University of Windsor

Final Report

Adaptive Cruise Control

Submitted by

Pruthvi Gabani (105159045)

Richa Singh Madnawat (105177573)

Priyanka Pradipbhai Nair (105202729)

Submitted to

Dr. Roozbeh Razavi-Far, University of Windsor

Date:25th March, 2020

CHAPTER NO.		DESCRIPTION	PAGE NUMBER
		Executive Summary	4
CHAPTER 1		Introduction	5
CHAPTER 2		Objective	5
CHAPTER 3		Requirements	6
	3.1	Hardware Requirements	6
	3.2.	Software Requirements	6
CHAPTER 4		Description	6
	4.1.	Five Input Buttons	6
	4.2.	Ultrasonic Sensors	6
	4.3.	Output	7
CHAPTER 5		Block Diagram	7
CHAPTER 6		Working	7
CHAPTER 7		Flowchart	8
CHAPTER 8		Development Procedure	9
	8.1.	Hardware development	9
	8.1.1.	Connect Liquid Crystal Display to Arduino	9
	8.1.2.	Connect Ultrasonic Sensor to Arduino	10
	8.1.3.	Connect Input Buttons to Arduino	10
	8.2.	Software Development	10
	8.2.1.	Installation of Add-ons for Arduino and MATLAB Setup	11
	8.2.2.	Liquid Crystal Display Setup	11
	8.2.3.	Ultrasonic Sensor Set-Up	12
	8.2.4.	Button-Arduino Pins Setup	12
	8.2.5.	Complete Model in Simulink Environment	15
CHAPTER 9		Codes for MATLAB function in Simulink Environment	15
	9.1.	Adaptive Cruise Control Code	15
	9.2.	Code for Numeric Display of Speed on LCD	17
	9.3.	Code for Blinking while Mode changes to Adaptive Cruise Control	18

FIGURE NO.	DESCRIPTION	PAGE NUMBER
1.	Adaptive Cruise Control	5
2.	Block diagram of the general Adaptive Cruise Control Model	7
3.	Flowchart	8
4.	Hardware	11
5.	LCD Pin Setup	12
6.	Ultrasonic Sensor Pin Setup	12
7.	Set-Speed Input Arduino Digital Pin	12
8.	Increase Input Arduino Digital Pin	13
9.	Decrease Input Arduino Digital Pin	13
10.	Adaptive Input Arduino Analog Pin	14
11.	Cancel Input Arduino Digital Pin	14
12.	Adaptive Cruise Control Simulink Model	15

TABLE NO.	DESCRIPTION	PAGE NUMBER
1.	LCD-Arduino connections	9
2.	Ultrasonic sensor to Arduino	10
3.	Button- Arduino Connections	10

EXECUTIVE SUMMARY:

The traffic on roads is increasing accidents have become a major concern for the people. Adaptive cruise control is widely known for its potential to reduce traffic congestion, road deaths, and pollution while increasing the driving comfort. Adaptive Cruise Control (ACC) slows down and speeds up the velocity of the car leading to automatically to keep pace with the car in front of the host car.

In this project, we built an Adaptive Cruise Control Model in the Simulink environment using MATLAB function. This is built on the Arduino UNO hardware connected to the Liquid Crystal Display which displays the change of speed as per the change in the modes of the car which is moving. The speed displayed on the LCD is zero when the system is initialized. After the initialization, we will increase the speed of the car till 40 and once the increase button is released the speed will automatically decrease. After this, we will set the speed constant which means the vehicle has entered the normal cruise control mode. When the vehicle is in the cruise control mode it moves with the constant speed. The Adaptive Cruise Control mode comes in the role when there is an object in front of the moving vehicle. The ultrasonic sensor detects the object in front and transmits the waves back to the vehicle. Once the vehicle enters the adaptive mode the speed increases and decreases according to the distance of the object from the car. If the object is close to the car the speed automatically decreases thus preventing a collision. And if the object is far the speed increases.

A vehicle with ACC adapts to changes in the speed of the vehicle that precedes it. The adaptive Cruise control system is a primary step toward autonomous driving in which the car controls its speed and distance from other vehicles

CHAPTER 1: INTRODUCTION:

The adaptive cruise control system is an advanced method of controlling the speed of the moving vehicle. It mainly reduces the driver's fatigue during a long trip. By using this intelligent control one can easily control the speed of the car according to the prevailing situation. The number of accidents can be avoided through timely threat recognition and appropriate collision avoidance. These types of sophistications are produced as embedded features in luxury cars. It's an early step toward autonomous driving in which the car controls its speed and distance from other vehicles.[1][3]

Adaptive cruise control (ACC) is an intelligent form of cruise control that slows down and speeds up automatically to keep pace with the car in front of us. It is also called active cruise control, autonomous cruise control, intelligent cruise control, or radar cruise control. The adaptive cruise control system is made up of a sensor that is fitted to the moving vehicle which constantly notes the speed of the car and provides the input to the processing unit.[2][3]

