

Project Report

SMART PARKING SYSTEM

Table of Contents

CONTENT	Page no.
• Abstract	3
• introduction	3
• Block diagram	4
• Components and it's description.	5
• Flow chart	6
• Referred code	7
• Code and Application	8
• Results	15
• Final output in keil	18
• Simulation in proteus	19
• Conclusion	24
• References	24

3. ABSTRACT:

Now days in many multiplex systems there is a severe problem for car parking systems. There are many lanes for car parking, so to park a car one must look for the all lanes. Moreover, there is a lot of men labour involved for this process for which there is lot of investment.

Hence in this project we have come up with a System which will effectively help in making the parking easy by showing the free slots. In this when you at parking site, you see the number of parking slots and available parking slots at the entry gate. it also makes the work of the security guards easy in finding the number of cars parked by showing the "cars parked:". This happens with the help of the sensors that will detect the presence of car and hence accordingly the parking slots availability is shown. To make it more precise and easier, LEDs are used which will notify the availability of free slots. In case if there are no slots available, then red LED glows, hence notifying the absence of free slots.

4. INTRODUCTION:

Smart parking system effectively helps in making the parking easy by showing the free slots. In the project we have used 8051 microcontroller. The port P2 is used as input port to sense the presence of vehicle. In a relay, the resistor absorbs access voltage given off when the relay is activated. This will protect any other components in the circuit from voltage spikes.

An LED (Light Emitting Diode) emits light when an electric current pass through it. The simplest circuit to power an LED is a voltage source with a resistor and an LED in series. In our circuit we have used 2 LEDs Red and green. The red led will glow to indicate the absence of free slot, and green led glows to indicate the presence of the parking slot. This is interfaced with the circuit using port 3.

The output is obtained at port 1 using an LCD display. This will LCD display will be displaying the free slots along with the number of parked cars, hence acting as an output through port 1 of 8051 microcontroller. We have made for 8 slots, which can be expanded to implement into reality.

So, whenever the sensor gets the input sensing the presence of a vehicle that is whenever the switch is pressed, there will be an immediate result that we will be able to see on the LCD display. The slot

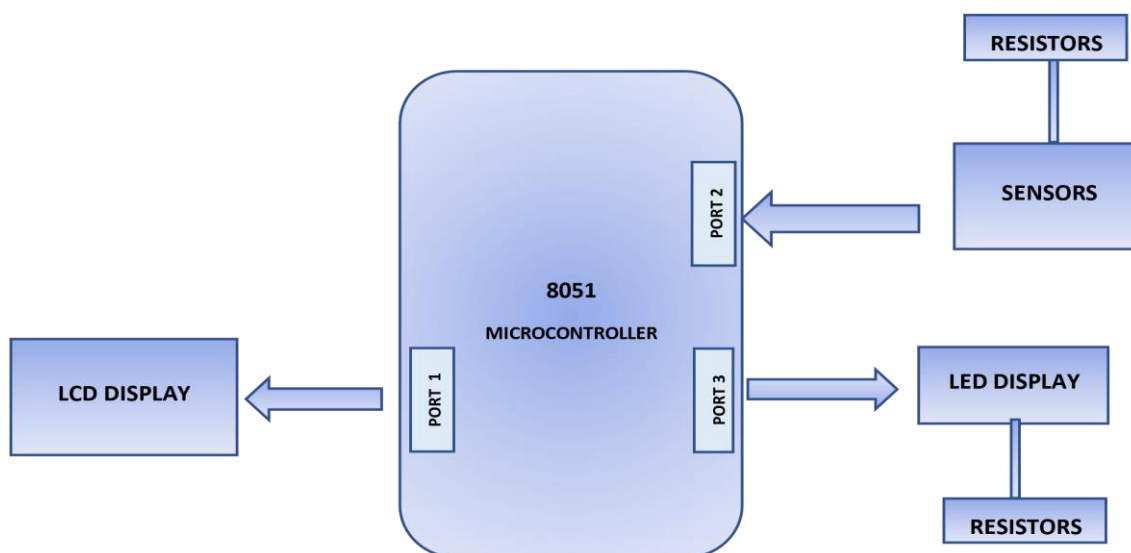
will show as occupied and the cars parked will get incremented by a count of 1. the same process will continue whenever the sensors are further detected, and hence further incrementing the count by 1. Similarly, when the car leaves the parking slot the count gets decremented by 1 and immediately it will also be reflected within the availability of free slots.

