Computational Methods and Modeling for Engineering Applications
GENG 8030-2 (4093)

University of Windsor

Project Report
on

"Implementation of Adaptive Cruise Control (ACC) System using Simulink on Arduino Uno Board"

Team Members

Jiten Rajendra Pandit……..105170936
Phani Tejaswini Karnati…….105192020
Shyam Lal Lakshmiramiya Sridhar ……...110005786

Submission Date: 25-Mar-2020

Instructor:
Roozbeh Razavi-Far

Table of Contents

List of Figures

Abbreviations

ACC – Adaptive Cruise Control
LCD – Liquid Crystal Display
GND – Ground
LED – Light Emitting Diode
RS – Register Select

1. Abstract

This paper accomplishes the function of adaptive cruise control or intelligent cruise control which is by sensing the distance of car running in front and adapt its cruise speed to maintain a safe distance from the vehicle ahead. This purpose is exhibited by the software and hardware combination of MATLAB Simulink and Arduino considering with the five important features like set speed, adaptive speed cancel, increase speed and decrease speed. The very important factors to improve the traffic flow and for safe and smart driving are constant time gap and proper safe distance which should be maintained correctly, and this purpose is solved by this project. This paper gives the information of adjusting a time interval, maximum-minimum speed and distance between two vehicles.

2. Introduction

Cruise Control is a feature that is present in almost all cars nowadays where the driver sets a travelling speed so that they can take their foot off the gas pedal, which is relaxing for long-distance travel when there are fewer cars on the road. But when the road is busy or when travelling in an urban area with frequent traffic signals, the purpose of cruise control is lost. That is when the Adaptive Cruise Control (ACC) comes handy for the drivers.

Figure 1 shows the working of the ACC system. The ACC can automatically adjust the speed of the car to match with the speed of the car in front. When the car ahead slows down, it can automatically match it and once the car in front accelerates or moves to a different lane, the speed of the car increases to the set speed.

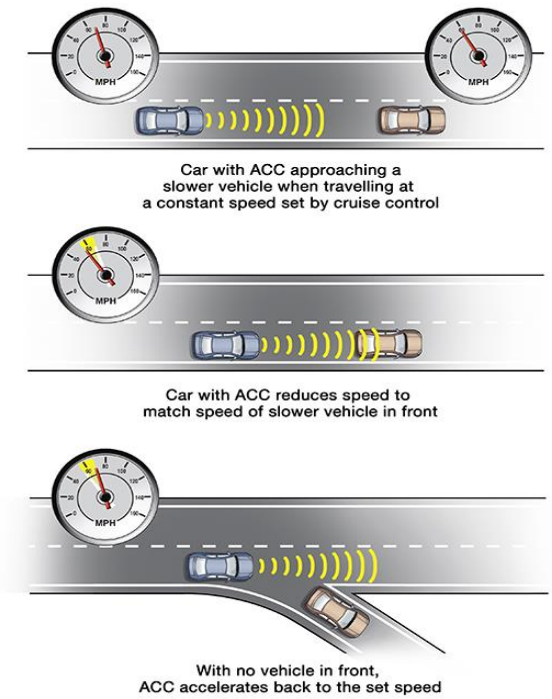

Figure 1: Working of ACC [1]

3. Description

There are 3 main parts in the project. They are as follows:

1. **Increase and Decrease the speed** of the vehicle using respective buttons.
2. Activate **cruise control mode** where a constant speed is set, but the vehicle speed is still adjustable. The mode can be quit by using a cancel button.
3. Activate **adaptive cruise control mode** where a constant speed is set and using an ultrasonic sensor, the distance between the vehicles is measured, and based on this value the vehicle speed is automatically adjusted. Here the increase and decrease speed buttons don't work. The mode can be quit by using a cancel button.

4. Development Procedure

To make our project a success, we have analyzed a series of work that needs to be completed. The development work needs to be monitored stepwise and therefore a thorough examination is mandatory. Successful implementation of the project development can be categorized into 4 project development phases.

1. Initialization

The first part of developing a successful project is to make sure that you are entirely sure of everything that needs to be done and that you have a clear objective and title for your project. So, we have figured out a rough idea of the project by reading different research papers related to the topic.

2. Resources Gathering

After having suitable knowledge related to the project, the resource gathering stage comes into action. In this stage, we have collected the suitable resources which would be useful in the project, for example,

Software: MATLAB, Simulink.

Hardware: Wires, Display Unit, Arduino Uno, Ultrasonic Sensor and many more.

3. Implementation

This stage is very crucial and challenging as the whole project is based on this stage. We face many challenges in term of changing double data type to string. Furthermore, we face difficulty in storing the value in memory block and etc. But, finally we have implement the project with proper coding and circuit design.

4. Testing

This is the final stage for the successful project. In this phase, we have test our code with the different test cases and there were small issues, we had solved those issue.

