SMART PARKING

Based on: Simulink and Arduino UNO

By:

Rajnish Tailor **110006068**
Dhaval Patel **105174255**

(TEAM 4)

Guidance of:

Dr. Rashid Rashidzadeh
(Professor)
and
Manya Mehta and Ankit Mehta
(G.A.)

Comp. Meth. & Modeling for Engineering (GENG-8030-1-R)
Faculty of Engineering University of Windsor, Canada
Winter 2020 Semester

Overview

Cars on the road are increasing day-by-day and so are the requirements to park them. Smart Parking is a system which is fixed inside the building/area, when activated can determine the number of parking spots inside as well as control the occupancy of the vehicle spots. The system designed in this project works on Simulink environment, a product of MathWorks, and is executed in the real-time with the help of an Arduino UNO micro-controller. The project consists of 2 basic modes;

- The Entry Mode: Entry and increase of occupancy
- The Exit Mode: Exit and decrease of occupancy

The entry exit is completely manual. The barrier lifting, everything is done on a trigger action of the buttons.

CONTENTS

List of Pictures

Introduction

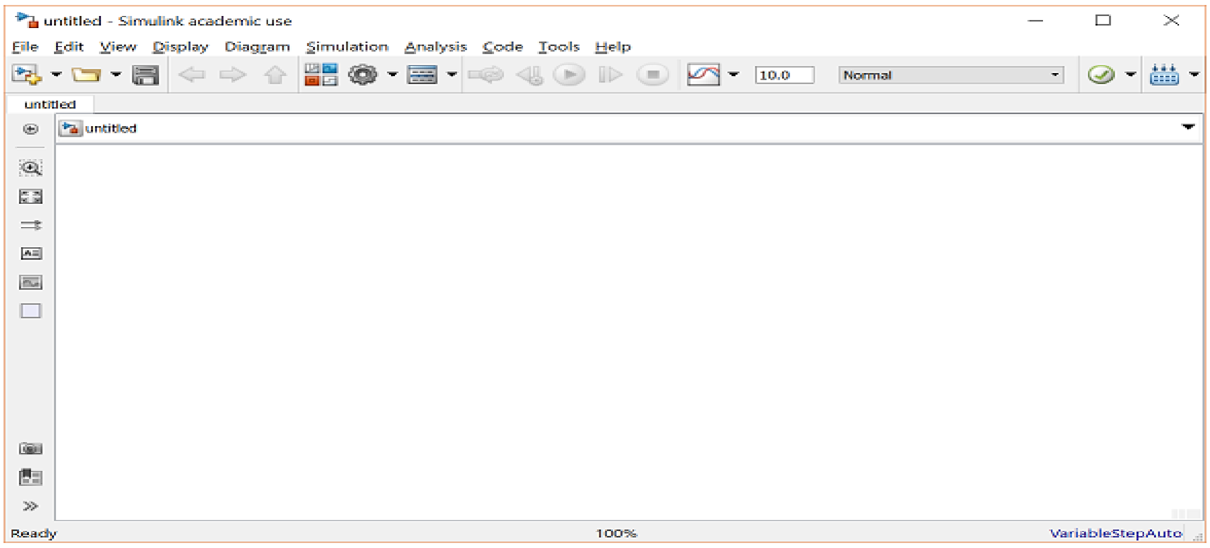
The system developed consists of two modes which are capable of manipulating the spots of the area the system is installed in. The modes have a specific task, which when selected change the number of spots. The entry mode decreases the number of spots on manual actions/button and the exit system increases the number of spots available.

The whole project runs on the Simulink background and is executed by the onboard embedded system Arduino. The manual system contains a logic set which increases and decreases the number of spots depending whether a vehicle is coming in or going out of the area where the system is installed in. The logic on respective trigger does the corresponding arithmetic with spots number as well as triggers the servo/barrier arm to turn in a particular direction.

Design and Methodology

2.1 Simulink

Simulink is a graphical simulation software in MATLAB. It can perform any action or run any program or task, but in a graphical format. It is as good as programming in MATLAB, rather better, as one can see and visualize the different sections and components used as well as monitor the signal flow. It contains a library of different blocks we can use to design a project. The blocks contain real time parameters as well as can perform a real time execution of the data. Each block has a specific purpose which is described in the description of the same. The description gives you the idea that what are different attributes of the block viz. data-type, inputs, outputs etc. With the help of all these an appropriate block can be chosen for a task.



untitled - Simulink academic use
File Edit View Display Diagram Simulation Analysis Code Tools Help
10.0
Normal
untitled
untitled
Ready
100%
VariableStepAuto

Fig. 2.1.1 Simulink Space [1]

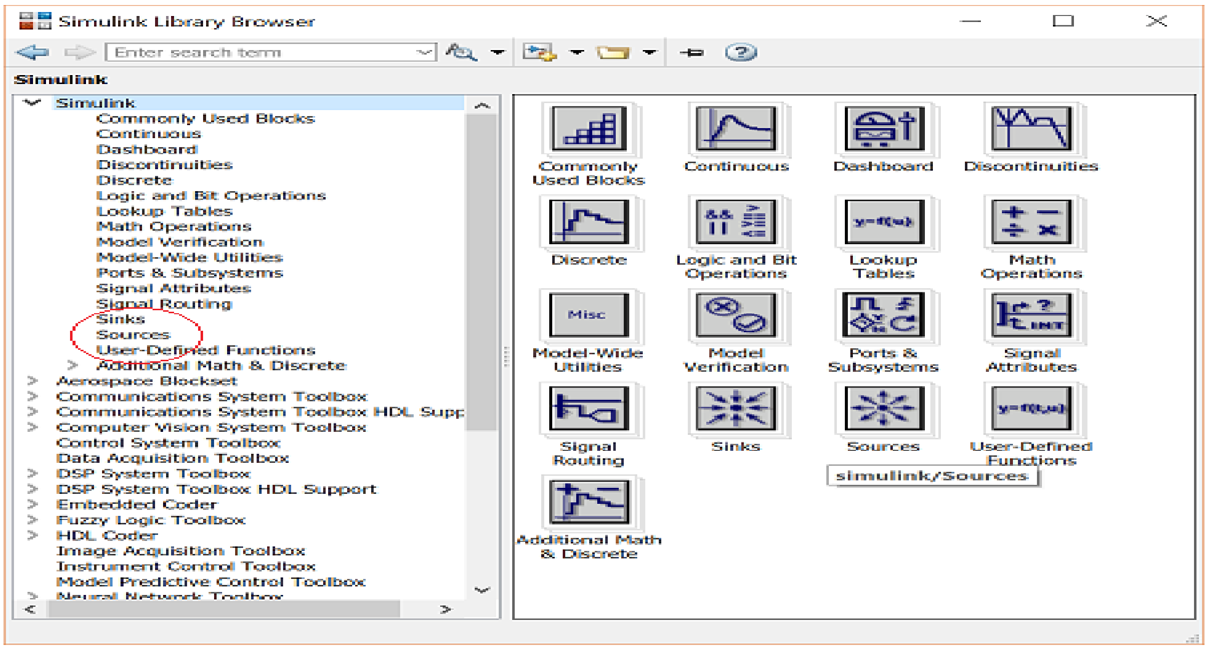


Fig. 2.1.2 Simulink Library [2]

2.2 Component Description

2.2.1 Hardware

- Arduino UNO: - The Arduino UNO is a microcontroller which can perform asks when programmed. It had a ATMEGA328P chipset from Atmel. The on-board system has a 16MHz crystal and is equipped with digital I/O, analog input, PWM output, serial communication and ICSP.

