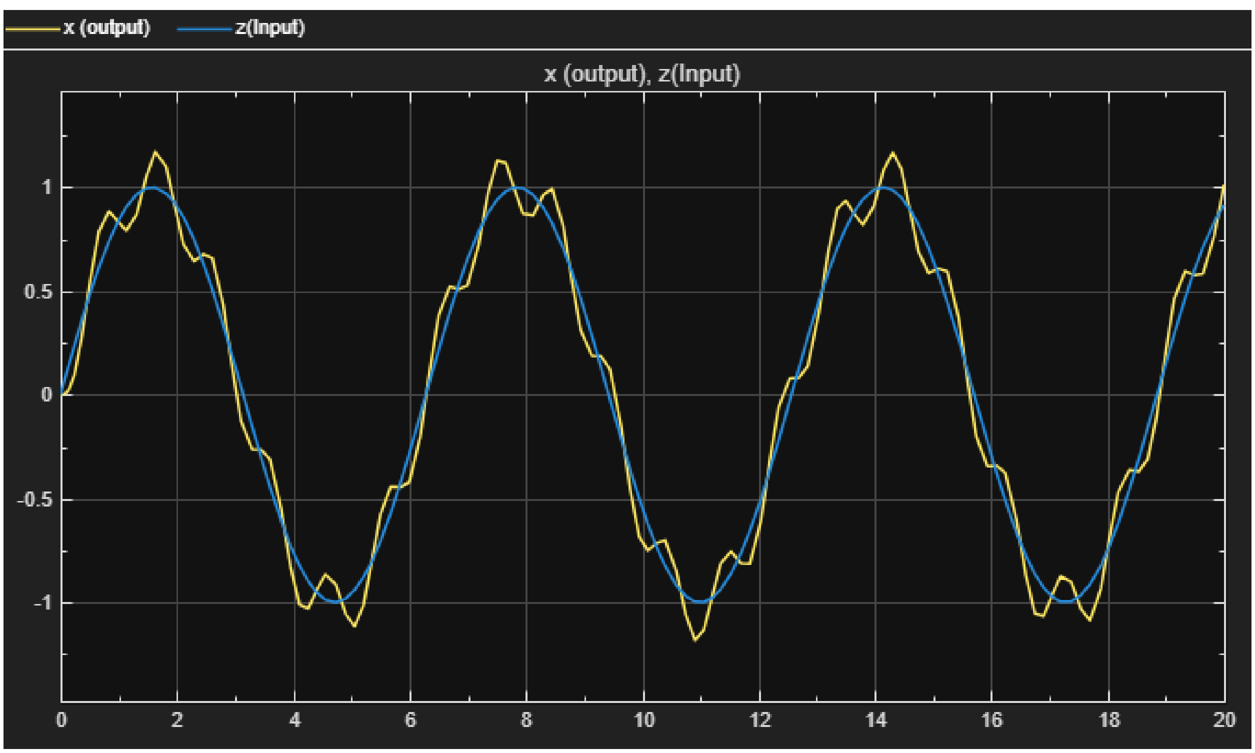
1. Open Simulink and create a new model.
2. Add a Sine Wave block and set amplitude = 1, frequency = 1 Hz.
3. Add Derivative block for $\dot{z}$.
4. Insert Gain blocks with b/m and c/m values.
5. Use a Sum block to add all forces: $+b/m*\dot{z} + c/m*z - b/m*\dot{x} - c/m*x$.
6. Connect output to two Integrators in series.
7. Feedback x and $\dot{x}$ through respective gains to the Sum block.
8. Use a Scope block to display both z and x.
9. Set solver to ode45, Stop time = 20 s.
10. Run the simulation and observe waveforms.

Observations :-

- The vehicle body displacement x(t) follows the road input z(t) with a small delay (phase lag).
- When damping is low, oscillations persist longer after excitation.
- Increasing b reduces oscillations and overshoot.
- Increasing c makes the system stiffer and increases the natural frequency.
- Near ωn (≈ 1.45 Hz), resonance may cause amplified oscillations.

Result:-

1. Natural frequency (ωn) = 9.13 rad/s (1.45 Hz).
2. Damping ratio (ζ) = 0.0548 (lightly damped).
3. The output x(t) lags behind z(t) slightly, showing spring-damper dynamics.
4. The system shows good correlation between theory and simulation plots.



Conclusion:-

- The quarter-car suspension model successfully simulates the dynamic response of a vehicle suspension system.
- Body displacement x(t) depends on the damping coefficient and spring stiffness.
- Proper damping ensures a smooth ride without excessive oscillations.
- Increasing stiffness raises natural frequency, making the ride less comfortable but more stable.