5. Summary of Work

- As our project is to integrate Arduino with MATLAB, we have learned about the basic language which is used in Arduino from online websites and how to download the Arduino support package for MATLAB from online tutorials.
- Group members discussed the flow of the project and split the work and set the timetable for achieving the objective.
- We ordered the Arduino kit and tested some basic projects such as blinking an LED, displaying "Hello World" on the LCD, switching LED ON and OFF with the help of push-button, etc. to get familiar with using Arduino Kit.
- We have worked on developing the logic for building the codes. The logic was developed in the form of a flow chart as it can reduce the time and error during code development.
- We have used Increase Speed, Decrease Speed, Set Speed, Adaptive Mode and Cruise Mode, Cancel buttons to perform various operations like increasing and decreasing the speed of the vehicle, setting a particular speed to be maintained and putting the vehicle in either adaptive or cruise mode or cancel any mode.
- We have used different variables such as **Adaptive flag**, **Current Speed**, **Set Speed**, **Cancel Flag** and **Cruise Flag** which are given to Adaptive Cruise Control function.
- And we used **blink State** and **Adaptive Flag** to give as input to blink function to make the LCD blink. Since LCD accepts only ASCII values, we have given ASCII values of the text to be printed on the LCD.
- And we used switch case to include all the cases in which we are going to print the speed of the vehicle using ASCII values.
- We used SIMULINK to connect all these variables and functions using various blocks available in it and MATLAB to implement the programming logic.
- Finally, we deployed our code into the Arduino after giving all the required connections.
- We have checked all the required conditions such as increasing, decreasing and setting the speed and also setting adaptive and cruise mode using Ultrasonic Sensor.

6. Components Used

Following components are used in the Adaptive Cruise Control.

6.1 Hardware Components

- Arduino Uno Board
- LCD screen
- Breadboard and Resistors
- Push Buttons
- Ultrasonic Sensor
- Wires

6.1.1 Arduino Uno Board

Arduino is an electronic device which is basically used for implementing for hardware and software. Arduino board are basically used to read inputs such as light on a sensor, activation of motor, turning LED light and many more. Arduino board cost is quite lower as compared to the other electronic devices. The Arduino software runs on Windows, Linux and many others operating system [1].

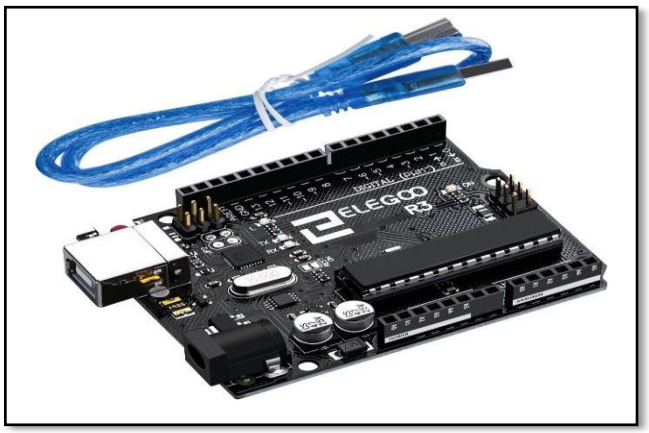

Figure 2: Arduino Uno Board [2]

- **General Pin Functions**

LED: There is a built-in LED driven by digital pin 13. When the pin is high value, the LED is on, when the pin is low, it is off.

Vin: The input voltage to the Arduino/Genuino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.

5V: This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 20V), the USB connector (5V), or the VIN pin of the board (7-20V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board.

3V3: A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND: Ground pins.

I/Oref: This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source, or enable voltage translators on the outputs to work with the 5V or 3.3V.

Reset: Typically used to add a reset button to shields that block the one on the board.

6.1.2 LCD Screen

The full form of LCD is Liquid Crystal Display. It has a 16-pin interface which is parallel. There are several pins such as register select (RS) pin, read/write (R/W) pin, enable pin, 8 data pins, power supply pin (Gnd and +5V) and LED backlight pin. The main working of the LCD screen is to display the data which create the image of what the data is display in the register [2]

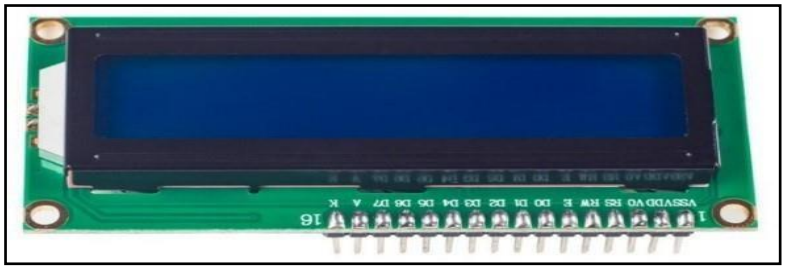

Figure 3: LCD screen [3]

Pin Configuration

Pin 1: Ground (0V)
Pin 2: Supply voltage; 5V (4.7V – 5.3V) [Vcc]
Pin 3: Contrast adjustment; the best way is to use a variable resistor such as a potentiometer. The output of the potentiometer is connected to this pin. Rotate the potentiometer knob forward and backwards to adjust the LCD contrast [Vo / VEE].
Pin 4: Selects command register when low, and data register when high RS
Pin 5: Low to write to the register; High to read from the register Read/write
Pin 6: Enable
Pin 7-Pin 14: 8-bit data pins DB0 - DB7
Pin 15: Backlight VCC (5V) [Led+]
Pin 16: Backlight Ground (0V) [Led-]

6.1.3 Breadboard and Resistors

Breadboard is a solder-less component which is basically used for the application of testing the circuit design [4]. The breadboard consists of spring clip contacts which is generally arranged in matrices with several blocks of clips connected together. The components such as resistors, jump wires, capacitors, transistors, LED and many other components are plugged in the pin to design the various circuits [5]. The figure shown below is the typical breadboard.

