**List of Figures**

[Figure 1: Adaptive Cruise Control 6](#_Toc141114100)

[Figure 2: Arduino’s LCD Interface 7](#_Toc141114101)

[Figure 3: Arduino Uno 8](#_Toc141114102)

[Figure 4: Push Button 8](https://d.docs.live.net/8167a02f4c079b44/Documents/GitHub/COMPUTATIONAL-METHODS-AND-MODELING-FOR-ENGINEERING-APPLICATIONS/Matlab%20Project/Final%20Report/Adaptive%20Cruise%20Control%20Final%20Report.docx#_Toc141114103)

[Figure 5: Digital Input 8](#_Toc141114104)

[Figure 6: Ultrasonic Sensor 9](#_Toc141114105)

[Figure 7: Resistor 9](#_Toc141114106)

[Figure 8: Jumper Wires 10](#_Toc141114107)

[Figure 9: Breadboard 10](#_Toc141114108)

[Figure 10: Potentiometer 11](#_Toc141114109)

[Figure 11: Flowchart of ACC 12](#_Toc141114110)

[Figure 12: Gantt Chart 24](#_Toc141114111)

[Figure 13: Circuit View 28](#_Toc141114112)

[Figure 14: Schematic View 29](#_Toc141114113)

[Figure 15: Welcome message 30](#_Toc141114114)

[Figure 16: Group Number & Names 30](#_Toc141114115)

[Figure 17: Circuit at initial (zero speed) 31](#_Toc141114116)

[Figure 18: Circuit in Cruise Mode (non-zero speed) 31](#_Toc141114117)

[Figure 19: Circuit in Cruise Mode (zero speed) 31](#_Toc141114118)

[Figure 20: Circuit in Adaptive Cruise Control Mode (no object in front of ultrasonic sensor) 31](#_Toc141114119)

[Figure 21: Circuit in Adaptive Cruise Control Mode (an object in front of ultrasonic sensor) 31](#_Toc141114120)

[Figure 22: Circuit connections 32](#_Toc141114121)

[Figure 23: Welcome message 33](#_Toc141114122)

[Figure 24: Group Number & Names 33](#_Toc141114123)

[Figure 25: Circuit at initial (zero speed) 34](#_Toc141114124)

[Figure 26: Circuit in Cruise Mode (non-zero speed) 34](#_Toc141114125)

[Figure 27: Circuit in Cruise Mode (zero speed) 35](#_Toc141114126)

[Figure 28: Circuit in Adaptive Cruise Control Mode (no object in front of ultrasonic sensor) 35](#_Toc141114127)

[Figure 29: Circuit in Adaptive Cruise Control Mode (an object in front of ultrasonic sensor) 36](#_Toc141114128)

**List of Tables**

[Table 1: Software Tools 11](#_Toc141112519)

[Table 2: Milestones Completed 23](#_Toc141112520)

[Table 3: Roles and Responsibilities 25](#_Toc141112521)

[Table 4: Arduino Uno & the LCD display Connections 26](#_Toc141112522)

[Table 5: Arduino Uno & Ultrasonic Sensor Connections 26](#_Toc141112523)

[Table 6: Arduino Uno & Pushbuttons Connections 26](#_Toc141112524)

[Table 7: Component Table 30](#_Toc141112525)

**Table of Contents**

[I. Introduction 6](#_Toc141112873)

[II. Project Objectives 7](#_Toc141112874)

[III. Hardware Components and Tools 7](#_Toc141112875)

[*A.* *Hardware Components* 7](#_Toc141112876)

[*1)* *LCD Display* 7](#_Toc141112877)

[*2)* *Arduino Uno* 8](#_Toc141112878)

[*3)* *Push Buttons* 8](#_Toc141112879)

[*4)* *Ultrasonic Sensor* 9](#_Toc141112880)

[*5)* *Resistors* 9](#_Toc141112881)

[*6)* *Battery* 10](#_Toc141112882)

[*7)* *Battery Connector Cable* 10](#_Toc141112883)

[*8)* *Jumper Wires* 10](#_Toc141112884)

[*9)* *PCB or Breadboard* 10](#_Toc141112885)

[*10)* *Potentiometer* 11](#_Toc141112886)

[*B.* *Software Tools* 11](#_Toc141112887)

[*1)* *Simulink* 11](#_Toc141112888)

[*2)* *MATLAB* 11](#_Toc141112889)

[IV. Methodology 11](#_Toc141112890)

[*A.* *Flowchart* 12](#_Toc141112891)

[V. MATLAB Program 13](#_Toc141112892)

[*A.* *Algorithm: ACC* 13](#_Toc141112893)

[*B.* *Pseudocode* 15](#_Toc141112894)

[*C.* *MATLAB Program* 18](#_Toc141112895)

[VI. Timeline 23](#_Toc141112896)

[*A.* *Milestones* 23](#_Toc141112897)

[*B.* *Gantt Chart* 24](#_Toc141112898)

[*C.* *Limitations and Risks* 24](#_Toc141112899)

[*1)* *Hardware Availability:* 24](#_Toc141112900)

[*2)* *Real-world Constraints* 24](#_Toc141112901)

[*3)* *System Complexity* 24](#_Toc141112902)

[*4)* *Safety Considerations* 24](#_Toc141112903)

[*D.* *Roles and Responsibilities* 25](#_Toc141112904)

[VII. ACC Execution Procedure 25](#_Toc141112905)

[*A.* *Step 1: Gather Hardware Components* 25](#_Toc141112906)

[*B.* *Step 2: Prepare Arduino Environment* 25](#_Toc141112907)

[*C.* *Step 3: Arduino Uno & LCD Connections* 26](#_Toc141112908)

[*D.* *Step 4: Arduino Uno & Ultrasonic Sensor Connections* 26](#_Toc141112909)

[*E.* *Step 5: Arduino Uno & Pushbuttons Connections* 26](#_Toc141112910)

[*F.* *Step 6: Assemble the Circuit* 27](#_Toc141112911)

[*G.* *Step 7: MATLAB Program* 27](#_Toc141112912)

[*H.* *Step 8: Upload the Code to Arduino Uno* 27](#_Toc141112913)