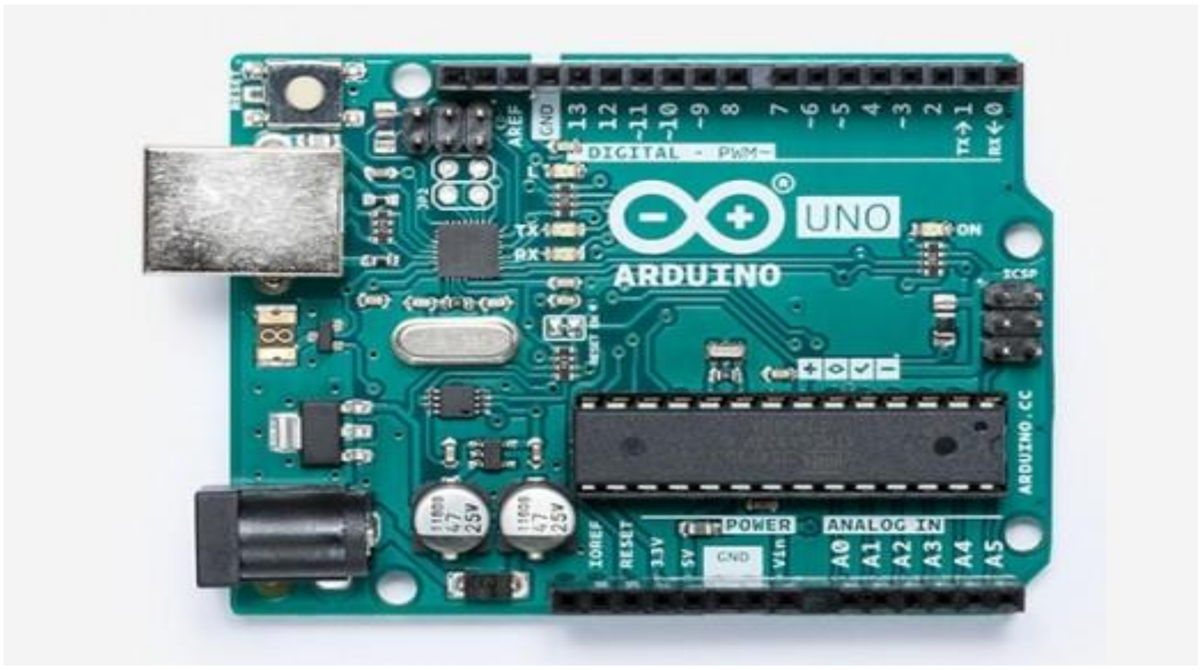


Fig. 2.2.1.1 Arduino UNO [3]

- Liquid Crysal Display (LCD): - There is a 16x2 LCD used to display the result of the process riggers by the user. It has 16 columns and 2 rows, also, a SoC which is used in 4bit feed procedure by the Arduino and a contrast control with a 10K potentiometer.

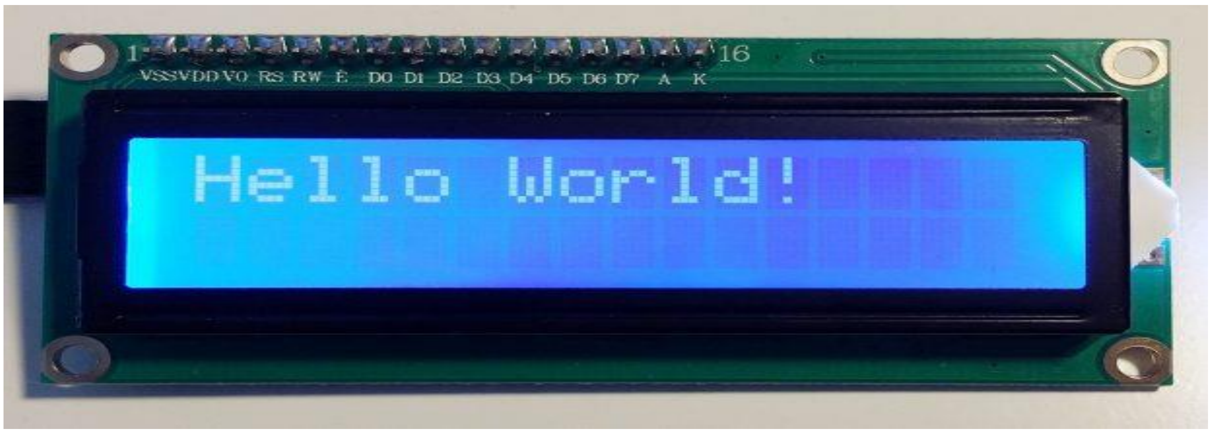


Fig. 2.2.1.2 16x2 LCD [4]

- Buttons: - They type of buttons used in this project are of push type. So, when you push the button the internal circuit gets triggered and the connection is completed. They are used to select a particular loop/sub-loop/subsystem inside the Simulink file.

Fig. 2.2.1.3 Push Button [5]

- Resistor: - A resistor is basically a passive electrical device to drop voltage or limit current. Here we have used 220Ohm resistor for the buttons and LCD and one 10K Ohm potentiometer for contrast variation for the LCD. The purpose of the resistor is o hold the value and limit overall current from supply to the Arduino pin.

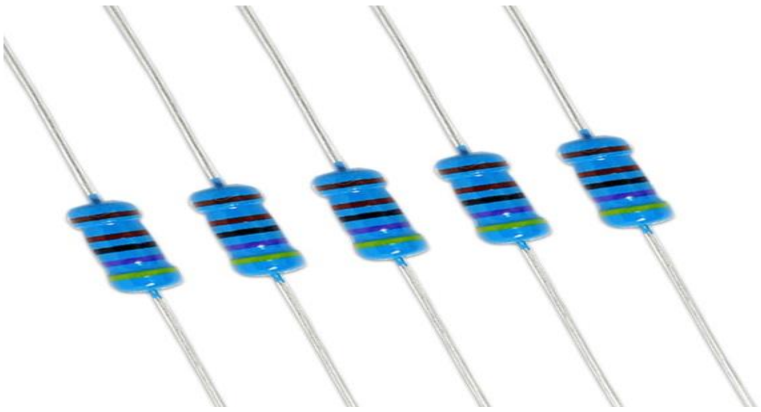


Fig. 2.2.1.4 Resistor [6]

Fig. 2.2.1.5 Potentiometer [7]

- RGB LED: - Light Emitting Diode (LED) is like a small bulb. It will glow up when a voltage is supplied to it. A resistor is always to be used limit the current otherwise it'll blowup the LED.

