**Computational Methods and Modelling for Engineering Applications**

**(GENG 8030 -1-R-2024W)**

# Final project report on Adaptive Cruise Control System

**by**

**Team 1**


| Sl. No | Name | Student ID | Department name |
|---|---|---|---|
| 1 | Badrinadh Ande | 110143113 | Electrical and Computer Engineering |
| 2 | Shoaib Akthar Shaik | 110126210 | Electrical and Computer Engineering |
| 3 | Venkata Sai Teja Pavuluri | 110132353 | Mechanical Engineering |


Submitted on:

**22 March 2024**

Submitted to:

**Professor: Yasser Alginahi**

## Table of contents



## List of figures and tables

Badrinadh Ande
Venkata Sai Teja Pavuluri
Shoaib Akthar Shaik

# 1. Introduction

Adaptive Cruise Control (ACC) is a sophisticated driver assistance system that automatically adjusts a vehicle's speed to maintain a safe following distance from the vehicle ahead [1]. It is an extension of conventional cruise control systems, which allow drivers to maintain a constant speed for their vehicle. However, ACC adds the ability to monitor the speed and distance of vehicles in front using ultrasonic sensors and then automatically adjusts the vehicle's speed accordingly. It's functionality, when activated, the system uses sensors to detect vehicles ahead and constantly adjusts the vehicle's speed to keep a safe distance. It can automatically accelerate, decelerate, or even brake the vehicle to maintain a safe following distance. ACC systems typically provide a variety of operating modes, including short-range, medium-range, and long-range modes, depending on the desired following distance. Some systems allow the driver to adjust the following distance manually.

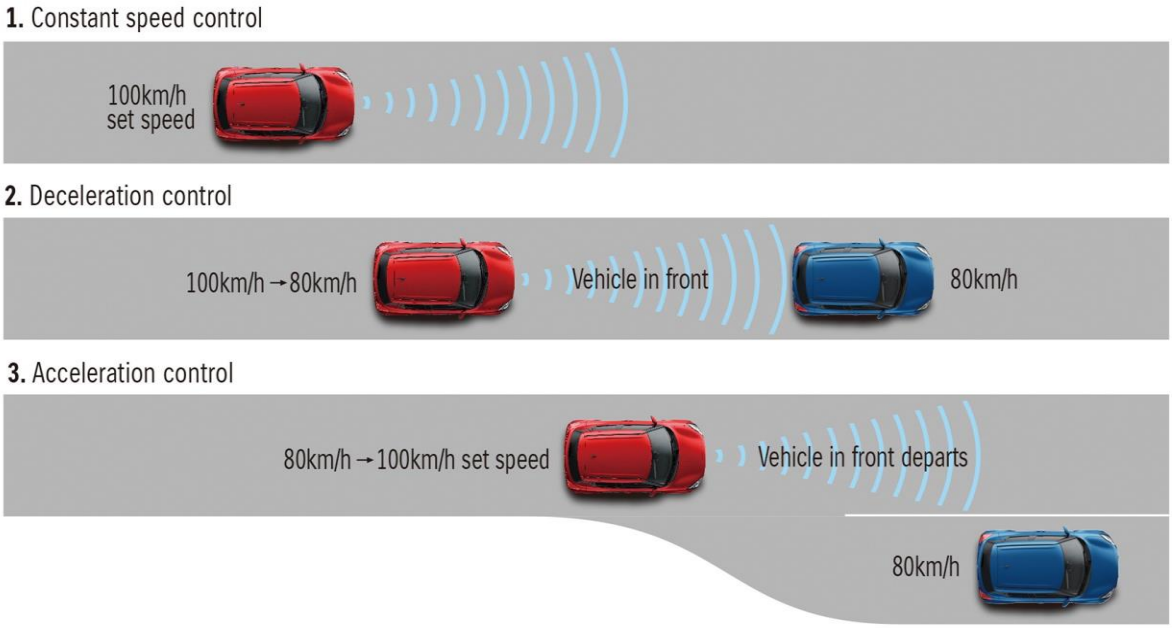



Fig 1: Illustration of Adaptive cruise control [2].

The outcomes of an Adaptive Cruise Control (ACC) system project using MATLAB typically include the creation of a functional ACC system capable of improving road

safety and traffic flow. This includes developing a reliable control algorithm, integrating sensors, designing a user-friendly interface, and validating the system through real-world testing.

Overall, adaptive cruise control represents a significant advancement in automotive technology, providing drivers with both safety and convenience benefits while also contributing to the evolution of autonomous driving systems. As automotive manufacturers continue to refine and improve ACC technology, it is expected to become more prevalent in modern vehicles, improving road safety and driving experiences.

## 2. Project Objective

The project aims to design, simulate, and optimize an Adaptive Cruise Control (ACC) system using MATLAB to enhance vehicle safety and comfort by automatically adjusting the vehicle's speed to maintain a safe distance from the preceding vehicle. By using Arduino Uno microcontroller, Push buttons, an Ultra sonic sensor, an LCD Display, and MATLAB for programming.

