**EXPERIMENT 10**

**TITLE:** **ADAPTIVE CRUISE CONTROL**

**AIM:** Conventional cruise control systems lack the capability to adapt to varying traffic conditions, leading to reduced safety and driver convenience. This project aims to design an Adaptive Cruise Control (ACC) system that automatically adjusts a vehicle’s speed based on the distance to the vehicle ahead, enhancing both driving comfort and road safety.

**LIST OF SOFTWARE COMPONENTS:**

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Software Block** | **Specification / Function** |
| 1 | Ultrasonic Sensor Block | for simulating distance measurement to the vehicle ahead. |
| 2 | **PID Controller Block** | for controlling throttle and brake based on input from the sensor. |
| 3 | Vehicle Dynamics Block | to simulate the motion of the vehicle in response to throttle and braking. |
| 4 | Speed Setpoint Block | for setting the desired cruise speed. |
| 5 | **Throttle Actuator Blocks** | to model acceleration and deceleration behaviour. |
| 6 | Scope Block | to visualize simulation results like speed, distance, etc. |

**LIST OF HARDWARE COMPONENTS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL NO.** | **Components** | **Quantity** | **Description** |
| 1. | Arduino IDE | 1 | Used for writing, compiling, and uploading |
| 2. | Motor driver | 1 | |  | | --- | | Handles PWM signals for controlling the **DC Motor** via the motor driver |  |  | | --- | |  | |
| 5. | Ultrasonic Sensor | few | For interfacing with the **Ultrasonic Sensor** to measure the distance to the vehicle ahead |
| 6. | Potentiometer | 1 | Reads analog values from the **Potentiometer** to simulate speed control or set a reference speed. |
| 7 | DC motor | 1 | Used for the rotation of the wheels |

**THEROTICAL BACKGROUND :**

Adaptive Cruise Control (ACC) is an advanced driver-assistance technology designed to enhance conventional cruise control systems by enabling vehicles to automatically adjust their speed based on traffic conditions. Unlike traditional cruise control, which maintains a constant speed set by the driver, ACC dynamically modulates the throttle and braking systems to keep a safe distance from the vehicle ahead.

The core working principle of ACC relies on real-time environment sensing, most commonly using ultrasonic sensors, radar, lidar, or cameras. These sensors measure the distance and relative speed of preceding vehicles. Based on this information, the system uses embedded control algorithms to adjust vehicle speed, either accelerating to the set speed or decelerating when the gap to the leading vehicle decreases below a safe threshold.

In embedded systems, such as the one implemented in this project, components like **ultrasonic sensors**, **DC motors**, **motor drivers**, and **microcontrollers** (e.g., ESP32 or Arduino) are integrated. The microcontroller processes distance data from the ultrasonic sensor and accordingly sends PWM signals to the motor driver to control the speed of the DC motor, simulating vehicle movement. A **potentiometer** may be used to represent the driver’s desired set speed.

The system's logic is coded using **Arduino IDE**, where sensor readings are continuously monitored and control decisions are executed in real time. This forms a basic but effective simulation of an ACC system, demonstrating how electronic components and software algorithms work together to enhance driving safety and comfort.