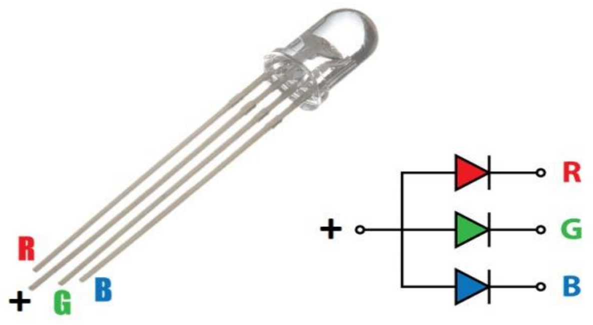


Fig. 2.2.1.6 RGB LED [8]

- Breadboard: - It is a solderless board with certain internal connections. We can spread out and connect different components in serial or parallel manner.

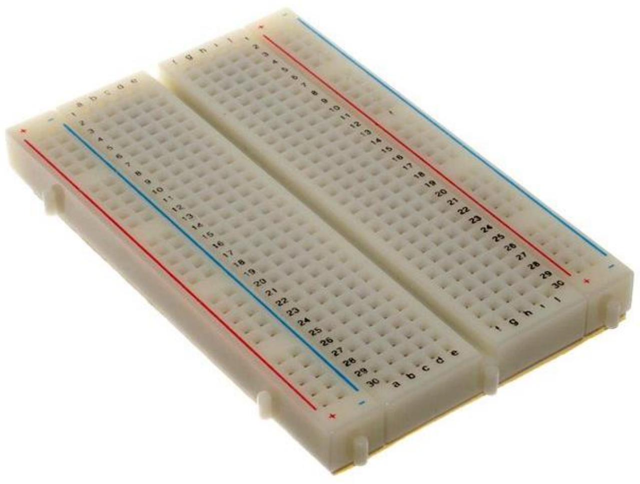


Fig. 2.2.1.7 Breadboard [9]

- **Servo Motor:** It is a motor in general, but, the shaft rotation can be controlled. The number can be supplied via a PWM output from the Arduino UNO.

Fig. 2.2.1.8 Servo Motor [10]

2.2.2 Software and Simulink Blocks

- **Blocks Input Pins**: - These are the blocks which can be used to receive a signal from the Arduino pins. The project has two inputs from the Arduino UNO. Each of them initiate a specific task/block. We have two different buttons like entry and exit.

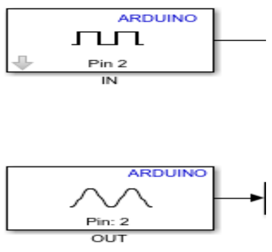


Fig. 2.2.2.1 Input Pin Blocks

- **Blocks Output Pins**: - These are the blocks which can be used to send a signal from the Arduino to the pins. We have two different outputs which are connected to two LEDs. These are being used as digital outputs for logic 1 and logic 0. The can have pa peripheral attaches to it. It will be an actuator.

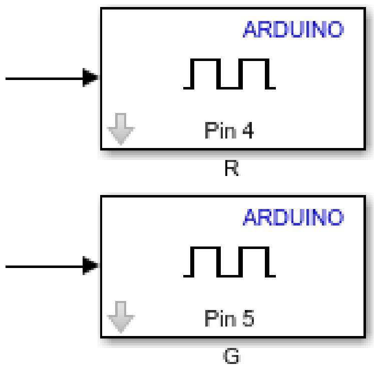


Fig. 2.2.2.2 Output Pin Blocks

- Counter and Logic: - This counter is used to count the time for which the arm remains open for the vehicle to go in or come out. This project uses the counter block is running at a sample time of 0.5. The following block is a comparator block. This comparator and counter combination keep the output 1 for 3 second in total and then 0 for rest.

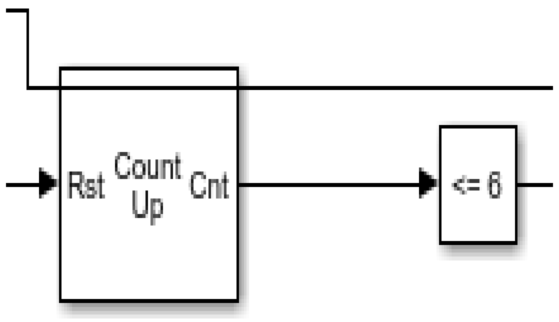


Fig. 2.2.2.3 Counter and Logic

- **Data Memory/Read/Write: -** Used for routing a signal in or out of particular block(s). Enables a user to use the same signal at multiple places. Here the block is used to save the spot variable. This stores the occupancy.

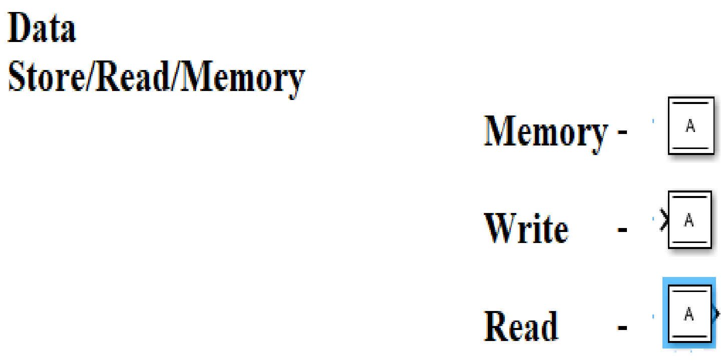


Fig. 2.2.2.4 Data Stores

- **LCD Block: -** This block is designed to display the content and makes it readable to the user. Also, there are certain other tasks which are programmed into it. This is programmed inside the S-block itself.