## 3. Project logic design and flow chart

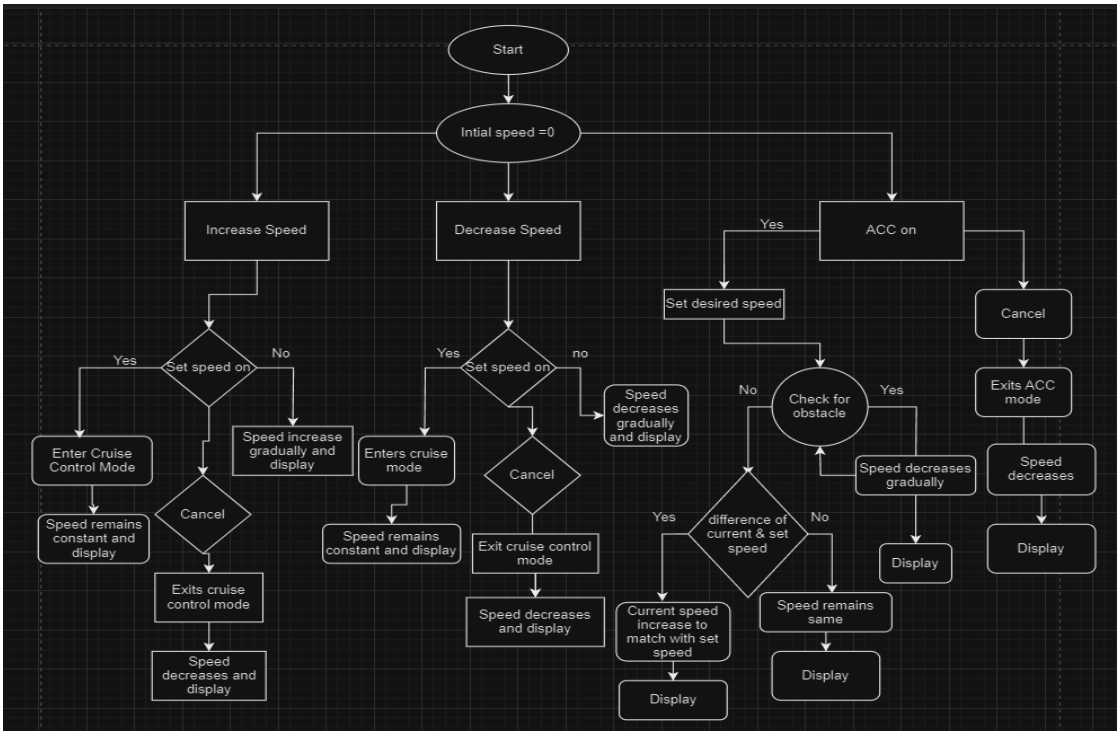
According to the given project description, the following logical tasks were considered to create the logical flow chart as shown in the below Fig 2.

### Logical tasks:

- Display a Welcome message, Project name, Team number, Project member’s name along with registration numbers respectively.
- Use the Increase_speed and Decrease_speed buttons to control the speed. However, ensure that the speed remains constant only when the Set_speed button is pressed; otherwise, it will change gradually over time.
- To activate cruise control mode, press the Set_speed button. The speed will remain constant. Allow the Set_speed to be adjusted using the Increase_speed and Decrease_speed buttons while in this mode. If you press the Cancel button, you will exit cruise control and gradually reduce your speed.

- Press the Adaptive_speed button to set and maintain a constant speed until a vehicle or obstacle is detected ahead. When an obstacle is detected, the speed is automatically reduced, and it is restored to the set speed once the road is clear. This mode is distinguished from cruise control mode by making the display blink. Disable the Increase_speed and Decrease_speed buttons while in adaptive cruise control mode, but keep the Cancel button functional. If the Cancel button is pressed, the blinking display will stop and the vehicle's speed will gradually decrease.