Hence the entire project will be effectively useful for showing the free slots for parking along with the count value.

This project can be used for parking system in any shopping mall, multiplex 2. Can be used for industries, commercial offices and educational institutes.

- Prevents waste of manpower when it's not enough for supervision.
- This project can be used in Hospitals.
- Easy to modify as per requirement.
- Economically feasible
- Simple circuitry
- Secured data collection
- Easy error detection.

5. BLOCK DIAGRAM:



SOFTWARES USED:

- Proteus
- Keil

COMPONENTS REQUIRED:

- LCD
- Microcontroller 8051
- LED
- Resistors and capacitors
- Sensors (represented as switches)

Description of components:

† MICRO CONTROLLER 8051

- 4KB on-chip program memory (ROM).
- 128 bytes on-chip data memory (RAM).
- Four register banks.
- 128 user defined software flags.
- 8-bit bidirectional data bus.
- 16-bit unidirectional address bus.
- 32 general purpose registers each of 8-bit.
- 16-bit Timers (usually 2, but may have more or less).
- Three internal and two external Interrupts.
- Four 8-bit ports, (short model have two 8-bit ports).
- 16-bit program counter and data pointer.

† LCD (16x2 Module)

- Operating Voltage is 4.7V to 5.3V. Current consumption is 1mA without backlight.
- Each character is build by a 5x8 pixel box.
- Can work on both 8-bit and 4-bit mode.
- It can also display any custom generated characters.
- Available in Green and Blue Backlight.

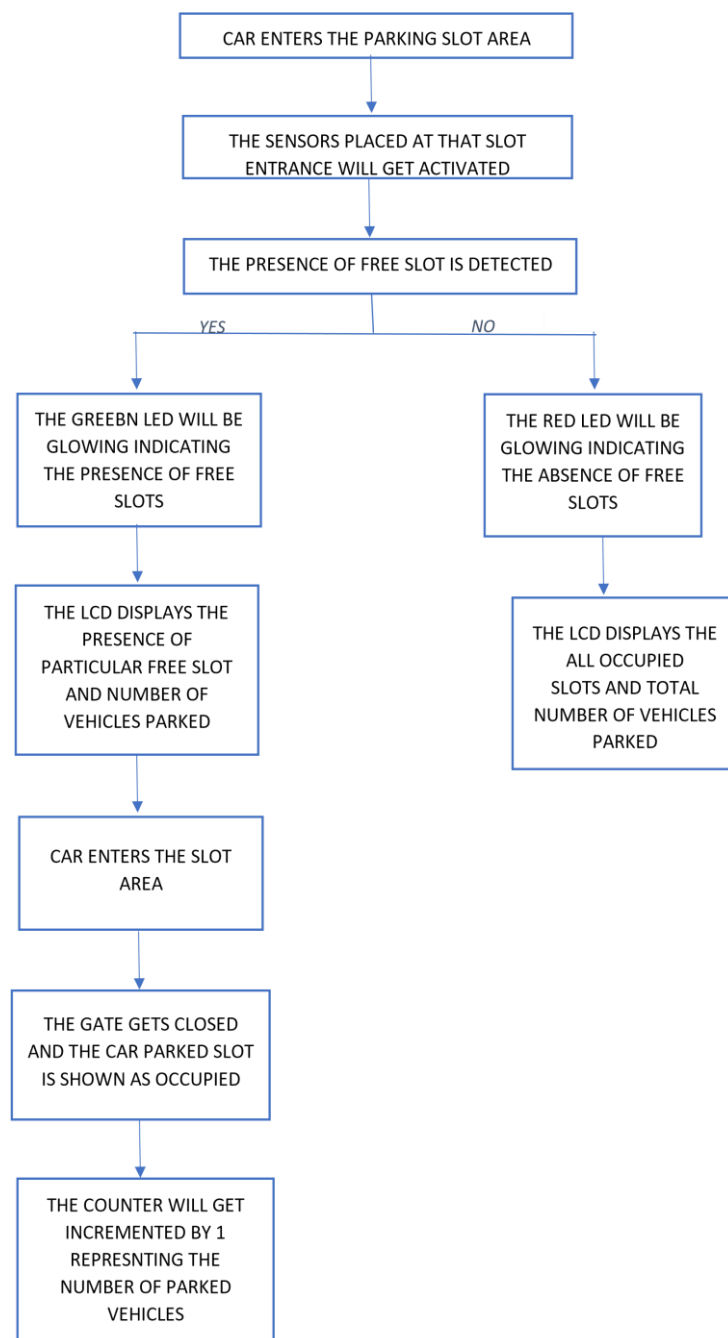
† Switches:

- Used to stop the flow of current.

