National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Embedded System Design

Automated Parking Lot

Project Report

Group Members

Momin Munir

Hasan Saeed

Saqib Javed

Abbas Ahmed Siddiqui

Ibrahim Zafar

Saad Abdul Majid

Instructor: Sir Awais Kamboh

Date: 17 Jan 2017

National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

INTRODUCTION

A very common problem today is faced in the parking lots today when drivers do not know if a parking space is available in the parking lot. They have to spend a lot of time and fuel as well to search for a space in the parking lot and this is very frustrating and time wasting. Automated Parking Lot aims to solve this issue very efficiently. Once implemented in a parking lot, the drivers will no longer have to drive themselves all around the place to look for a space. The will be informed before they enter whether a space exists, via a big LCD screen, and also where exactly is an empty slot.

ABSTRACT

Our project smartly tackles the parking problem that occurs anywhere where managing parking is an issue. Entry and exit barriers are controlled through multi-level IR sensors, 2 sensors at each barrier. 2 sensors are placed in order to detect the presence of vehicle and also the size of vehicle according to which the barrier will be lifted to a particular height. The counts status is updated on LCD display. Switching of parking lights is done through LDR and through presence of vehicle as well. We will also be doing multiplexing to support 16 or more vehicle slots through a single controller. This will help the car entering to know where the empty car spaces are which will help in reducing the fatigue and time for the driver. The project is just a downsized model. When implemented in the real world, a large LCD screen will be used to display the status of the parking lot and the spaces which are empty and occupied.

MOTIVATION AND PROBLEM STATEMENT

"The typical driver spends 106 days of their life searching for a parking space, according to a new survey."-Telegraph

The average motorist wastes a total of 2,549 hours circling the streets searching for a space whether it is on the school run, the local high street or a supermarket or airport car park. Across the UK, it takes an average of six minutes and 45 seconds to find a suitable space – but this is just the average, according to the survey by Parkatmy House.
For those in London, it takes 20 minutes to find somewhere to park thanks to yellow lines, meters and areas restricted to resident parking.

A survey of drivers done by the Telegraph found that 81 per cent say it often takes 20 minutes or more to find somewhere to park, with 45 per cent describing parking as their biggest motoring headache.

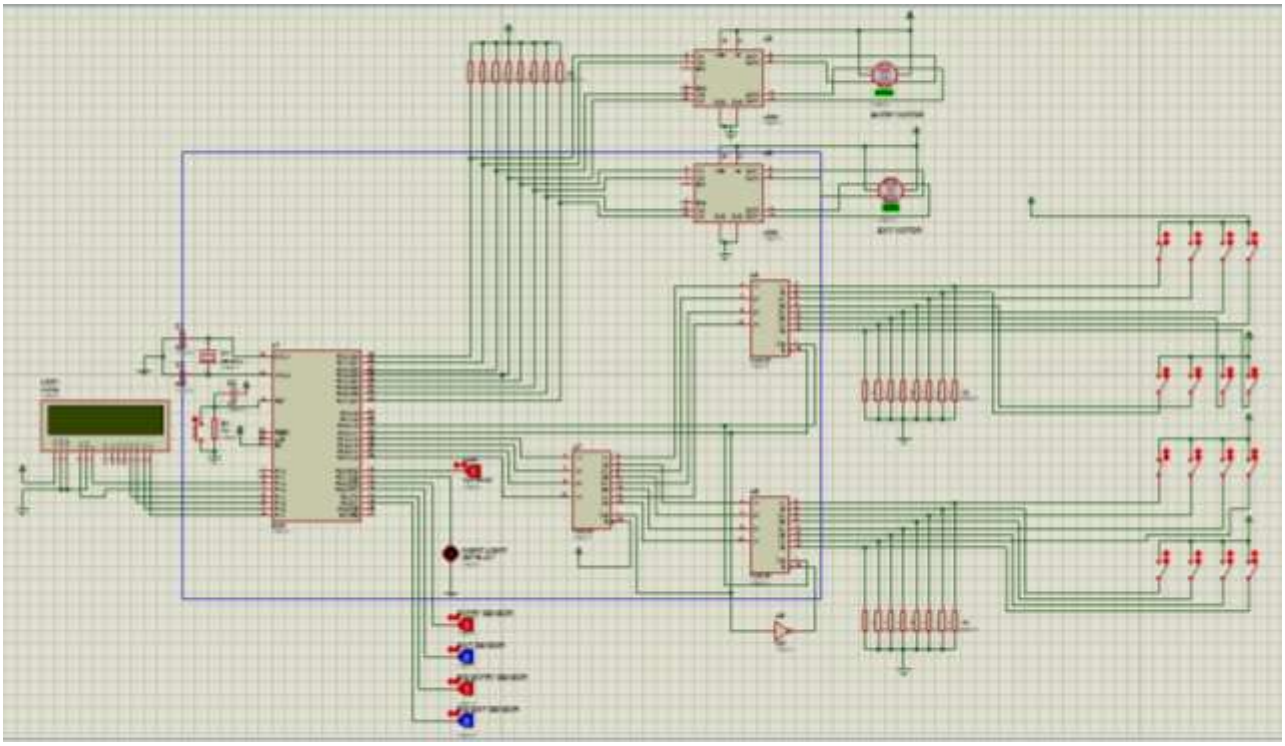
An article in March's Time Magazine cites research done by UCLA urban-planning professor Donald Shoup to show how much waste is generated by cruising for parking.

"Studies suggest that up to 30% of downtown drivers may just be looking for a place to park their cars. Shoup's research, conducted on 15 blocks near the UCLA campus, illustrates how a little cruising can add up. He and his students found that the average time a driver spent hunting for parking was 3.3 min.; the average distance covered was a half-mile. That means that over the course of a year, the search for parking around just the Los Angeles campus would add up to 950,000 miles of travel, along with 47,000 gallons of wasted gas and 730 tons of greenhouse gas emissions. "In a day, the amount of cruising was more than the distance across the U.S.," Shoup says.