## Flow Chart:



## Block Diagram

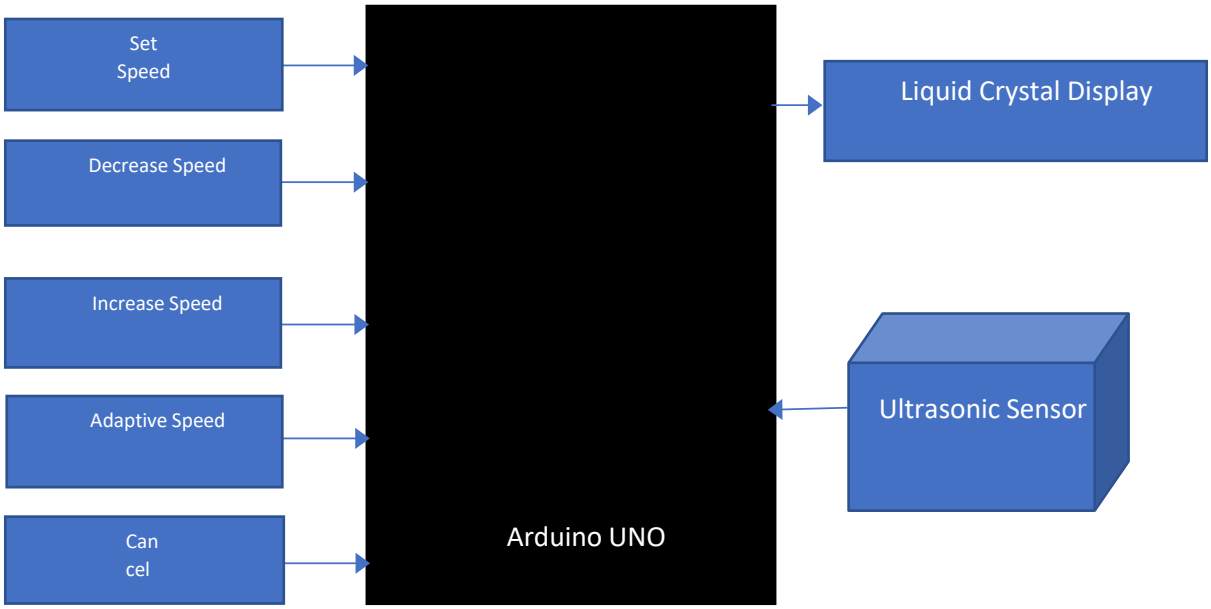



Fig 2: Flow chart and Block Diagram

## 4. Pseudo Code

- Initialize Arduino connection and components: ultrasonic sensor, LCD display.
- Display welcome message and project details on the LCD.
- Initialize variables:

  - incrementButton, decrementButton, cancelButton, setSpeedButton, adaptiveCruiseButton (for button states)

  - counter (for control mode)

  - speed (current speed)

  - blink (for mode indication)

  - distance (from ultrasonic sensor)

  - speedLimit (maximum speed limit)
- Enter main control loop:

  while true:
- Read button states and distance from ultrasonic sensor.
- Control speed based on buttons and control mode:

  - If counter == 0: (Default mode)

  - If incrementButton pressed, increase speed.

  - If decrementButton pressed, decrease speed.

  - Else, decrease speed by default.

  - If counter == 1: (Set speed mode)

- Maintain current speed.
- Allow speed adjustment with buttons.

- If counter == 2: (Adaptive cruise control mode)
- Adjust speed based on obstacle distance:
- Slow down if obstacle detected.
- Speed up if no obstacle.
- Cap speed at maximum limit.

- Control mode selection based on button presses:
  - Enter set speed mode if setSpeedButton pressed.
  - Adjust speed if already in set speed mode.
  - Activate adaptive cruise control mode if adaptiveCruiseButton pressed.
  - Exit any active mode if cancelButton pressed.
- Ensure speed is non-negative.
- Update LCD display based on control mode:
  - If counter == 2 (Adaptive cruise control mode):
  - Display "ACC MODE" and current speed.

  - If counter == 1 (Set speed mode):
  - Display "CRUISE MODE" and current speed.

  - Otherwise:
  - Display "NORMAL MODE" and current speed.
- End of loop.

## 5. Project Development

- **Circuit diagram:** To visualize the project objective, a circuit diagram, as shown in Fig 3 below, was created on the TechieYan Technologies website, incorporating all of the required hardware components.

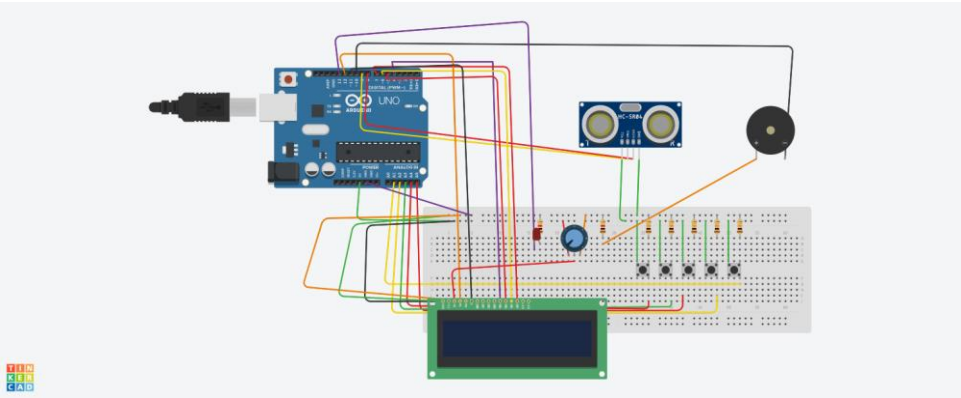


Fig 3: Adaptive cruise control circuit diagram

- **Components:** After designing the circuit, an analysis was performed to identify and obtain the necessary hardware components for the project.

| Sl. No | Component |
|---|---|
| 1 | Arduino Uno R3 |
| 2 | Jumper Cables |
| 3 | LCD display |
| 4 | Breadboard |
| 5 | Ultrasonic sensor |
| 6 | Push buttons |
| 7 | Resistors |
| 8 | Potentiometer |
| 9 | LED light |
| 10 | Active buzzer |