† Resistor:

- Used to oppose the flow of current.
- It protects from the damage of component directly supply of pow

FLOW CHART:



8) CODE AND APPLICATION:

ASM CODE:

```
GREEN_LED EQU P3.1
RED_LED EQU P3.2 org
0000h
mov p1,#0h //make port1(LCD port) as output
MOV P2,#0FFH
clr p3.4 //make RS pin as output clr
p3.6 //make E pin as output MOV
R4,#00h //r1 as 8 vehicle counter
MOV R5,#00h //r2 for switch to 4 line
SETB RED_LED //turn off RED LED CLR
GREEN_LED // turn GREEN LED a call
delay_1s //1s delay
MOV DPTR,#INIT_COMMANDS //Initialize LCD to 8 bit 5x7 matrix
ACALL LCD_CMD
MOV DPTR,#LINE1 //clear screen and go to line 1 position 1
ACALL LCD_CMD
MOV DPTR,#TEXT1 //display introduction
ACALL LCD_DISP
acall delay_1s //4s delay
acall delay_1s acall delay_1s
acall
delay_1s
UP:MOV R5,#00h
MOV DPTR,#LINE1 //go to line 1
ACALL LCD_CMD
MOV DPTR,#PARKED_VEHICAL //print the vehicle count
ACALL LCD_DISP
MOV A,#0x30
ADD A,R4
ACALL write
MOV R4,#08h

MOV A,p2 //check all slots are used or not
//ANL A,#0xFF
```

```

CJNE A,#0x00,SKIP      //if all slots are used then print on lcd
MOV DPTR,#LINE2
ACALL LCD_CMD
MOV DPTR,#TEXT2
ACALL LCD_DISP
CLR RED_LED           //turn off RED LED
SETB GREEN_LED // turn GREEN LED acall
delay_1s acall delay_1s MOV R4,#08h
SJMP UP

SKIP:MOV DPTR,#LINE2 //if free slots are available then check available slots

ACALL LCD_CMD
MOV DPTR,#FREE
ACALL LCD_DISP

MOV DPTR,#LINE3
ACALL LCD_CMD
MOV A,p2              //check s1 slot
RRC A
JNC NEXT1             //if slot are free print slot name on lcd
SETB RED_LED          //turn off RED LED
CLR GREEN_LED         // turn GREEN LED
DEC R4
INC R5
MOV DPTR,#S1
ACALL LCD_DISP
NEXT1:MOV A,p2         //check s2 slot
RRC A
RRC A
JNC NEXT2             //if slot are free print slot name on lcd
SETB RED_LED          //turn off RED LED
CLR GREEN_LED         // turn GREEN LED
DEC R4
INC R5

```



```

MOV DPTR,#S2 ACALL
LCD_DISP
NEXT2:MOV A,p2          //check s3 slot
RRC A
RRC A
RRC A
JNC NEXT3              //if slot are free print slot name on lcd
SETB RED_LED           //turn off RED LED
CLR GREEN_LED          // turn GREEN LED
DEC R4
INC R5
MOV DPTR,#S3
ACALL LCD_DISP
NEXT3:MOV A,p2          //check s4 slot
RRC A
RRC A
RRC A
RRC A
JNC NEXT4
SETB RED_LED           //turn off RED LED
CLR GREEN_LED          // turn GREEN LED
DEC R4
INC R5
MOV DPTR,#S4           //if slot are free print slot name on lcd
ACALL LCD_DISP
NEXT4:MOV A,p2          //check s4 slot
RRC A
RRC A
RRC A
RRC A
RRC A
JNC NEXT5
SETB RED_LED           //turn off RED LED
CLR GREEN_LED          // turn GREEN LED
DEC R4
INC R5
CJNE R5,#5,CONT2
MOV DPTR,#LINE4
ACALL LCD_CMD