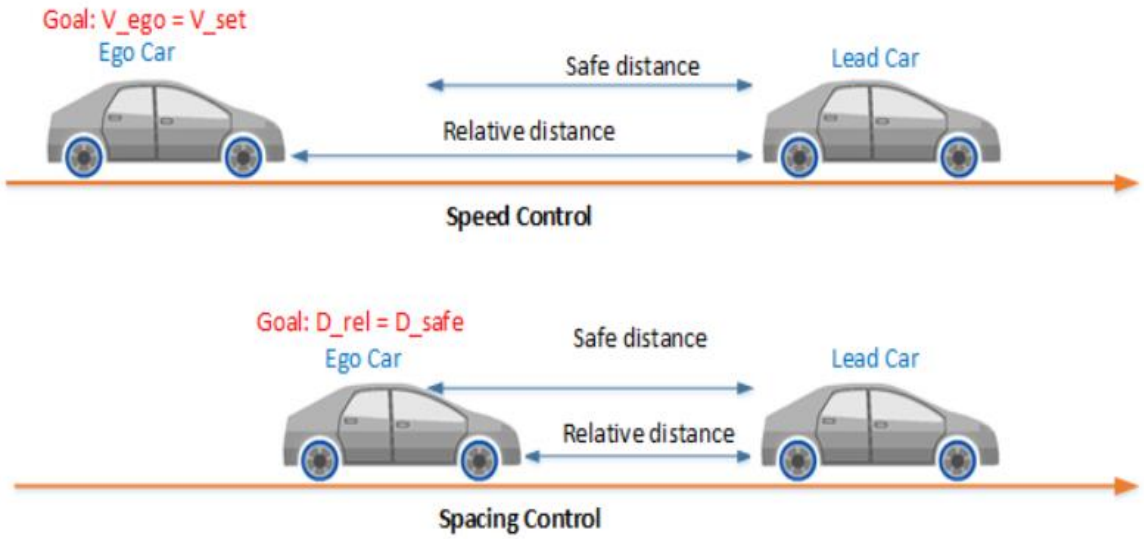

Fig.1: Adaptive Cruise Control [4]

CHAPTER 2: OBJECTIVE:

The main objective is to build an Adaptive Cruise Control model that increases and decreases the speed of the host car according to the distance between the lead car and the host car. This will automatically adjust the vehicle speed to maintain a safe distance from vehicles ahead. The entire design, implementation, and testing of the project is carried out in the hardware, that is, Arduino UNO using MATLAB and Simulink environment.

CHAPTER 3: REQUIREMENTS:

3.1. HARDWARE REQUIREMENTS:

1. Arduino UNO - 1
2. Button - 5
3. Ultrasonic sensor - 1
4. Liquid Crystal Display (16-pin Interface) – 1
5. 10K ohm potentiometer - 1
6. 220-ohm resistors
7. Hook-up wires
8. Breadboard

3.2. SOFTWARE REQUIREMENTS:

1. MATLAB Support Package for Arduino Hardware.
2. Simulink Support Package for Arduino Hardware.
3. Arduino Engineering Kit Hardware Support.
4. Simulink library for Arduino Liquid Crystal Display.

CHAPTER 4: DESCRIPTION:

The Adaptive Cruise Control comprises of the following:

4.1. Five Input Buttons

- Set Speed Button: It is used for Normal Cruise Control mode.
- Increase Button: It is used to increase speed.
- Decrease Button: It is used to decrease the speed.
- Adaptive Button: It is used to switch to Adaptive Cruise Control mode.
- Cancel Button: It is used to cancel the Cruise Control mode.

4.2. Ultrasonic Sensors

- This is used to read the distance when the vehicle is in Adaptive Cruise Control Mode.

Distance = (Time*Speed of Sound)/2

4.3. Output

- Liquid Crystal Display: A 16*2 LCD Display is used to display the speed of the host car which shows the varying speed in different modes.

CHAPTER 5: BLOCK DIAGRAM:

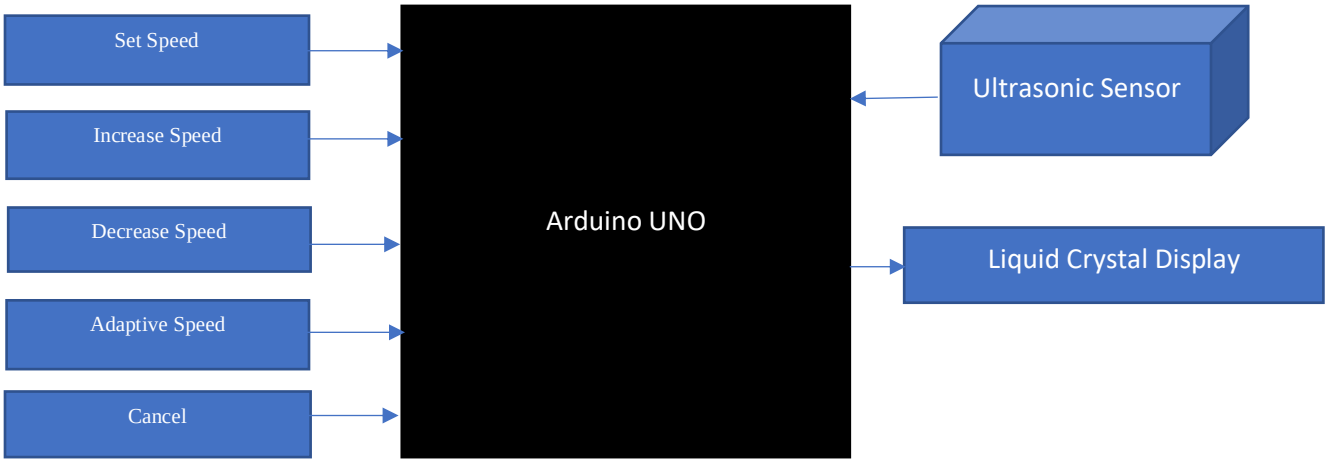

Fig.2. Block diagram of the general Adaptive Cruise Control Model

CHAPTER 6: WORKING:

The basic working of the system starts with the initialization of the system with the display of speed zero on the LCD screen. Once the increase button is pressed there should be an increase of speed from the initialization stage displayed on the display screen. Further, the pressing decrease button should decrease speed. Since the set speed button is off the speed will not remain constant and will decrease if the increase button is released. Pressing the set speed button will fix the speed at a constant leading the vehicle to enter the normal cruise control mode. To change the speed in cruise control mode the increase and decrease button are operative. If the cancel button is active the system exits the cruise control mode and the speed starts decreasing slowly. When the Adaptive button is pressed the speed is set to a constant and is held at the same constant till there is no object or vehicle in front. Once the object is detected the speed starts decreasing and as the distance between the object and the host car increases the speed increases. The display blinks when the system changes the mode from cruise control to adaptive cruise control. To exit the adaptive cruise control mode, the cancel button is pressed which stops the blinking and the systems speed starts decreasing.

CHAPTER 7: FLOWCHART:

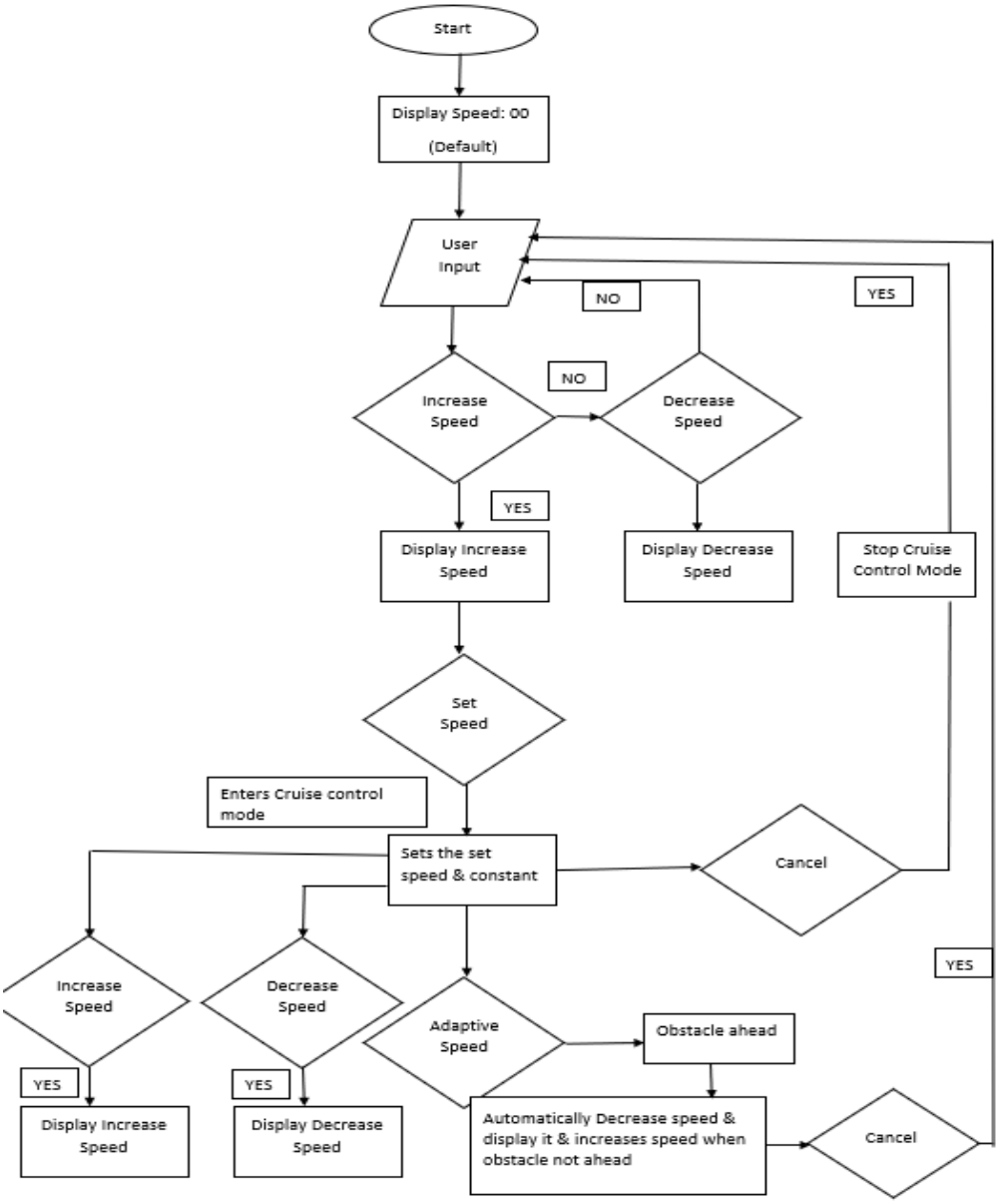

Fig.3: Flowchart of Working of ACC

CHAPTER 8: DEVELOPMENT PROCEDURE:

8.1. HARDWARE DEVELOPMENT:

8.1.1. CONNECT LIQUID CRYSTAL DISPLAY TO ARDUINO: This process requires the following components:

- Solderless Breadboard
- Arduino UNO
- USB Cable
- Potentiometer (10K Ohm)
- LCD Screen (16*2)
- Jumper Cables
- Resistor (220 Ohm)

We will use 16 by 2-character LCD that we will be used to display symbols. Each character is off by default and is a matrix of small dots of liquid crystal. These dots make up the numbers and letters that we display on screens. The LCDs have a parallel interface to control the display, the microcontroller manipulates several interface pins at once.[5]

Following are the pins in the interface:

1. **Register Select (RS)**: This pin controls the LCD's memory where we write the data. To hold data on the screen we can use data register. The instruction register is used to look for instructions on what to do next. 2. Read/Write (R/W) pin that selects reading mode or writing mode

2. **Read or Write (R or W):** This pin selects the mode for reading and writing.

