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

Project Report
on

"Implementation of Obstacle Avoiding Robot using Simulink on Arduino Uno Board"

Team Members

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

Submission Date: 13-April-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
RN – Random Number

1. Abstract

In mobile robotics, obstacle avoidance is one of the most important aspects. Robot movement would be very restrictive and fragile if there is no concept of obstacle avoidance. A robotic vehicle that has an intelligence built in it such that whenever an obstacle comes in its path it redirects itself is proposed here.

The objective of the project is to use Simulink environment to develop and implement an obstacle avoiding robot that will move straight until it detects an obstacle, which then will maneuver to a perpendicular direction until no obstacle is detected. This is demonstrated though the software and hardware combination of MATLAB, Simulink and Arduino Uno Board. To detect any obstacle ahead of it, an ultrasonic sensor is used. If the distance between the robot and the obstacle is very less, the robot stops and checks for new distance using Servo motor and Ultrasonic sensor by scanning in the left and right directions. The robot will turn to the left if the distance towards the right side is less than the left side.

2. Introduction

Robotics is an advancing and interesting field. With the advancement of technology, the applications of robotics are increasing. The concept of the mobile robot is advancing, and the number of mobile robots and the complexities associated with those robots are getting increased with various applications. There are various types of mobile robot navigation techniques like map interpreting, path planning, and self-localization. An obstacle avoiding robot that senses and avoids collision with unexpected obstacles is a type of autonomous mobile robot.

3. Description

We have implemented an obstacle avoiding robot, which uses two motors, the servo motor to move only in the left or right directions and the DC motor only moves in the forward direction. Only one of these motors will operate at a time. The conditions on which these motors operate are:

1. Forward moving direction: The ultrasonic sensor will be constantly monitoring for upcoming obstacles. This condition will only be true if no obstacles are detected in front of the robot. The DC motor will spin constantly, and the servo motor will remain still at a neutral position. The top line of the LCD will display “DC Motor RPM:” and the bottom line of the LCD will display the numerical RPM of the motor, this value may be arbitrary. The LED is off in this condition.

2. Sideways moving direction: The ultrasonic sensor detects an obstacle in front of the robot, the LED will light up and the robot will stop moving forward (DC motor is no longer spinning). The servo arm will now rotate from the neutral posi6on to the left or right direc6on and stay there un6l the obstacle is no longer detected. The direction of movement direction is decided randomly. The top line of the LCD will display “ALERT”, and the bottom line of the LCD will indicate which direction the robot is moving. This needs to correlate with the direction the servo arm moves.

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.

- **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.

- **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.

- **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. But finally, we have implemented the project with proper coding and circuit design.

- **Testing**

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

5. Summary of Work

- As our project is to integrate Arduino with MATLAB, we learnt about the basic language which is used in Arduino from online websites and 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 tested some basic projects such as blinking an LED, displaying "Hello World" on the LCD, using DC motor, servo motor, Ultrasonic sensor 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.
- In MATLAB, we have used certain add-ons such as
    - MATLAB Support Package for Arduino Hardware
    - Simulink Support Package for Arduino Hardware
    - Arduino Engineering Kit Hardware Support
    - Simulink Library for Arduino Liquid Crystal Display.
- We have given 5V supply for Ultrasonic sensor, LCD module, L293D motor driver chip and servo motor.
- 3.3V supply for DC motor and Ground for Ultrasonic sensor, LCD module, L293D motor driver chip and servo motor.
- We have used a main function to make use of Ultrasonic sensor to control DC motor, servo motor and LED based on the obstacle in front of the sensor.
- And LCD function to display the status of the servo motor arm direction when an object is detected.
- 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 and checked all the conditions.

6. Components Used

Following components are used in the Obstacle Avoidance Robot.

6.1. Hardware Components

- Arduino Uno x1
- LCD Display x1
- Servo Motor x1
- Bread board and 220-ohm resistor
- Ultrasonic Sensor x1
- 3-6V DC Motor x1
- L293D Motor Driver Chip x1
- LED x1 Arduino Uno Board

6.2. Software Components

- MATLAB
- SIMULINK

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.

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.

Blocks used in Simulink

Following blocks are used in the Simulink Model of Obstacle Avoiding Robot.

- Ultrasonic Sensor
- LCD Display
- Data Store Memory
- Data Store Read
- Data Store Write
- Uniform Random Number Generator
- Digital Output
- Continuous Servo Write

Ultrasonic Sensor:

The Ultrasonic Sensor block outputs the distance between the Hardware-connected ultrasonic sensor and the nearest object in front of the sensor. The block generates the distance in metres as a measure of double precision. Unless the target is located outside the sensor detection range, the block outputs 0. And when you simulate a model comprising the Ultrasonic Sensor block without connecting the hardware, the block produces zero as output.[1]

Figure 1: Ultrasonic Sensor [1]

LCD Display:

Liquid Crystal Display is used to display certain content that we want. This block support two uint8-type 1-D arrays. In the top row, the input values for line 1 are displayed. Line 2 input values are shown in the bottom line. [2]

Figure 2: LCD Display [2]

Data Store Memory:

The Data Store Memory block identifies and initializes a specified shared data store, a memory area that can be accessed by Data Store Read and Data Store Write blocks defining the same name for the data store. The location of the Data Store block that identifies a data store decides which data store blocks can access the data store.