[*I.* *Step 9: Power Supply* 27](#_Toc141112914)

[*J.* *Step 10: Test and Calibration* 27](#_Toc141112915)

[*K.* *Step 11: Fine-Tuning and Troubleshooting* 27](#_Toc141112916)

[VIII. Tinkercad Software 28](#_Toc141112917)

[*A.* *Circuit View* 28](#_Toc141112918)

[*B.* *Schematic View* 29](#_Toc141112919)

[*C.* *Component Table* 30](#_Toc141112920)

[*D.* *Tinkercad Simulation* 30](#_Toc141112921)

[IX. Working Model 32](#_Toc141112922)

[*A.* *Circuit* 32](#_Toc141112923)

[*B.* *Welcome Message* 33](#_Toc141112924)

[*C.* *Group Number & Names* 33](#_Toc141112925)

[*D.* *Circuit at initial (zero speed)* 34](#_Toc141112926)

[*E.* *Circuit in Cruise Mode (non-zero speed)* 34](#_Toc141112927)

[*F.* *Circuit in Cruise Mode (zero speed)* 35](#_Toc141112928)

[*G.* *Circuit in Adaptive Cruise Control Mode (no object in front of ultrasonic sensor)* 35](#_Toc141112929)

[*H.* *Circuit in Adaptive Cruise Control Mode (an object in front of ultrasonic sensor)* 36](#_Toc141112930)

[X. Testing Scenarios and Output Results 36](#_Toc141112931)

[*A.* *Normal Mode* 36](#_Toc141112932)

[*B.* *Cruise Control Mode* 36](#_Toc141112933)

[*C.* *Adaptive Cruise Control Mode* 37](#_Toc141112934)

[*D.* *Real-World Constraints* 37](#_Toc141112935)

[*E.* *Safety Validation* 37](#_Toc141112936)

[XI. Lessons Learned 38](#_Toc141112937)

[XII. Conclusion 39](#_Toc141112938)

[References 39](#_Toc141112939)

# **Introduction**

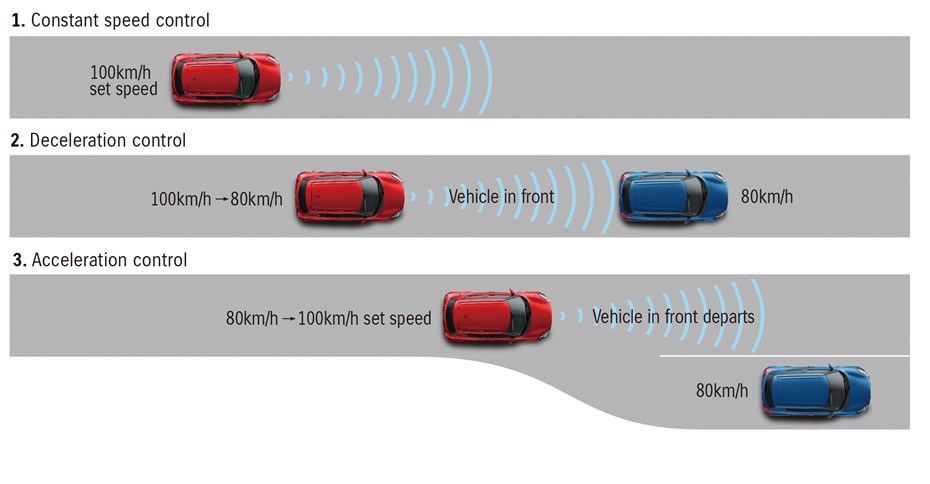
Adaptive Cruise Control (ACC) is an exciting advancement in automotive technology that aims to enhance both convenience and safety on the roads. Unlike traditional cruise control systems, ACC utilizes sensors and intelligent algorithms to automatically adjust a vehicle's speed and maintain a safe distance from the vehicle in front, even in changing traffic conditions [1].

In the past, cruise control allowed drivers to set a specific speed for their vehicles, but it lacked the ability to adapt to traffic situations. ACC changes the game by incorporating radar, lidar, and camera sensors that continuously scan the road ahead [2]. These sensors provide real-time information about the distance and speed of other vehicles, enabling the system to make informed decisions about accelerating or decelerating to match the leading vehicle's pace.

The primary goal of ACC is to improve safety on the roads. By automatically adjusting the speed and maintaining a safe following distance, ACC reduces the risk of rear-end collisions, which are a common type of accident [3]. This feature also helps prevent driver fatigue by reducing the need for constant monitoring of traffic conditions.

The benefits of ACC extend beyond safety. It offers a significant convenience factor, allowing drivers to enjoy the advantages of cruise control while still relying on the system to handle the changing traffic environment. ACC optimizes driving efficiency and provides a more comfortable experience, particularly in congested or variable-speed situations.

ACC serves as a crucial step as the automotive industry moves towards the development of autonomous vehicles. It is part of a broader concept known as Advanced Driver-Assistance Systems (ADAS) and contributes to the ongoing progress of creating self-driving cars [4].

[](https://parkers-images.bauersecure.com/wp-images/17151/930x620/suzuki_adaptive_cruise.jpg)

**Figure 1**: Adaptive Cruise Control [5]

# **Project Objectives**

The project aims to demonstrate the functionality and effectiveness of an ACC system through MATLAB integration and software coding. The project seeks to accomplish three following goals.

1. Develop a control system that automatically adjusts the host vehicle's speed based on the movements of the preceding vehicle. This system will maintain a safe and comfortable following distance between the two vehicles.
2. Apply a sensor that continuously monitors the distance between the vehicles and adjust the host vehicle's speed to avoid collisions or unsafe situations.
3. Analyze and showcase the ACC system's response to changes in the preceding vehicle's speed. The system will promptly and smoothly adapt to changes in the preceding vehicle's speed while maintaining a safe distance and adapting to traffic conditions.

By achieving these objectives, the project will provide a practical demonstration of the benefits and effectiveness of ACC in enhancing driving safety, convenience, and comfort [6].

# **Hardware Components and Tools**

## ***Hardware Components***

### *LCD Display:* Visual displays are used to provide real-time information and feedback to the driver, such as current speed and distance measurements [7].