Table 1: Components

- **Hardware Components:**

a. **Arduino Uno R3:** The Arduino Uno R3 is a microcontroller board that includes an ATmega328P microcontroller running at 16 MHz. It has 14 digital input/output pins, 6 analog input pins, and a USB interface for programming and communication [3]. It can be powered by either USB or an external power supply. The board has a reset button, LEDs for power and pin 13, an ICSP header for programming, and headers for connecting external components. It's popular for

prototyping and DIY projects because of its simplicity, versatility, and compatibility with a diverse ecosystem of shields and accessories. Overall, Arduino is a versatile platform that allows users to quickly and easily build interactive electronic projects and prototypes. Its open-source nature, simplicity, and extensive ecosystem have made it a popular choice among makers, educators, and professionals all over the world.

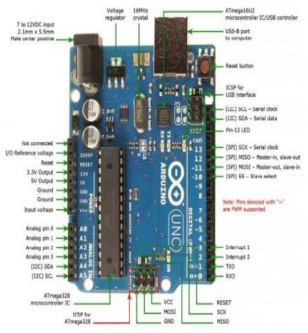


Fig 4: Arduino Uno R3

b. **Liquid Crystal Display:** The LCD 1602 is a widely used alphanumeric liquid crystal display module with 16 characters per line and two lines. It is widely used in electronic projects to display text-based data. It's controlled via a parallel interface, compatible with microcontroller platforms like Arduino, and includes features like adjustable contrast settings and custom character support.

| Pin | Function | Connected to |
|---|---|---|
| 1 | VSS | GND |
| 2 | VDD | 5V |
| 3 | A1 | Increase button |
| 4 | A2 | Decrease button |
| 5 | A3 | Cruise button |
| 6 | A4 | Adaptive cruise control button |
| 7 | A5 | Cancel button |

| | | |
|---|---|---|
| 8 | D4 | Data pins |
| 9 | D5 | Data pins |
| 10 | D6 | Data pins |
| 11 | D7 | Data pins |
| 12 | D10 | Active buzzer |
| 13 | D12 | Register select pin |
| 14 | D13 | LED light |
| 15 | A | 5V |
| 16 | K | GND |

Fig 5: Liquid crystal display and connections

c. **Jumper Cables:** Jumper cables are electrical wires with connectors on each end that are used to make temporary connection between components in electronic circuits. They come in a variety of lengths, colours, and wire gauges and are widely used for prototyping, testing, and debugging electronic projects.

Fig 6: Jumper cables

d. **Breadboard:** A breadboard is a versatile tool used for electronic prototyping. It is a rectangular plastic board with a grid of holes that allows electronic components to be inserted and connected without the need for soldering. Jumper wires are used to connect components, and power rails along the edges make it easy to connect power and ground. Breadboards are reusable and enable quick and simple experimentation with electronic circuits.

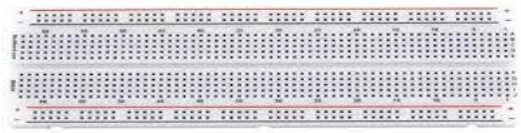


Fig 7: Breadboard

e. **Ultrasonic sensor:** An ultrasonic sensor sends out high-frequency sound waves and measures how long it takes for the waves to bounce off an object and return to the sensor [4]. This time measurement is used to determine the distance to the object. Ultrasonic sensors are widely used for distance measurement, object detection, and proximity sensing in robotics, automotive systems, industrial automation, and consumer electronics. They operate without contact, have a wide detection range, and are reliable in a variety of environments.


| Pins | Functions | Connected to |
|---|---|---|
| 1 | VSS | 5V |
| 2 | TRIG | D9 |
| 3 | ECHO | D8 |
| 4 | GND | GND |

Fig 8: Ultrasonic sensor and connections

f. **Push buttons:** Push buttons are electromechanical devices that allow for momentary control in electronic circuits. When pressed, they temporarily complete a circuit to allow for user input or control. They come in a variety of configurations, including normally open and normally closed, and are used in applications such as power switches, input controls, and reset buttons. Push buttons are popular in electronics because they are long-lasting, versatile, and easy to use.

Fig 9: Push buttons

g. **Resistors**: Resistors are passive components that limit the flow of electric current through a circuit. They come in a variety of types, sizes, and resistance values (measured in ohms). Electronic circuits use resistors to control current,

reduce voltage, set biasing conditions, and divide voltages. They are essential components in almost every electronic device and system.


Fig 10: 10k resistor

h. **Potentiometer:** A potentiometer is a three-terminal variable resistor that allows you to manually adjust the resistance within a specific range [5]. The resistance between the input and output terminals can be adjusted by rotating a knob or sliding a lever. Potentiometers are used in electronic circuits to control volume, brightness, motor speed, and voltage. They are available in a variety of types, resistance values, and taper characteristics.