- Unless the Data Store Memory block is in the top-level structure, Blocks of Data Store Read and Data Store Write will reach the data store anywhere in the layout.
- When the Data Store Memory block is in a subsystem, Data Store Read and Data Store Write blocks will access the data storage in the same subsystem or any subsystem below it throughout the layout hierarchy.[3]

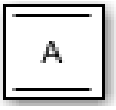

Figure 3: Data Store Memory [3]

Data Store Read:

The Data Store Read block copies the data to its source from the designated data server. More than one block of Data Store Read will read from the same data source. The location of the Data Store Memory block describing the data store specifies the data register from which the data is being read.[4]

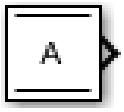

Figure 4: Data Store Read [4]

Data Store Write:

The Data Store Write block copies the value into the designated data store at its entry. Any write operation performed by a Data Store Write block writes over the data store and removes the previous material. The location of the Data Store Memory block or signal entity that identifies the data store specifies the data store to which this block writes to.[5]

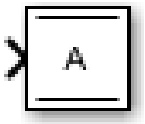

Figure 5: Data Store Write [5]

Uniform Random Number:

The Uniform Random Number block produces random numbers randomly spread over an interval you define. The Random Number block is used to produce naturally distributed random numbers. With any Uniform Random Number Block, you can create a repeatable sequence with the same non-negative seed and parameters. Every time a simulation begins the seed resets to the specified value.[6]

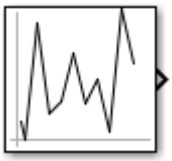

Figure 6: Uniform Random Number [6]

Digital Output:

The Digital Output is used to set the Arduino Hardware to the logical value of a digital pin.

- Depending on the voltage of the circuit, sending 1 to the block input sets the logical value of the digital pin HIGH at 5 V or 3.3 V.
- Sending 0 to the block input sets the logical value of the digital pin LOW to 0 V. [7]

Figure 7: Digital Output [7]

Continuous Servo Write:

Continuous Servo Write sets the direction and speed of a servo motor on continuous rotation:

- Sending -90 to block input generates the highest rotation rate in one direction.
- Sending 90 to block input generates the highest rotation rate in opposite direction.
- Sending 0 to servo motor stops the block entry.
- Sending out-of-range values to the block input, such as -95 or 200, has the same result as submitting the minimum or maximum input values.[8]

Figure 8: Continuous Servo Write [8]

7. Control for each component

- The Ultrasonic sensor has two pins namely Trigger pin and the Echo pin. Trigger pin is to send the sound wave and the Echo pin is to receive the sound wave. The parameters of both trigger and echo pins appear only when you set the parameter of number of signal pins to 2.
- In the LCD Display, for a label combination and its value, a maximum of 16 characters are shown. Any output that reaches 16 characters is truncated on the LCD panel. To turn the LCD backlight on or off, we can use a scalar input 'Bklgt' and to display a value on the LCD screen, we can use 'Val'. The block accepts two 1-D arrays of type uint8. The input values for Line1 get displayed in the top row and the values for Line2 get displayed in the bottom row.
  Parameters: (rs, en, d0, d1, d2, d3) :: (6, 7, 8, 9, 10, 11)
- In the Data Store Read, values from a data store, output with the similar datatype and the dimensions as in the data store will be our outputs. Both real and complex signals are supported by this block. You can choose to output the entire data store, or only selected data store elements.
- In the Data Store Write, the values to write to the specified data store will be our input values. Any real or complex signal is accepted by the Data Store Write block. An array of buses can be used with this block.
- In the Uniform Random Number, the output signal of generated uniformly distributed random numbers over an interval we specify will be our outcome. The 'double' datatype is allowed in this block.
- We should assign different pin numbers for Digital output and Continuous Servo Write because if we give same pin number, it may cause resource management conflicts.

8. Overall Logic

- The first step will be the initialization of all pins and variables.
- Then the output from the Ultrasonic sensor is continuously taken into the MATLAB function.
- When an object is not detected in front of the Ultrasonic sensor, the DC motor turns ON, LED turns OFF and the servo motor arm will be in center position and the function keeps checking for the object in front of the Ultrasonic sensor.
- If an object is detected in front of the Ultrasonic sensor, DC motor turns OFF, LED turns ON and a random number (RN) generator input is obtained is taken into the MATLAB function and the RN is locked as long as the object found so that the arm of the servo motor can be kept in the respective random direction.
- If the generated random number (RN) is greater than zero, the servo motor should be rotated 900 clockwise and if not, the servo motor should rotate 900 anti-clockwise.
- After the rotation of this servo motor, we should check whether the obstacle is still in place or not. If the object remains, the servo motor arm should be retained in the same position and if not, it should return to the center(original) position.

9. Functions Blocks Explanation

Following are the functions used in Obstacle Avoiding Robot.

- Main Function
- LCD Function

Main Function:

Main function is used to make use of Ultrasonic sensor to control DC motor, Servo motor and LED based on the obstacle in front of the sensor.

- Read Ultrasonic sensor data and calculate the distance of the obstacle from the robot.
- Keep DC motor turned ON, Servo motor in its initial position and LED OFF until an obstacle is detected by the sensor.
- When an obstacle is detected, turn OFF the DC motor, turn the Servo motor either clockwise or anti-clockwise based on a Random Generator block in Simulink and turn on LED.
- When the obstacle is still in front, retain the position of the Servo motor with DC motor turned OFF and LED turned ON.
- When the obstacle is absent, turn ON DC motor and bring the Servo motor arm to the initial position with LED turned OFF.