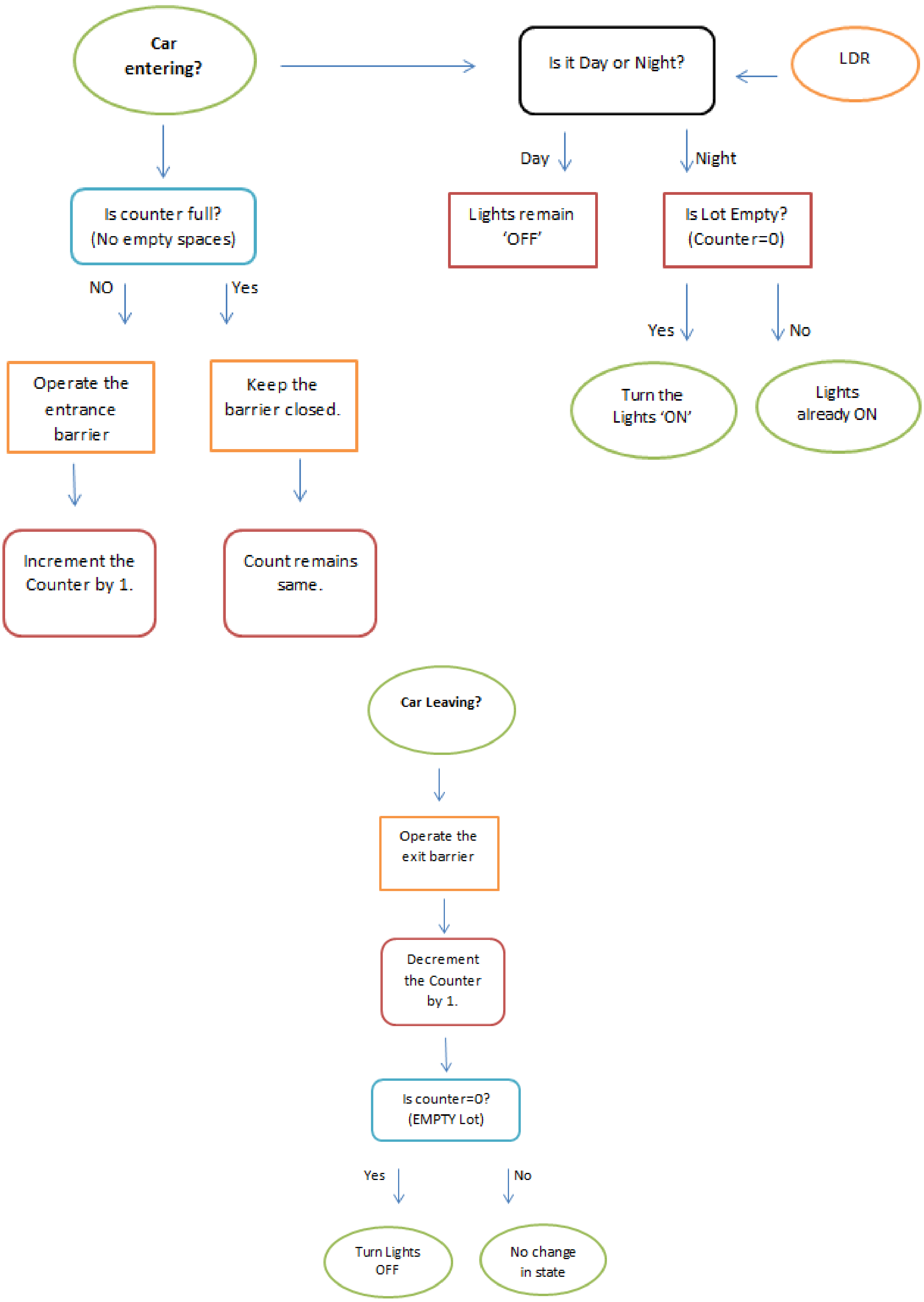
OBJECTIVES

- Controlling the Stepper Motor to lift to an angle corresponding to the type of vehicle entering the parking lot.
- Showing available parking spaces using demultiplexing
- Displaying the amount of cars present in the parking lot using an LCD
- Smart lighting using automated lighting system using LDR
- Simulation the circuit on Software (Proteus)
- Implementing the circuit on Hardware using an 8052 microcontroller

HARDWARE AND SCHEMATIC DIAGRAM



CONTROL FLOW DIAGRAM



HARDWARE COMPONENTS USED

1. 16 LED's
2. LDR
3. 16x2 LCD
4. 2 Stepper Motors
5. 8052 Microcontroller
6. Switches for Parking Spaces
7. 4 IR sensors
8. Three 4 bit 2x1 De-Multiplexers for multiplexing parking spaces
9. Resistors for LEDs
10. Stepper Motor driving IC (L293D)
11. Crystal Oscillator
12. NOT gate
13. Wires
14. Capacitors for reset circuit