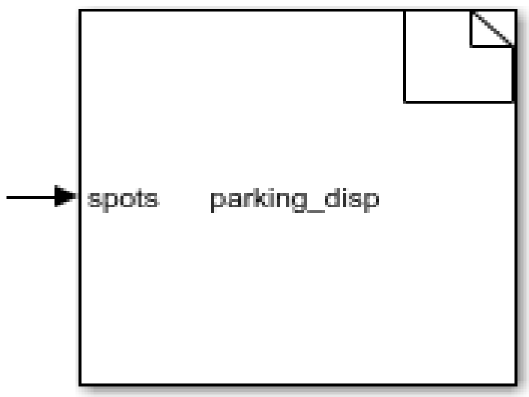
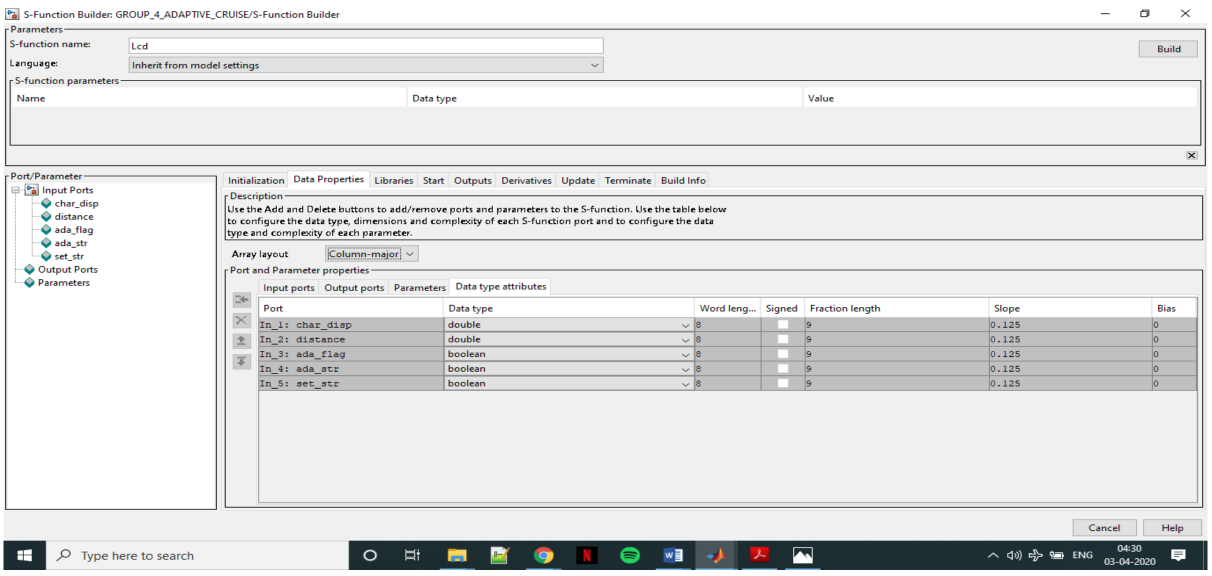


Fig. 2.2.2.5 LCD Module


S-Function Builder: GROUP_4_ADAPTIVE_CRUISE/S-Function Builder

Parameters

S-function name: Lcd

Language: Inherit from model settings

S-function parameters

Name	Data type	Value

Port/Parameter
- Input Ports
  - char_disp
  - distance
  - ada_flag
  - ada_str
  - set_str
- Output Ports
- Parameters

Initialization | Data Properties | Libraries | Start | Outputs | Derivatives | Update | Terminate | Build Info

Description

Use the Add and Delete buttons to add/remove ports and parameters to the S-function. Use the table below to configure the data type, dimensions and complexity of each S-function port and to configure the data type and complexity of each parameter.

Array layout: Column-major

Port and Parameter properties

Input ports | Output ports | Parameters | Data type attributes

Port	Data type	Word leng...	Signed	Fraction length	Slope	Bias
In_1: char_disp	double	8	☐	9	0.125	0
In_2: distance	double	8	☐	9	0.125	0
In_3: ada_flag	boolean	8	☐	9	0.125	0
In_4: ada_str	boolean	8	☐	9	0.125	0
In_5: set_str	boolean	8	☐	9	0.125	0

Fig. 2.2.2.6 Internal S-Block

- **Servo: -** The block forwards the angle to the servo motor connected to the PMW pin of the Arduino. The angle is divided between 0 and 180 degrees. Covering a semicircle. Negative angles have to be brought down to the numbers between the range.

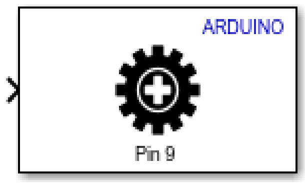


Fig. 2.2.2.7 Servo Write

- **Rise Detector: -** This block sets the input as 1, when it detects a rise. Rise basically when the signal transitions from logic ‘0’ to logic ‘1’. As this ia a positive ruse detector, only the event of the upward transition is detected. Further, irrespective of a fall or constant level of 0 or 1, the output stays zero

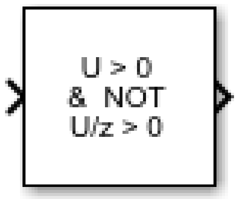


Fig. 2.2.2.8 Rise Detector

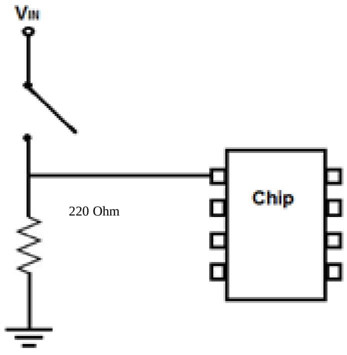
Implementation

3.1 Module 1

The project can be divided majorly into two modules, the hardware and the software. The hardware consisting all the connections and testing of the same using a different coding logic. This ensures that the connections are firm and secure and won't be altered further as well as signal flow becomes more realizable.

3.1.1 Hardware

- First the buttons were connected in a pull down configuration. So the by default value the pin holds is logic '0' or ground. And when the button is pressed the connection is completed and the value changes to logic '1' or Vcc. This is sensed by the Arduino and further execution is done.



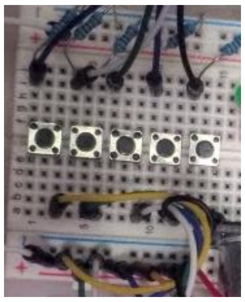


Fig. 3.1.1.1 Buttons

- Connection to the LCD was done by connecting the 6 pins of the LCD rs, en, D4-D7 to Arduino pins 8-13 respectively. The strings and the data will be provided by the Simulink and will be transferred to the LCD via the Arduino for it to display.

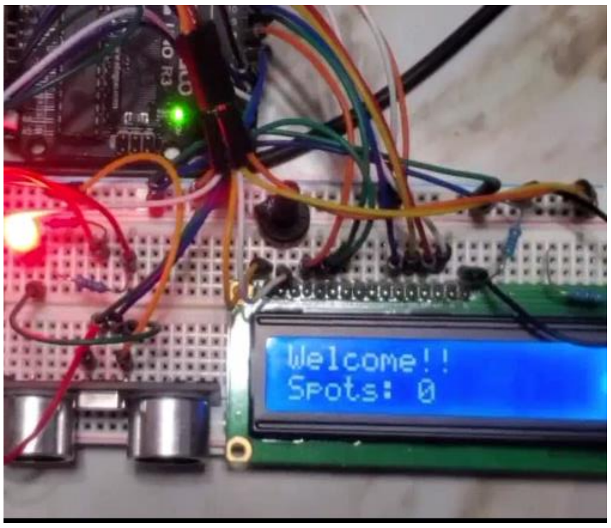


Fig. 3.1.1.2 LCD

- Connected the Servo to the Arduino. The trigger and echo pins were connected to the pins 4 and 5 of Arduino respectively. The distance value is sent to the Simulink for further processes. Ultrasound waves are used for the purpose of calculating the distance.

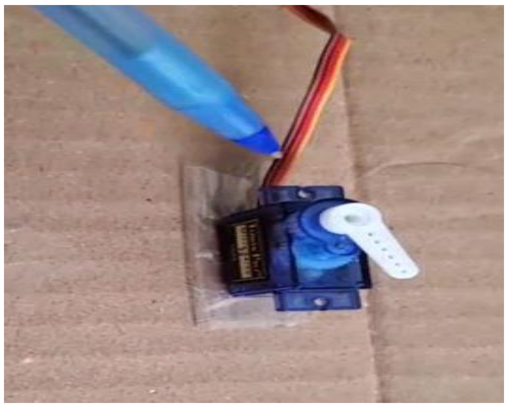


Fig. 3.1.1.3 Servo Motor

- Final assembly resulted in a proper working system for the software to execute the designed program. With the buttons, LCD and the RBG LED all working simultaneously.