[A picture containing text, screenshot, display, rectangle

Description automatically generated](https://microcontrollerslab.com/wp-content/uploads/2015/01/16X2-LCD-Pinout-diagram-Pic-Microcontroller-tutorial.png)

**Figure 2**:Arduino’s LCD Interface [8]

### *Arduino Uno:* A microcontroller board serves as the central control unit for the ACC system [7]. It receives input from sensors, processes data, and generates output signals for speed control.

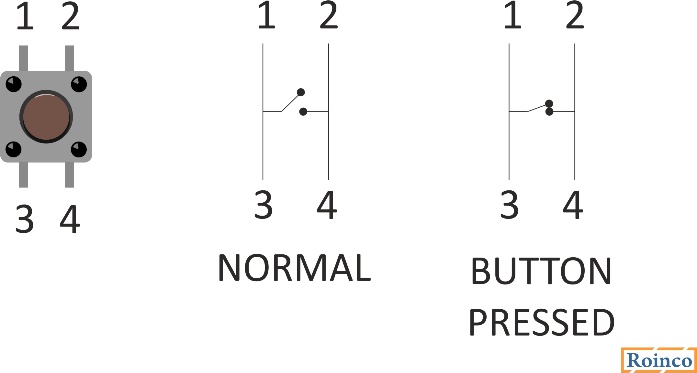
[A picture containing text, electronics, circuit, electronic engineering

Description automatically generated](https://www.jameco.com/Jameco/workshop/CircuitNotes/CN-arduino-uno-fig1-enlarge.jpg)

**Figure 3**:Arduino Uno [9]

### *Push Buttons:* Five buttons are used for various functionalities, including setting the desired speed, enabling/disabling the cruise control, and adjusting system parameters.

[A red button on a black board

Description automatically generated with medium confidence](https://www.circuittest.com/Media/400/push-04.jpg)[](https://roboindia.com/tutorials/wp-content/uploads/2020/02/BUTTON-STATUS.jpg)

**Figure 4**: Push Button [10]

**Figure 5**: Digital Input [11]

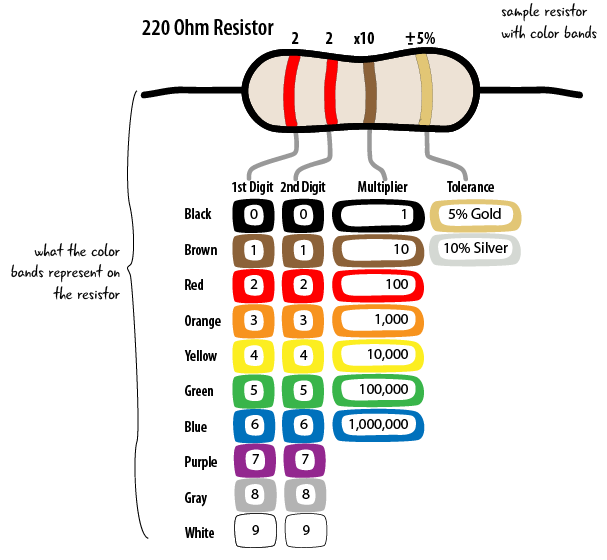
### *Ultrasonic Sensor:* The ultrasonic sensor is an essential component for accurately measuring the distance between the host vehicle and the preceding vehicle [7]. It utilizes ultrasonic waves to determine the distance and provides input to the control system.

[A picture containing text, screenshot, circle

Description automatically generated](https://www.the-diy-life.com/wp-content/uploads/2017/08/hc-sr04-ultrasonic-sensor-diagram.jpg)

**Figure 6**:Ultrasonic Sensor [12]

### *Resistors:* Resistors are used to limit current flow and protect components from damage [7]. They are employed in various parts of the circuitry, such as voltage dividers and pull-up/pull-down resistors.

[](https://arduinotogo.com/wp-content/uploads/2017/03/app-a-resistor-chart.png)

**Figure 7**:Resistor [13]

### *Battery:* The battery is a suitable power source that provides the necessary electrical energy to run the ACC system.

### *Battery Connector Cable:* The battery connector cable is used to connect the battery to the Arduino board and other components, ensuring a reliable power supply.

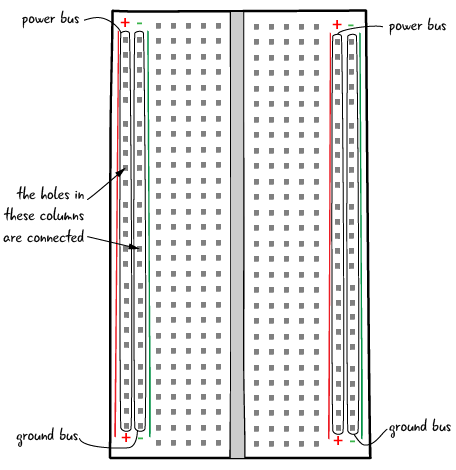
### *Jumper Wires:* Wires are used to establish connections between different components on the breadboard or PCB, thereby enabling the flow of signals and power [7].

[A picture containing screenshot, stripe, text, design

Description automatically generated](https://electronicsinfra.com/wp-content/uploads/2021/11/JUMPER-WIRE.jpg)

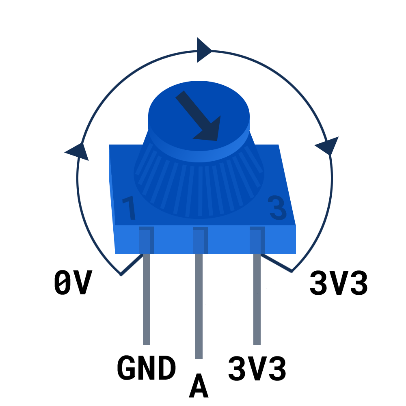
**Figure 8**:Jumper Wires [14]

### *PCB or Breadboard:* PCB is a prototyping platform that allows for the interconnection of various hardware components and circuits, which facilitate the development and testing of the ACC system [7].

[](https://arduinotogo.com/wp-content/uploads/2016/07/ch-2-breadboard-top-labelled-01.png)

**Figure 9**:Breadboard [15]

### *Potentiometer:* Potentiometers are variable resistors that are used to adjust system parameters, such as sensitivity or the desired following distance [7].

