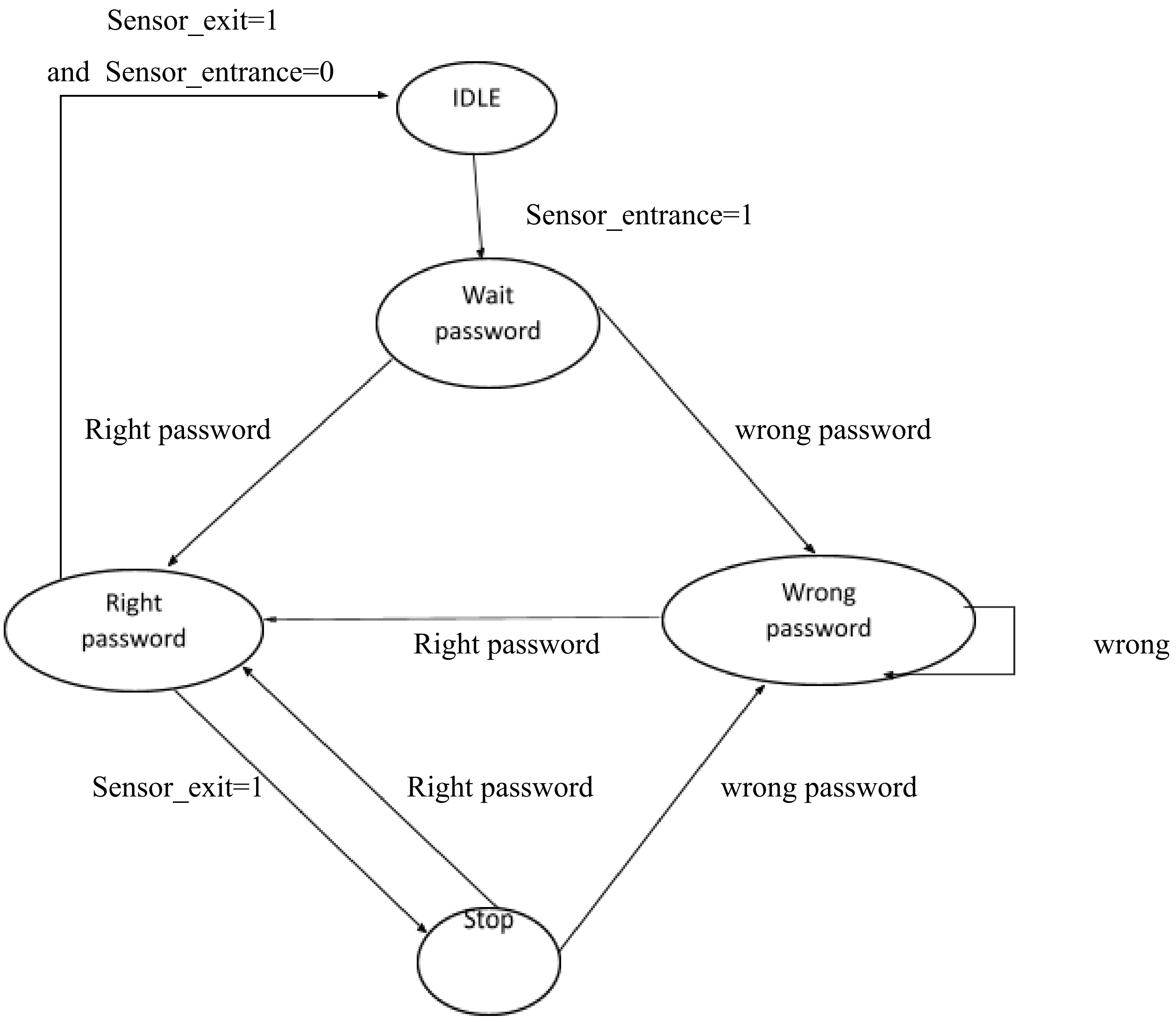
INTRODUCTION:

In this project, our aim is to implement a digital car parking system in Verilog and show its circuit implementation in proteus.