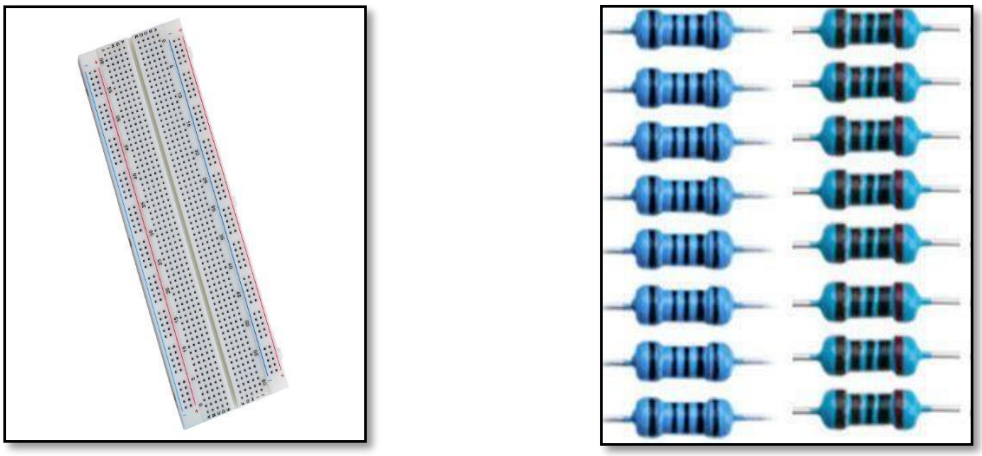

Figure 4: Breadboard and Resistors [6] [7]

6.1.4 Pushbutton

The basic principle of pushbutton is to join two points in the circuit when the button is pressed. The three wires are connected in the Arduino board. When the pushbutton is not pressed at that time there will be no connection in between the two legs of pushbutton, due to which the pin is connected to +5V and it is read as “High”. On the other hand, when the pushbutton is pressed at that time it makes connection between the two legs and the pin gets connected to ground and it is read as “Low” [8].

Figure 5: Pushbutton [8]

6.1.5 Ultrasonic Sensor

Ultrasonic sensor uses SONAR to determine the distance of an object just like the bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package from 2 cm to 400 cm or 1” to 13 feet.

Figure 6: Ultrasonic Sensor [9]

6.2 Software Components

- MATLAB
- SIMULINK

6.2.1 MATLAB

We can use MATLAB to create applications and models, develop algorithms, analyze data, etc. MATLAB is provided with the language, built-in math functions, applications that enable us to explore various approaches to get a solution quickly. MATLAB uses MATLAB language, a matrix-based language in which the most natural expression of computational mathematics is allowed.

6.2.2 Simulink

A MATLAB-based graphical programming environment that is used for modeling, simulating and analyzing multi-domain dynamical systems is called Simulink. A graphical block diagramming tool and a customizable set of block libraries is the primary interface of Simulink. A tight integration with the rest of the MATLAB environment can be offered by Simulink. It can be either scripted from MATLAB or can drive MATLAB.