Task for Students

1. Run the simulation with b = 100, 300, 1000 and note changes in x(t).
2. Change stiffness c = 15000, 25000, 40000 and record effects on oscillation frequency.
3. Replace the Derivative block with a filtered derivative (transfer function method).

Questions for Oral:-

1. Write the governing equation for the quarter-car suspension system.
2. Define natural frequency and damping ratio.
3. How does damping affect the ride comfort?
4. What are the functions of the spring and damper?
5. What is the difference between z(t) and x(t)?

EXPERIMENT NO. 8

Aim: - To build a Simulink model that transmits and receives CAN messages using Vehicle Network Toolbox blocks, and to analyze correct message packing, transmission, reception, and unpacking.

Objectives: -.

- Understand how to use CAN Configuration, CAN Pack, CAN Transmit, CAN Receive, and CAN Unpack blocks in Simulink.
- Build the transmit side model and then the receive side model using virtual CAN channels.
- Simulate message flow from transmit to receive, unpack the data, and visualize it.
- Demonstrate message integrity and timing consistency over CAN.

Software Requirements: -

- MATLAB & Simulink (with Vehicle Network Toolbox)
- Vehicle Network Toolbox license
- Virtual CAN channel support (MathWorks Virtual CAN)
- PC capable of MATLAB/Simulink

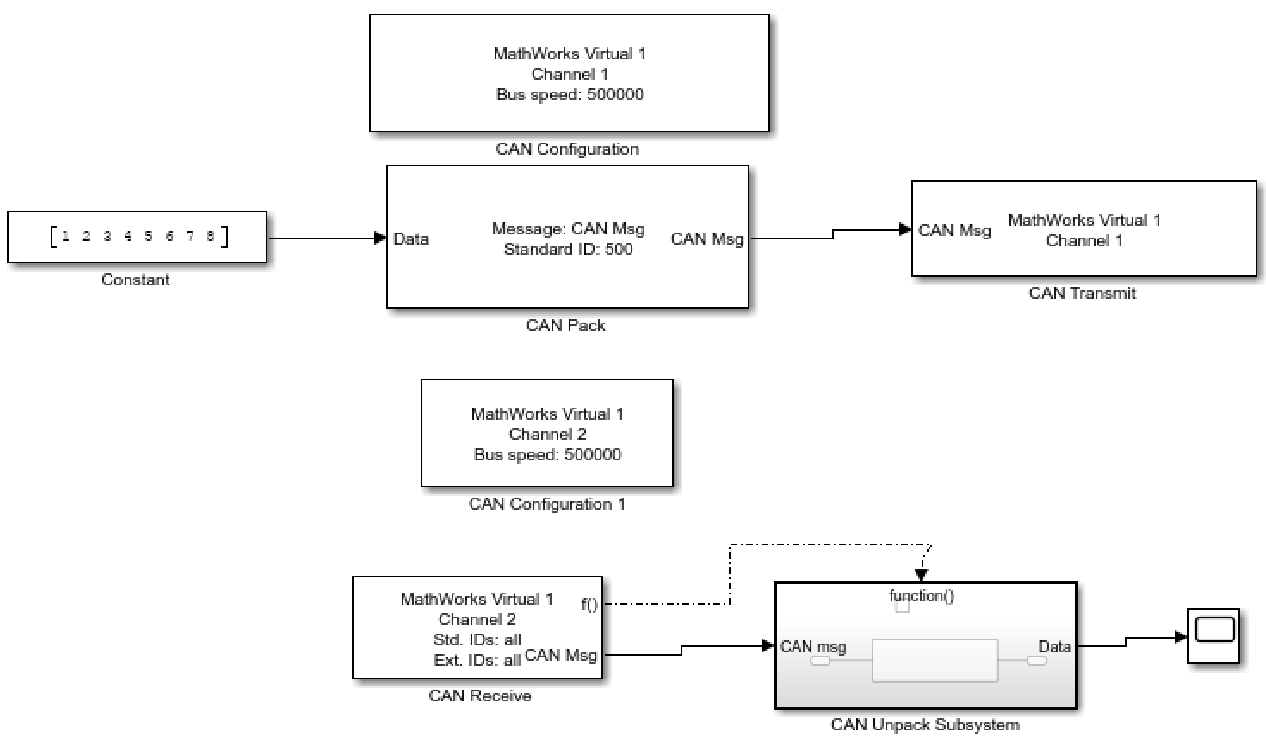
Theory:-

- **Controller Area Network (CAN):** robust vehicle network protocol for message-based communication.
- **Message format:** IDs (standard 11-bit or extended 29-bit), data bytes (0–8), frame control bits and CRC.
- **Packing/unpacking:** CAN Pack assembles data into a CAN message structure. CAN Unpack (inside a subsystem) extracts raw data from the message.
- **Virtual CAN channel:** MathWorks provides a software-only CAN interface (Virtual CAN) for simulation without hardware.
- **Timing and sample times:** Messages are transmitted at discrete intervals − sample times must be consistent between transmit and receive paths.