LIST OF SOFTWARE USED

1) Proteus
2) Keil

LIBRARIES

3) Stdio.h
4) Reg51.h

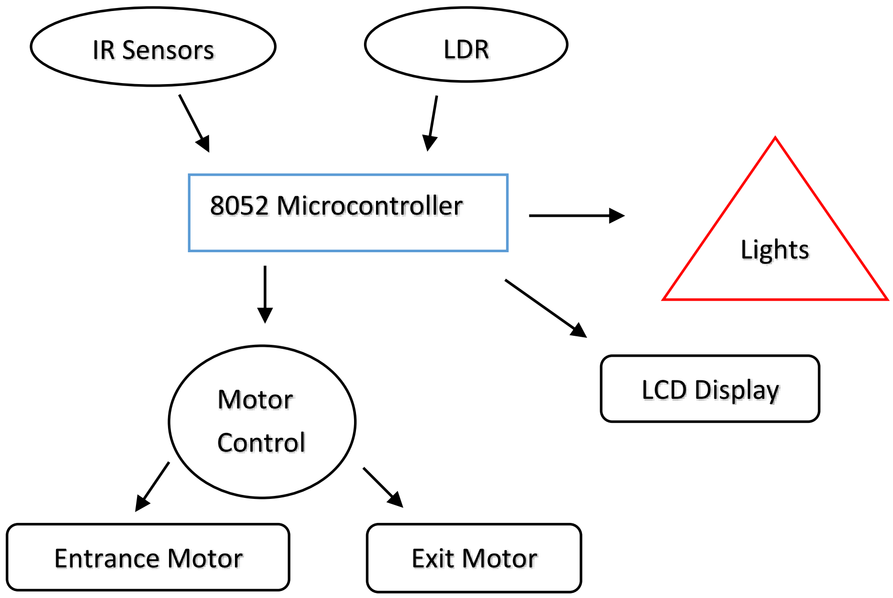
DESIGN and WORKING

At the start/after reset the parking lot is in a state where there are no cars in the parking lot. All the LEDs show that all the spaces are available. The LCD reads a message that says No cars in the lot. In front of the entrance barrier there are 2 infra-red sensors positioned. When a car enters the IR sensors detect it. The second IR sensor is placed at a little height above the first one. This IR sensor is used in determining whether it's a car or a truck/bus that is entering. If the vehicle is a truck for example, because of its height it will reflect back the rays of both IR sensors instead of just the lower one. Using the information from these 2 sensors the barrier is lifted with the use of the stepper motor and the elevation angles depends on the type of vehicle. If the vehicle is a higher one the barrier will have a larger angle of elevation and vice versa. When the barrier lifts and then shuts it is an indication that a car has entered the lot and the count on the LCD screen increments by one. Moving ahead, every parking space is represented by a switch button. When the vehicle parks in a slot, that slots' switch turns on and the LED corresponding to that space lights up. The Microcontroller is informed about which slot exactly has been occupied by a de-multiplexing process.

The leaving process of a car from the lot works in a similar way. The IR sensors in this case have a similar arrangement that is one being at some height above the other. However unlike the entrance arrangement the IR sensors are placed prior to the exit barrier. When a vehicle approaches to leave the parking lot the IR sensors detect it. They also detect whether it is a large or a small vehicle depending on the height of the vehicle. They relay this information to the barrier and the barrier lifts with an elevation angle depending on the height of the vehicle. When the vehicle has exited the barrier shuts down again and the count on the LCD screen decrements by one.