MAIN IDEA:

In the entrance of the parking system, there is a sensor which is activated to detect a vehicle coming. Once the sensor is triggered, a password is requested to open the gate. If the entered password is correct, the gate would open to let the vehicle get in. Otherwise, the gate is still locked. If the current car is getting in the car park being detected by the exit sensor and another car comes, the door will be locked and requires the coming car to enter passwords.

The flowchart of our project is given below:



Here, we can see there is 5 states which are

Idle, Wait password, Right password, Wrong Password and Stop.

Actually, its working follows Moore type sequential circuit because output just depends on the current state.

When there is no car in front of the parking gate that state indicates “Idle” state. There is a sensor named sensor_entrance in the parking gate. When a car crosses this gate, this sensor_entrance will be activated (sensor_entrance=1) and we will enter our next stage “Wait password”. In this stage the system will wait for a valid password. After entering the correct password, the gate will open and the car will enter the parking area. There will be another sensor named sensor_exit in the parking area. After parking the car at its own slot, the sensor_exit=1 and then the current state will be “Stop”.

If in the “Wait password” state , someone enters the wrong password the gate will not open and then the current state will be “Wrong password” until the current password is entered.

When a car wants to exit from the parking, sensor_exit keeping in its parking slot will activate and after entering the correct password the parking gate will open and the car exits from parking.

VERILOG CODE WITH COMMENTS:

```
module parking_system(
                input clk,reset_n,
 input sensor_entrance, sensor_exit,
 input [1:0] password_1, password_2,
 output wire GREEN_LED,RED_LED,
 output reg [6:0] HEX_1, HEX_2
   );
 parameter IDLE = 3'b000, WAIT_PASSWORD = 3'b001, WRONG_PASS = 3'b010, RIGHT_PASS = 
3'b011,STOP = 3'b100;
 // Moore FSM : output just depends on the current state
 reg[2:0] current_state, next_state;
 reg[31:0] counter_wait;
 reg red_tmp,green_tmp;
 // Next state
```

```
always @(posedge clk or negedge reset_n)//asynchronus clk

begin

if(~reset_n)

current_state = IDLE;

else

current_state = next_state;

end

// counter_wait

always @(posedge clk or negedge reset_n)

begin

if(~reset_n)

counter_wait <= 0;

else if(current_state==WAIT_PASSWORD)

counter_wait <= counter_wait + 1;

else

counter_wait <= 0;

end

// change state


always @(*)

begin

case(current_state)

IDLE: begin

     if(sensor_entrance == 1)

next_state = WAIT_PASSWORD;
```

```
else
next_state = IDLE;
end
WAIT_PASSWORD: begin
if(counter_wait <= 3)
next_state = WAIT_PASSWORD;
else
begin
if((password_1==2'b01)&&(password_2==2'b10))
next_state = RIGHT_PASS;
else
next_state = WRONG_PASS;
end
end
WRONG_PASS: begin
if((password_1==2'b01)&&(password_2==2'b10))
next_state = RIGHT_PASS;
else
next_state = WRONG_PASS;
end
RIGHT_PASS: begin
if(sensor_entrance==1 && sensor_exit == 1)
next_state = STOP;
else if(sensor_exit == 1)
next_state = IDLE;
```

```
else
next_state = RIGHT_PASS;
end
STOP: begin
if((password_1==2'b01)&&(password_2==2'b10))
next_state = RIGHT_PASS;
else
next_state = STOP;
end
default: next_state = IDLE;
endcase
end
// LEDs and output, change the period of blinking LEDs here
always @(posedge clk)
begin
case(current_state)
IDLE: begin
green_tmp = 1'b0;
red_tmp = 1'b0;
HEX_1 = 7'b1111111; // off
HEX_2 = 7'b1111111; // off
end
WAIT_PASSWORD: begin
green_tmp = 1'b0;
red_tmp = 1'b1;
```

```
HEX_1 = 7'b000_0110; // E

HEX_2 = 7'b010_1011; // n

end

WRONG_PASS: begin

green_tmp = 1'b0;

red_tmp = ~red_tmp;

HEX_1 = 7'b000_0110; // E

HEX_2 = 7'b000_0110; // E

end

RIGHT_PASS: begin

green_tmp = ~green_tmp;

red_tmp = 1'b0;

HEX_1 = 7'b000_0010; // 6

HEX_2 = 7'b100_0000; // 0

end

STOP: begin

green_tmp = 1'b0;

red_tmp = ~red_tmp;

HEX_1 = 7'b001_0010; // 5

HEX_2 = 7'b000_1100; // P

end

endcase

end

assign RED_LED = red_tmp  ;

assign GREEN_LED = green_tmp;
```