Working Principle:

- **Transmit side:**
  - A data source (e.g. Constant block) provides raw data (e.g. an array of 8 bytes).
  - CAN Pack block packages this raw data into a CAN message struct (with ID, length, data payload).
  - CAN Transmit block sends the message on a specified CAN channel (virtual).
  - A CAN Configuration block configures the channel (bus speed, device, acknowledge mode).
- **Receive side:**
  - A separate CAN Configuration block is used for receive channel.
  - CAN Receive block monitors the CAN channel and outputs received message struct and triggers.
  - A Function-Call Subsystem containing a CAN Unpack block extracts the raw data from the message struct.
  - This unpacked data is sent to a Scope or display block for visualization.
- The virtual CAN channel ensures messages sent by the transmit block become available to the receive block

Circuit Diagram :-



Model Connections & Blocks:

Transmit Part Blocks

- CAN Configuration
- Constant (raw data source)
- CAN Pack
- CAN Transmit

Transmit Part Connections & Parameter Settings

1. Connect Constant → CAN Pack → CAN Transmit.
2. Configure CAN Configuration block:
    - Device: MathWorks Virtual 1 (Channel 1)
    - Bus speed: 500000
    - Acknowledge mode: Normal
3. Configure CAN Pack:
    - Data source: raw data
    - Identifier type: Standard (11-bit)
    - Identifier: 500
    - Length: 8 bytes
4. Configure Constant block:
    - Value: [1 2 3 4 5 6 7 8]
    - Sample time: 0.01 s
    - Output data type: uint8

Receive Part Blocks

- CAN Configuration (for receive)
- CAN Receive
- Function-Call Subsystem containing CAN Unpack
- Scope

Receive Part Connections & Parameter Settings

1. Connect CAN Receive output to the input of the Function-Call Subsystem.
2. Inside the subsystem, place CAN Unpack which outputs raw data.
3. Connect output of subsystem to Scope.
4. Configure receive-side CAN Configuration:
    - Device: MathWorks Virtual 1 (Channel 2)

- Bus speed: 500000
- Acknowledge mode: Normal

5. Configure CAN Receive block:
    - Device: Virtual 1 (Channel 2)
    - Sample time: 0.01
    - Number of messages per time step: all
6. Configure CAN Unpack:
    - Data output: raw data
    - Identifier type: Standard (11-bit)
    - Identifier: 500
    - Data length: 8 bytes

Final Integration & Simulation

- Save the model.
- In Configuration Parameters, set solver to discrete (or fixed-step) so message timing is discrete.
- Run simulation.
- Open Scope to view received data over time.
- The transmitted data `[1 2 3 4 5 6 7 8]` at each 0.01 s should appear in the Scope as unpacked output.