DIAGRAMS

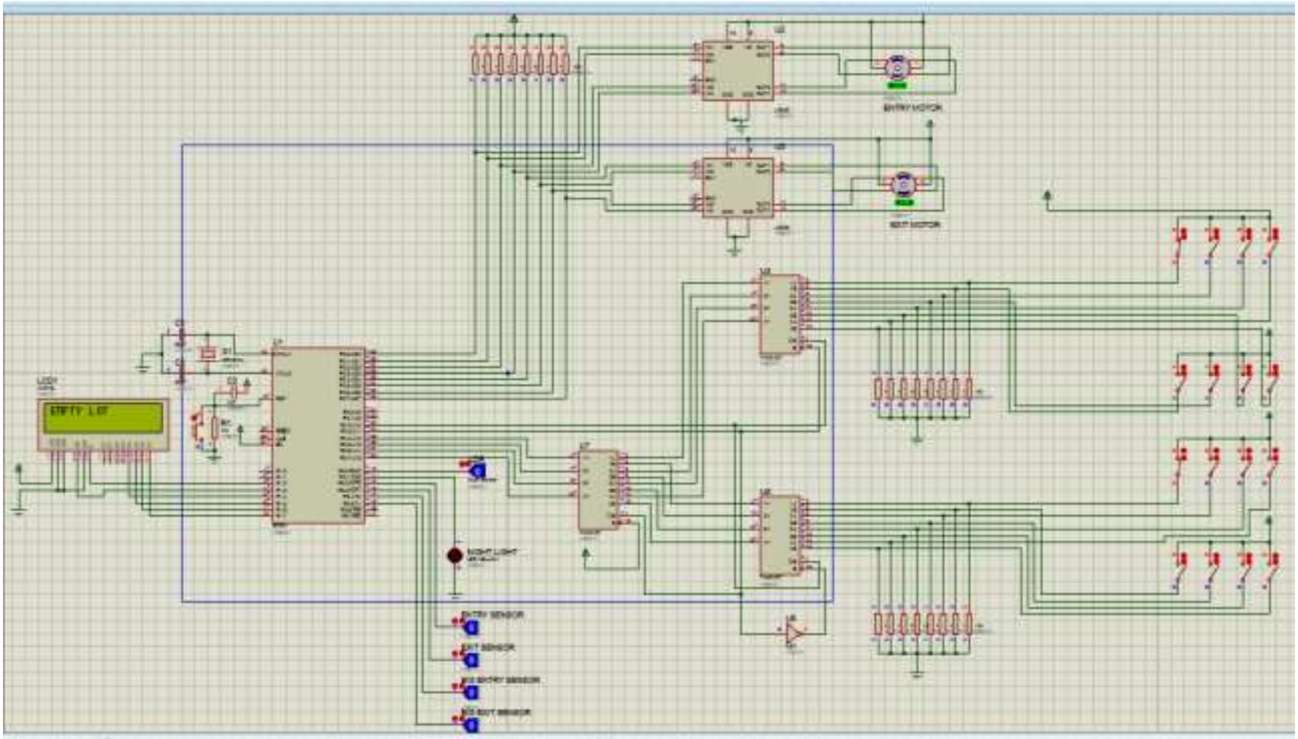
BLOCK DIAGRAM



SIMULATION RESULTS

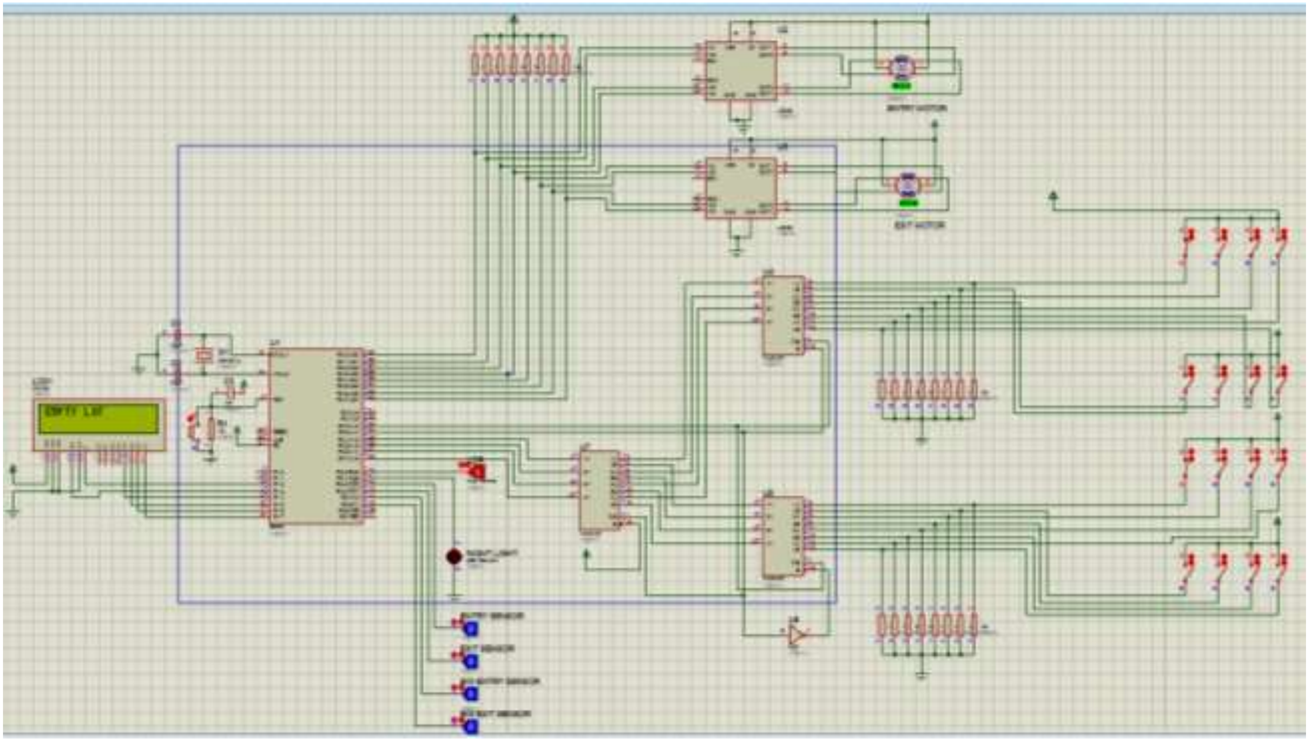
1. **When code start:**

No car and Lights Off



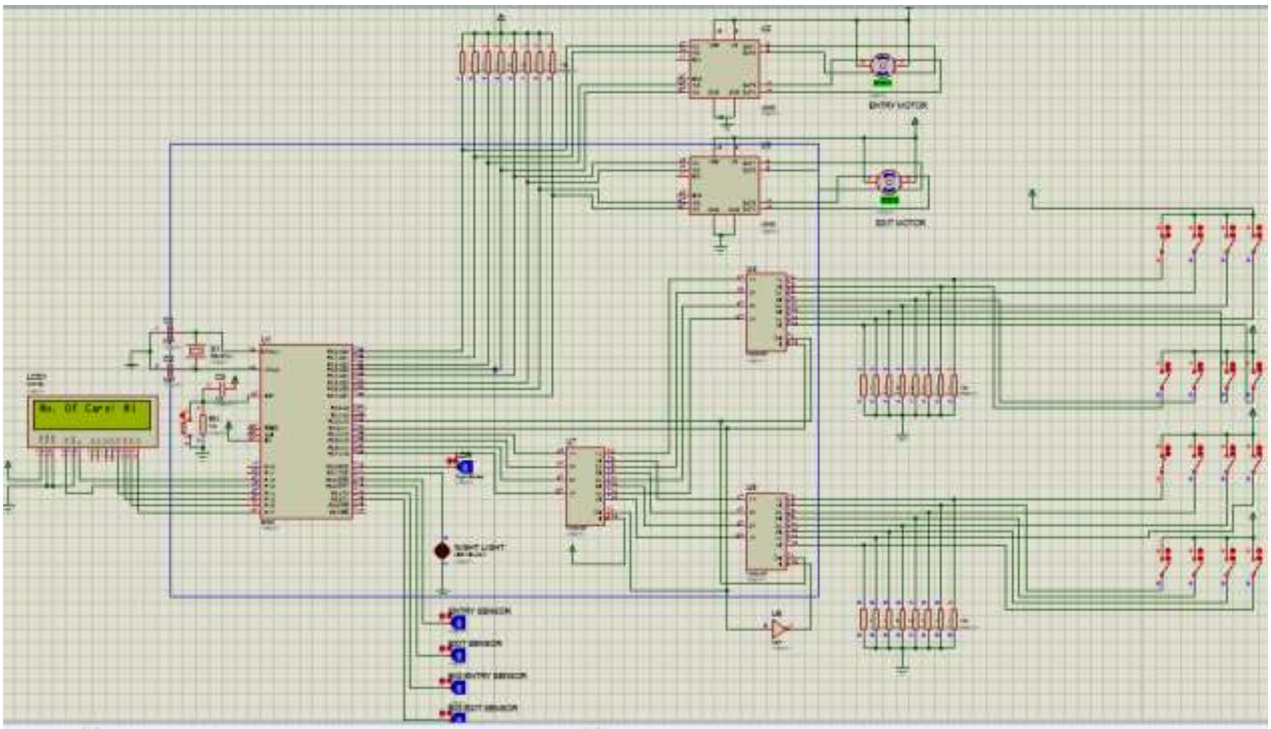
2. **When empty lot and It's night:**

   No car so are Lights Off automatically even it's night.