LCD Function:

LCD function is used to display the status of the Servo motor arm direction when an object is detected.

- When the obstacle is not found, display ‘DC Motor RPM:’ in the first line of LCD Display and an arbitrary constant (Eg:200) in the second line.
- When the Servo motor arm is rotated clockwise, display ‘Turned Right’ in the second line of LCD Display with the first line displaying ‘Alert!!’

- When the Servo motor arm is rotated anti-clockwise, display ‘Turned Left’ in the second line of LCD Display with the first line displaying ‘Alert!!’.

10. Code Explanation

Main Function:

Refer Appendix A.

Following are the input variables used in the main function of Obstacle Avoiding Robot.

- sensor_time - time taken for the ultrasonic waves to reach the receiver end of the ultrasonic sensor.
- random_val - Random value between -1 and +1 generated by Continuous Random Generator block in Simulink.
- lock_in - Locking flag to retain the position of the Servo motor arm.
- locked_random_val_in - Locked random value (to retain the position of the Servo motor arm).
- servo_dir_right_in - Flag to display the direction of the Servo motor.

Following are the output variables used in the main function of Obstacle Avoiding Robot.

- m1 - DC Motor Terminal 1.
- m2 - DC Motor Terminal 2.
- led_out - Signal to turn on LED.
- servo_out - Signal to the control pin of Servo motor.
- servo_dir_right - Flag to represent the direction of Servo motor arm.
- lock_out - Locking flag to retain the position of the Servo motor arm.
- locked_random_val - Updated locked random value (to retain the position of the Servo motor arm).

1. Initially, the distance of the obstacle in front of the ultrasonic sensor is calculated using the formula

   Distance = (sensor_time*343)/2

   (Speed of sound in air - 343 m/s)
2. And then we must check whether the obstacle is in front of the robot or not using if statement and if its true, set the two DC terminals to zero and turn ON the LED.
3. We must check whether the locking flag to retain the position of the servo motor arm is zero or not. If yes, locking the random value to retain the servo motor arm in its position as long as the obstacle is present is done by assigning random_val to locked_random_val, servo_dir_right_in to servo_dir_right and the signal to the control pin of servo motor is set to zero. If no, the locked_random_val_in is assigned to locked_random_val.
4. We must check whether the locked_random_val_in value is greater than zero or not. If yes, the rotation of the servo motor arm either in clockwise or anti-clockwise based on random value is done.
5. If the signal to the control pin of the servo motor is 90, then the arm turns left and the servo_dir_right sets to 1. And if the signal is -90, then the arm turns right and the servo_dir_right sets to 0.

6. At the end of all these conditions, we must set the locking flag to retain the position of the servomotor to 1.
7. If the obstacle is not detected in front of the ultrasonic sensor, it enters the else loop in which the first DC terminal sets to 1 and the second DC terminal sets to 0, led_out and servo_out to 0, locked_random_val_in to locked_random_val, servo_dir_right_in to servo_dir_right and finally the lock_out variable to 0.

LCD function:

Refer Appendix B.

Following are the input variables used in Obstacle Avoiding Robot.

- stop - Stop signal denoting object detected.
- servo_dir_right - Flag to display the direction of the Servo motor.

Following are the output variables used in Obstacle Avoiding Robot.

- line1 - Characters to display to line 1 of LCD.
- line2 - Characters to display to line 2 of LCD.

1. Initially, we must check whether an object is detected or not. If no object is detected, line1 displays ‘DC Motor RPM’ and line2 displays numerical RPM of the motor (ASCII values are used to display these characters).
2. Else the object is detected and the line1 displays ‘ALERT!!’ message and checks for the direction of the arm.
3. If the arm turns right, the line2 displays ‘Turning right’ or else ‘Turning left’ (ASCII values are used to display these characters).

11. Commands Used in MATLAB Function

Following are the commands used in Obstacle Avoiding Robot.

- **if-else Statement**
  The basic structure for the use of this statement is:

```
if logical expression
    Statement group 1
else
    Statement group 2
end
```

Every if statement must have an accompanying end statement. The end statement marks the end of the statements that are to be executed if the logical expression is true.