Procedure

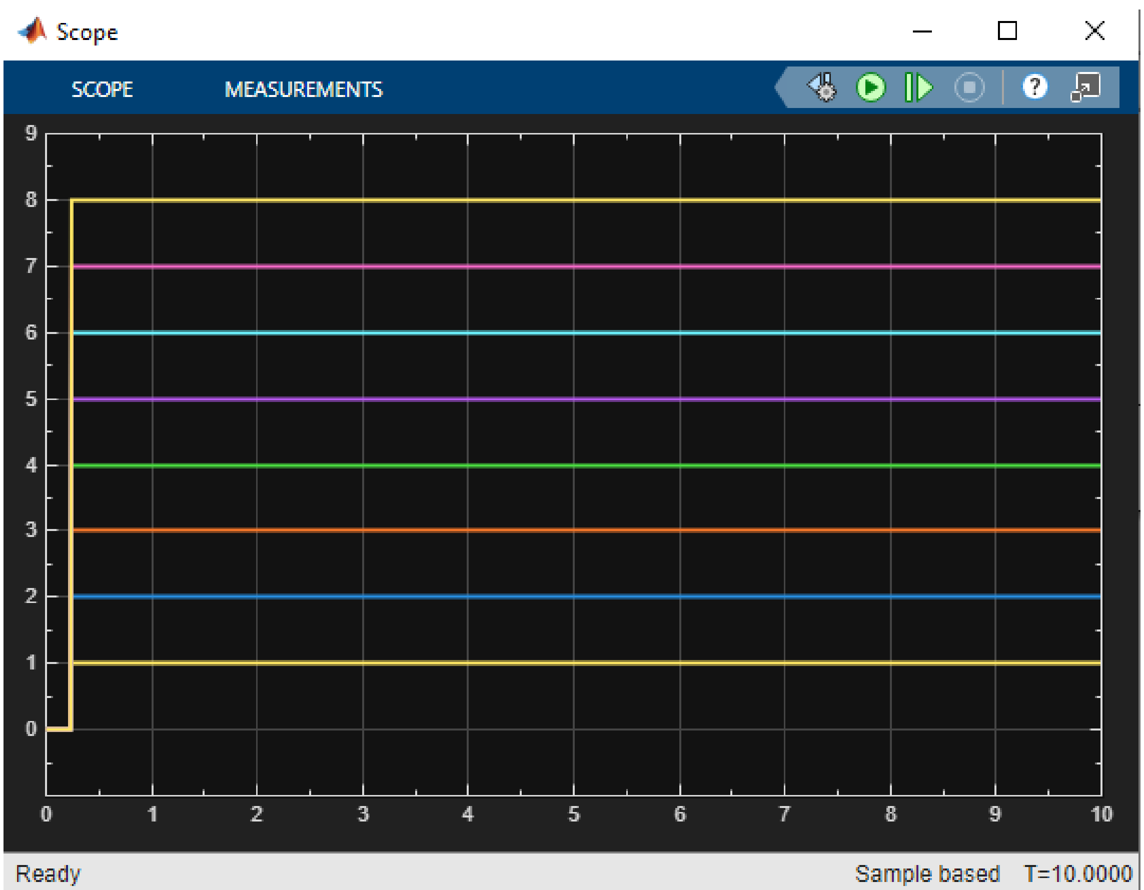
- Start Simulink → Blank Model → save with an appropriate name.
- Open Library Browser → navigate to Vehicle Network Toolbox → CAN Communication.
- Drag CAN Configuration, CAN Pack, CAN Transmit into the model.
- Drag a Constant block from Simulink common blocks.
- Wire Constant → CAN Pack → CAN Transmit.
- Add a CAN Configuration block (for transmit) (note: this block does not connect to others but configures channel).
- Configure all transmit-side blocks as per parameter values above.
- Now add receive-side: CAN Configuration (for receive), CAN Receive, Function-Call Subsystem, CAN Unpack, Scope.
- Wire CAN Receive → Function-Call Subsystem → Scope.
- Configure receive-side blocks as per parameter values.
- Set simulation to discrete solver (or fixed-step) in Model Settings.

- Save the model again.
- Run simulation.
- Open Scope to view data. Use autoscale if needed.
- Verify that the transmitted data appears correctly in the receive side.

Observations :-

- At each time step, the transmitted array `[1 2 3 4 5 6 7 8]` should appear in the Scope as received data.
- There should be no data corruption (same values).
- Timing should match—data should update every 0.01 s.

Result:-



- Successful transmission and reception of raw data via CAN model.
- Plots showing correct data over simulation time.
- Confirmation that CAN Pack and CAN Unpack blocks function as expected.
- Verification that message parameters (ID, length, sample time) are correctly applied.

Conclusion:-

- The CAN communication model in Simulink using Vehicle Network Toolbox is working.
- Pack, transmit, receive, and unpack steps are correctly implemented.
- Virtual CAN allows simulation without hardware.
- This experiment demonstrates the basics of message-based communication useful in automotive embedded systems.
- Students gain insight into message configuration, timing, and data integrity in CAN

Task for Students

1. Change the identifier from 500 to another value (e.g., 300) and verify that only matching receive unpack block receives.
2. Use multiple transmitters with different IDs and filter in receiver side to process only selected IDs.
3. Alter data values over time (e.g., using a Signal Generator or Ramp block instead of constant).

Questions for Oral:-

1. What is the role of the CAN Pack block?
2. How does CAN Unpack differ from Pack?
3. Explain standard vs extended identifier in CAN.
4. What is the point of a virtual CAN channel?
5. Why use discrete solver or fixed-step in this CAN model?
6. What happens if transmit and receive sample times mismatch?