endmodule

EXPLANATION OF CODE:

Here, we have 6 inputs which are clock, resetn (for reset the whole system, active low), Sensor_entrance, Sensor_exit and 2 passwords named password_1 and Password_2.

We define password_1=01 and password_2=10 as correct password.

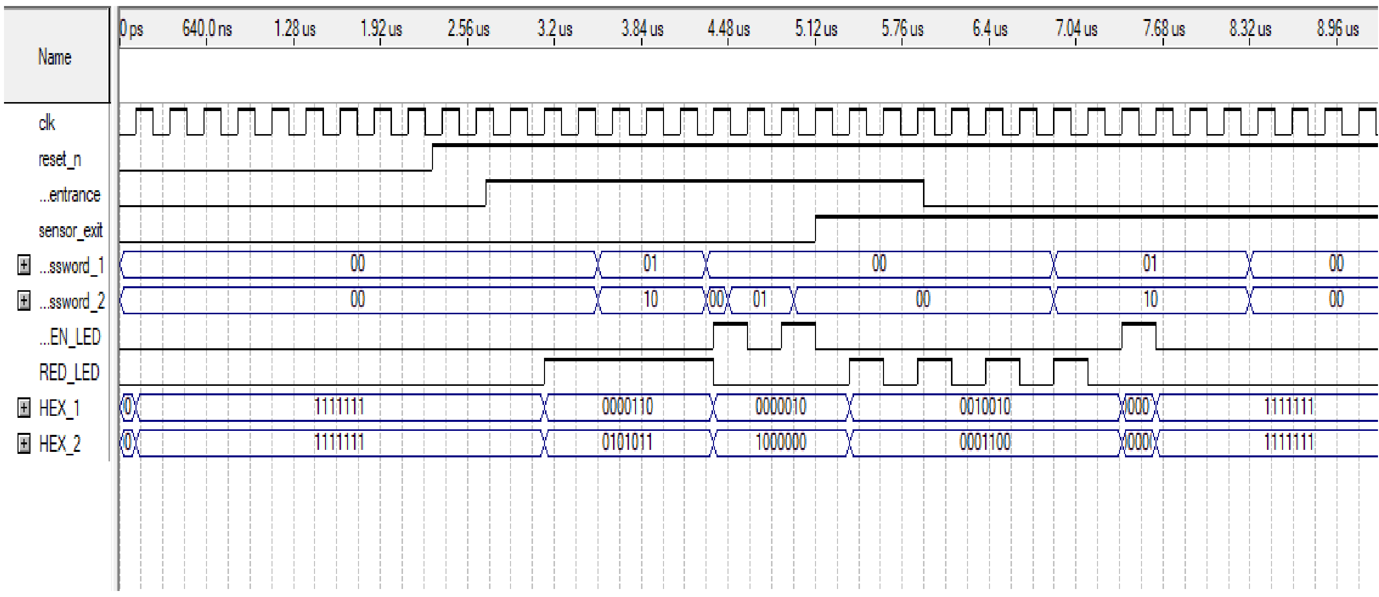
Outputs of our code is 2 led light one is green led and other a red led. And other 2 outputs Hex_1 and Hex_2 are indicated the current state of the system.