7. Flowchart

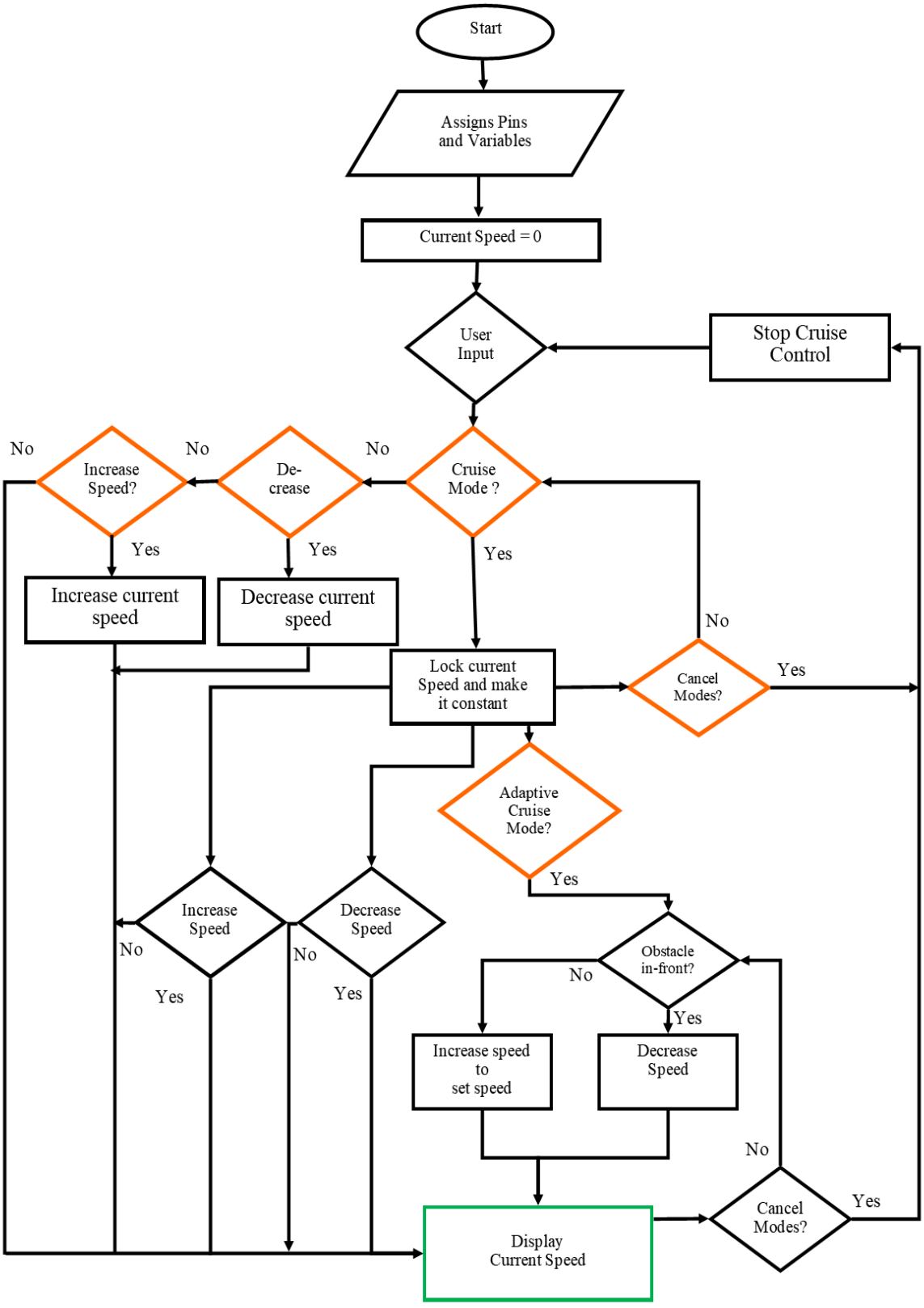

Figure 7: Flowchart of Adaptive Cruise Control

8. ACC Circuit Diagram

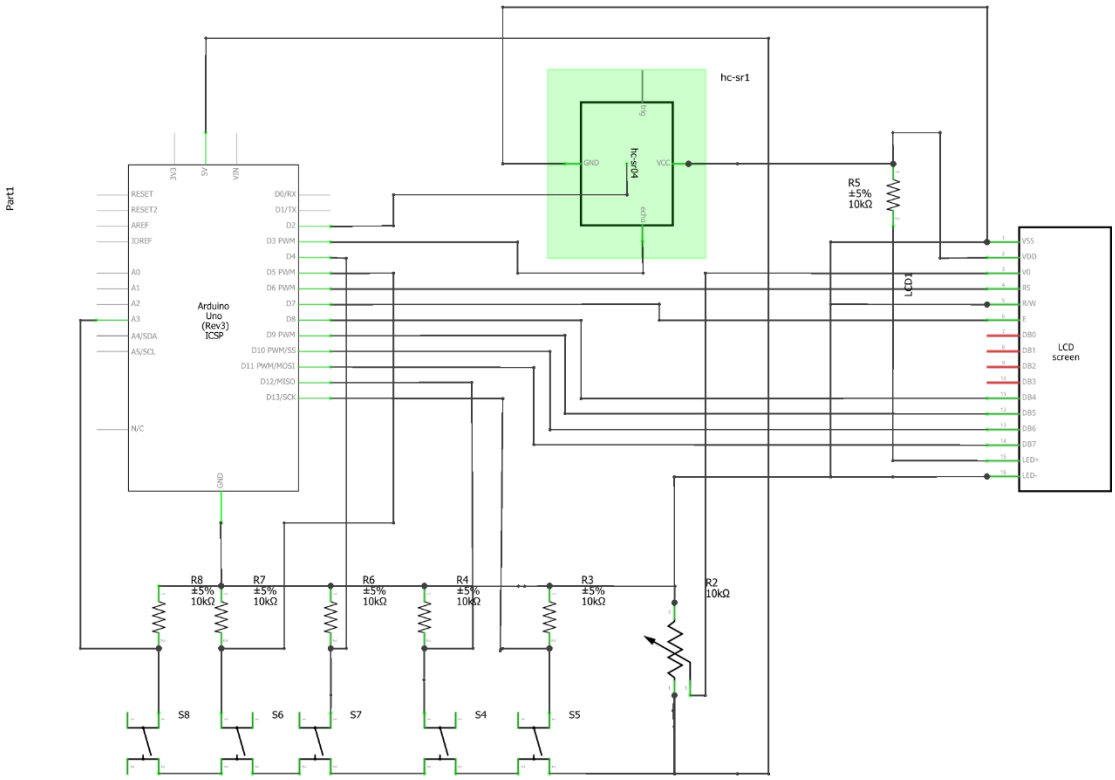

Figure 8: Schematic diagram of Adaptive Cruise Control

9. ACC Connection Diagram

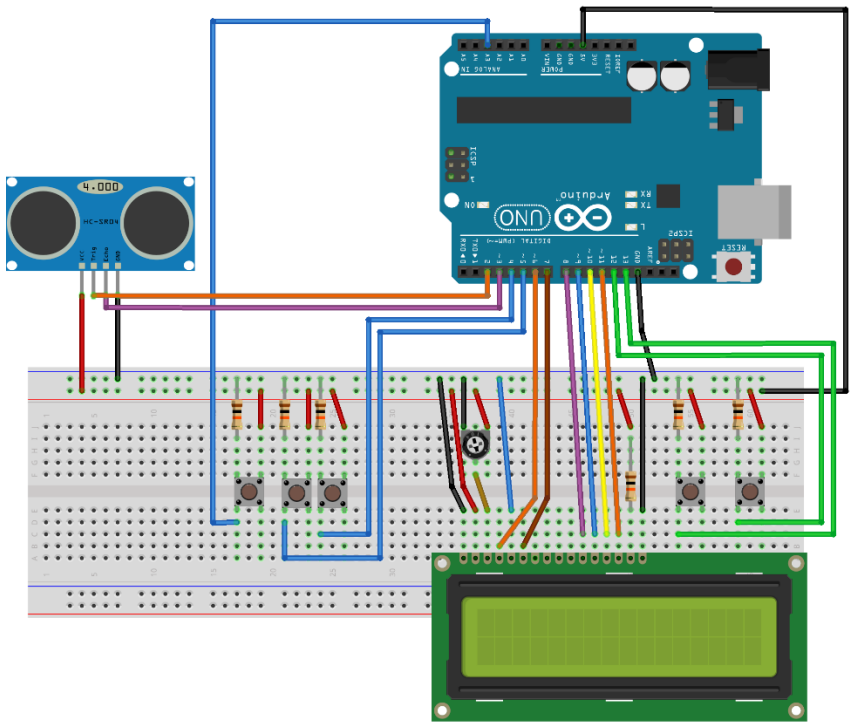

Figure 9: Breadboard Connection diagram of Adaptive Cruise Control

10. SIMULINK Block Diagram