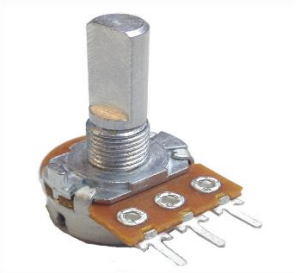


Fig 11: Potentiometer

- **Software:** We have used MATLAB for this project, MATLAB is a high-level programming language and interactive environment designed for numerical computation and data analysis. It includes built-in functions for mathematical calculations, matrix operations, plotting, and visualization. MATLAB includes a number of toolboxes for specialized tasks and is widely used in academia, research, engineering, and industry for scientific computing, modelling, and simulation. It has a user-friendly interface, supports multiple platforms, and integrates with other software applications.

## 6. MATLAB code

```
% Clear command window, workspace, and close all figures
clc;
clear all;
close all;

% Initialize Arduino connection with necessary libraries
arduinoObj =
arduino('COM5','Uno','Libraries',{'Ultrasonic','ExampleLCD/LCDAddon'},'ForceBuildO
n',true);

% Initialize ultrasonic sensor and LCD display
ultrasonicSensor = ultrasonic(arduinoObj,'D9','D8');
lcd =
addon(arduinoObj,'ExampleLCD/LCDAddon','RegisterSelectPin','D12','EnablePin','D11','
DataPins',{'D4','D5','D6','D7'});
initializeLCD(lcd);

% Display welcome message on LCD
printLCD(lcd,'WELCOME');
pause(2);

% Clear LCD and print project details
clearLCD(lcd);
printLCD(lcd,'ADAPTIVE');
printLCD(lcd,'CRUISE CONTROL');
pause(2);
clearLCD(lcd);
printLCD(lcd,'MATLAB');
printLCD(lcd,'Team 1');
pause(2);
clearLCD(lcd);
printLCD(lcd,'BADRINADH ANDE');
printLCD(lcd,'110143113');
```

```
pause(2);
clearLCD(lcd);
printLCD(lcd,'SHOAIB AKTHAR');
printLCD(lcd,'110126210');
pause(2);
clearLCD(lcd);
printLCD(lcd,'SAI TEJA');
printLCD(lcd,'110132353');
pause(2);
clearLCD(lcd);

% Initialize variables for button states, speed, and control mode


incrementButton = 0;
decrementButton = 0;
cancelButton = 0;
setSpeedButton = 0;
adaptiveCruiseButton = 0;
counter = 0;
speed = 0;
blink = 0;

% Main control loop
while true

    % Read button states and distance from ultrasonic sensor
    incrementButton = readVoltage(arduinoObj,'A1');
    decrementButton = readVoltage(arduinoObj,'A2');
    cancelButton = readVoltage(arduinoObj,'A5');
    setSpeedButton = readVoltage(arduinoObj,'A3');
    adaptiveCruiseButton = readVoltage(arduinoObj,'A4');
    distance = readDistance(ultrasonicSensor);
```

```
% Control speed based on button presses and control mode
if counter == 0
    if incrementButton >= 1.0
        speed = speed + 1; % Increase speed
        disp(speed)
    elseif decrementButton >= 1.0
        speed = speed - 1; % Decrease speed
        pause(0.5);
    else
        speed = speed - 1; % Default speed decrease
    end
elseif counter == 1
    speed; % Maintain current speed when setting speed
elseif counter == 2

    % Adjust speed based on distance from obstacle
    if distance < 0.5
        speed = speed - 1; % Slow down if obstacle detected
        writeDigitalPin(arduinoObj,'D13',1)
        writeDigitalPin(arduinoObj,"D10",0);
    else
        speed = speed + 1; % Increase speed if no obstacle
        writeDigitalPin(arduinoObj,'D13',0)
        writeDigitalPin(arduinoObj,"D10",1);
    end
    if speed > speedLimit
        speed = speedLimit; % Cap speed at maximum limit
    end
end

% Control mode selection based on button presses
if setSpeedButton >= 2.0
```

```
    counter = 1; % Enter set speed mode
elseif incrementButton >= 2.0 && counter == 1
    speed = speed + 1; % Increase speed during set speed mode
elseif decrementButton >= 2.0 && counter == 1
    speed = speed - 1; % Decrease speed during set speed mode
elseif adaptiveCruiseButton >= 2.0 % Changed variable name here
    % Activate adaptive cruise control mode
    printLCD(lcd,'Adaptive Cruise');
    printLCD(lcd,'Control ON');
    pause(2);
    counter = 2;
    blink = 1;
    speedLimit = speed; % Set speed limit for cruise control
elseif cancelButton >= 2.0
    blink = 0;
    counter = 0; % Exit any active mode and return to default
end

% Ensure speed does not drop below zero
if speed < 0
    speed = 0;
end

% Update LCD display based on control mode
if blink == 1
    % Display adaptive cruise control mode
    printLCD(lcd,'ACC MODE');
    x1 = 'CURRENT SPEED:';
    y1 = char(string(num2str(speed)));
    printLCD(lcd, [x1 y1]);
    pause(0.5);
elseif counter == 1
    % Display set speed mode
```

```
        printLCD(lcd,'CRUISE MODE');
        x1 = 'CURRENT SPEED:';
        y1 = char(string(num2str(speed)));
        printLCD(lcd, [x1 y1]);
        pause(0.5);
    else
        % Display normal mode
        printLCD(lcd,'NORMAL MODE');
        x1 = 'CURRENT SPEED:';
        y1 = char(string(num2str(speed)));
        printLCD(lcd, [x1 y1]);
        pause(0.5);
    end
end
```