3. **Enable Pin (EN):** Facilitate writing to the registers.

4. **Data Pins(D0-D7):** There are eight data pins. The states of these pins (high or low) are the bits that we are writing to a register when we write or the values we are reading when we read.

5. **Display contrast pin (Vo)**

6. **Power supply pins (+5V and Ground)**

7. **LED Backlight**: Pins that powers the LCD, control the display contrast, and turn on and off the LED backlight.[5]

PIN	FUNCTION	CONNECTS TO:
1.	VSS	GND
2.	VDD	5V
3.	V0	Potentiometer Input
4.	RS	Arduino Digital Pin 12
5.	RW	GND

6.	EN	Arduino Digital Pin 11
7.	D0	No Connection
8.	D1	No Connection
9.	D2	No Connection
10.	D3	No Connection
11.	D4	Arduino Digital Pin 5
12.	D5	Arduino Digital Pin 4
13.	D6	Arduino Digital Pin 3
14.	D7	Arduino Digital Pin 2
15.	A	5V
16.	K	GND

Table.1: LCD-Arduino connections [5]

8.1.2. CONNECT ULTRASONIC SENSOR TO ARDUINO:

The ultrasonic sensor works by sending sound waves from the transmitter, which then bounce off an object and then return to the receiver. We can determine how far away the object is by the time it takes for the sound waves to get back to the sensor.

PIN	**FUNCTION**	**CONNECTS** TO:
1.	VSS	5V
2.	TRIG	Arduino Digital Pin 9
3.	ECHO	Arduino Digital Pin 10
4.	GND	GND

Table.2: Ultrasonic sensor to Arduino

8.1.3. CONNECT INPUT BUTTONS TO ARDUINO:

Hardware requirements for the process are mentioned below:

- Push buttons – 5
- Resistor (220 Ohm) – 5
- Jumper Wires

Connect one of the Arduino ground pins to the breadboards ground pin and then connect the 5V of the Arduino to the 5V of the breadboard to complete the circuit. Connect a jumper wire from the 5-volt pin to one side of the pushbutton and follow the same for all the five buttons. Further, connect a 220-ohm resistor from the ground pin to the other side of the button and follow the same for all the other buttons.

BUTTON	FUNCTION	CONNECT TO:
1.	Cancel Button	Arduino Digital Pin 6
2.	Set Button	Arduino Digital Pin 7
3.	Increase Speed Button	Arduino Digital Pin 13
4.	Decrease Speed Button	Arduino Digital Pin 8