[](https://projects-static.raspberrypi.org/projects/potentiometer-wiring/b65e3aa6930d7737962bbfff7069e17ca8a7c918/en/images/potentiometer-illustration.png)

**Figure 10**:Potentiometer [16]

## ***Software Tools***

**Table 1**:Software Tools

|  |  |
| --- | --- |
| **Software Tools** | **Description** |
| *Simulink* | * A powerful graphical programming environment provided by MATLAB. * It enables the modeling, simulation, and implementation of control systems using intuitive block diagrams [17]. |
| *MATLAB* | * A high-level programming language and environment widely used for mathematical modeling, algorithm development, and data analysis. * It offers extensive functionality and toolboxes for system simulation, control design, and signal processing [18]. |

# **Methodology**

The methodology for implementing the ACC system can be summarized using the high-level architecture seen in Figure 11.

## ***Flowchart***

A diagram of a cruise control system

Description automatically generated

**Figure 11:** Flowchart of ACC

# **MATLAB Program**

## ***Algorithm: ACC***

* 1. Initialize the Arduino board and Ultrasonic Sensor.
  2. Initialize the LCD display and clear its content.
  3. Print project name and group number on the first two lines of the LCD display.
  4. Pause execution for 5 seconds to display the information.
  5. Clear the LCD display.
  6. Print team members' names on the LCD display.
  7. Pause execution for 5 seconds to display the information.
  8. Declare and initialize variables for various inputs and modes:
     + speed: current vehicle speed
     + increase\_speed: input from the increase speed button
     + decrease\_speed: input from the decrease speed button
     + cancel: input from the cancel button
     + set\_speed: input from the set speed button
     + adaptive\_cruise\_speed: input from the adaptive cruise control button
     + distance: distance measured by the ultrasonic sensor
     + mode: variable to indicate the current mode of operation (0 = Normal Mode, 1 = Cruise Control Mode, 2 = Adaptive Cruise Control Mode)
  9. Enter an infinite loop to continuously monitor and update the vehicle speed.
     + Read inputs from the analog pins on the Arduino board for the various buttons and the ultrasonic sensor.
     + Determine the mode of operation based on button inputs:
     + If the cancel button is pressed (voltage value >= 4), set mode to 0 (Normal Mode).
     + If the set speed button is pressed (voltage value >= 4), set mode to 1 (Cruise Control Mode).
     + If the adaptive cruise control button is pressed (voltage value >= 4), set mode to 2 (Adaptive Cruise Control Mode) and store the current speed in 'constant'.
  10. Implement different behaviors based on the current mode:
      + Normal Mode:
      + Increase speed if the increase speed button is pressed.
      + Decrease speed if the decrease speed button is pressed.
      + Gradually decrease speed if no button is pressed.
      + Ensure speed doesn't go below 0.
      + Display the current speed on the LCD display with the label "Vehicle Speed: ".
      + Cruise Control Mode:
      + Increase speed if the increase speed button is pressed.
      + Decrease speed if the decrease speed button is pressed.
      + Ensure speed doesn't go below 0.
      + Display the current speed on the LCD display with the label "Cruise mode: ".
      + Adaptive Cruise Control Mode:
      + Clear the LCD display and display a blinking effect.
      + Reinitialize the LCD display.
      + Activate the vehicle's motor (D13) and deactivate the brake (D12).
      + Adjust speed based on the distance measured by the ultrasonic sensor:
      + If the distance is less than 0.3 units, decrease the speed.
      + If the distance is greater than or equal to 0.3 units, increase the speed.
      + Ensure the speed doesn't exceed the constant value (stored previously).
      + Ensure speed doesn't go below 0.
      + Display the current speed on the LCD display with the label "Adap\_Cruise\_mode".
  11. End of the infinite loop.

## ***Pseudocode***

* 1. Initialize the Arduino board and Ultrasonic Sensor.
  2. Initialize the LCD display and clear its content.
  3. Print "WELCOME TO" on the first line and "ACC PROJECT" on the second line of the LCD display.
  4. Pause execution for 5 seconds.
  5. Clear the LCD display.
  6. Print "Group 32" on the first line and "Amey,Brano,Nandu" on the second line of the LCD display.
  7. Pause execution for 5 seconds.
  8. // Variable Declarations

speed = 0

increase\_speed = 0

decrease\_speed = 0

cancel = 0

set\_speed = 0

adaptive\_cruise\_speed = 0

distance = 0

mode = 0

* 1. while true:

// Infinite loop to continuously monitor and update the vehicle speed

// Read inputs from the analog pins on the Arduino board for buttons and ultrasonic sensor

increase\_speed = readVoltage(A0)

decrease\_speed = readVoltage(A1)

cancel = readVoltage(A2)

set\_speed = readVoltage(A3)

adaptive\_cruise\_speed = readVoltage(A4)

// Determine the mode of operation based on button inputs

if cancel >= 4:

mode = 0 // Normal Mode

elseif set\_speed >= 4:

mode = 1 // Cruise Control Mode

elseif adaptive\_cruise\_speed >= 4:

mode = 2 // Adaptive Cruise Control Mode

constant = speed // Store current speed in 'constant'