## 7. Functional explanation of the code

This MATLAB code is intended to build an adaptive cruise control system with an Arduino board, an ultrasonic sensor, and an LCD display. Here is a functional description of the code:

### a. Initialization:

- The code begins by clearing the command window, workspace, and any open figures.
- It configures the Arduino connection with the libraries required for the ultrasonic sensor and LCD display.

### b. Ultrasonic Sensor and LCD Initialization:

- It sets up the ultrasonic sensor and LCD display connected to the Arduino board.

### c. Welcome Message:

- It shows a welcome message on the LCD screen.

### d. Project Details Display:

- It clears the LCD and prints project information such as "ADAPTIVE CRUISE CONTROL", team member names, and IDs.

### e. Variable Initialization:

- The variables for button states, speed, control mode, and blink status are initialized.

- These variables track button presses, current speed, control modes, and blink status.

## f. Main Control Loop:

- Inside an infinite loop, the code reads the button states and distance from the ultrasonic sensor on a continuous basis.
- It adjusts the system speed based on the number of button presses and the control mode.

## g. Speed Control:

- Depending on the control mode, pressing specific buttons causes the code to increase or decrease the speed.
- The adaptive cruise control mode adjusts the speed based on the distance between obstacles detected by the ultrasonic sensor.

## h. Control Mode Selection:

The code allows the selection of various control modes:

- "Set Speed" mode: Allows the user to specify the desired speed.
- "Adaptive Cruise Control" mode: Enables adaptive cruise control, which adjusts the speed based on distance from obstacles.
- Otherwise, it displays "NORMAL MODE" along with the current speed.

## i. LCD Display Update:

- The LCD display is constantly updated to reflect the current mode and speed.
- If adaptive cruise control mode is enabled, it displays additional information about the mode.

## j. Safety Precautions:

- The code ensures that the speed does not fall below zero and caps it at a maximum value.

## k. LED Indication:

- Depending on the mode and sensor readings, LED indicators turn on or off to provide visual feedback.

## l. Endless Loop:

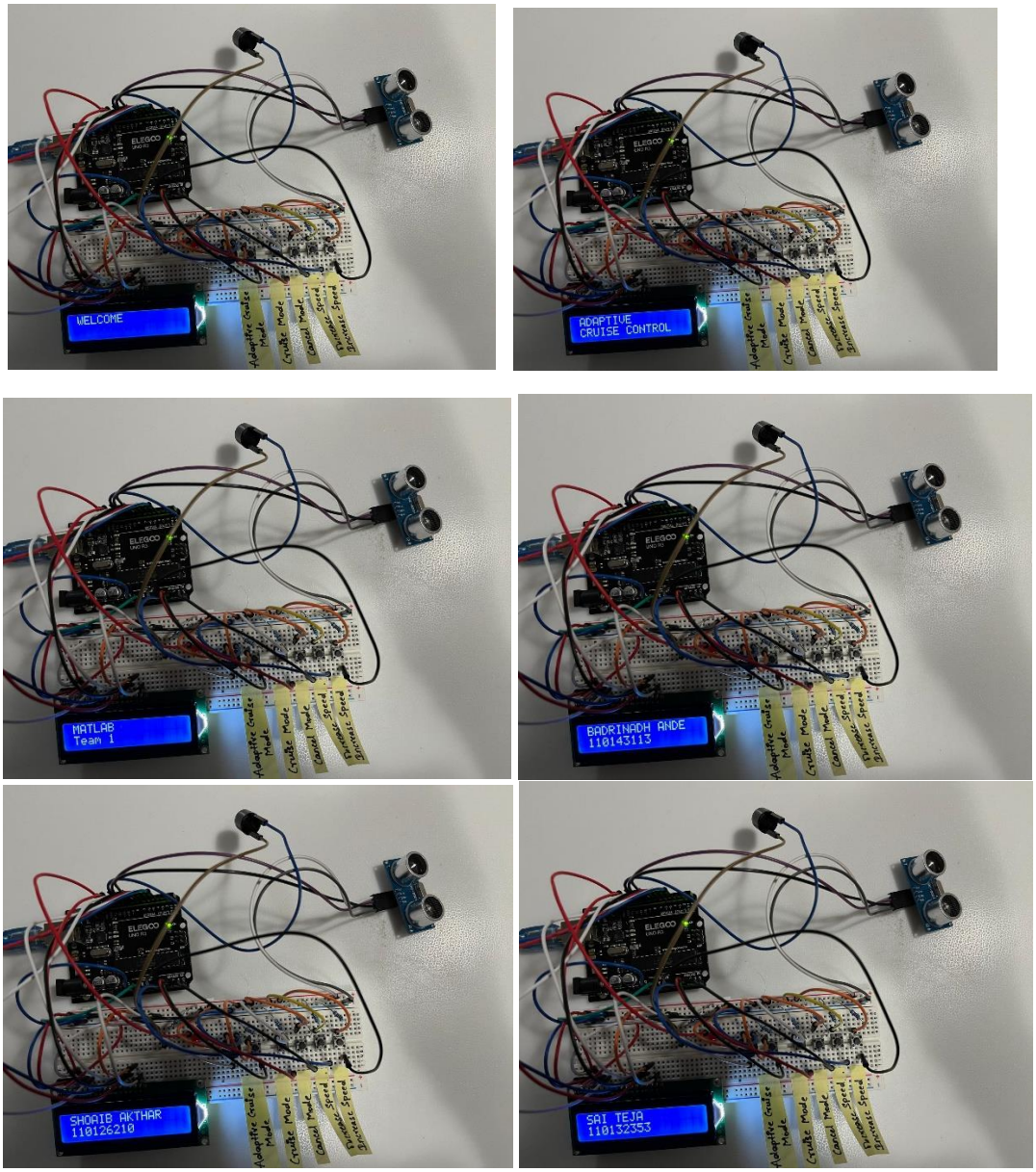
- The control loop will run indefinitely until the program is terminated externally.