5.	Adaptive Cruise Button	Arduino Analog Pin 3

Table.3: Button- Arduino Connections

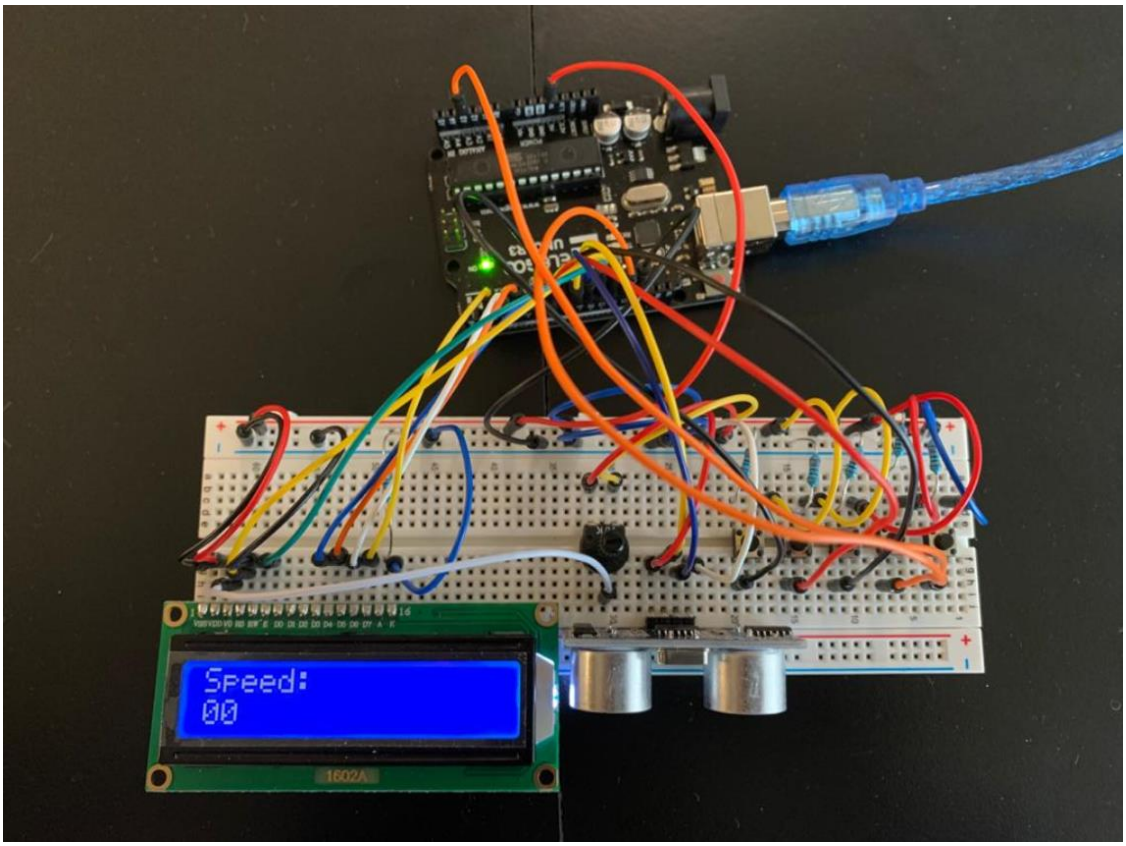

Fig.4. Hardware

8.2. SOFTWARE DEVELOPMENT:

8.2.1. INSTALLATION OF ADD-ONS FOR ARDUINO AND MATLAB SETUP:

To build the model for adaptive cruise control in MATLAB Simulink environment we need to install several add-ons as they provide various blocks used in the model. We can install these add-ons from the home tab in MATLAB by clicking the add-ons options. These add-ons help to link hardware and software by assigning the connections of hardware to the blocks available in these packages. Below are the Add-ons used for building the adaptive cruise control:

1. MATLAB Support Package for Arduino Hardware.
2. Simulink Support Package for Arduino Hardware.
3. Arduino Engineering Kit Hardware Support.
4. Simulink library for Arduino Liquid Crystal Display.