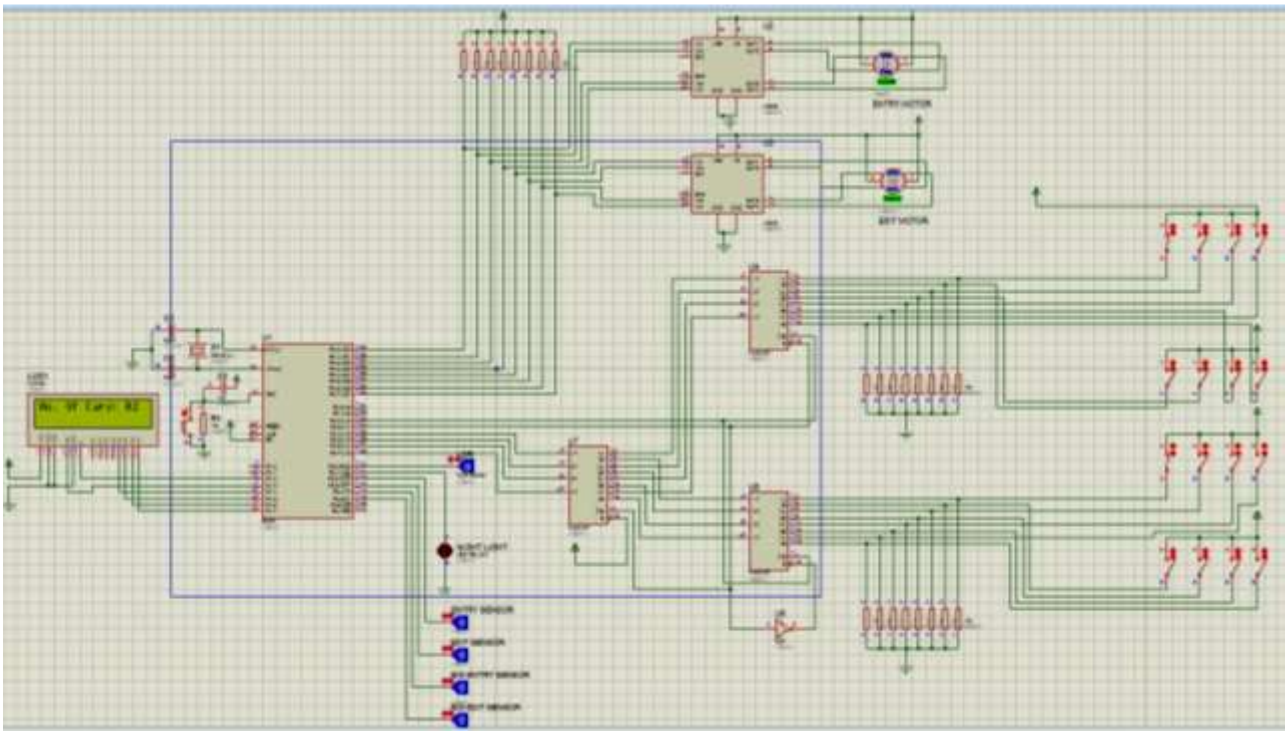
3. **When car enters:**

   No. of cars++(incremented),entry motor rotates and Lights still Off as LDR input=0(day time)



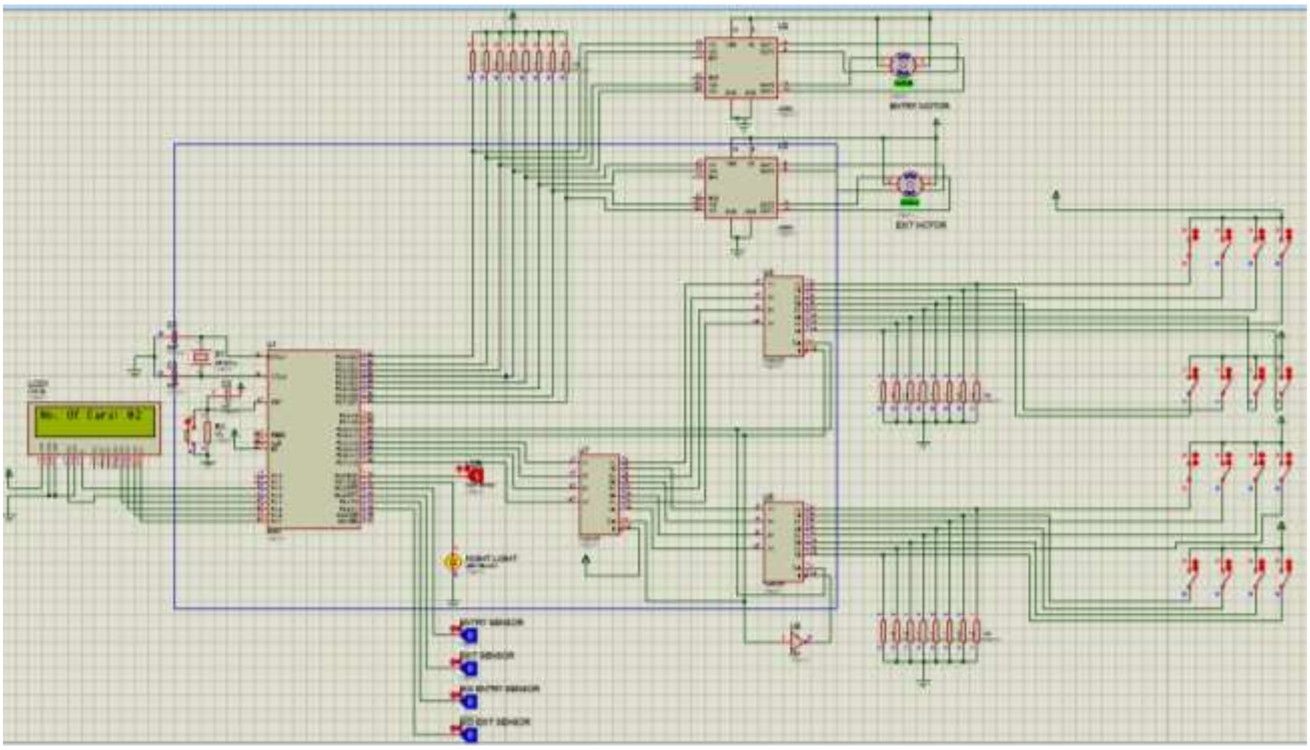
4. When another car enters:

No. of cars++, entry motor rotates and Lights still Off as LDR input=0(day time).