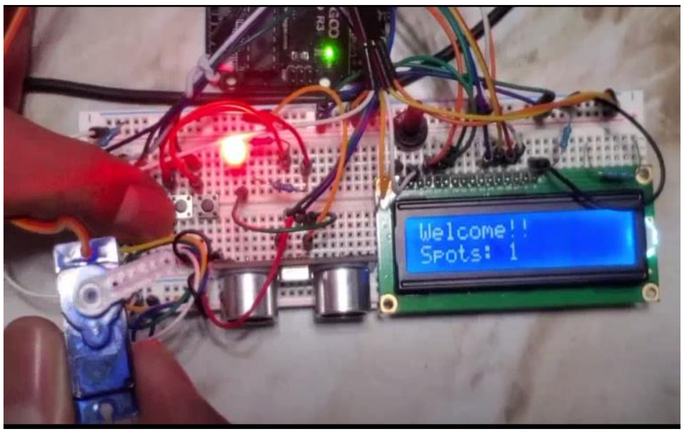


Fig. 3.1.1.4 Final Assembly

- The entry mode was tested to make sure that the servo only acts once per button press and not more even if the button is pressed and held. The number of spots increases per press.

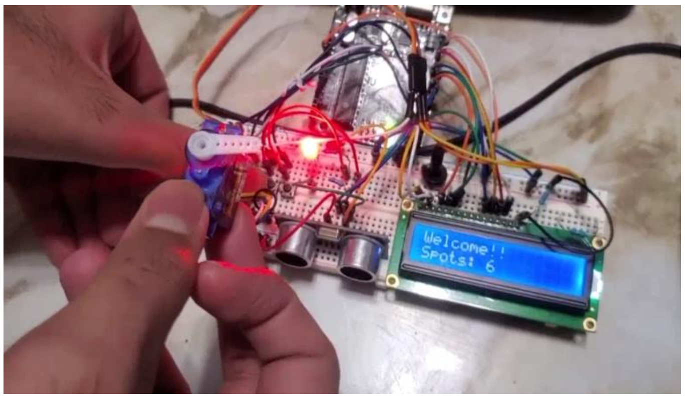


Fig. 3.1.1.5 Entry Demo

- The exit mode was tested to make sure that the servo only acts once per button press and not more even if the button is pressed and held. The number of spots decreases per press.

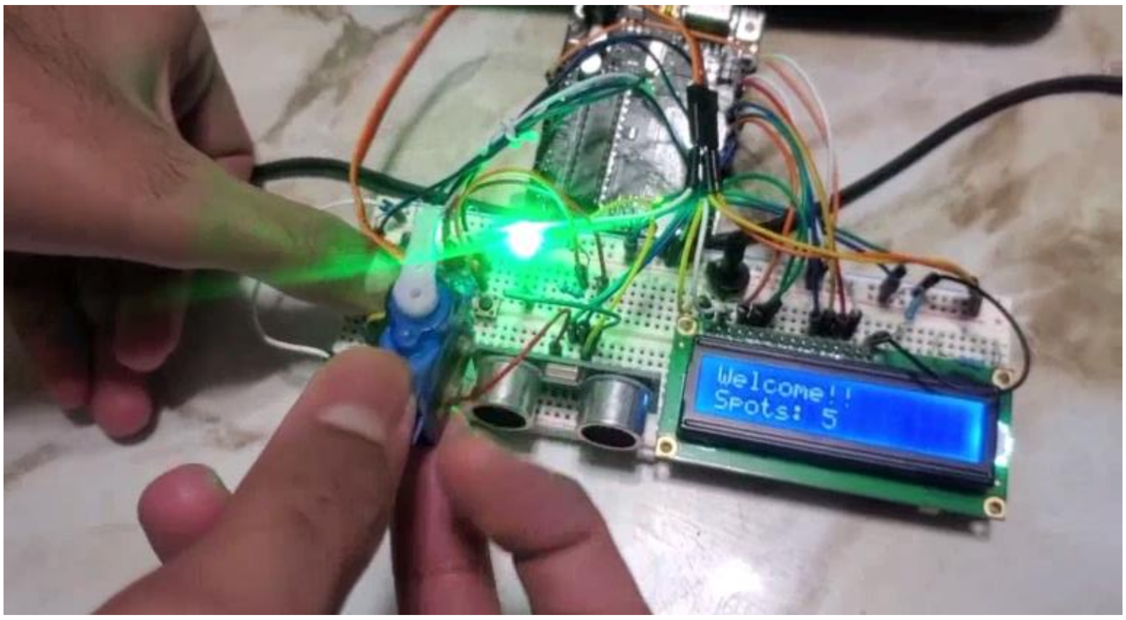


Fig. 3.1.1.6 Exit Demo

- When the total number of spots reaches '13', the message changes from 'Welcome' to 'Please come back later'. Also, on next press nothing happens and the arm doesn't move either.

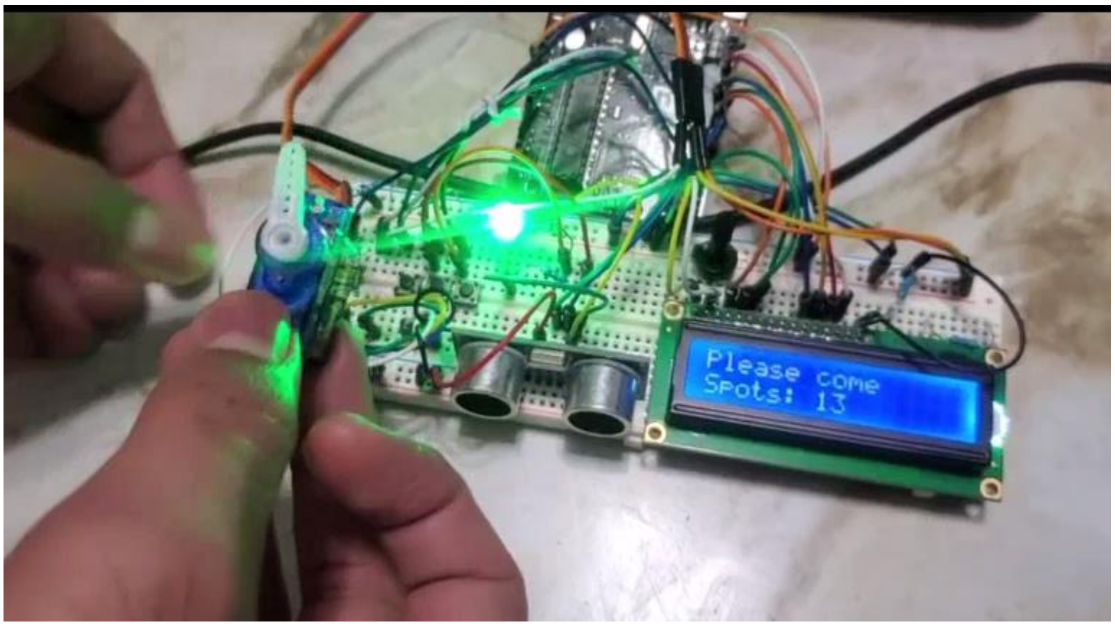


Fig. 3.1.1.7 Occupancy Message

3.2 Module 2

This module consists of 3 sections which are the 3 layers of the software part of the program. The Simulink model is responsible for all the processing and the calculations. Which are then forwarded or received by/to the Arduino board.