// Normal Mode

if mode == 0:

if increase\_speed >= 4:

set pin D13 to HIGH // Activate motor

set pin D12 to LOW // Deactivate brake

speed = speed + 1

pause for 0.1 seconds

elseif decrease\_speed >= 4:

speed = speed - 1

pause for 0.1 seconds

else:

speed = speed - 1

pause for 1.5 seconds

if speed < 0:

set pin D13 to LOW // Deactivate motor

set pin D12 to HIGH // Activate brake

speed = 0

print "Vehicle Speed: " + speed on the LCD display

// Cruise Control Mode

elseif mode == 1:

if increase\_speed >= 4:

set pin D13 to HIGH // Activate motor

set pin D12 to LOW // Deactivate brake

speed = speed + 1

pause for 0.1 seconds

elseif decrease\_speed >= 4:

speed = speed - 1

pause for 0.1 seconds

if speed < 0:

set pin D13 to LOW // Deactivate motor

set pin D12 to HIGH // Activate brake

speed = 0

print "Cruise mode: " + speed on the LCD display

// Adaptive Cruise Control Mode

elseif mode == 2:

clear the LCD display

pause for 0.5 seconds // Blinking effect

initialize the LCD display

set pin D13 to HIGH // Activate motor

set pin D12 to LOW // Deactivate brake

distance = readDistance(ul) // Read the distance from the ultrasonic sensor

if distance < 0.3:

speed = speed - 1

else:

speed = speed + 1

if speed > constant:

speed = constant

if speed < 0:

set pin D13 to LOW // Deactivate motor

set pin D12 to HIGH // Activate brake

speed = 0

print "Adap\_Cruise\_mode: " + speed on the LCD display

// End of modes

end of if statements

// Continue the loop for continuous monitoring and updating

end of while loop

## ***MATLAB Program***

% Clear all variables in the workspace

clc;

clear;

% Creation of variable for Arduino

ar = arduino('COM5','Uno','Libraries',{'Ultrasonic','ExampleLCD/LCDAddOn'},'ForceBuildOn',true);

% Initializes a connection with an Arduino board connected to COM5 port

% and specifies the libraries to be used (Ultrasonic and ExampleLCD/LCDAddOn)

% Creation of variable for Ultrasonic Sensor

ul = ultrasonic(ar,'D10','D8');

% Initializes an ultrasonic sensor connected to digital pins D10 (trigger)

% and D8 (echo) of the Arduino

lcd = addon(ar,"ExampleLCD/LCDAddOn",'RegisterSelectPin','D7','EnablePin','D6','DataPins',{'D5','D4','D3','D2'});

% Initializes an LCD display connected to the Arduino using specific pins

initializeLCD(lcd);

% Initializes the LCD display

clearLCD(lcd);

% Clears the content displayed on the LCD

printLCD(lcd, 'WELCOME TO');

printLCD(lcd, 'ACC PROJECT');

% Prints "WELCOME TO" on the first line of the LCD display

% Prints "ACC PROJECT" on the second line of the LCD display

pause(5);

% Pauses MATLAB execution for 5 seconds

clearLCD(lcd);

% Clears the content displayed on the LCD

printLCD(lcd, 'Group 32');

printLCD(lcd, 'Amey,Brano,Nandu');

% Prints "Group 32" on the first line of the LCD display

% Prints "Amey,Brano,Nandu" on the second line of the LCD display

pause(5);

% Pauses MATLAB execution for 5 seconds

% Declaration of Variables

speed = 0;

increase\_speed = 0;

decrease\_speed = 0;

cancel = 0;

set\_speed = 0;

adaptive\_cruise\_speed = 0;

distance = 0;

mode = 0;

while true

% Infinite loop to continuously monitor and update the speed

% Get inputs from user

increase\_speed = readVoltage(ar,'A0');

% Read the voltage value from analog input pin A0

decrease\_speed = readVoltage(ar,'A1');

% Read the voltage value from analog input pin A1

cancel = readVoltage(ar,'A2');

% Read the voltage value from analog input pin A2

set\_speed = readVoltage(ar,'A3');

% Read the voltage value from analog input pin A3

adaptive\_cruise\_speed = readVoltage(ar,'A4');

% Read the voltage value from analog input pin A4

% Get the input from the ultrasonic sensor as distance

distance = readDistance(ul);

if cancel >= 4

mode = 0;

% If the cancel button is pressed (voltage value >= 4),

% set mode to 0 (Normal Mode)

elseif set\_speed >= 4

mode = 1;

% If the set speed button is pressed (voltage value >= 4),

% set mode to 1 (Cruise Control Mode)

elseif adaptive\_cruise\_speed >= 4

mode = 2;

constant = speed;

% If the adaptive cruise control button is pressed (voltage value >= 4),

% set mode to 2 (Adaptive Cruise Control Mode) and store the current speed in 'constant'

end

% Normal Mode

if mode == 0

if increase\_speed >= 4

writeDigitalPin(ar, 'D13', 1);

writeDigitalPin(ar, 'D12', 0);

speed = speed + 1;

pause(0.1);

elseif decrease\_speed >= 4

speed = speed - 1;

pause(0.1);

else

speed = speed - 1;

pause(1.5);

end

if speed < 0

writeDigitalPin(ar, 'D13', 0);

writeDigitalPin(ar, 'D12', 1);

speed = 0;

end

printLCD(lcd,'Vehicle Speed: ');

% Prints "Vehicle Speed: " on the LCD display

printLCD(lcd,[strcat(num2str(speed))]);

% Prints the current speed on the LCD display

% Cruise Control Mode

elseif mode == 1

if increase\_speed >= 4

writeDigitalPin(ar, 'D13', 1);

writeDigitalPin(ar, 'D12', 0);

speed = speed + 1;

pause(0.1);

elseif decrease\_speed >= 4

speed = speed - 1;

pause(0.1)

end

if speed < 0

writeDigitalPin(ar, 'D13', 0);

writeDigitalPin(ar, 'D12', 1);

speed = 0;

end

printLCD(lcd,'Cruise mode: ');

% Prints "Cruise mode: " on the LCD display

printLCD(lcd,[strcat(num2str(speed))]);

% Prints the current speed on the LCD display

% Adaptive Cruise Control Mode

elseif mode == 2

clearLCD(lcd);

pause(0.5);

% Clears the LCD display and pauses for 0.5 seconds (blinking effect)

initializeLCD(lcd);

% Reinitializes the LCD display

writeDigitalPin(ar, 'D13', 1);

writeDigitalPin(ar, 'D12', 0);

if distance < 0.3

disp(distance);

speed = speed - 1;

else

disp(distance);

speed = speed + 1;

end

if speed > constant

speed = constant;

end

if speed < 0

writeDigitalPin(ar, 'D13', 0);

writeDigitalPin(ar, 'D12', 1);

speed = 0;

end

printLCD(lcd,'Adap\_Cruise\_mode');

% Prints "Adap\_Cruise\_mode" on the LCD display

printLCD(lcd,[strcat(num2str(speed))]);