For our five different states, we have following desired outputs:

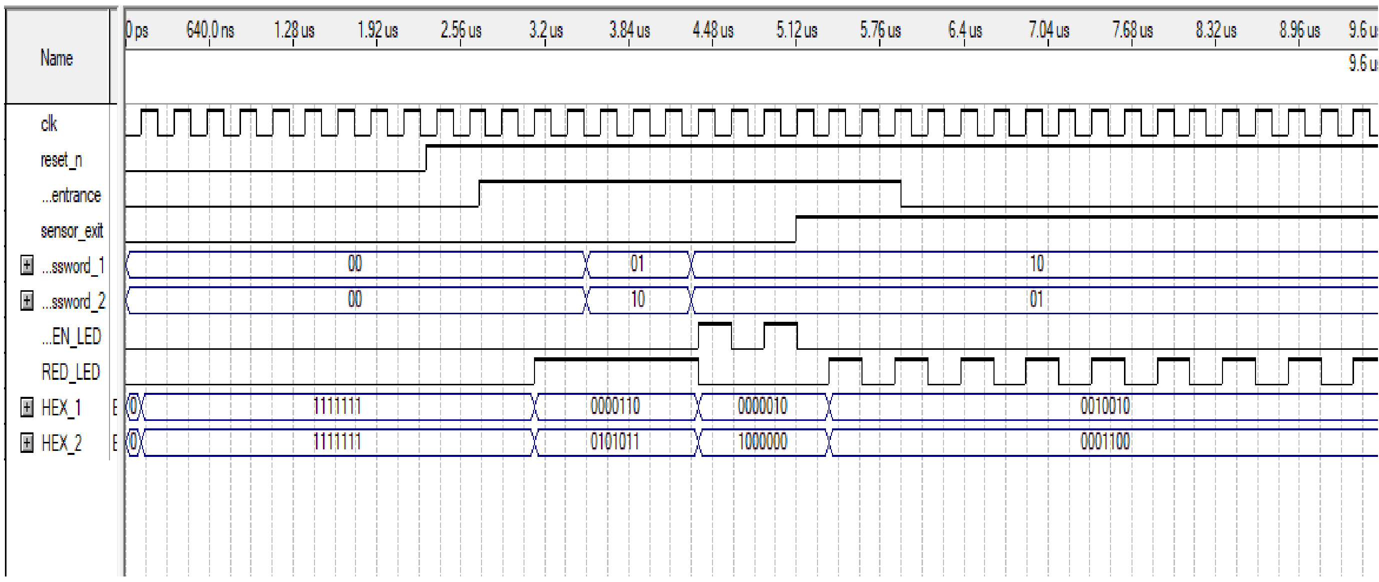
Current state	Outputs			
	Green led	Red led	Hex_1	Hex_2
Idle	0	0	111 1111	111 1111
Wait password	0	1	000 0110	010 1011
Wrong password	0	~r	000 0110	000 0110
Right password	~g	0	000 0010	100 0000
Stop	0	~r	001 0010	000 1100

OUTPUTS:

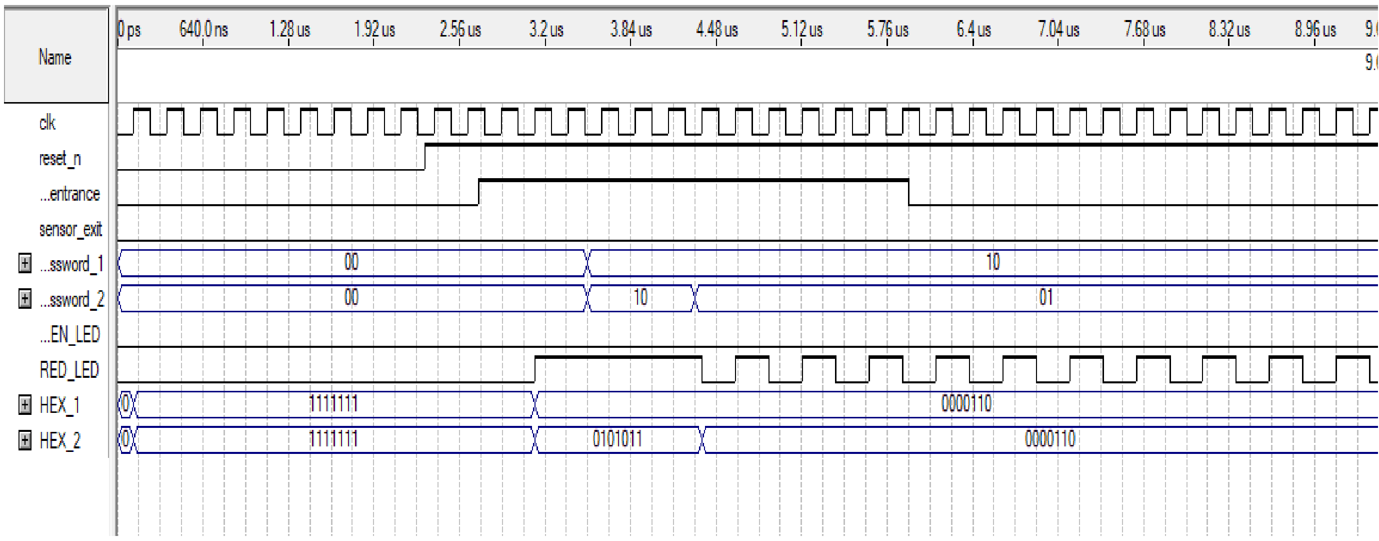
1. For idle-wait password-right password-stop-right password-idle