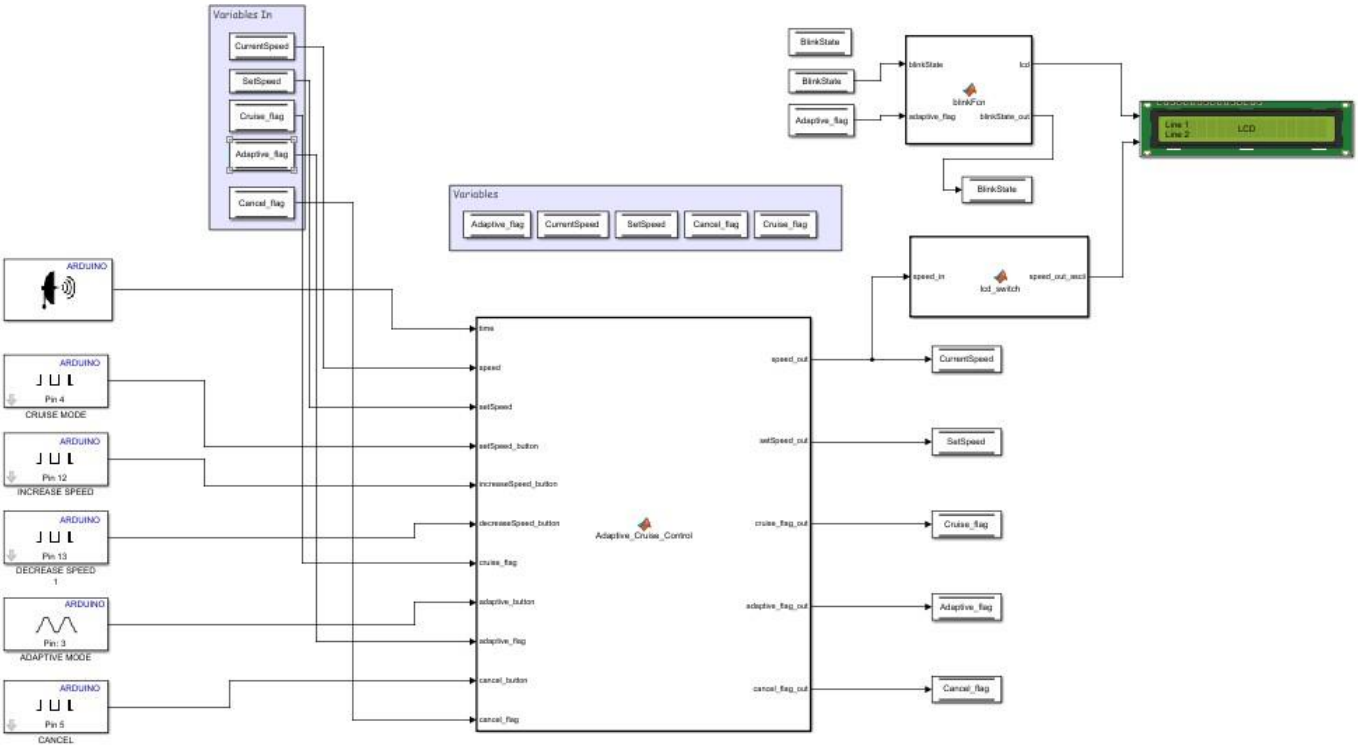

Figure 10: Simulink Block Diagram of Adaptive Cruise Control

11. Working of Adaptive Cruise Control

When we press the **Increase Speed** button, the speed will get increased. And if we press the **Decrease Speed** button, the speed will get decreased. If we are not pressing **Set Speed** button, the speed won't remain constant and decreases over time.