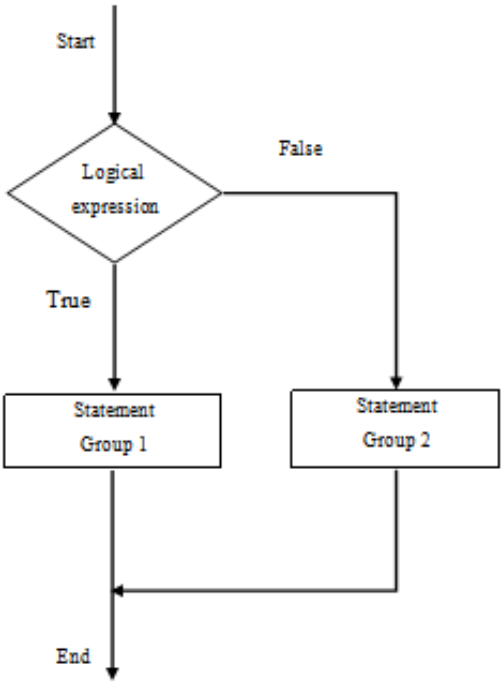

Figure 9: Flowchart of If-Else condition

12. Flowchart

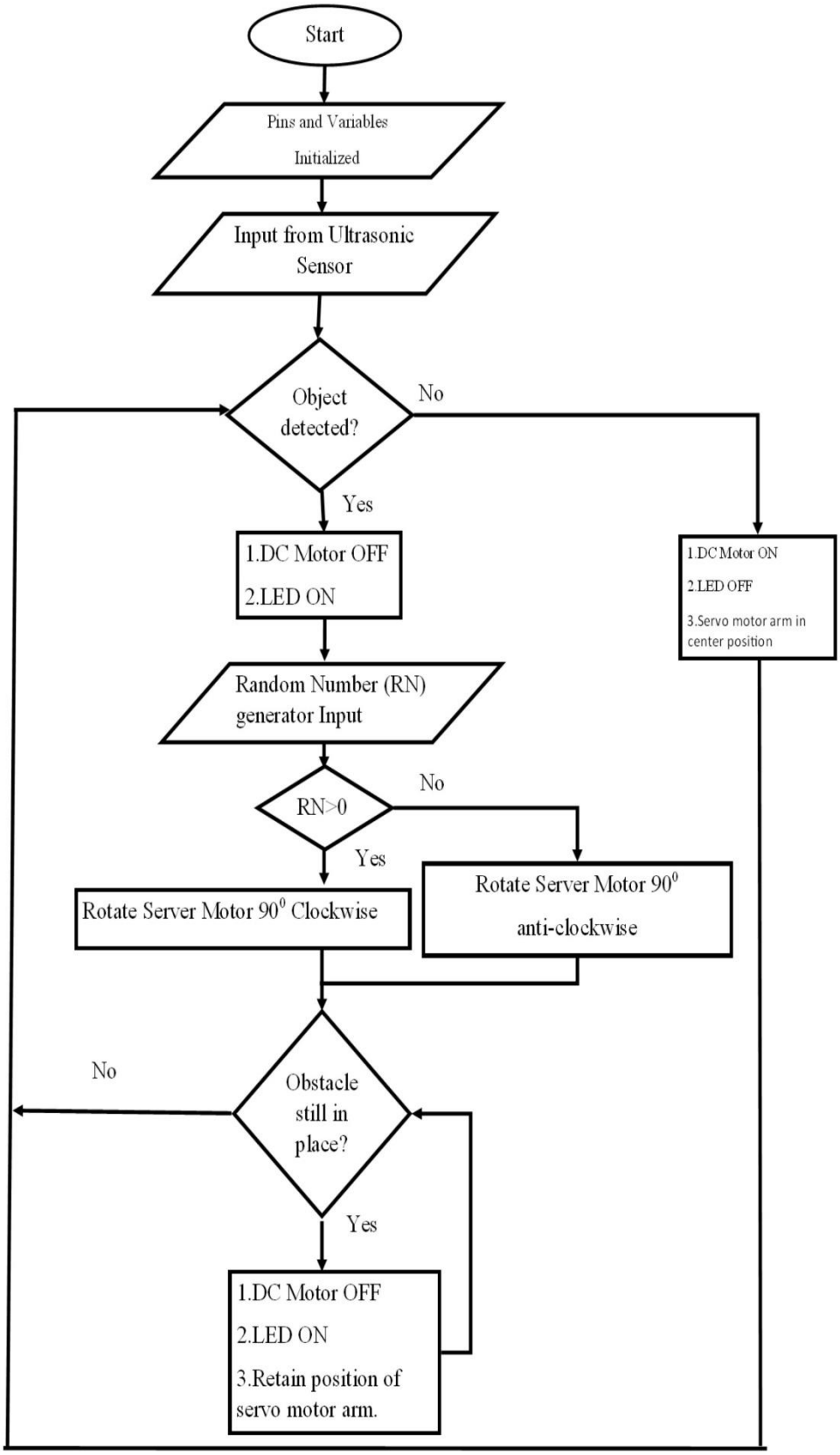

Figure 10: Flowchart of Obstacle Avoiding Robot

13. Circuit Diagram

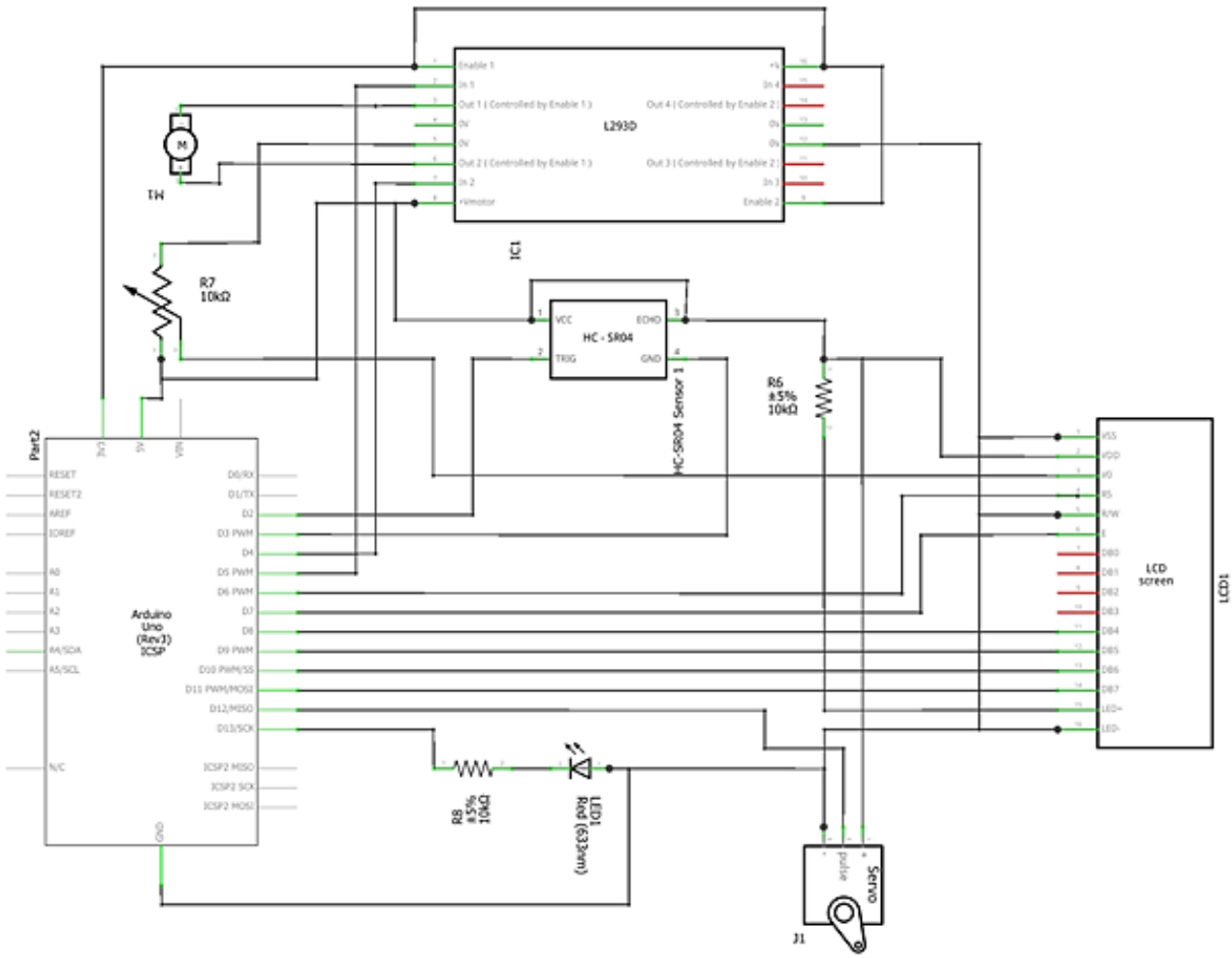

Figure 11: Circuit diagram

14. Connection Diagram

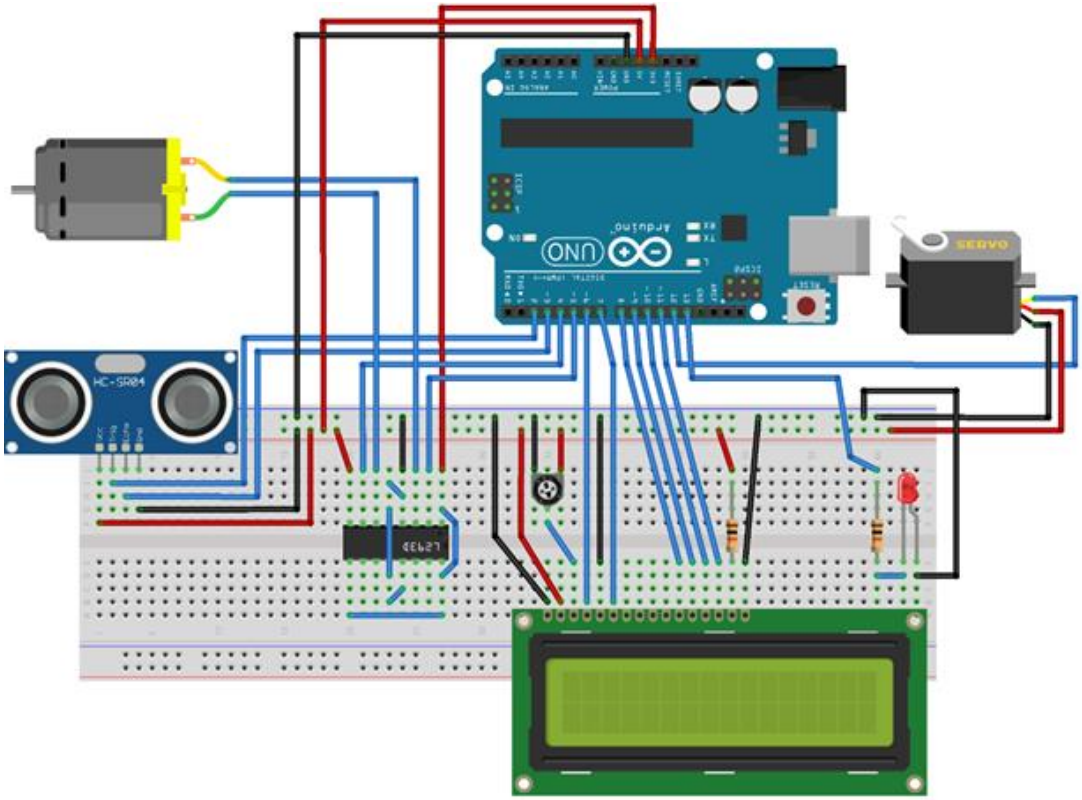

Figure 12: Connection diagram

15. Simulink Block Diagram

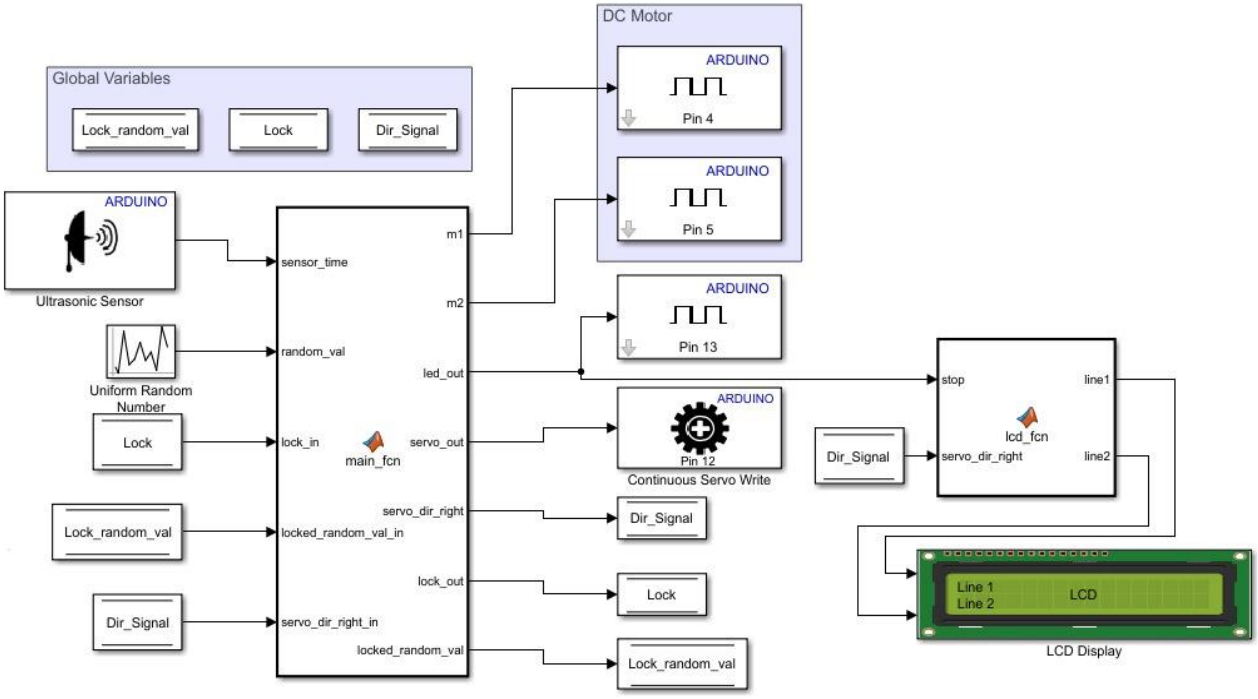

Figure 13: Simulink Block Diagram

16. Analysis of the Results