**SIMULINK MODEL:**

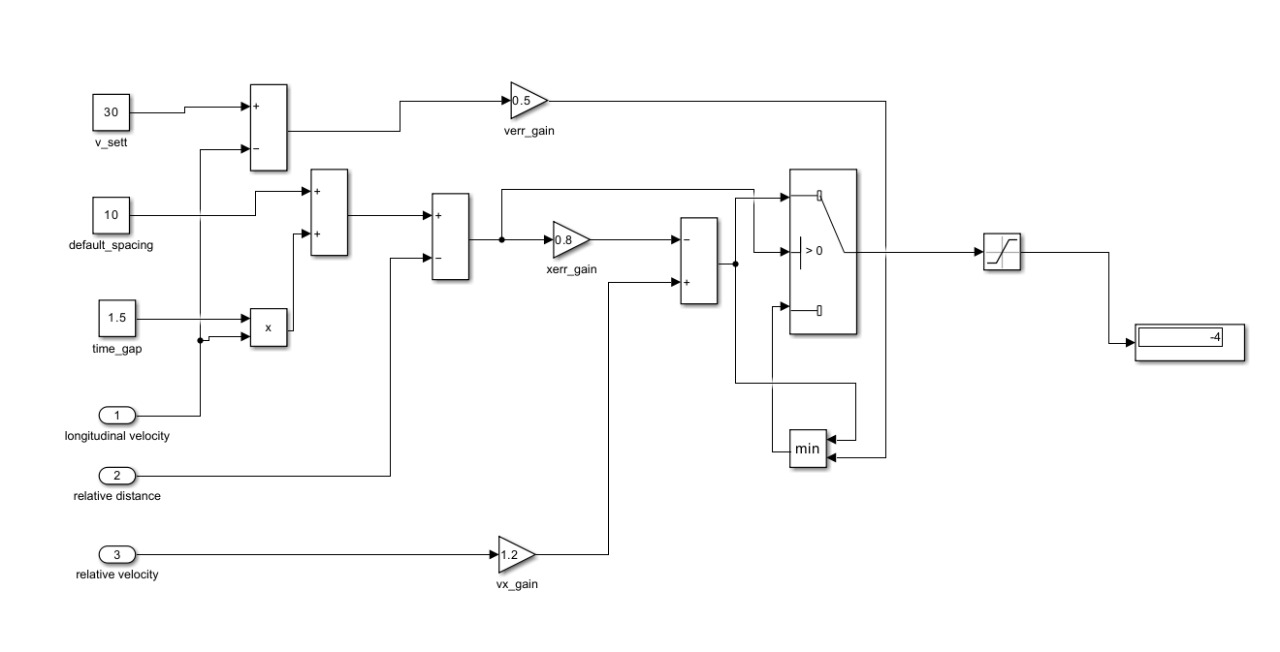


Figure: Simulink model

**Simulation Details:**

 The Adaptive Cruise Control system was **simulated to test automatic speed regulation** based on the distance to the vehicle ahead.

 An **ultrasonic sensor** was used to detect the distance between vehicles in the simulation setup.

 A **DC motor** represented the vehicle's motion, with speed adjusted dynamically.

 A **potentiometer** simulated the set cruise speed input by the driver.

 **Arduino IDE** was used to program and control the hardware logic for the simulation.

 The system **slowed down or stopped** the motor when an object (vehicle) was too close, maintaining a safe following distance.

 Simulation demonstrated **real-time response** to changing distances, validating the ACC concept.