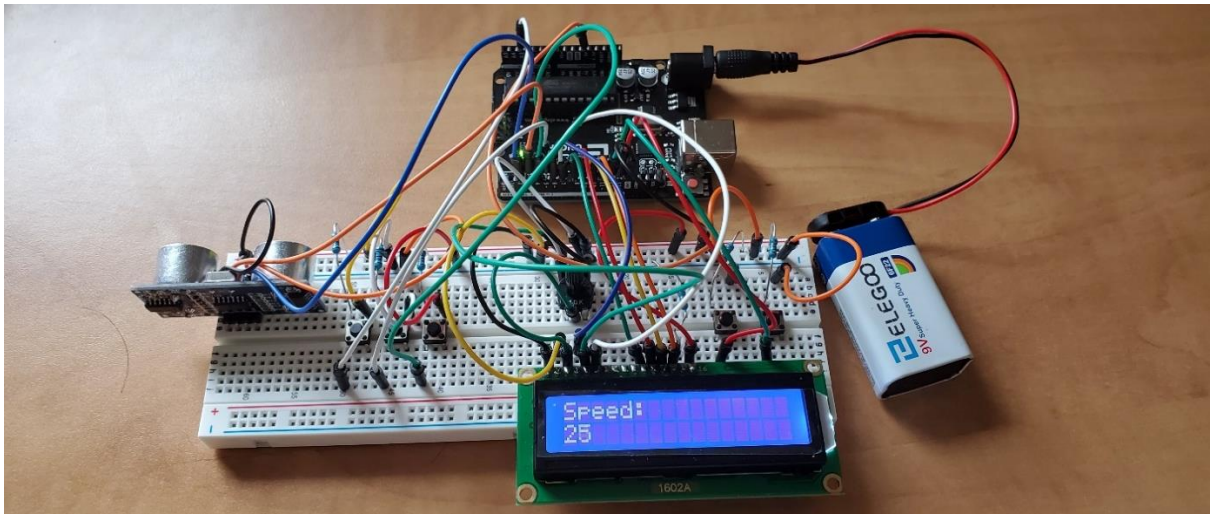

Figure 11: Car in Normal Mode

When we press the **Set speed** button the system will enter the cruise control mode where the speed is held constant. In this mode, the **Increase speed** button and the **Decrease speed** button are still functional and can be used to change the **Set speed**. If we press the **Cancel** button, the system quits the cruise control mode where the speed decreases slowly.

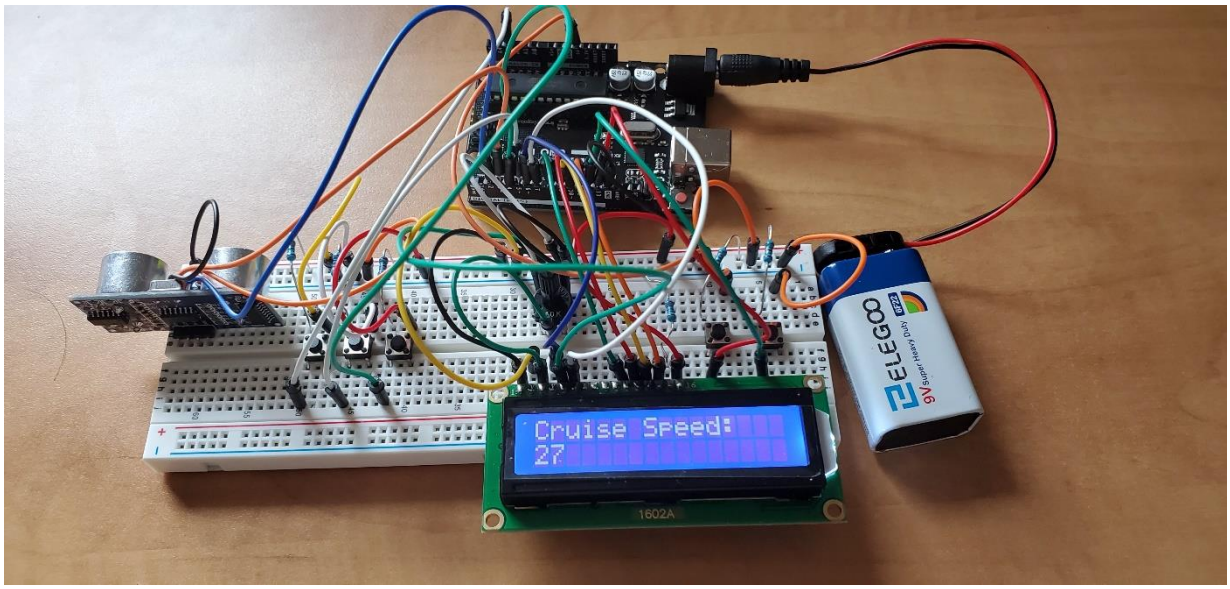

Figure 12: Car in Cruise Mode

When we press the **Adaptive speed** button, the speed is set and held constant until a vehicle shows up at the front or an object is detected where the speed decreases automatically. When the road becomes clear, the speed increases to reach the set speed again. In the adaptive cruise control mode, the display must blink to differentiate this mode from the cruise mode. In this mode, the **Increase speed** button and the **Decrease speed** button does not function, but the Cancel button can still be used to quit the adaptive cruise mode. If we press the Cancel button, the display stops blinking and the vehicle speed begins to slow down.