3.2.1 Layer 1

- This is the foundation layer which makes sure of the latching and triggering of the entire project. Also as it is the root layer it contains the memory blocks, due to which they are accessible by all the elements of Simulink. The pulse generator and LCD module are present too in the same layer.

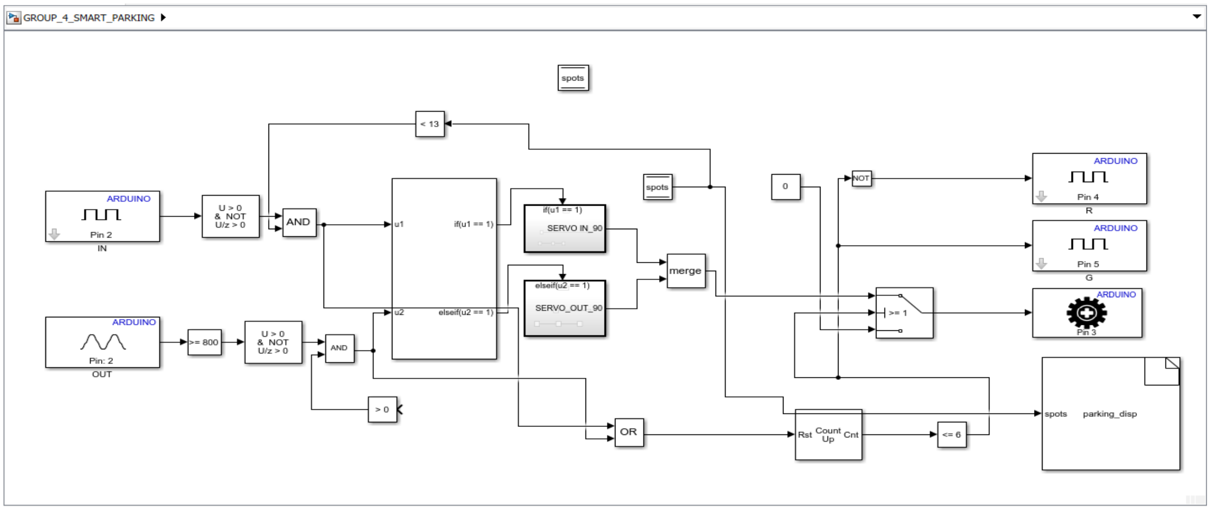


Fig. 3.2.1.1 Layer 1

3.2.2 Layer 1a

- The layer 1a is simply an increment part where the spot is increased and turns the arm once for opening.

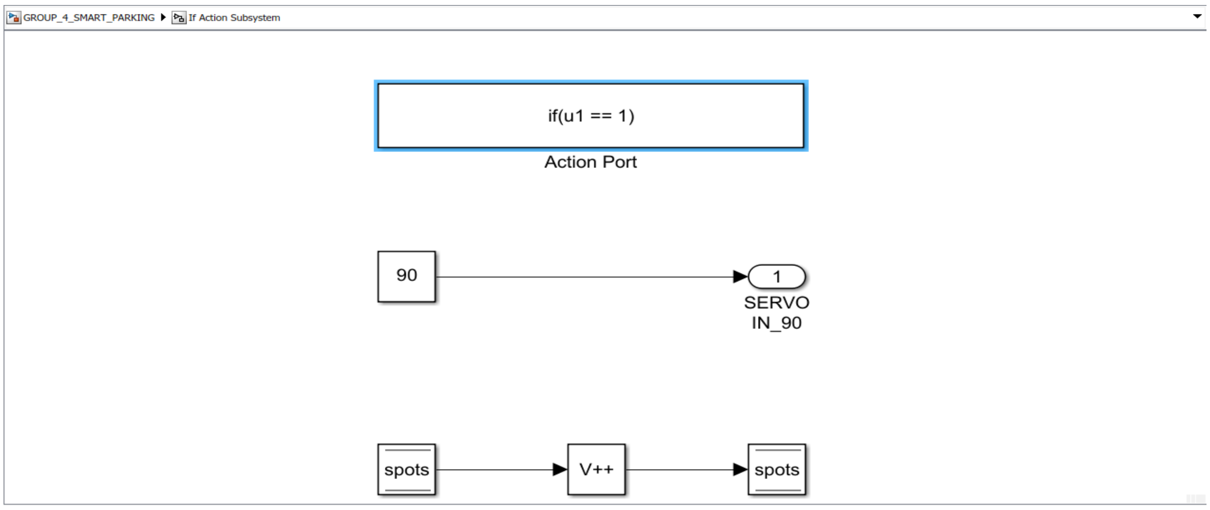


Fig. 3.2.1.2 Layer 1a

3.2.3 Layer 1b

- The layer 1b is simply a decrements part where the spot is increased and turns the arm once for opening.

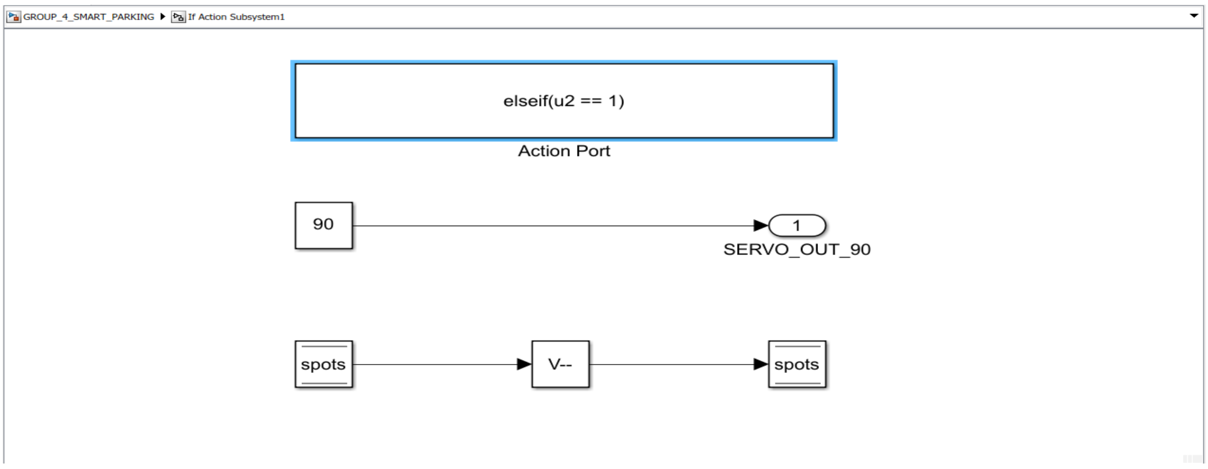


Fig. 3.2.1.3 Layer 1b

Challenges Faced

This section will be describing the challenges faced while designing the project. There were many obstacles which were hurdled through in order to get the complete result. These majorly include LCD, Signal routing etc.