Our main objective is to develop and implement an obstacle avoiding robot that will move straight until it detects an obstacle, which then will maneuver to a perpendicular direction until no obstacle is detected. Initially, the Ultrasonic sensor will constantly monitor for obstacles. If no obstacles are detected in front of the robot, this condition will be true. The servo motor will remain at a neutral position and the **DC** motor will spin constantly. The top line of the **LCD** will display "**DC Motor RPM**" and the bottom line of the **LCD** will display the numerical **RPM** of the motor. The **LED** is off in this condition.

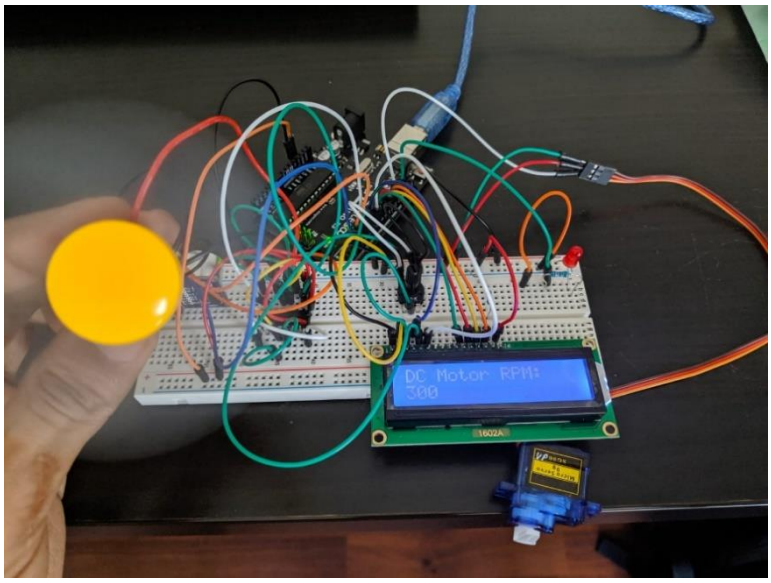

Figure 14: DC motor spinning

When the Ultrasonic sensor detects an obstacle in front of the robot, the **LED** will light up and the **DC** motor stops spinning. The servo arm will now rotate from the neutral position to the left direction and stay there until the obstacle is no longer detected. The top line of the **LCD** will display "**ALERT!!**" and the bottom line of the **LCD** will display "**Turning Left**"