Overall, this code serves as the foundation for an adaptive cruise control system that allows the user to select a desired speed, activates adaptive cruise control to

adjust speed in response to obstacles, and provides real-time feedback via an LCD display.

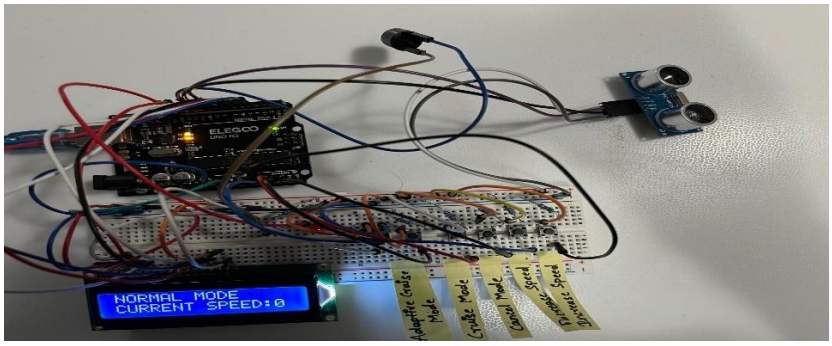
## 8. Testing Scenarios

- **Initial Stage**



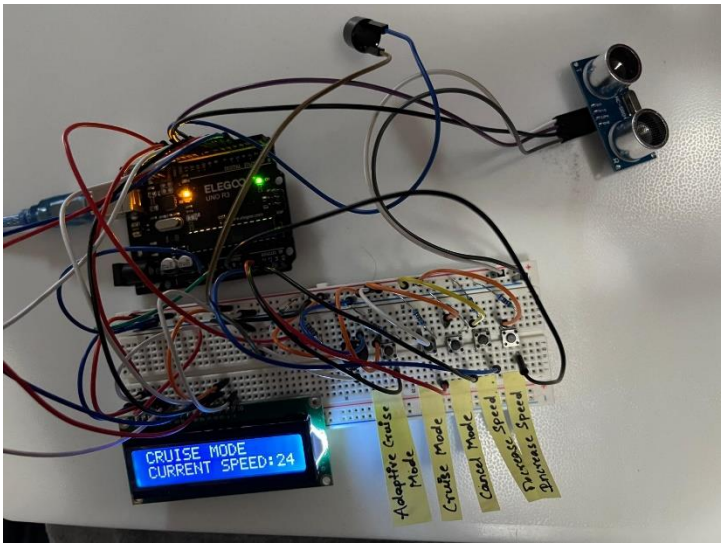

a. This initial display provides essential project details, including the project name, team number, and the names of team members along with their respective student IDs.

- **Normal mode**




a. In normal mode, the system operates without any special features activated.
b. The user manually controls the speed of the vehicle using buttons.
c. No automated adjustments are made based on sensor inputs.

- **Cruise mode**




a. In cruise mode, the system allows the user to set a desired speed for the vehicle.
b. Once set, the vehicle maintains this speed until manually adjusted by the user.
c. This mode is ideal for highway driving, where consistent speeds are desirable.