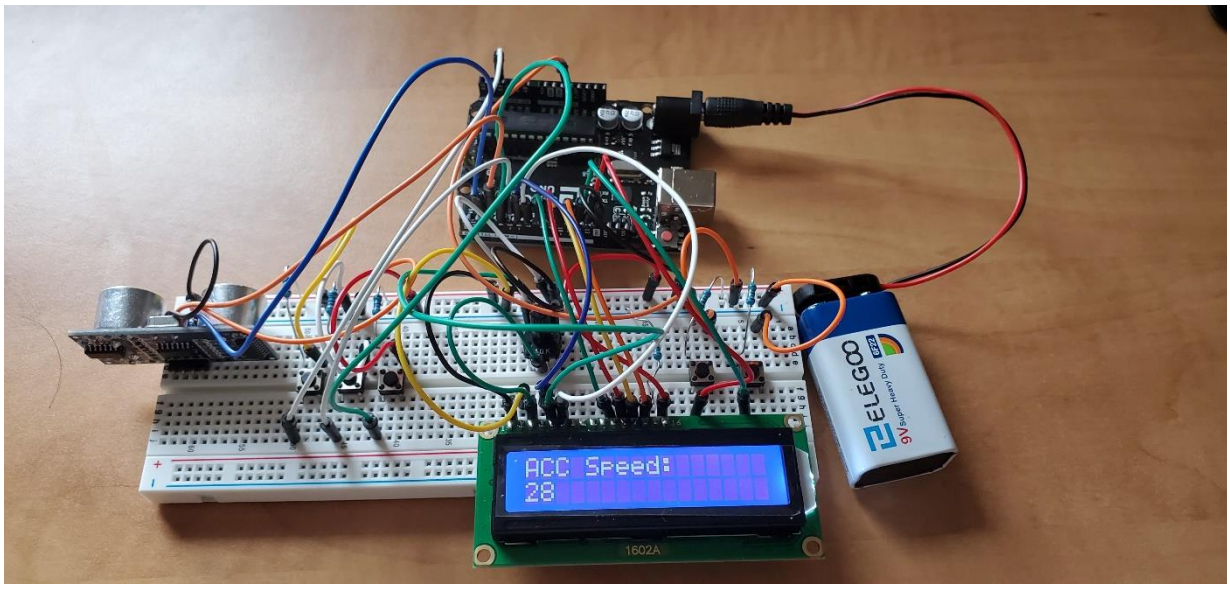

Figure 13: Car in Adaptive Cruise Control Mode

12. MATLAB program for Adaptive Cruise Control

Adaptive_Cruise_Control:

```
% Program Adaptive_Cruise_Control.m:
% Adjusts the speed of a car based on the obstacle in front along with
% normal cruise mode(maintaining a constant speed)
% Created on Feb 22, 2020 by SHYAM LAL LAKSHMIRAMIYA SRIDHAR, JITEN
% RAJENDRA PANDIT,PHANI TEJASWINI KARNATI

% Input Variables:
% time = time taken for the ultrasonic waves to reach the receiver end of
% the ultrasonic sensor.
% speed = current speed of the vehicle.
% setSpeed = locked speed of the vehicle in cruise mode.
% setSpeed_button = input from cruise mode button.
% increaseSpeed_button = input from increase speed button.
% decreaseSpeed_button = input from decrease speed button.
% cruise_flag = flag to represent the vehicle in cruise mode.
% adaptive_button = input from the adaptive cruise button.
% adaptive_flag = flag to represent the vehicle in Adaptive Cruise mode.
% cancel_button = input from the cancel mode button
% cancel_flag = flag to represent the vehicle is not in any driving mode.

% Output Variables:
% speed_out = updated current speed of the vehicle.
% setSpeed_out = updated locked speed of the vehicle in cruise mode.
% cruise_flag_out = updated flag to represent the vehicle in cruise mode.
% adaptive_flag_out = updated flag to represent the vehicle in Adaptive
Cruise mode.
% cancel_flag_out = updated flag to represent the vehicle is not in any
driving mode.

function [speed_out, setSpeed_out,cruise_flag_out, adaptive_flag_out,
cancel_flag_out] =
Adaptive_Cruise_Control(time,speed,setSpeed,setSpeed_button,increaseSpeed_bu
tton,
decreaseSpeed_button,cruise_flag,adaptive_button,adaptive_flag,cancel_button
,cancel_flag)
    % Calculating the distance of the obstacle in front of the ultrasonic
sensor.
    distance=(time*343)/2;

    % Setting flags for different driving modes.
    if cancel_button % Checking cancel button input and resetting all other
modes flags.
        cancel_flag_out=1;
        adaptive_flag_out=0;
        setSpeed_out=setSpeed;
        cruise_flag_out=0;
    elseif adaptive_button>600 % Checking adaptive button input and setting
the adaptive cruise mode flags.
        setSpeed_out=speed;
        adaptive_flag_out=1;
        cancel_flag_out=0;
        cruise_flag_out=0;
```

```
    elseif setSpeed_button % Checking set button input and setting the
normal cruise mode flags.
        setSpeed_out=speed; % Locking the current speed
        cruise_flag_out=1;
        cancel_flag_out=cancel_flag;
        adaptive_flag_out=0;
    else % Retaining flag values when not in any mode.
        setSpeed_out=setSpeed;
        cruise_flag_out=cruise_flag;
        adaptive_flag_out=adaptive_flag;
        cancel_flag_out=cancel_flag;
    end

    if increaseSpeed_button && ~adaptive_flag
        speed_out = speed+1; % Increasing Speed
    elseif decreaseSpeed_button && speed>0 && ~adaptive_flag
        speed_out = speed-1; % Decreasing Speed
    elseif cruise_flag % Cruise Mode
        if increaseSpeed_button && ~adaptive_flag % increase speed and lock.
            speed_out = speed+1;
            setSpeed_out = speed;
        elseif decreaseSpeed_button && speed>0 && ~adaptive_flag % decrease
speed and lock.
            speed_out = speed-1;
            setSpeed_out = speed;
        else
            speed_out = speed;
            setSpeed_out = setSpeed;
        end
    elseif adaptive_flag % Adaptive Cruise Mode
        if (distance<20) && (speed>0)
            speed_out=speed-2; % Decreasing speed by 2 if the obstacle is
very near to the car.
        elseif (distance>=20) && (distance<30) && (speed>0)
            speed_out=speed-1; % Decreasing speed by 1 if the obstacle is
near to the car.
        else % Bringing the speed to the set speed if no obstacle is found.
            if setSpeed<1
                speed_out=speed;
            else
                if speed<setSpeed
                    if distance<=30
                        speed_out=speed;
                    else
                        speed_out=speed+1;
                    end
                elseif speed>setSpeed && speed>0
                    speed_out=speed-1;
                else
                    speed_out=speed;
                end
            end
        end
    elseif cancel_flag || cancel_flag_out % Bring the current speed to 0.
        if speed>0
            speed_out=speed-1;
        else
            speed_out=speed;
            cancel_flag_out=0;
        end
    else % Bring the current speed to 0 if not in any mode.
```

```
        if speed>0
            speed_out=speed-1;
        else
            speed_out=speed;
        end
    end
end
```