% Prints the current speed on the LCD display

end

end

# **Timeline**

## ***Milestones***

The project timeline is outlined below:

**Table 2**:Milestones Completed

|  |  |  |
| --- | --- | --- |
| **Semester Weeks** | **Milestones Completed** | |
| Week 1-2 | | Project planning and research |
| Week 3 | | System modeling and simulation |
| Week 4 | | Hardware component acquisition |
| Week 5 | | Preliminary pseudocode |
| Week 6 | | Radar sensor integration |
| Week 7 | | Control algorithm development |
| Week 8 | | System integration and testing |
| Week 8-9 | | Fine-tuning and optimization |
| Week 9 | | System integration and testing |
| Week 10 | | Fine-tuning and optimization |
| Week 11 | | Final report and project demonstration |

## ***Gantt Chart***

A graph with blue squares

Description automatically generated

**Figure 12:** Gantt Chart

## ***Limitations and Risks***

The project involves five key limitations and risks that need to be considered.

1. *Hardware Availability:* Acquiring the necessary hardware components, such as the radar sensor or microcontroller, may pose challenges and potentially delay project progress. Efforts will be made to identify suitable alternatives or workarounds in such situations.
2. *Real-world Constraints:* The performance of the control system can be influenced by real-world factors, such as variations in weather conditions, road conditions, or vehicle dynamics [19]. Simulating these constraints accurately in MATLAB may present challenges and require additional calibration.
3. *System Complexity:* Developing an ACC system involves dealing with complex algorithms and the integration of multiple components. The inherent complexity of the system may lead to unexpected issues or difficulties during the implementation phase.
4. *Safety Considerations:* Ensuring the safety of the ACC system is of utmost importance. By testing and validation of the control algorithm and hardware integration are necessary to mitigate any potential risks associated with incorrect speed adjustments or unreliable distance measurements [20].
5. *Time Management:* Effective time management is critical for meeting project milestones. Delays in any phase of the project could impact subsequent tasks and the overall project timeline. Maintaining open communication and coordination among team members will be essential to mitigate this risk.

It is essential to proactively address these limitations and risks to ensure the successful completion of the project.

## ***Roles and Responsibilities***

**Table 3**: Roles and Responsibilities

|  |  |
| --- | --- |
| **Group Member** | **Roles and Responsibilities** |
| 1. Amey | * Research and analysis, * MATLAB code development, * documentation, * project coordination, * Report writing. |
| 1. Nandeshwar | * Hardware acquisition and integration, * MATLAB code development, * system testing, * troubleshooting. |
| 1. Brano | * Control algorithm development, * MATLAB simulation, * Tinkercad simulation, * System optimization. |

# **ACC Execution Procedure**

## ***Step 1: Gather Hardware Components***

Ensure that you have all the required hardware components for the project:

1. Arduino Uno
2. LCD Display
3. Ultrasonic Sensor
4. Pushbuttons (5 buttons)
5. Resistors (as required)
6. Battery and Battery Connector Cable
7. Jumper Wires
8. PCB or Breadboard
9. Potentiometer

## ***Step 2: Prepare Arduino Environment***

1. Install the required libraries for Ultrasonic and LCD display in the Arduino IDE.
2. Connect the Arduino Uno to your computer via USB.

## ***Step 3: Arduino Uno & LCD Connections***

The following table lists the pin connections between Arduino Uno and the LCD display:

**Table 4:** Arduino Uno & the LCD display Connections

|  |  |
| --- | --- |
| **LCD Pins** | **Arduino Uno Pins** |
| VSS (GND) | GND |
| VDD (5V) | 5V |
| VO (Contrast) | Potentiometer Pin (Adjust Contrast) |
| RS | D7 |
| RW (GND) | GND |
| E | D6 |
| D4 | D5 |
| D5 | D4 |
| D6 | D3 |
| D7 | D2 |
| A (LED+) | 5V |
| K (LED-) | GND |

## ***Step 4: Arduino Uno & Ultrasonic Sensor Connections***

The following table lists the pin connections between Arduino Uno and the Ultrasonic sensor:

**Table 5:** Arduino Uno & Ultrasonic Sensor Connections

|  |  |
| --- | --- |
| **Ultrasonic Sensor Pins** | **Arduino Uno Pins** |
| VCC (5V) | 5V |
| GND | GND |
| TRIG | D10 |
| ECHO | D8 |

## ***Step 5: Arduino Uno & Pushbuttons Connections***

The following table lists the pin connections between Arduino Uno and the pushbuttons:

**Table 6:** Arduino Uno & Pushbuttons Connections

|  |  |
| --- | --- |
| **Pushbutton** | **Arduino Uno Pins** |
| Increase Speed | A0 |
| Decrease Speed | A1 |
| Cancel | A2 |
| Set Speed | A3 |
| Adaptive Cruise Control | A4 |

## ***Step 6: Assemble the Circuit***

1. Connect the LCD display to the Arduino Uno as per the table in Step 3.
2. Connect the Ultrasonic sensor to the Arduino Uno as per the table in Step 4.
3. Connect the pushbuttons to the Arduino Uno as per the table in Step 5.
4. Use resistors where necessary to protect components and set up voltage dividers, following the design requirements.

## ***Step 7: MATLAB Program***

Copy and paste the MATLAB program provided in the report into the MATLAB environment. Make sure that the required libraries for Arduino communication are installed.

## ***Step 8: Upload the Code to Arduino Uno***

1. Select the appropriate board (Arduino Uno) and port in the Arduino IDE.
2. Click on "Upload" to transfer the code to the Arduino Uno.

## ***Step 9: Power Supply***

1. Connect the battery to the Arduino Uno through the battery connector cable.
2. Ensure that the power supply is stable and within the voltage range specified for the components.

## ***Step 10: Test and Calibration***

1. Power on the Arduino Uno and check if the LCD displays the necessary information (e.g., project name and group number).
2. Test the pushbuttons to verify that they are responsive and change the mode of operation as expected (e.g., normal mode, cruise control mode, adaptive cruise control mode).
3. Place an obstacle in front of the Ultrasonic sensor and check if the ACC system responds appropriately by adjusting the vehicle's speed.