- **LCD**:

  - **Challenge:** MATLAB doesn't have its own LCD module for external simulation, which can be used in the project. It does contain and LCD block but only for normal/in-system simulation and can't be deployed in the Arduino board. LCD stands for, which we will be interfacing with our Arduino hardware setup. As the professor asked if we can use LCD with Simulink and Arduino, we decided to proceed further and give it a try, even though as a matter of fact we knew it would not be easy.

  - **Challenge Handled:** Construction of the LCD module using the S-Block Builder in Simulink. This enabled us to transfer numeric data from the model to the LCD via Arduino. Although, strings couldn't be transmitted, as it a restriction of the Simulink software and incompatibility between the S-Block builder and string library modules

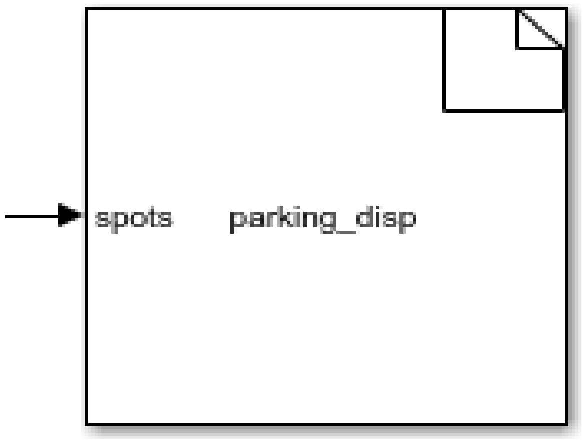


Fig. 4.1 LCD

- **Data Routing**

  - **Challenge:** Just like any other technical project, this adaptive model is a combination of hardware as well as software too. A con of doing it on Simulink is that unlike Arduino IDE, we cannot write a variable, assign a value to it and use it in or model as and when we require. But, in this project, certain blocks/sub-systems are supposed to be triggered only when a certain button is pressed.

- **Challenge Handled:** We used the memory blocks and assigned each of them to different sub-systems as we required. Each come with a set of read, write and store sections. The read and memory is used once while write is used multiple times as per the requirement

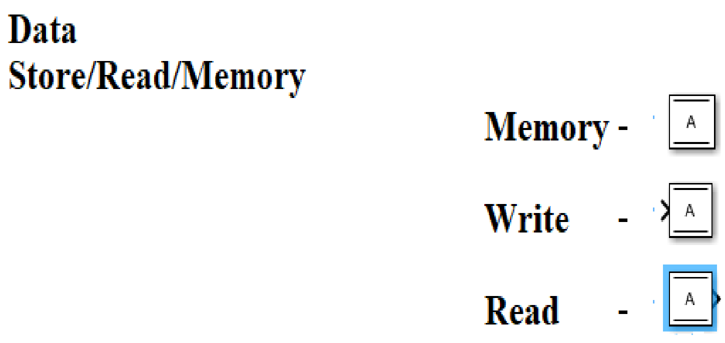


Fig. 4.2 Data Store

➢ One Shot Execution

- **Challenge:** It was challenging to develop a system which would only once per press of the button. The user should have no constriction on the button pressing timing. But something was required to ignore the pressing time and only count the instant of occurrence.

- **Challenge Handled:** The rise positive block makes sure that only the occurrence is recorded and output is produced during the same.

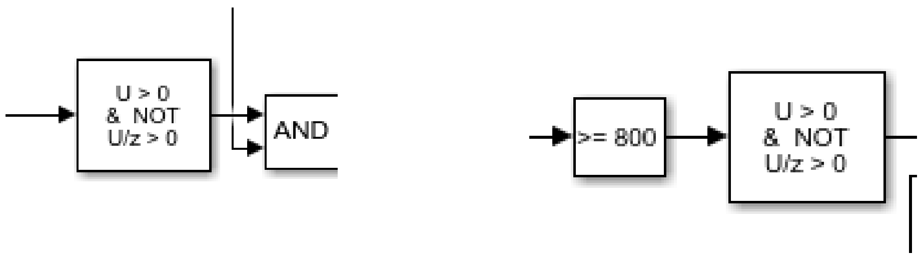


Fig. 4.3 Positive Rise Detector

➢ Arm Rotation Time

- **Challenge:** The model turns the arm of the servo motor by sending the signal once on the corresponding PWM pin of the Arduino. Now we need something to turn back the arm to rest position after a certain amount of time, even though the button is released. Which made it complicated, as Simulink doesn’t support ‘pause()’ function. Which could make it wait for the desires time interval before the arm comes back.

- **Challenge Handled:** We introduced a combination of counter and comparator block. The counter is reset every time the button is pressed and starts counting 6 seconds from them. This count input is fed to a counter which triggers a switch and makes sure that after the desired interval the arm comes back to the resting position.