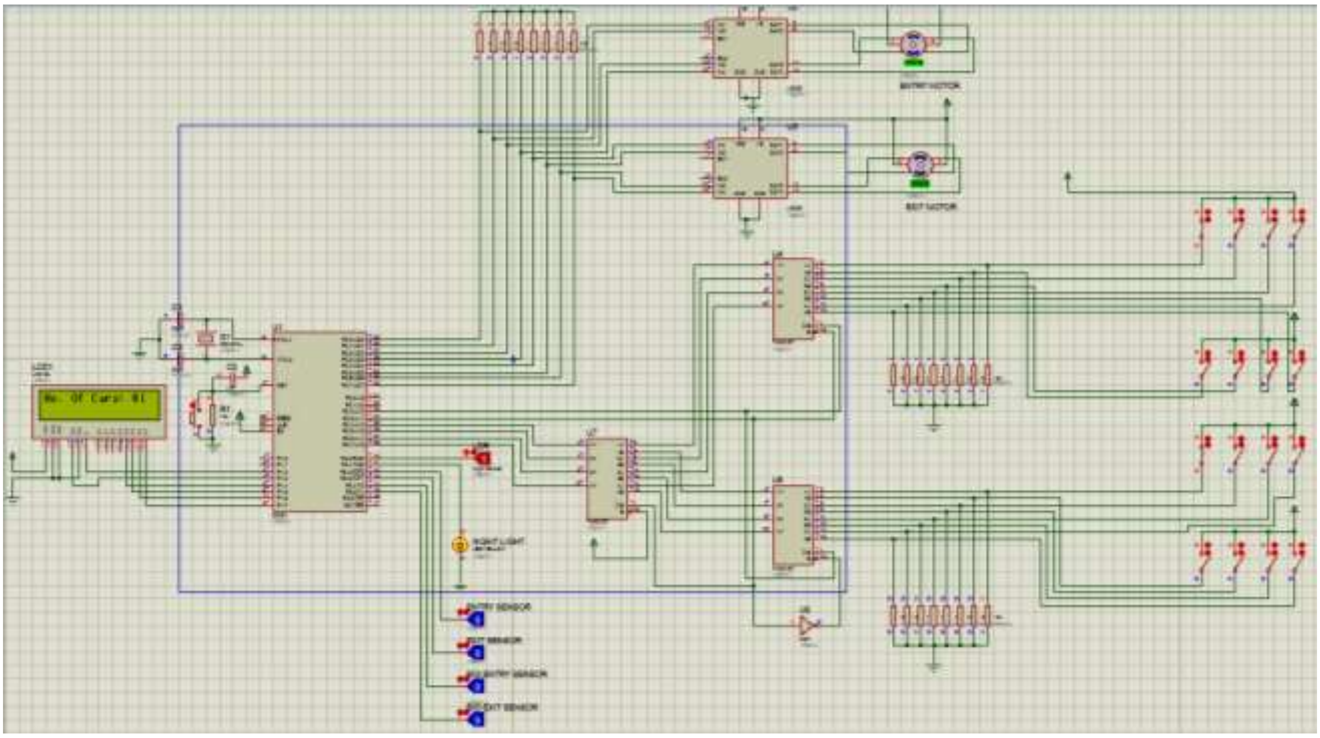
5. When there are cars and it's night:

Cars in parking and it's night, so lights ON.



6. When a car exit:

No. of cars--, exit motor rotates and as there is one car remaining so lights still on.