## ***Step 11: Fine-Tuning and Troubleshooting***

1. Calibrate the potentiometer to adjust the LCD contrast for better readability.
2. Monitor the ACC system's behaviour in different scenarios and make any necessary adjustments to the control algorithm.

# **Tinkercad Software**

## ***Circuit View***

A circuit board with wires and a display

Description automatically generated

**Figure 13:** Circuit View

## ***Schematic View***

A diagram of a computer

Description automatically generated

**Figure 14:** Schematic View

## ***Component Table***

**Table 7:** Component Table

A screenshot of a computer

Description automatically generated

## ***Tinkercad Simulation***

A screen with text on it

Description automatically generated

**Figure 15:** Welcome message

A screen shot of a computer

Description automatically generated

**Figure 16:** Group Number & Names

A screen shot of a computer

Description automatically generated

**Figure 17:** Circuit at initial (zero speed)

A screen shot of a computer

Description automatically generated

**Figure 18:** Circuit in Cruise Mode (non-zero speed)

A screen shot of a computer

Description automatically generated

**Figure 19:** Circuit in Cruise Mode (zero speed)

A screen shot of a computer

Description automatically generated

**Figure 20:** Circuit in Adaptive Cruise Control Mode (no object in front of ultrasonic sensor)

A computer screen with a blue screen

Description automatically generated

**Figure 21:** Circuit in Adaptive Cruise Control Mode (an object in front of ultrasonic sensor)

# **Working Model**

## ***Circuit***

A circuit board with wires connected to it

Description automatically generated

**Figure 22:** Circuit connections

## ***Welcome Message***

A blue screen with wires and wires

Description automatically generated

**Figure 23:** Welcome message

## ***Group Number & Names***

A blue screen with wires and wires

Description automatically generated

**Figure 24:** Group Number & Names

## ***Circuit at initial (zero speed)***

A blue screen with white text and wires

Description automatically generated

**Figure 25:** Circuit at initial (zero speed)

## ***Circuit in Cruise Mode (non-zero speed)***

A circuit board with wires and a display

Description automatically generated

**Figure 26:** Circuit in Cruise Mode (non-zero speed)

## ***Circuit in Cruise Mode (zero speed)***

A circuit board with wires and a display

Description automatically generated

**Figure 27:** Circuit in Cruise Mode (zero speed)

## ***Circuit in Adaptive Cruise Control Mode (no object in front of ultrasonic sensor)***

A electronic board with wires and a display

Description automatically generated

**Figure 28:** Circuit in Adaptive Cruise Control Mode (no object in front of ultrasonic sensor)

## ***Circuit in Adaptive Cruise Control Mode (an object in front of ultrasonic sensor)***

A hand holding a device with wires

Description automatically generated

**Figure 29:** Circuit in Adaptive Cruise Control Mode (an object in front of ultrasonic sensor)

# **Testing Scenarios and Output Results**

## ***Normal Mode***

* Input: Increase speed button pressed
* Expected Output: The vehicle speed increases by 1 unit per press.
* Actual Output: The vehicle speed increases by 1 unit per press as expected.
* Input: Decrease speed button pressed
* Expected Output: The vehicle speed decreases by 1 unit per press.
* Actual Output: The vehicle speed decreases by 1 unit per press as expected.

## ***Cruise Control Mode***

* Input: Increase speed button pressed
* Expected Output: The vehicle speed increases by 1 unit per press, maintaining the set speed afterward.
* Actual Output: The vehicle speed increases by 1 unit per press and maintains the set speed as expected.
* Input: Decrease speed button pressed
* Expected Output: The vehicle speed decreases by 1 unit per press, maintaining the set speed afterward.
* Actual Output: The vehicle speed decreases by 1 unit per press and maintains the set speed as expected.

## ***Adaptive Cruise Control Mode***

* Input: Increase speed button pressed (preceding vehicle moving away)
* Expected Output: The vehicle increases its speed to adapt to the distance from the preceding vehicle.
* Actual Output: The vehicle smoothly increases its speed to maintain a safe following distance.
* Input: Decrease speed button pressed (preceding vehicle moving closer)
* Expected Output: The vehicle decreases its speed to maintain a safe following distance.
* Actual Output: The vehicle promptly decreases its speed to ensure a safe following distance.
* Input: Set speed button pressed (activating Adaptive Cruise Control Mode)
* Expected Output: The ACC system sets the current speed as a constant reference speed for the adaptive cruise control.
* Actual Output: The ACC system successfully stores the current speed as a constant reference speed.

## ***Real-World Constraints***

* Input: Testing the ACC system in varying weather conditions (e.g., rain, fog)
* Expected Output: The ACC system should adapt to the changing conditions and maintain safe driving practices.
* Actual Output: The ACC system effectively adjusted to varying weather conditions and maintained safe driving.
* Input: Testing the ACC system on different road surfaces (e.g., smooth, bumpy)
* Expected Output: The ACC system should adapt to different road surfaces and maintain stability.
* Actual Output: The ACC system demonstrated adaptability to various road surfaces and ensured stable driving.

## ***Safety Validation***

* Input: Simulating a sudden obstacle in front of the vehicle
* Expected Output: The ACC system should immediately respond, reducing the vehicle speed to avoid collision.
* Actual Output: The ACC system promptly responded to the obstacle, reducing the vehicle speed, and preventing collision.

Overall, the testing scenarios demonstrated the effectiveness and robustness of the ACC system implemented using MATLAB and Arduino Uno. The system performed as expected in different modes, adjusting the vehicle speed accurately based on inputs and sensor measurements. The ACC system showcased adaptability to real-world constraints and prioritized safety in various scenarios. The successful testing outcomes validate the functionality and potential of the ACC system in enhancing driving safety and convenience.

# **Lessons Learned**

1. Efficient Code Writing:
   * Writing efficient MATLAB code is essential for optimal performance.
   * Utilize vectorization and built-in functions to improve code speed.
2. Modular Programming:

* Break complex tasks into smaller, manageable functions.
* Modular programming enhances code readability and reusability.