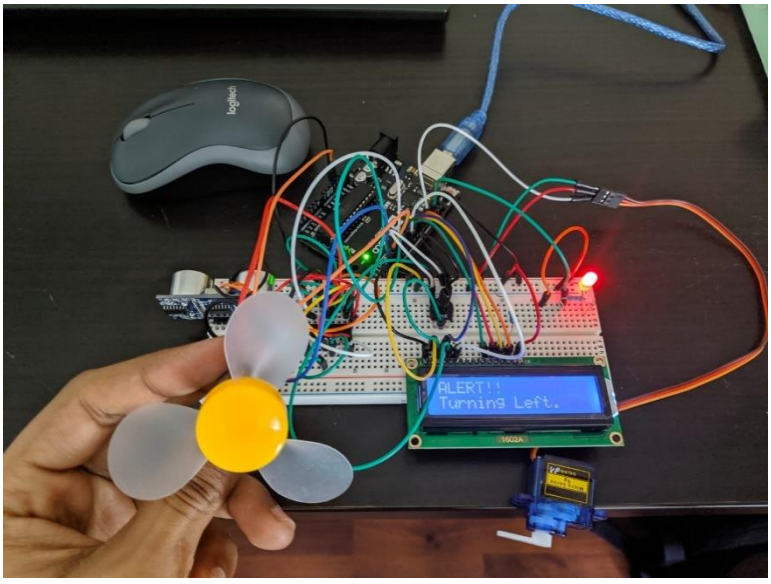

Figure 15: Obstacle detected and the servo arm turning left

When the Ultrasonic sensor detects an obstacle in front of the robot, the **LED** will light up and the **DC** motor stops spinning. The servo arm will now rotate from the neutral position to the right direction and stay there until the obstacle is no longer detected. The top line of the **LCD** will display "**ALERT!!**" and the bottom line of the **LCD** will display "**Turning Right**"

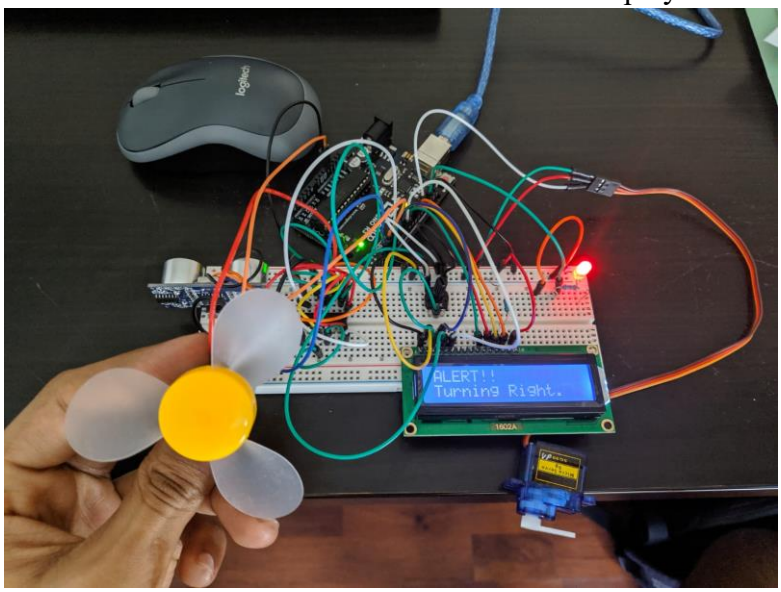

Figure 16: Obstacle detected and the servo arm turning right

17. Conclusion

We were able to build a prototype of an Obstacle Avoiding Robot in Simulink environment using MATLAB on Arduino Uno board and test all possible scenarios successfully. This model can be made more realistic if we had modified it to mimic a small robotic car but because of the ongoing pandemic we couldn't take it further.

18. References

[1] "Ultrasonic Sensor," Arduino. [Online]. Available: https://www.mathworks.com/help/supportpkg/arduinoio/ultrasonic-sensor.html [Accessed: 13-Apr-2020].

[2] "LCD," Arduino. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/49901-matlab-arduino-lcd-display-system. [Accessed: 13-Apr-2020].

[3] "Data Store Memory," Arduino. [Online]. Available: https://www.mathworks.com/help/simulink/slref/datastorememory.html. [Accessed: 13-Apr-2020].

[4] "Data Store Read," Arduino. [Online]. Available: https://www.mathworks.com/help/simulink/slref/datastoreread.html. [Accessed: 13-Apr-2020].

[5] "Data Store Write," Arduino. [Online]. Available: https://www.mathworks.com/help/simulink/slref/datastorewrite.html. [Accessed: 13-Apr-2020].

[6] "Uniform Random Number," Arduino. [Online]. Available : https://www.mathworks.com/help/simulink/slref/uniformrandomnumber.html. [Accessed: 13-Apr-2020].

[7] "Digital Output," Arduino. [Online]. Available : https://www.mathworks.com/help/supportpkg/freedomboard/ref/digitaloutput.html. [Accessed: 13-Apr-2020].