CODE

```
#include <reg52.h>
#include <stdio.h>

#define lcdport P1
#define scanport P2
sbit s0 = scanport^2;
sbit s1 = scanport^3;
sbit rs = lcdport^3;
sbit en  = lcdport^2;
sbit sublot0 = P3^6;
sbit sublot1 = P3^7;
sbit sublot2 = P1^1;
sbit sublot3 = P1^0;


unsigned int i,j,k,ang,tim;
unsigned char z;
unsigned char count=0;
unsigned char temp=0;
sbit LDR= P3^0;                    //input from LDR
sbit Led=P3^1;                                                                            //Output for LED
sbit big_entry = P3^4;
sbit big_exit = P3^5;
sfr ldata = 0x90;
sfr tcon = 0x89;
//configuring port 1 for LCD
```

```
unsigned char emp1[]= "EMPTY LOT ";
unsigned char fu[]= "FULL LOT ";
unsigned char noc[]= "No. Of Cars: ";
int value[2] = {0};
//integer -> seperate  numerical characters
int* div_num(unsigned int t)
{
        if(t<10)
        {
                value[0] = t;
                value[1] = 0;
        }
        if(t>=10)
        {
                value[1] = t/10;
                value[0] = t%10;
        }
        return value;
}


void dela(int a)//delay for lcd
{
  int i;
  for(i=0;i<a;i++);   //null statement
}


void Delay(unsigned int x)
 // Generic function for 10ms Delay
{
        for(i=0;i<x;i++)
        for (j=0;j<10000;j++);
}
void Delay123(unsigned int x)
// Generic function for 1ms Delay(for motor )
{
        for(i=0;i<x;i++)
        for (j=0;j<1000;j++);
}

void scan_sublots()
{
        scanport |= 0xf0;
        temp=0;

        //scan sublot-lot 0
        s0 = 0;
        s1 = 0;
        Delay123(150);     //Demux delay increased from 30->300
        temp = scanport;//scanned values on higher bits
        temp = temp&0xf0;//Mask lower bits
        if(temp)
        sublot0 = 1;
        if(!temp)
        sublot0 = 0;

        //scan sublot-lot 1
        scanport |= 0xf0;
        s0 = 1;
```

```
        s1 = 0;
        Delay123(150);   //Demux delay increased from 30->300
        temp = scanport;//scanned values on higher bits
        temp = temp&0xf0;//Mask lower bits
        if(temp)
        sublot1 = 1;
        if(!temp)
        sublot1 = 0;

        //scan sublot-lot 2
        scanport |= 0xf0;
        s0 = 0;
        s1 = 1;
        Delay123(150);   //Demux delay increased from 30->300
        temp = scanport;//scanned values on higher bits
        temp = temp&0xf0;//Mask lower bits
        if(temp)
        sublot2 = 1;
        if(!temp)
        sublot2 = 0;


        //scan sublot-lot 3
        scanport |= 0xf0;
        s0 = 1;
        s1 = 1;
        Delay123(150);   //Demux delay increased from 30->300
        temp = scanport;//scanned values on higher bits
        temp = temp&0xf0;//Mask lower bits
        if(temp)
        sublot3 = 1;
        if(!temp)
        sublot3 = 0;
        temp=0;

}



void gate_operate_1(unsigned int ang,unsigned int tim)
{
        for(z=0;z<(ang*2);z++)//Change multiplier according to motor
        {
        P0=0x30;    //00000011
        Delay123(3);
        P0=0x20;    //00000010
        Delay123(3);
        P0=0x60;    //00000110
        Delay123(3);
        P0=0x40;    //00000100
        Delay123(3);
        P0=0xc0;    //00001100
        Delay123(3);
        P0=0x80;    //00001000
        Delay123(3);
        P0=0x90;    //00001001
        Delay123(3);
        P0=0x10;    //00000001
        Delay123(3);
        }
```

```
		Delay(tim);			//Barrier opens for seconds(Choose)
		for(z=0;z<(ang*2);z++)//Change multiplier according to motor
		{
		P0=0x10;	//00000011
		Delay123(3);
		P0=0x90;	//00000010
		Delay123(3);
		P0=0x80;	//00000110
		Delay123(3);
		P0=0xc0;	//00000100
		Delay123(3);
		P0=0x40;	//00001100
		Delay123(3);
		P0=0x60;	//00001000
		Delay123(3);
		P0=0x20;	//00001001
		Delay123(3);
		P0=0x30;	//00000001
		Delay123(3);
		}	//Motor in opposite direction to close the barrier
		Delay123(3)
	P0=0x00;
}

void gate_operate_2(unsigned int ang, unsigned int tim)
{
		for(z=0;z<(ang*2);z++)//Change multiplier according to motor
		{
		P0=0x03;	//00000011
		Delay123(3);
		P0=0x02;	//00000010
		Delay123(3);
		P0=0x06;	//00000110
		Delay123(3);
		P0=0x04;	//00000100
		Delay123(3);
		P0=0x0c;	//00001100
		Delay123(3);
		P0=0x08;	//00001000
		Delay123(3);
		P0=0x09;	//00001001
		Delay123(3);
		P0=0x01;	//00000001
		Delay123(3);
		}
		Delay(tim);			//Barrier opens for seconds(Choose)
		for(z=0;z<(ang*2);z++)//Change multiplier according to motor
		{
		P0=0x01;	//00000011
		Delay123(3);
		P0=0x09;	//00000010
		Delay123(3);
		P0=0x08;	//00000110
		Delay123(3);
		P0=0x0c;	//00000100
		Delay123(3);
		P0=0x04;	//00001100
		Delay123(3);
		P0=0x06;	//00001000
		Delay123(3);
```

```
            P0=0x02;        //00001001
            Delay123(3);
            P0=0x03;        //00000001
            Delay123(3);
            }
            Delay123(3);
            P0=0x00;
}


void lcdcommand(unsigned char a)
{
  rs = 0;           // This is command

  lcdport &= 0x0F;                  // Make P2.4 to P2.7 zero
  lcdport |= (a&0xF0);    // Write Upper nibble of data

  en  = 1;          // => en = 1
  dela(150);
  en  = 0;          // => en = 0
  dela(150);

  lcdport &= 0x0F;                  // Make P2.4 to P2.7 zero
  lcdport |= ((a<<4)&0xF0);// Write Lower nibble of data

  en  = 1;          // => en = 1
  dela(150);
  en  = 0;          // => en = 0
  dela(150);
}

void data1(unsigned char b)
{
  rs = 1;           // This is data

  lcdport &= 0x0F;                  // Make P2.4 to P2.7 zero
  lcdport |= (b&0xF0);    // Write Upper nibble of data

  en  = 1;          // => en = 1
  dela(150);
  en  = 0;          // => en = 0
  dela(150);

  lcdport &= 0x0F;                  // Make P2.4 to P2.7 zero
  lcdport |= ((b<<4)&0xF0);// Write Lower nibble of data

  en  = 1;          // => en = 1
  dela(150);
  en  = 0;          // => en = 0
  dela(150);
}
void init(void)
{
 ///////////// Reset process from datasheet /////////
   dela(15000);

            lcdport &= 0x0F;                  // Make P2.4 to P2.7 zero
            lcdport |= (0x30&0xF0);    // Write 0x3

            en  = 1;          // => en = 1
```

```
            dela(150);
            en  = 0;            // => en = 0
            dela(150);

    dela(4500);

            lcdport &= 0x0F;                    // Make P2.4 to P2.7 zero
            lcdport |= (0x30&0xF0);    // Write 0x3

            en  = 1;            // => en = 1
            dela(150);
            en  = 0;            // => en = 0
            dela(150);

    dela(300);

            lcdport &= 0x0F;                    // Make P2.4 to P2.7 zero
            lcdport |= (0x30&0xF0);    // Write 0x3

            en  = 1;            // => en = 1
            dela(150);
            en  = 0;            // => en = 0
            dela(150);

    dela(650);

            lcdport &= 0x0F;                    // Make P2.4 to P2.7 zero
            lcdport |= (0x20&0xF0);    // Write 0x2

            en  = 1;            // => en = 1
            dela(150);
            en  = 0;            // => en = 0
            dela(150);

            dela(650);

  //////////////////////////////////////////////////////
  lcdcommand(0x28);   //function set
  lcdcommand(0x0c);   //display on,cursor off,blink off
  lcdcommand(0x01);   //clear display
  lcdcommand(0x06);   //entry mode, set increment
}

void emptylot(void)
{
//Empty lot message
            if (count==0)
            {
            lcdcommand(0x01);
   for(z=0;z<9;z++)
   {
                              data1(emp1[z]);
   }
            lcdcommand(0x0c);
            }
}

void fulllot(void)
{
  if (count==16)
```

```
                {
   lcdcommand(0x01);
   for(z=0;z<8;z++)
   {
                        data1(fu[z]);
   }
            lcdcommand(0x0c);
            }
}

void display_no(void)
{
//Display no of cars
                lcdcommand(0x01);
                for(z=0;z<13;z++)
                {
                        data1(noc[z]);
                }
                div_num(count);
                for(z= 0 ; z < 2 ; z++ )
                data1(value[1-z]+48);
                lcdcommand(0x0c);
}

//*********************Interrupt for enter*****************//
void Gate_Enter() interrupt 0                   //Interrupt INT0 p3.2, address=0003H
{
            if(count<=15)
            {
            unsigned char temp=count;
            IE &= 0x00;
            ang = 10;
            if(big_entry)
            ang = 50;
            gate_operate_1(ang,50);


  count++;
            temp=count+48;
  switch (temp)
  {
   case (48):
   {
            emptylot();
            break;
   }
   case (64):
   {
   fulllot();
   break;
   }
   default:
   {
            display_no();
            break;
   }
  }
            TCON &= 0xf5;
            IE |= 0x85;
 }d
```

```
 }
 //******************Interrupt for exit******************//
void Gate_Exit() interrupt 2                    //Interrupt INT1 p3.3 , address=0013H
{
            if (count>=1)
            {
unsigned char temp=count;
            ang = 10;
            if(big_exit)
            ang = 50;
            IE &= 0x00;
            gate_operate_2(ang,50);

  count--;
            temp=count+48;
  switch (temp)
  {
   case (48):
   {
            emptylot();
            break;
   }
   case (64):
   {
   fulllot();
   break;
   }
   default:
   {
            display_no();
            break;
   }
  }

            TCON &= 0xf5;
            IE |= 0x85;
 }
}
            //************************main*******************************//
void main()
{

            unsigned char wel[]= "    WELCOME ";
            P0=0x00;     //output configure for motor
  P1=0x00;                // P3 and P2 as output ports
  P3=0x00;
            P2=0xff;

  init();
for(z=0;z<12;z++)
 data1(wel[z]);
lcdcommand(0x0c);
Delay(20);
scan_sublots();
emptylot();
  TCON=0x05;                 //Edge sensitive                          //enabling interrupts
  IE=0x85;
            LDR=1;     //Configure input
  while (1)
  {
```

```
 if (count==0)
                    {
                              Led=0;
                    }
                    else
                    {
   if(LDR==1 && count > 0)        //Check for LDR Input, 1=high R =Night
                              {         Led=1;
                              }
                              else if(LDR==0)
                              {         Led=0;
                              }
                    }
                    scan_sublots();
 }
}
```