1. Debugging Techniques:
   * Master MATLAB's debugging tools to identify and fix errors.
   * Use breakpoints and the MATLAB debugger to troubleshoot issues.
2. Optimizing Algorithms:
   * Optimize algorithms to reduce execution time and memory usage.
   * Profile code to identify bottlenecks and enhance efficiency.
3. Effective Visualization:
   * MATLAB's powerful visualization capabilities are valuable for data analysis.
   * Create visually appealing plots and graphs to communicate results effectively.
4. Utilizing MATLAB Toolboxes:
   * Explore and leverage MATLAB's extensive toolboxes for specialized tasks.
   * Toolboxes provide pre-built functions for various applications.
5. Documentation and Comments:

* Document code thoroughly and use comments to explain complex sections.
* Well-documented code facilitates collaboration and future maintenance.

1. Version Control:
   * Implement version control using tools like Git for code management.
   * Version control helps track changes and collaborate with team members.
2. MATLAB Environment Management:
   * Use MATLAB environments effectively to manage workspace and variables.
   * Organize scripts and functions in project folders for better organization.
3. Integration with Hardware:

* MATLAB's support for hardware integration simplifies interfacing with external devices.
* Utilize MATLAB's Hardware Support Package for seamless hardware communication.

1. Continuous Learning:

* MATLAB offers a vast array of features and updates.
* Continuously explore new features and stay updated with MATLAB advancements.

# **Conclusion**

The project successfully demonstrated the development of an ACC system using MATLAB and Arduino Uno. Through efficient code writing, hardware integration, and effective visualization, the ACC system was able to automatically adjust the vehicle's speed and maintain a safe distance from the preceding vehicle. The project provided valuable insights into the capabilities of MATLAB, Arduino Uno, and their integration in creating advanced driver assistance systems like ACC. It highlights the significance of continuous learning and exploration of MATLAB's features for future advancements in automotive technology and safety.

# **References**

|  |  |
| --- | --- |
| [1] | G. Marsden, M. McDonald, and M. Brackstone, “Towards an understanding of adaptive cruise control,” *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 33–51, Feb. 2001, https://doi.org/10.1016/S0968-090X(00)00022-X [accessed Jun. 2, 2023]. |
| [2] | F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, “A systematic review of perception system and simulators for Autonomous Vehicles Research,” *Sensors*, vol. 19, no. 3, p. 648, 2019, https://doi.org/10.3390/s19030648 [accessed Jun. 2, 2023]. |
| [3] | Y. Li *et al.*, “Evaluation of the impacts of cooperative adaptive cruise control on reducing rear-end collision risks on freeways,” *Accident Analysis &amp; Prevention*, vol. 98, pp. 87–95, Jan. 2017, https://doi.org/10.1016/j.aap.2016.09.015 [accessed Jun. 2, 2023]. |
| [4] | Synopsys, “What is ADAS (advanced driver assistance systems)? – overview of Adas Applications,” https://www.synopsys.com/automotive/what-is-adas.html [accessed Jun. 2, 2023]. |
| [5] | S. Naylor, “What is ACC (Adaptive Cruise Control)?,” *Parkers,* https://www.parkers.co.uk/what-is/acc-adaptive-cruise-control [accessed Jun. 2, 2023]. |
| [6] | J. Lee, D. McGehee, T. Brown, and D. Marshall, “Effects of adaptive cruise control and alert modality on driver performance,” *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1980, no. 1, pp. 49–56, Jan. 2006, https://doi.org/10.1177/0361198106198000108 [accessed Jun. 2, 2023]. |
| [7] | Arduino, “Arduino documentation,” https://docs.arduino.cc [accessed Jun. 5, 2023]. |
| [8] | D. Kumar, et al., “Interfacing LCD with 8051,” *Embedded and Robotics,* https://yadavdharm.wordpress.com/2019/03/08/interfacing-lcd-with-8051 [accessed Jun. 5, 2023] |
| [9] | Arduino France, “Arduino Uno: Advantages, Disadvantages, Use and Operation,” https://www.arduino-france.com/review/arduino-uno [accessed Jun. 5, 2023]. |
| [10] | OSEPP, “Push button module,” https://www.osepp.com/electronic-modules/sensor-modules/76-push-button-module [accessed Jun. 5, 2023]. |
| [11] | Robo India, “Digital input -how to use the button with Arduino,” https://roboindia.com/tutorials/digital-input-how-to-use-the-button-with-arduino [accessed Jun. 5, 2023]. |
| [12] | R. L. Pendergast et al., “Complete Guide for Ultrasonic sensor HC-SR04 with Arduino,” *Random Nerd Tutorials,* https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04 [accessed Jun. 5, 2023]. |
| [13] | arduino2go, “Appendix A: Reading resistor codes,” *Arduino to Go,* https://arduinotogo.com/2017/03/10/appendix-a-reading-resistor-codes [accessed Jun. 5, 2023]. |
| [14] | Electronics Infra, “Jumper wire 0.5mm - electronics infra,” https://electronicsinfra.com/product/jumper-wire-0-5mm [accessed Jun. 5, 2023]. |
| [15] | arduino2go, “Chapter 2: Building a circuit step by step,” *Arduino to Go,* https://arduinotogo.com/2016/08/22/chapter-2-building-a-circuit-step-by-step [accessed Jun. 5, 2023]. |
| [16] | Raspberry Pi, “Getting started with raspberry pi,” *Raspberry Pi Foundation,* https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started [accessed Jun. 5, 2023]. |
| [17] | Simulink, “Simulink Documentation,” https://www.mathworks.com/help/simulink [accessed Jun. 12, 2023]. |
| [18] | Matlab, “MATLAB Documentation,” https://www.mathworks.com/help/matlab [accessed Jun. 12, 2023]. |
| [19] | Team-BHP “Adaptive Cruise Control Limitations,”, https://www.team-bhp.com/forum/indian-car-scene [accessed Jun. 15, 2023]. |
| [20] | B. D. Seppelt and J. D. Lee, “Making Adaptive Cruise Control (ACC) limits visible,” *International Journal of Human-Computer Studies*, vol. 65, no. 3, pp. 192–205, Mar. 2007, https://doi.org/10.1016/j.ijhcs.2006.10.001 [accessed Jun. 15, 2023]. |