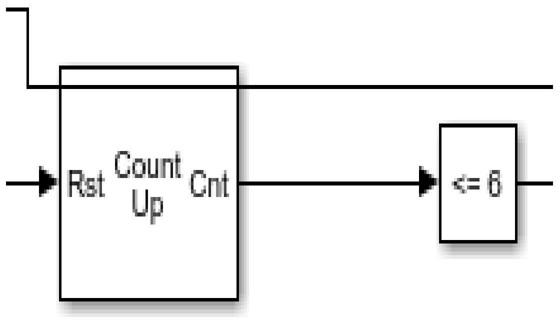


Fig. 4.4 Counter and Logic

Functioning

5.1 Block Diagram of Project

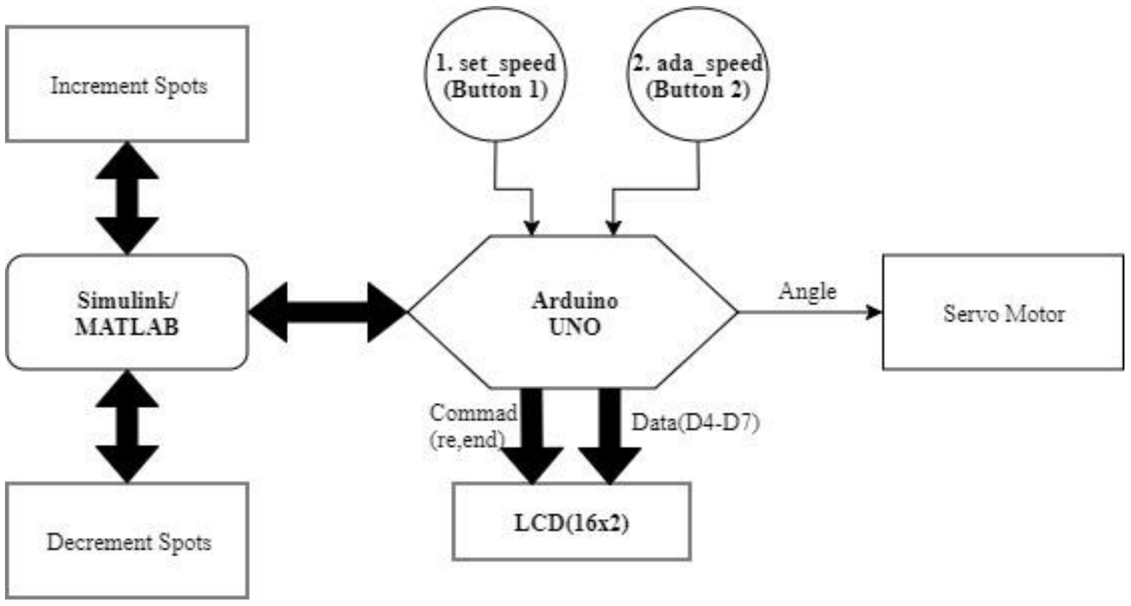


Fig. 5.1 Block Diagram [10]

5.2 Flowchart of Adaptive Cruise Control

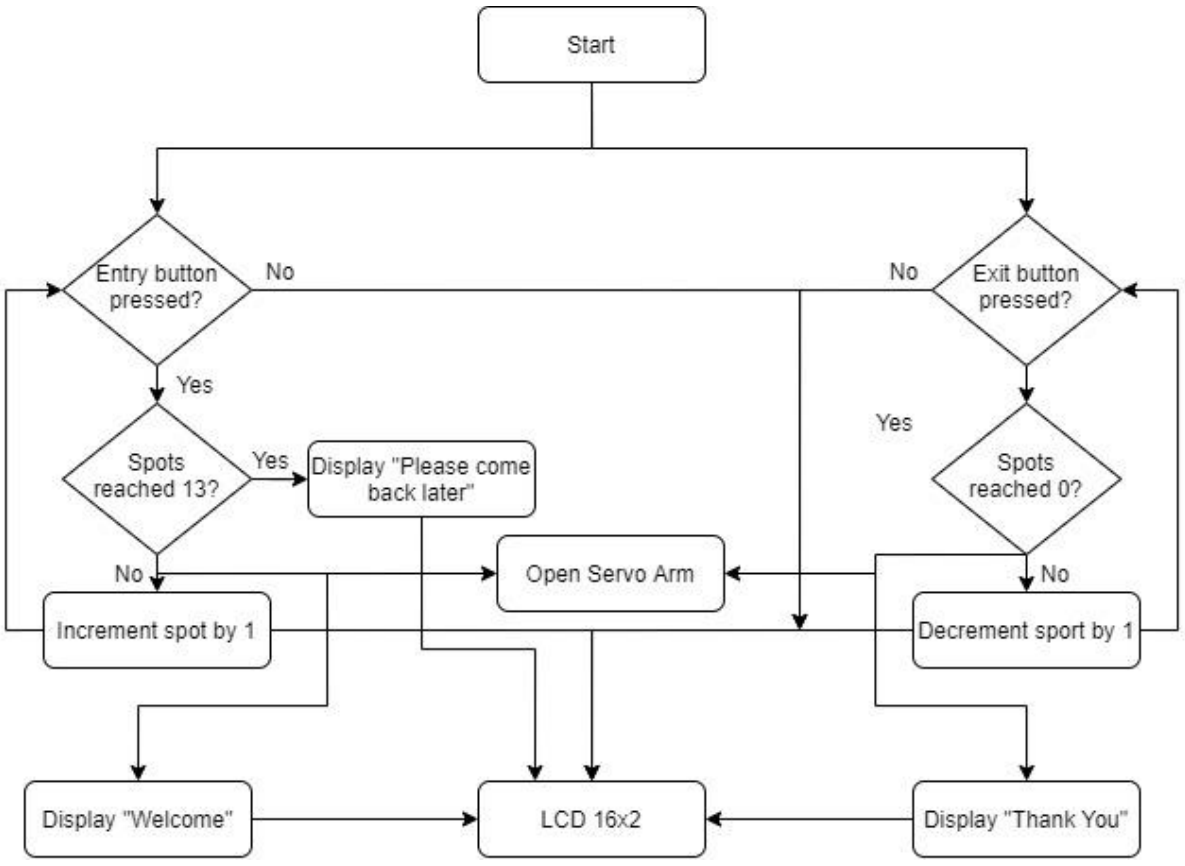


Fig. 5.2 Flowchart [10]

5.3 LCD S-Block Code

- So we need to create a S-Block which will run as an LCD module there was some in-programming needed to be done. So the S-Block code was written to display execute considering other flags.

```
#ifndef MATLAB_MEX_FILE
   lcd.begin(16,2);
   lcd.setCursor(0,1);
   lcd.print(F("Spots: "));
   lcd.print(spots[0]);
   if (spots[0]==13)
   {
      lcd.setCursor(0,0);
      lcd.print(F("Please come"));
      delay(600);
      lcd.setCursor(0,0);
      lcd.print(F("back later     "));
      delay(600);
   }
   else
   {
      lcd.setCursor(0,0);
      lcd.print(F("Welcome!!      "));
   }

#endif
```

Results

The Simulink now can display the number of spots occupied and depending upon the occupancy and switch between displays of ‘Welcome’ and ‘Please come back later’. Also, the arm stays up for three seconds, for demonstration purposes only. This system is perfectly capable of being implemented in real life. All we have to do is increase the wait time of the system. This can be achieved by making a simple increase in the comparator block.

References

[1] http://ctms.engin.umich.edu/CTMS/index.php?aux=Basics_Simulink
[2] https://create.arduino.cc/projecthub/products/arduino-uno-rev3
[3] https://oneguyoneblog.com/2016/08/08/geekcreit-uno-16x2-lcd-display/
[4] https://www.sparkfun.com/products/97
[5] Res https://www.pchcables.com/63-1039.html
[6] https://www.adafruit.com/product/562
[7] https://en.wikipedia.org/wiki/Breadboard#/media/File:400_points_breadboard.jpg
[8]https://www.google.com/search?q=RGb+LED&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjE9KvmiOLoAhUjlXIEHQhgBLQQ_AUoAXoECA4QAw&biw=1366&bih=657#imgrc=9bVDx1SS_McfJM&imgdii=eK6HEABWi0p1oM

[9]https://www.google.com/search?q=servo+motor&tbm=isch&source=iu&ictx=1&fir=-G6hbn0R3pH7gM%253A%252Czmx6SRRoUcjwhM%252C%252Fm%252F0858kcn&vet=1&usg=AI4_-kSrAgz5_Pyq25GKXQz0a7KqG_6zxw&sa=X&ved=2ahUKEwjd6pW0i-LoAhURVd8KHZu8AHkQ_B0wIHoECAMQAw#imgrc=Zh2tdeycmbW_QM&imgdii=lV_Kt4IE6nU2mM

[10] https://app.diagrams.net/