[8] "Continuous Servo Write," Arduino. [Online]. Available : https://www.mathworks.com/help/supportpkg/arduino/ref/continuousservowrite.html. [Accessed: 13-Apr-2020].

19. Appendix

Appendix A:

Main Function:

```
% Program main_fcn.m:
% Makes use of Ultrasonic sensor to control DC motor, Servo motor and LED
% based on the obstacle in front of the sensor.
%    1. Read ultrasonic sensor data and calculate the distance of the     obstacle from the robot.
%    2. Keep DC motor turned ON, Servo motor in its initial position and LED OFF until an obstacle is detected by the sensor.
%    3. When an obstacle is detected, turn OFF the DC motor, turn the Servo motor either clockwise or anti-clockwise based on a Random Generator block in Simulink and turn ON LED.
%    4. When the obstacle is still in-front, retain the position of the Servo motor with DC motor turned OFF and LED turned ON.
%    5. When the obstacle is absent, turn ON DC motor and bring the Servo motor arm to the initial position with the LED turned OFF.
% Created on Apr 02, 2020 by SHYAM LAL LAKSHMIRAMIYA SRIDHAR, JITEN
% RAJENDRA PANDIT, PHANI TEJASWINI KARNATI

% Input Variables :
% sensor_time = time taken for the ultrasonic waves to reach the receiver end of
% the ultrasonic sensor.
% random_val = Random value between -1 and +1 generated by Continues Random Generator block in Simulink,
% lock_in = Locking flag to retain the position of the Servo motor arm.
% locked_random_val_in = Locked random value (to retain the position of the Servo motor arm).
% servo_dir_right_in = Flag to display the direction of the Servo motor.
% arm in the LCD Display.

% Output Variables:
% m1 = DC Motor Terminal 1.
% m2 = DC Motor Terminal 2.
% led_out = Signal to turn on LED.
% servo_out = Signal to the control pin of Servo motor.
% servo_dir_right = Flag to represent the direction of Servo motor arm.
% lock_out = Locking flag to retain the position of the Servo motor arm.
% locked_random_val = Updated locked random value (to retain the position of the Servo motor arm).

function [m1,m2,led_out,servo_out,servo_dir_right, lock_out,locked_random_val] = main_fcn(sensor_time, random_val, lock_in, locked_random_val_in, servo_dir_right_in)
% Calculating the distance of the obstacle in front of the ultrasonic sensor.
% Speed of sound in air = 343 m/s.
distance=(sensor_time*343)/2;

if distance < 20 % Checking for an obstacle in front of the robot.
   m1=0;
   m2=0;
   led_out=1;
   if lock_in==0 % Locking the random value to retain the servo motor arm in its position as long as the obstacle is present.
      locked_random_val=random_val;
      servo_dir_right=servo_dir_right_in;
      servo_out= [0];
   else
      locked_random_val=locked_random_val_in;
      if locked_random_val_in>0 % Rotating the Servo motor arm either clockwise or anti-clockwise based on random value.
         servo_out= [90]; % Turn arm to left
```

```
        servo_dir_right=1;
      else
        servo_out= [-90]; % Turn arm to right
        servo_dir_right=0;
      end
    end
    lock_out=1;
else
    m1=1;
    m2=0;
    led_out=0;
    servo_out= [0];

    servo_dir_right=servo_dir_right_in;
    lock_out=0;
end
end
```

Appendix B:

LCD Module Function:

```
% Program lcd_fcn.m:
% Displays the status of the Servo motor arm direction when an object is detected.
%    1. When the obstacle is not found, display 'DC Motor RPM:' in the first line of LCD Display and an
arbitrary constant (Eg:300) in the second line.
%    2. When the Servo motor arm is rotated clockwise, display 'Turned Right.' in the second line of LCD
Display with the first line displaying 'Alert!!'.
%    3. When the Servo motor arm is rotated anti-clockwise, display 'Turned Left.' in the second line of LCD
Display with the first line displaying 'Alert!!'.
% Created on Apr 02, 2020 by SHYAM LAL LAKSHMIRAMIYA SRIDHAR, JITEN
% RAJENDRA PANDIT, PHANI TEJASWINI KARNATI

% Input Variables :
% stop = Stop signal denoting object detected.
% servo_dir_right = Flag to display the direction of the Servo motor.

% Output Variables:
% line1 = Characters to display to line 1 of LCD.
% line2 = Characters to display to line 2 of LCD.

function [line1, line2] = lcd_fcn (stop, servo_dir_right)
if ~stop % If no object is detected.
    line1= [[68], [67], [32], [77], [111], [116], [111], [114], [32], [82], [80], [77], [58], [32]];
    line2= [[51], [48], [48], [32], [32], [32], [32], [32], [32], [32], [32], [32], [32], [32]];
else
    line1= [[65], [76], [69], [82], [84], [33], [33], [32], [32], [32], [32], [32], [32], [32]];
    if servo_dir_right % if object is detected, and the arm is turned towards right.
        line2= [[84], [117], [114], [110], [105], [110], [103], [32], [76], [101], [102], [116], [46], [32]];
    else % if object is detected, and the arm is turned towards left.
        line2= [[84], [117], [114], [110], [105], [110], [103], [32], [82], [105], [103], [104], [116], [46]];
    end
end
end
```