8.2.2. LIQUID CRYSTAL DISPLAY SETUP:

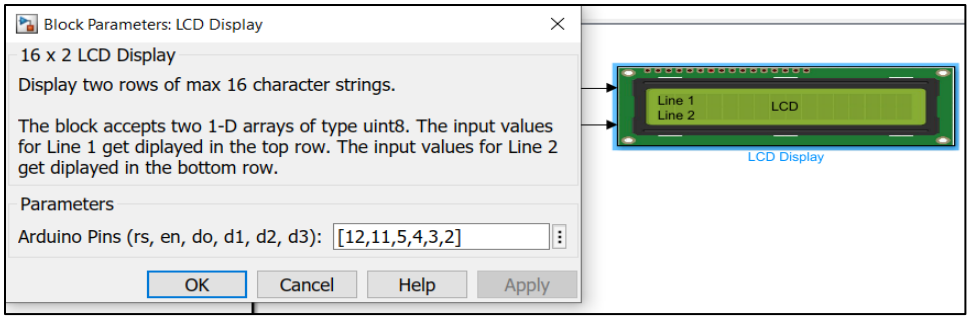

Fig.5: LCD Pin Setup

8.2.3. ULTRASONIC SENSOR SET-UP:

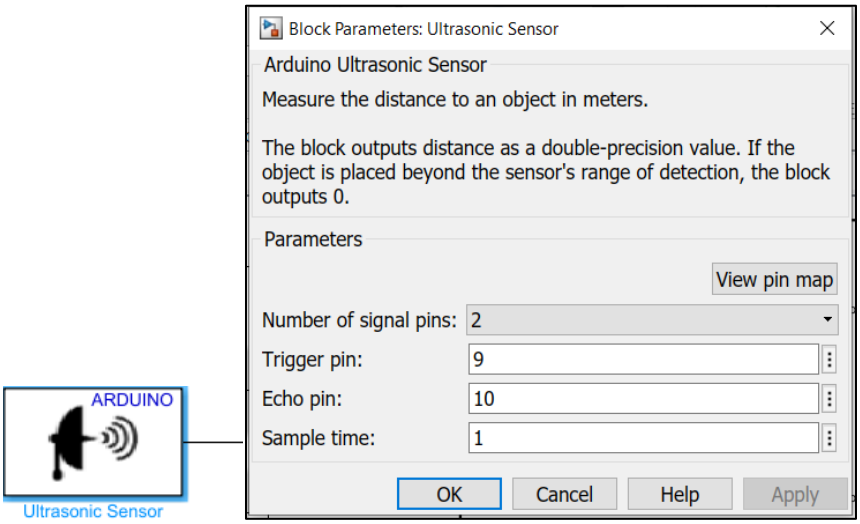

Fig.6: Ultrasonic Sensor Pin Setup

8.2.4. BUTTON-ARDUINO PINS SETUP:

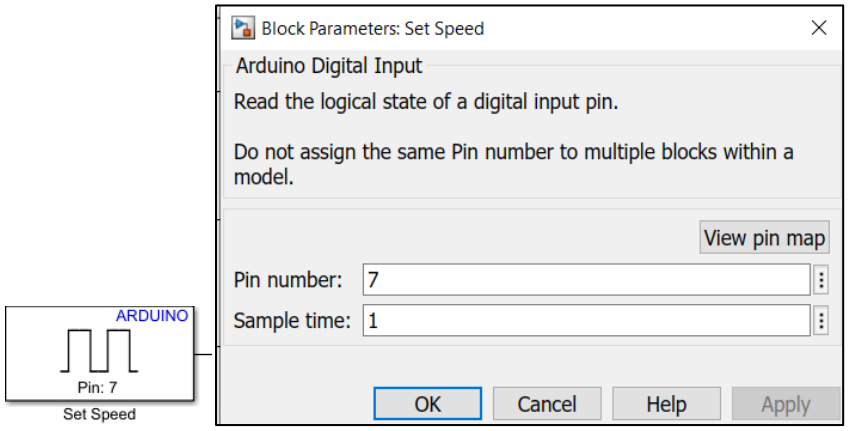

Fig.7: Set-Speed Input Arduino Digital Pin

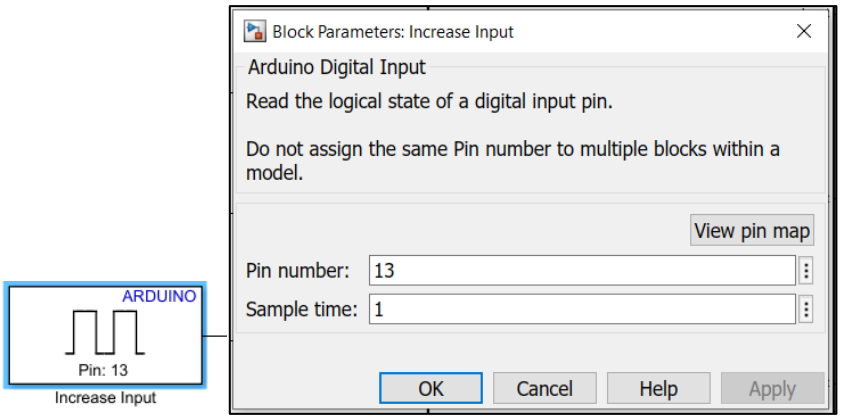

Fig.8: Increase Input Arduino Digital Pin