2. For idle-wait password-right password-stop



3. For idle-wait password-wrong password

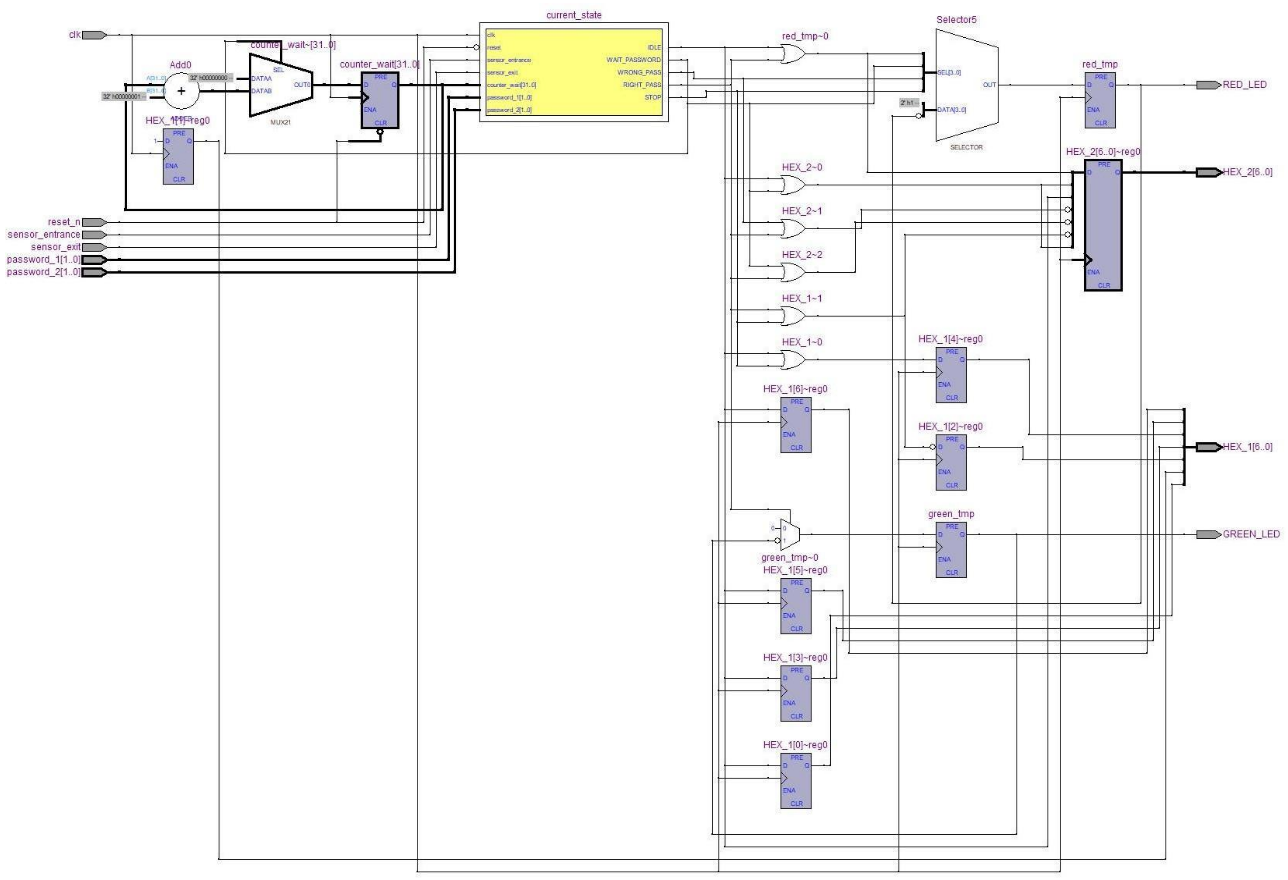


There is a error in our code as entering correct password after wrong password, current state doesn’t change.

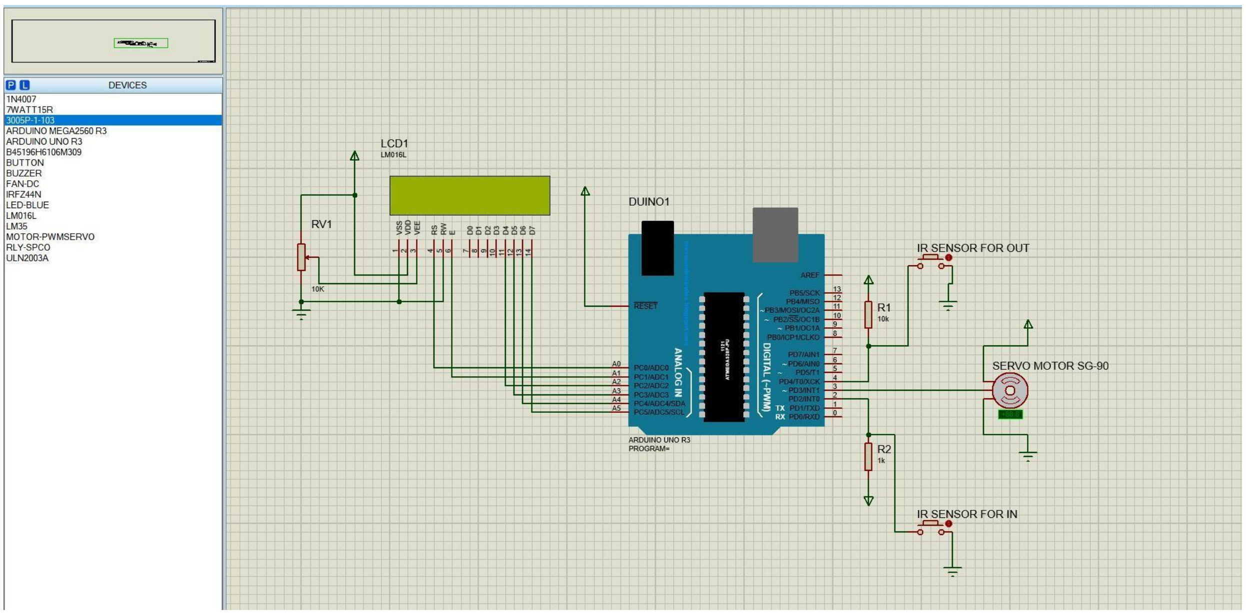
Part B:

Proteus Implementation of Smart Parking System :

Schematic netlist generated from Verilog :



Design Schematic:



<u>Arduino Code and explanation :</u>

Code:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(A0, A1, A2, A3, A4, A5);
#include <Servo.h>

Servo myservo1;


int ir_s1 = 2;
int ir_s2 = 4;


int Total = 5;
int Space;


int flag1 = 0;
int flag2 = 0;


void setup() {
pinMode(ir_s1, INPUT);
pinMode(ir_s2, INPUT);


myservo1.attach(3);
myservo1.write(100);


lcd.begin(16, 2);

lcd.setCursor (0,0);
lcd.print("  Car
Parking      "); lcd.setCursor
(0,1); lcd.print("
             System        ");
delay (2000);

lcd.clear();


Space = Total;

}


void loop(){


if(digitalRead (ir_s1) == LOW && flag1==0){
if(Space>0){flag1=1;
if(flag2==0){myservo1.write(0); Space = Space-1;}
}else{

lcd.setCursor (0,0);
lcd.print(" Sorry not Space
"); lcd.setCursor (0,1);
lcd.print("    Available
"); delay (1000);

lcd.clear();

}

}


if(digitalRead (ir_s2) == LOW && flag2==0){flag2=1;
if(flag1==0){myservo1.write(0); Space = Space+1;}

}


if(flag1==1 && flag2==1){
delay (1000);
myservo1.write(100);
flag1=0, flag2=0;

}


lcd.setCursor (0,0);


lcd.print("Total Space: ");
lcd.print(Total);


lcd.setCursor (0,1);
lcd.print("Have  Space: ");
lcd.print(Space);

}
```

Explanation:

1. We initialized the library with the numbers of required interface pins
2. Included the servo library and initialized our myservo1.
3. Inside void setup() , pinMode() function is used to configure a specific pin to behave either as an **input** or an **output**
4. We attach the Servo with a variable and write 100 which is the angle of the shaft in degrees.
5. lcd.begin() will Initializes the interface to the LCD screen, and specify the dimensions (width and height) of the display to be (16,2)
6. To write to the display we need to position the LCD cursor and print required text in it.

7. Inside the void loop(), DigitalRead function reads the value from a digital pin and sets sensor1 & sensor2 with appropriate logic. It outputs HIGH & LOW and based on this output we execute the entrance and exit of the car.

Application sectors of this project:

- Garages
- Parking startups
- Buildings

Advantages:

- Inefficient Parking Management Worsens Traffic
- Smart Parking as a Solution to Traffic Woes
- Smart Parking Eliminates Distractions
- Smart Parking Reduces Cruising Time
- Benefits to Stakeholders

Extended activities :

- Incorporating number plate detection which will help secure entrance to homes and office buildings .
- Implementing Smart navigation system towards parking lot
- Web based service can be implemented integrating real-time data of available spaces in Dhaka.

Conclusion:

Smart parking technology combines the use of sensors, street lights, smart navigation systems, and online payment platforms to relay information to drivers and parking lot operators. The real-time data collected from the sensor system is translated into actionable insights on smart parking applications. These are then used by drivers to take the least congested route, get an overview of the parking options, and make payment at the touch of a button.

Creating more **parking spaces** won't necessarily solve traffic congestion in densely populated cities like Dhaka. Instead, we need better management of the existing spaces. **Smart parking technology** holds the key to easing traffic by smoothening the parking process for both drivers and operator