- **Adaptive cruise mode**

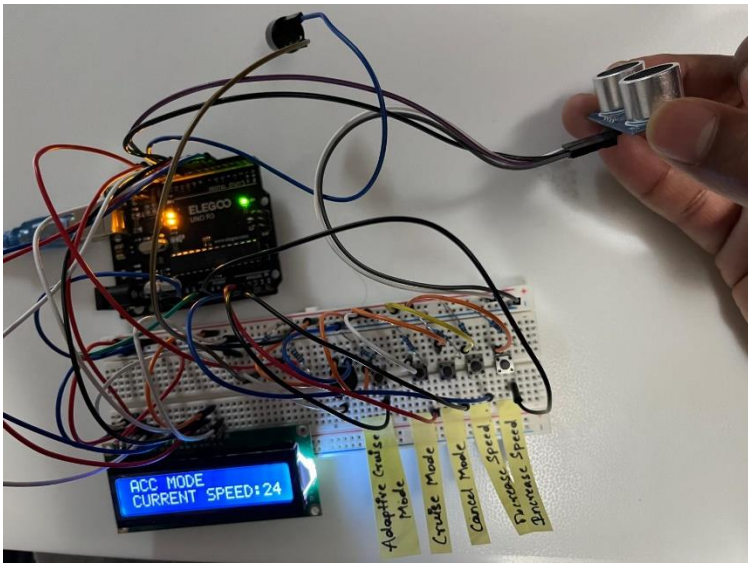



Fig 12: Images of testing scenarios

a. Adaptive cruise mode utilizes sensor inputs to automatically adjust the vehicle's speed based on the distance to other vehicles or obstacles.
b. The system maintains a safe following distance by slowing down or speeding up as necessary.
c. This mode enhances safety and convenience, particularly in heavy traffic or unpredictable driving conditions.

## 9. Responsibilities, Timeline and Limitations/risks

| Team members | Responsibilities |
|---|---|
| • Badrinadh Ande | • Worked on circuit connections.<br>• Contributed to final report.<br>• Created and optimized MATLAB code.<br>• Scheduled meetings and assigned tasks accordingly. |
| • Shoaib Akthar Shaik | • Conducted extensive research on MATLAB code.<br>• Formulated a comprehensive flowchart.<br>• Contributed significant contributions to the final report.<br>• Managed testing and troubleshooting efforts. |

| | |
|---|---|
| • Venkata Sai Teja Pavuluri | • Conducted extensive research on components.<br>• Collected necessary data and components.<br>• Performed extensive research on Arduino technology.<br>• Made significant contributions to the final report. |
| • **Limitations/risks** | • In summary, key limitations and risks when implementing an Adaptive Cruise Control (ACC) project in MATLAB include hardware compatibility issues, sensor accuracy, signal processing delays, and testing and validation requirements. Addressing these aspects requires careful planning, design, and testing to ensure the safety, dependability, and effectiveness of the ACC system. |

**Table 2**

| Milestones Completed | Duration (as per semester weeks) |
|---|---|
| Gather all the information Required | Week 4 |
| Creating an outline & making a initial draft for the primary project report | Week 4 |
| Revising and editing the draft | Week 5 |
| Finalizing the draft | Week 5 |
| Getting hands on the Arduino Kit | Week 5 |
| Getting Familiarized with Arduino and MATLAB | Week 6 |
| Circuit Design and connections | Week 7 |
| Implementation and Testing | Week 8 |
| Documentation and finalizing the project | Week 8 |
| Review the final project | Week 9 |
| Presentation | Week10 |

**Table 3**

## 10. Lessons learned

Through the Adaptive Cruise Control (ACC) project, we learned some important lessons. Firstly, we realized how essential it is to do thorough research and planning, especially when dealing with different hardware and sensors. Knowing what each part can and can't do is key to making everything work smoothly. Plus, we saw how crucial it is to communicate well and divide tasks among team members. Working together to solve problems and figure out issues like delays and testing was super important. We also learned that testing and making sure everything works safely is a big part of any project like this. Overall, this project taught us a lot about planning carefully, teamwork, and paying attention to the small details in complicated projects.

## 11. Conclusion

The adaptive cruise control system project, which utilizes MATLAB and Arduino, successfully integrates hardware components and software programming to produce a functional system. It improves safety by adjusting vehicle speed in response to detected obstacles, provides a user-friendly interface via an LCD display, offers flexibility through various control modes, and is highly customizable for a wide range of applications. The project serves as an educational resource and has the potential for further development in automotive automation.

## 12. References

[1] "Adaptive Cruise Control," Wikipedia, https://en.wikipedia.org/wiki/Adaptive_cruise_control (accessed Mar. 20, 2024).

[2] P. V. Phanindra, "Adaptive Cruise Control : Skill-lync," Skill Lync, https://skill-lync.com/student-projects/project-2-adaptive-cruise-control-279 (accessed Mar. 20, 2024).

[3] "Uno R3," docs.arduino.cc, https://docs.arduino.cc/hardware/uno-rev3/ (accessed Mar. 20, 2024).

[4] D. Jost, "What is an ultrasonic sensor?," Fierce Electronics, https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor (accessed Mar. 20, 2024).

[5] "Potentiometer: Resistor types: Resistor guide," EEPower, https://eepower.com/resistor-guide/resistor-types/potentiometer/# (accessed Mar. 20, 2024).