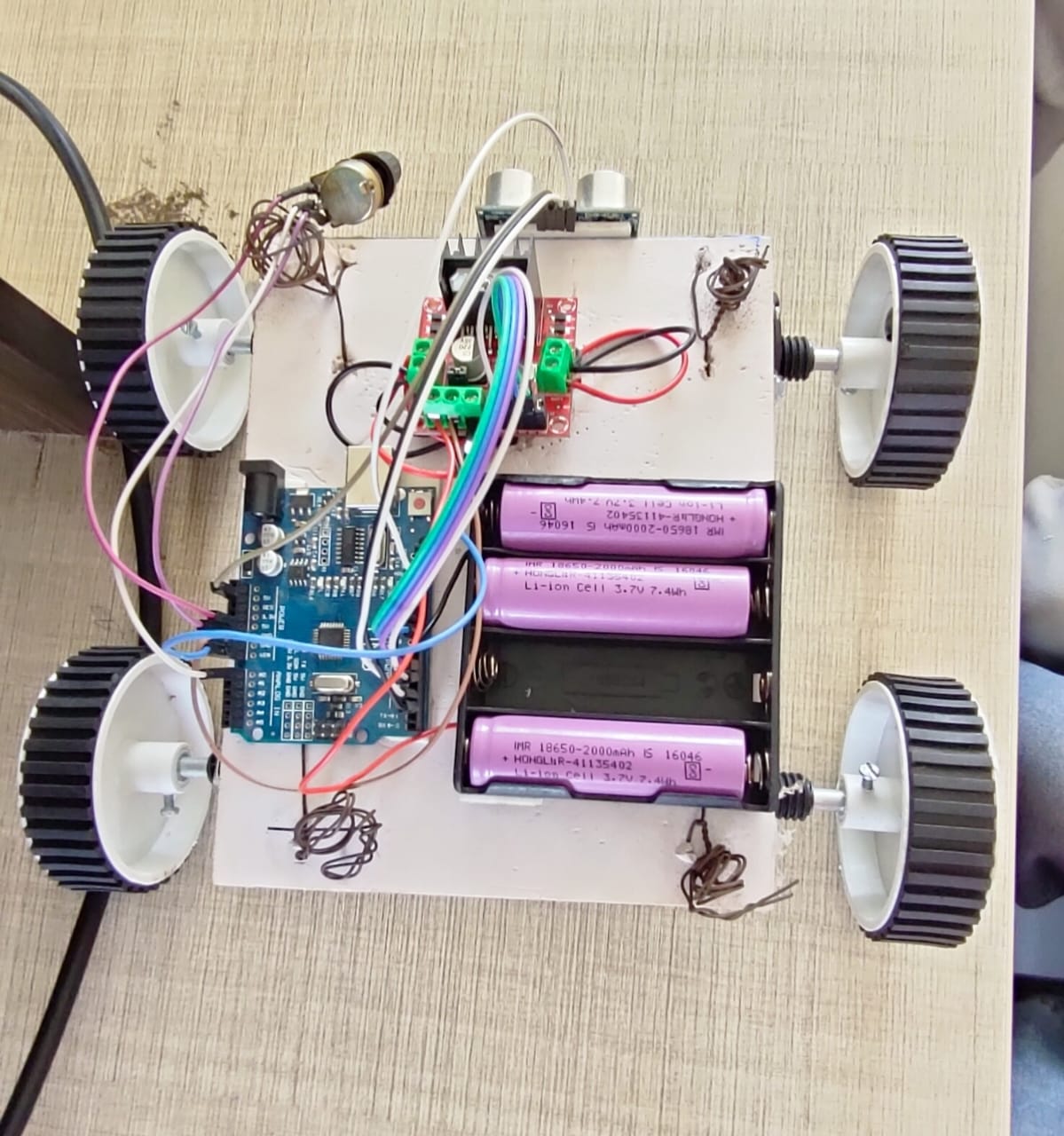


Figure: Hardware Implementation of Adaptive Cruise Control

**Hardware Details:**

* **Arduino UNO** – Main controller for processing and control.
* **Ultrasonic Sensor (HC-SR04)** – Measures distance to the object ahead.
* **L298N Motor Driver** – Controls motor speed and direction.
* **DC Motors (2x)** – Drive the wheels based on control signals.
* **Potentiometer** – Sets desired speed (cruise control input).
* **Li-ion Batteries (3x 3.7V)** – Power supply for motors and Arduino.
* **Chassis & Wheels** – Base frame for mounting components and movement.
* **Jumper Wires** – Used for all interconnections.

**CODE:**

// Motor Pins

const int MR1 = 5; // Right Motor Forward (PWM)

const int MR2 = 6; // Right Motor Backward

const int ML1 = 9; // Left Motor Backward

const int ML2 = 10; // Left Motor Forward (PWM)

// Potentiometer and Ultrasonic Pins

const int potPin = A0;

const int trigPin = 2;

const int echoPin = 3;

long duration;

int distance;

int baseSpeed = 0;

int adjustedSpeed = 0;

void setup() {

Serial.begin(9600);

pinMode(MR1, OUTPUT);

pinMode(MR2, OUTPUT);

pinMode(ML1, OUTPUT);

pinMode(ML2, OUTPUT);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

Stop();

}

void loop() {

// Read potentiometer and map to base speed

int potValue = analogRead(potPin); // 0–1023

baseSpeed = map(potValue, 0, 1023, 0, 255); // 0–255

// Measure distance using ultrasonic

distance = getDistance();

// Adjust speed relative to how close the object is

if (distance <= 15) {

adjustedSpeed = 0; // Stop immediately if object is very close

} else if (distance > 15 && distance < 100) {

// Linearly scale from 0.1x to 1.0x based on distance

float factor = (distance - 15) / 85.0; // range from 0 to 1

adjustedSpeed = baseSpeed \* factor;

} else {

adjustedSpeed = baseSpeed; // Object far enough, go at full speed

}

// Debug info

Serial.print("Distance: ");

Serial.print(distance);

Serial.print(" cm | Base Speed: ");

Serial.print(baseSpeed);

Serial.print(" | Adjusted Speed: ");

Serial.println(adjustedSpeed);

// Drive forward with adjusted speed

analogWrite(MR1, adjustedSpeed);

digitalWrite(MR2, LOW);

analogWrite(ML2, adjustedSpeed);

digitalWrite(ML1, LOW);

delay(100);

}

int getDistance() {

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

long duration = pulseIn(echoPin, HIGH, 20000); // timeout to avoid hang

return duration \* 0.034 / 2; // in cm

}

void Stop() {

digitalWrite(MR1, LOW);

digitalWrite(MR2, LOW);

digitalWrite(ML1, LOW);

digitalWrite(ML2, LOW);

}

**CONCLUSION:**

The Adaptive Cruise Control system enhances driving safety and comfort by intelligently adjusting vehicle speed based on traffic conditions. By integrating sensor technology with automated control, it reduces driver workload and contributes to the advancement of semi-autonomous driving systems. It not only improves road safety by maintaining a safe following distance but also optimizes fuel efficiency through smoother acceleration and braking. The successful implementation of ACC marks a significant step toward fully autonomous vehicle technologies in the future.

|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
|  |
|  |  |

|  |
| --- |
|  |
|  |  |

|  |
| --- |
|  |
|  |  |