Blink_Function:

```
% Program blinkFcn.m:
% Blink the LCD Display for differentiating ACC Mode.
% Created on Feb 22, 2020 by SHYAM LAL LAKSHMIRAMIYA SRIDHAR, JITEN
% RAJENDRA PANDIT,PHANI TEJASWINI KARNATI

% Input Variables:
% blinkState = flag to represent the blinking toggle value.
% adaptive_flag = flag to represent the vehicle in Adaptive Cruise mode.

% Output Variables:
% lcd = input to the Line 2 of LCD Display.
% blinkState_out = update flag to represent the blinking toggle value.
function [lcd,blinkState_out]  =
blinkFcn(blinkState,adaptive_flag,cruise_flag)

    if blinkState==0 && adaptive_flag % display blank screen in ACC mode.

lcd=[[32],[32],[32],[32],[32],[32],[32],[32],[32],[32],[32],[32],[32],[32]];
        blinkState_out=1;
    elseif blinkState==1 && adaptive_flag % display current speed in ACC
mode.

lcd=[[65],[67],[67],[32],[83],[112],[101],[101],[100],[58],[32],[32],[32]];
        blinkState_out=0;
    elseif cruise_flag % display current speed in Cruise Control mode.

lcd=[[67],[114],[117],[105],[115],[101],[32],[83],[112],[101],[101],[100],[5
8]];
        blinkState_out=blinkState;
    else % display current speed in other modes/ no mode.

lcd=[[83],[112],[101],[101],[100],[58],[32],[32],[32],[32],[32],[32],[32]];
        blinkState_out=blinkState;
    end
end
```

LCD_Switch:

```
% Input Variables:
% speed_in = Integer value of current Speed.

% Output Variables:
% speed_out_ascii = corresponding speed value in ASCII format.


function speed_out_ascii = lcd_switch(speed_in)
switch speed_in
    case 0
```

```
        speed_out_ascii = [[48],[48]];
    case 1
        speed_out_ascii = [[48],[49]];
    case 2
        speed_out_ascii = [[48],[50]];
    case 3
        speed_out_ascii = [[48],[51]];
    case 4
        speed_out_ascii = [[48],[52]];
    case 5
        speed_out_ascii = [[48],[53]];
    case 6
        speed_out_ascii = [[48],[54]];
    case 7
        speed_out_ascii = [[48],[55]];
    case 8
        speed_out_ascii = [[48],[56]];
    case 9
        speed_out_ascii = [[48],[57]];
    case 10
        speed_out_ascii = [[49],[48]];
    case 11
        speed_out_ascii = [[49],[49]];
    case 12
        speed_out_ascii = [[49],[50]];
    case 13
        speed_out_ascii = [[49],[51]];
    case 14
        speed_out_ascii = [[49],[52]];
    case 15
        speed_out_ascii = [[49],[53]];
    case 16
        speed_out_ascii = [[49],[54]];
    case 17
        speed_out_ascii = [[49],[55]];
    case 18
        speed_out_ascii = [[49],[56]];

      .
      .
      .
      .
      .
      .
      .
      .
      .
      .
    case 98
        speed_out_ascii = [[57],[56]];
    case 99
        speed_out_ascii = [[57],[57]];
    otherwise
        speed_out_ascii = [[48],[48]];
end
```

13. Conclusion

Adaptive Cruise Control is the Level 1 in autonomous driving system. Though there are various sensors like the RADAR, LIDAR, GPS and SONAR, that can be used in an autonomous driving system, use of Ultrasonic sensor is still a good choice considering the scope of the project.

14. References

[1] "Introduction," Arduino. [Online]. Available: https://www.arduino.cc/en/guide/introduction. [Accessed: 23-Mar-2020].

[2] "LCD," Arduino. [Online]. Available: https://www.arduino.cc/en/Tutorial/HelloWorld. [Accessed: 23-Mar-2020].

[3] "LCD Screen Module," Wiki. [Online]. Available: http://wiki.sunfounder.cc/index.php?title=LCD1602_Module. [Accessed: 23-Mar-2020].

[4] H. Barrag, "Breadboard \ Wiring," *Wiring cover*. [Online]. Available: http://wiring.org.co/learning/tutorials/breadboard/. [Accessed: 23-Mar-2020].

[5] "Encyclopedia," breadboard Definition from PC Magazine Encyclopedia. [Online]. Available: https://www.pcmag.com/encyclopedia/term/breadboard. [Accessed: 23-Mar-2020].

[6] "ELEGOO Breadboard 830 Point Solderless Prototype PCB Board Kit," Elegoo Industries. [Online]. Available: https://www.elegoo.com/product/elegoo-3pcs-mb102-breadboard-830-point-solderless-prototype-pcb-board-kit/. [Accessed: 23-Mar-2020].

[7] "Elegoo UNO Project Basic Starter Kit with Tutorial and UNO R3 for Arduino," Elegoo Industries. [Online]. Available: https://www.elegoo.com/product/elegoouno-project-basic-starter-kit-with-tutorial-and-uno-r3-for-arduino/. [Accessed: 23-Mar-2020].

[8] "Pushbutton 4 pin Tactile-Micro Switch - Small SPST," Evelta Electronics. [Online]. Available: https://www.evelta.com/pushbutton-4-pin-tactile-microswitch-small/. [Accessed: 23-Mar-2020].

[9] "Ultrasonic Sensor," https://www.robotistan.com/. [Online]. Available: https://www.robotistan.com/hc-sr04-ultrasonik-mesafe-sensoru. [Accessed: 23-Mar-2020].