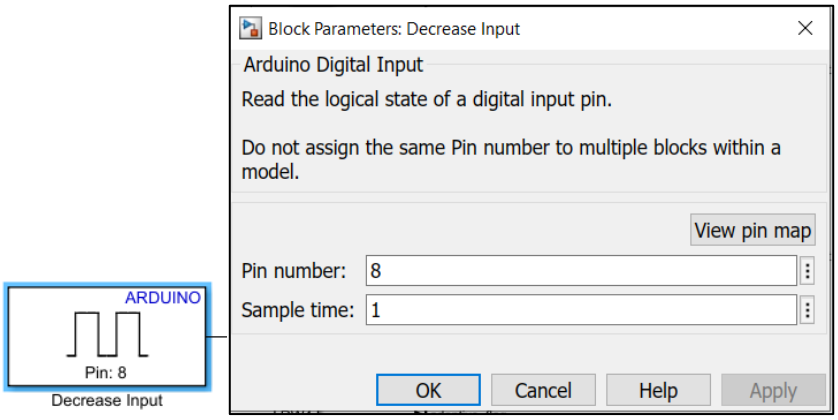

Fig.9: Decrease Input Arduino Digital Pin

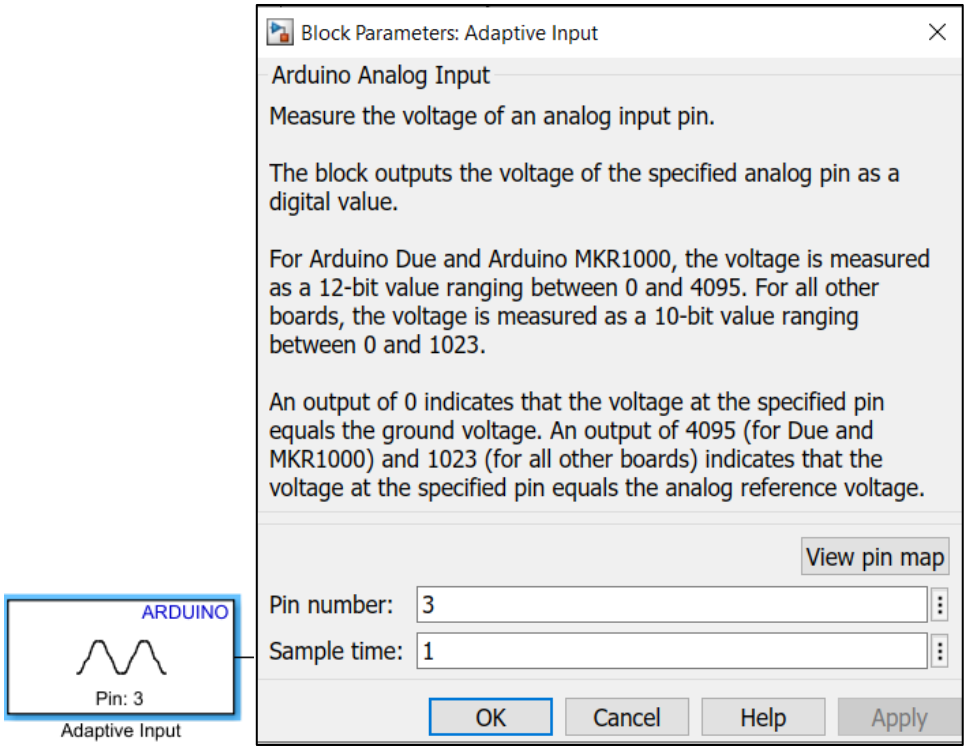

Fig.10: Adaptive Input Arduino Analog Pin

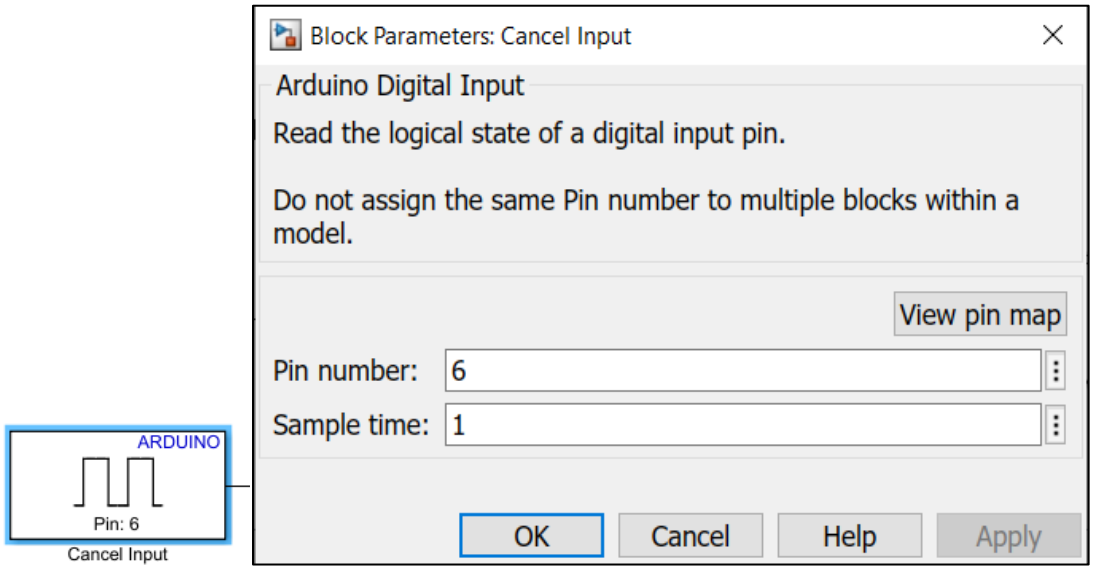

Fig.11: Cancel Input Arduino Digital Pin

8.2.5. COMPLETE MODEL IN SIMULINK ENVIRONMENT:

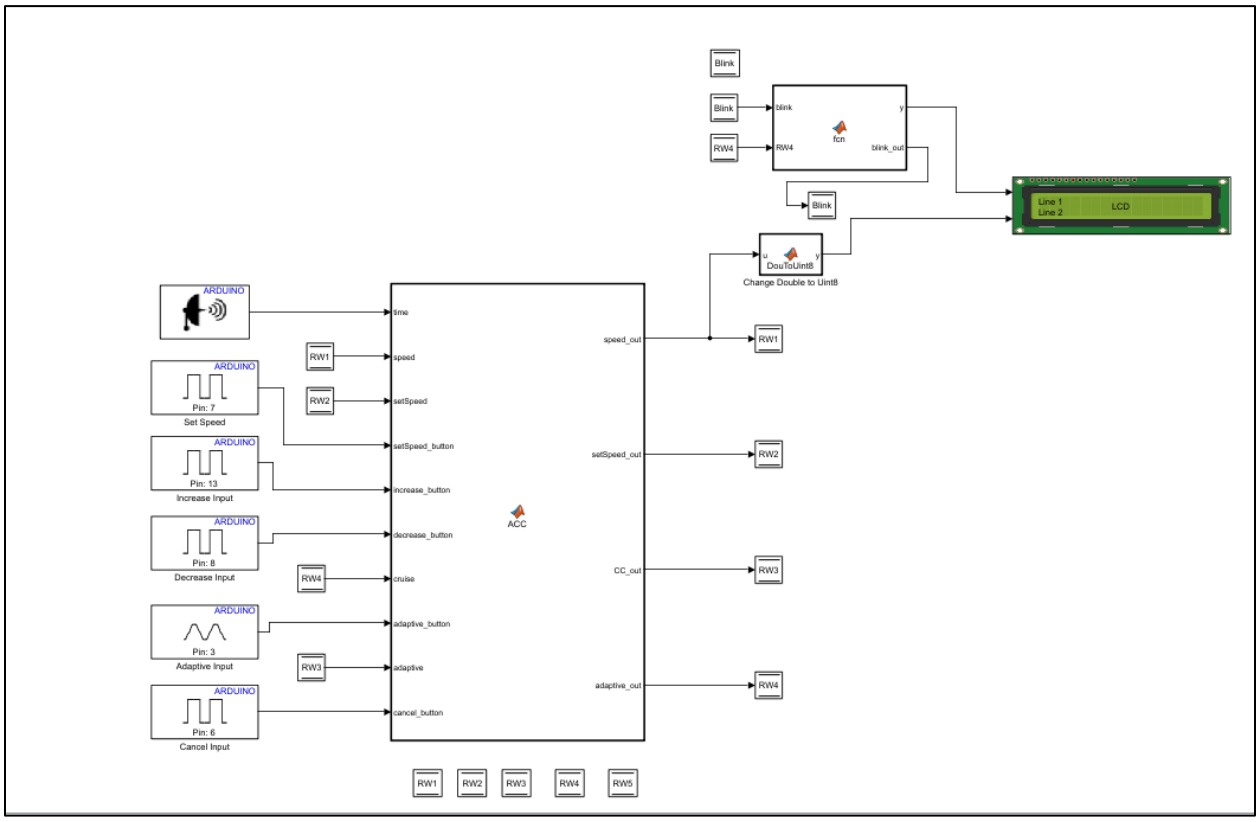

Fig.12: Adaptive Cruise Control Simulink Model

CHAPTER 9: CODES FOR MATLAB FUNCTION IN SIMULINK ENVIRONMENT:

9.1 ADAPTIVE CRUSIE CONTROL CODE:

```
function [speed_out, setSpeed_out,CC_out, adaptive_out] = ACC (time, speed, setSpeed,
setSpeed_button, increase_button, decrease_button, cruise, adaptive_button, adaptive,
cancel_button)

    distance=(time*343)/2;

    if cancel_button
        adaptive_out=0;
        setSpeed_out=setSpeed;
        CC_out=0;
    elseif adaptive_button>800
        setSpeed_out=speed;
        adaptive_out=1;
        CC_out=0;
```

```
    elseif setSpeed_button
        setSpeed_out=speed;
        CC_out=1;
        adaptive_out=0;
    else
        setSpeed_out=setSpeed;
        CC_out=cruise;
        adaptive_out=adaptive;
    end

    if increase_button == 1 && ~adaptive_out == 1
        speed_out = speed+1;
    elseif decrease_button == 1 && speed>0 && ~adaptive_out == 1
        speed_out = speed-1;
    elseif CC_out ==1
        if increase_button == 1  && ~adaptive_out == 1
            speed_out = speed+1;
        elseif decrease_button == 1 && speed>0 && ~adaptive_out == 1
            speed_out = speed-1;
        else
            speed_out = speed;
        end
    elseif adaptive_out == 1
        if (distance<20) && (speed>0)
            speed_out=speed-1;
        else
                if speed<setSpeed && distance>=20
                    speed_out= speed+1;
                else
                    speed_out=speed;
                end
        end
    else
        if speed>0
            speed_out=speed-1;
        else
            speed_out=speed;
        end
    end
end
```

9.2. CODE FOR NUMERIC DISPLAY OF SPEED ON LCD:

```
function y = DouToUint8(u)
switch u
   case 0
      y = [[48],[48]];
   case 1
      y = [[48],[49]];
   case 2
      y = [[48],[50]];
   case 3
      y = [[48],[51]];
   case 4
      y = [[48],[52]];
   case 5
      y = [[48],[53]];
   case 6
      y = [[48],[54]];
   case 7
      y = [[48],[55]];
   case 8
      y = [[48],[56]];
   case 9
      y = [[48],[57]];
   case 10
      y = [[49],[48]];
   case 11
      y = [[49],[49]];
   case 12
      y = [[49],[50]];
   case 13
      y = [[49],[51]];
   case 14
      y = [[49],[52]];
   case 15
      y = [[49],[53]];
   case 16
      y = [[49],[54]];
   case 17
      y = [[49],[55]];
   case 18
      y = [[49],[56]];
   case 19
      y = [[49],[57]];
   case 20
      y = [[50],[48]];
   case 21
```

```
    y = [[50],[49]];
case 22
    y = [[50],[50]];
case 23
    y = [[50],[51]];
case 24
    y = [[50],[52]];
case 25
    y = [[50],[53]];
case 26
    y = [[50],[54]];
case 27
    y = [[50],[55]];
case 28
    y = [[50],[56]];
case 29
    y = [[50],[57]];
case 30
    y = [[51],[48]];
case 31
    y = [[51],[49]];
case 32
    y = [[51],[50]];
case 33
    y = [[51],[51]];
case 34
    y = [[51],[52]];
case 35
    y = [[51],[53]];
case 36
    y = [[51],[54]];
case 37
    y = [[51],[55]];
case 38
    y = [[51],[56]];
case 39
    y = [[51],[57]];
case 40
    y = [[52],[48]];
case 41
    y = [[52],[49]];
case 42
    y = [[52],[50]];
case 43
    y = [[52],[51]];
case 44
```

```
        y = [[52],[52]];
    case 45
        y = [[52],[53]];
    case 46
        y = [[52],[54]];
    case 47
        y = [[52],[55]];
    case 48
        y = [[52],[56]];
    case 49
        y = [[52],[57]];
    case 50
        y = [[53],[48]];
    otherwise
        y = [[48],[48]];
end
```

9.3. CODE FOR BLINKING WHILE MODE CHANGES TO ADAPTIVE CRUISE CONTROL:

```
function [y,blink_out]  = fcn(blink,RW4)

    if blink==0 && RW4
        y=[[32],[32],[32],[32],[32],[32]];
        blink_out=1;
    elseif blinkState==1 && RW4
        y=[[83],[112],[101],[101],[100],[58]];
        blink_out=0;
    else
        y=[[83],[112],[101],[101],[100],[58]];
        blink_out=blink;
    end
end
```

References:

[1]. Duc Lich Luu ; Ciprian Lupu ; Thien Van Nguyen, "Design and Simulation Implementation for Adaptive Cruise Control Systems of Vehicles" 2019 22nd International Conference on Control Systems and Computer Science (CSCS) [Online]. Doi: 10.1109/CSCS.2019.00008. [Accessed: 20 Mar. 2020].

[2]. P. Shakouri, J. Czeczot and A. Ordys, "Adaptive Cruise Control System using Balance Based Adaptive Control technique," 2012 17th International Conference on Methods & Models in Automation & Robotics (MMAR), Miedzyzdrojie, 2012, pp. 510-515.May 2006. [Online]. doi: 10.1109/MCS.2006.1636309. [Accessed: 21 Mar. 2020].

[3]. W. Pananurak, S. Thanok and M. Parnichkun, "Adaptive cruise control for an intelligent vehicle," 2008 IEEE International Conference on Robotics and Biomimetics, Bangkok, 2009, pp. 1794-1799. [Online]. doi: 10.1109/ROBIO.2009.4913274. [Accessed: 22 Mar. 2020].

[4]. MathWorks," Adaptive Cruise Control System", Documentation. [Accessed: 22 Mar. 2020].

[5]. SM," Hello World", Arduino tutorials>Examples from libraries>Liquid Crystal>Hello World. [Accessed: 22 Mar. 2020].