BENEFITS

Automated parking lot has some very important benefits and it makes life very convenient for the drivers. The first benefit as stated in the beginning as well is that it relieves the mental tension and frustration of drivers who look manually for a parking space in a parking lot. With the introduction of the automated parking lot the drivers will be at a lot more ease and a lot of time will also be saved for them.

Secondly, the amount of fuel saved will also be huge and that being said, the pollution done by the exhaust fumes of a vehicle will also greatly reduce. As mentioned previously, in a research done by UCLA urban-planning professor on 15 blocks near the UCLA campus, 47,000 gallons of gas were wasted in the course of a year looking for a parking space in the 15 block area only. Moreover 730 tons of greenhouse gas emissions were released in the same time.

Another benefit although small in comparison to the ones mentioned above is the amount of electricity saved by opening the barrier to an amount required depending on the vehicle. If the 2 infra-red sensor arrangement was not applied the barrier would always have a minimum opening elevation angle depending on the largest height allowed of a vehicle to enter. This way even if a small car was entering the barrier would lift all the way up to the fixed elevation angle although it doesn’t need to. Therefore the Automated parking lot has catered this problem as well in its design and a significant amount of electricity and power is saved.

The Automated Parking lot doesn’t only tell the driver before entering that a parking space is available. It tells the driver exactly which slot is empty and hence the driver will know exactly where to go. This way an extra amount of time and fuel is saved.

The automated lighting system helps in saving electricity. Normally the lighting would be done manually. The issue with this is that if the lights are not switched off when needed(when the parking lot is empty etc) the light will be wasted for no reason. The

Automated Parking lot solves the lighting problem by catering for this condition and switches on the lights only when the daylight is not sufficient using an LDR and also switching off the lights if there are no cars in the lot even if it is night time.

PROJECT LIMITATIONS

As case will most of problem solving projects, the Automated Parking lot also does have its limitations. The parking lot model that we have designed has a capacity of only 16 vehicles. However when the design is implemented for a larger number of vehicles it shouldn't be a problem as the only modification done to the system will be in the de-multiplexing part. Depending on the new size of the lot more de-multiplexers will be added and all the spaces will be accommodated.

Another limitation is that because we are using only one micro controller, it can only handle one interrupt at a time. So cars cannot simultaneously enter and exit.

The de- multiplexing ICs we used cause a delay because their switching is slow. Moreover we are sending a lot of data on the same data bus so it slows down the circuit. We need to add delays in the program to cater for the time wasted by the ICs

PROBLEMS FACED

While building the circuit a problem arose when the infra-red sensors would detect the presence on an object even when it was at a significant angle. This meant that a lot of unnecessary detections were being made. The problem was solved by changing the sensitivity of the sensors (decreasing it). Further the sensor was taking 2 pulses and hence increased count by 2 each time. We solved it by clearing the interrupt register flag after giving pulse. More problems were faced when the software circuit was implemented on hardware. The connections on the breadboard on some locations were disconnected and this was a lengthy process in finding out using the probe. We faced problem while multiplexing the parking space and were not able to get correct pulses at the controller pin. We adjusted it by sending delayed pulses to the controller during polling.

COST

Name	Quantity	Cost/Rs
ICs	-	580
Motors	2	500
Wires		240
Sensors	4	650
Bread boards	7	1400
LCD	1	290
LED,Resistors	-	50
Switches	25	250
Microcontroller and beds	2	310
Misc		500
	TOTAL	4770

FUTURE IMPROVEMENTS

The Automated parking lot system is a relatively basic solution. A lot of improvements and modifications can be done upon this original design to improve the efficiency of the project. The possible improvements can be:

- ✓ Increasing the capacity of the parking lot.
- ✓ Better Multiplexing Scheme.
- ✓ Using a better and faster microcontroller that is capable of handling more data.
- ✓ Using more than one entrance and exit barriers to avoid congestion and queues on the entrance.
- ✓ Adding security features such as using facial recognition to assure that the driver entering in a vehicle is the same as the one exiting with the same vehicle.
- ✓ Adding on to the complexity by adding floors into Lot.