```

CONT2:MOV DPTR,#S5 //if slot are free print slot name on lcd

ACALL LCD_DISP

NEXT5:MOV A,p2 //check s4 slot

RRC A

RRC A

RRC A

RRC A

RRC A

RRC A

JNC NEXT6

SETB RED_LED //turn off RED LED

CLR GREEN_LED // turn GREEN LED

DEC R4

INC R5

CJNE R5,#5,CONT3

MOV DPTR,#LINE4

ACALL LCD_CMD

CONT3:MOV DPTR,#S6 //if slot are free print slot name on lcd

ACALL LCD_DISP

NEXT6:MOV A,p2 //check s4 slot

RRC A

RRC A

RRC A

RRC A

RRC A

RRC A

RRC A

JNC NEXT7

SETB RED_LED //turn off RED LED

CLR GREEN_LED // turn GREEN LED

DEC R4

INC R5

CJNE R5,#5,CONT4

MOV DPTR,#LINE4

ACALL LCD_CMD

CONT4:MOV DPTR,#S7 //if slot are free print slot name on lcd

ACALL LCD_DISP

NEXT7:MOV A,p2 //check s4 slot

```

RRC A
RRC A

RRC A
RRC A
RRC A RRC
A
RRC A
RRC A
JNC NEXT8
SETB RED_LED          //turn off RED LED
CLR GREEN_LED          // turn GREEN LED
DEC R4
INC R5
CJNE R5,#5,CONT5
MOV DPTR,#LINE4
ACALL LCD_CMD
CONT5:MOV DPTR,#S8      //if slot are free print slot name on lcd
ACALL LCD_DISP NEXT8:
acall delay_1s acall
delay_1s
LJMP UP                //go to check the slots

command:  //function to send command mov p1,a clr p3.4

setb p3.6
acall delay

```

clr p3.6 acall

delay acall

delay ret

write: //function to send data

mov p1,a setb p3.4 setb p3.6 acall delay clr p3.6

acall delay acall delay

ret

LCD_DISP: CLR A //function to send display string

MOVC A,@A+DPTR

JZ EXIT1

INC DPTR

ACALL write

SJMP LCD_DISP

EXIT1: RET

LCD_CMD: CLR A MOVC //function to send commands A,@A+DPTR

JZ EXIT2

INC DPTR

ACALL command

SJMP LCD_CMD

EXIT2: RET

delay: mov

r0,#1ch rep:

djnz r0,rep

ret

delay_1s: mov //function for 1s delay

r3,#08h df1s: mov r2,#0ffh d1s: mov r1,#0ffh de1s:

djnz r1,de1s

djnz r2,d1s

djnz r3,df1s ret

TEXT1: DB " PARKING SYSTEM",0

TEXT2: DB " NO FREE SLOTS",0

FREE: DB " Free slots",0

S1: DB "S1 ",0

S2: DB "S2 ",0

S3: DB "S3 ",0

S4: DB "S4 ",0

S5: DB "S5 ",0

S6: DB "S6 ",0

S7: DB "S7 ",0

S8: DB "S8 ",0

INIT_COMMANDS: DB 38h,06h,0ch,01H,80H,0

LINE1: DB 01H,06H,06H,80H,0

LINE2: DB 0C0H,0

LINE3: DB 094H,0

LINE4: DB 0D4H,0

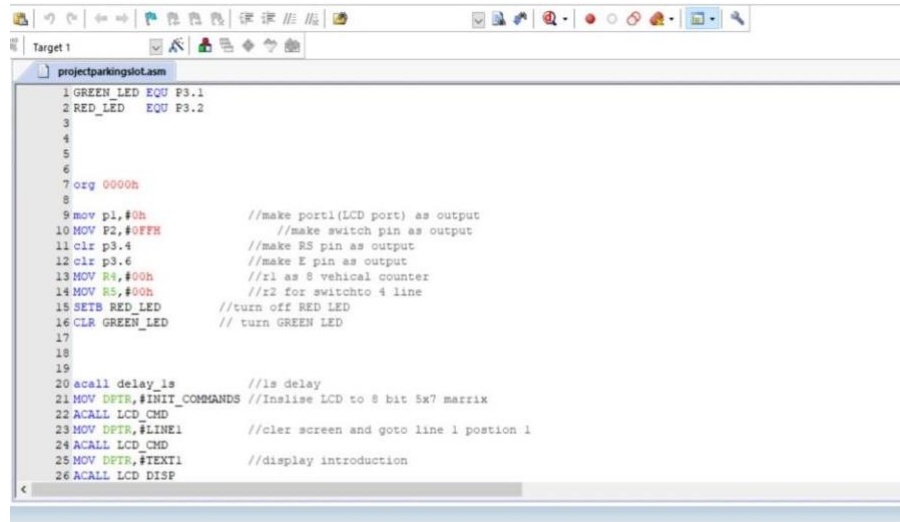
CLEAR: DB 01H,0

PARKED_VEHICAL: DB " parked vehicles:",0

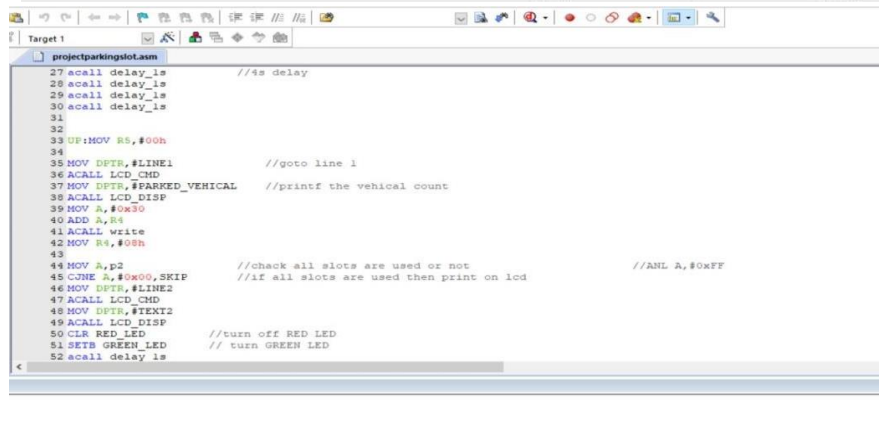
End

9 RESULTS:

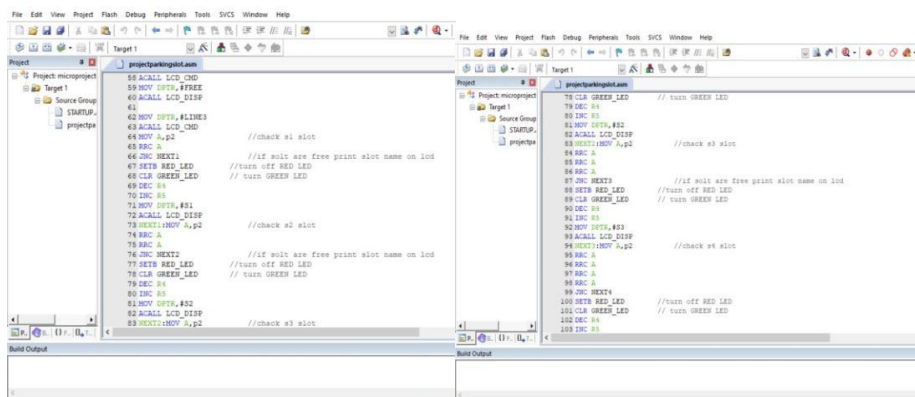
(i) Keil software implementation:



```
1 GREEN_LED EQU P3.1
2 RED_LED EQU P3.2
3
4
5
6
7 org 0000h
8
9 mov p1,#0h //make port1(LCD port) as output
10 MOV P2,#0FFh //make switch pin as output
11 clr p3.4 //make RS pin as output
12 clr p3.6 //make E pin as output
13 MOV R4,#00h //r1 as 8 vehical counter
14 MOV R5,#00h //r2 for switcho 4 line
15 SETB RED_LED //turn off RED LED
16 CLR GREEN_LED // turn GREEN LED
17
18
19
20 acall delay_1s //1s delay
21 MOV DPTR,#INIT_COMMANDS //Inslise LCD to 8 bit 5x7 Marrix
22 ACALL LCD_CMD
23 MOV DPTR,#LINE1 //cler screen and goto line 1 postion 1
24 ACALL LCD_DISP
25 MOV DPTR,#TEXT1 //display introduction
26 ACALL LCD_DISP
```



```
27 acall delay_1s //4s delay
28 acall delay_1s
29 acall delay_1s
30 acall delay_1s
31
32
33 UP:MOV R5,#00h
34
35 MOV DPTR,#LINE1 //goto line 1
36 ACALL LCD_CMD
37 MOV DPTR,#PARKED_VEHICAL //printf the vehical count
38 ACALL LCD_DISP
39 MOV A,R4
40 ADD A,R4
41 ACALL write
42 MOV R4,#00h
43
44 MOV A,p2 //check all slots are used or not //ANL A,#0xFF
45 CNE A,#000,SKIP //if all slots are used then print on lcd
46 MOV DPTR,#LINE2
47 ACALL LCD_CMD
48 MOV DPTR,#TEXT2
49 ACALL LCD_DISP
50 CLR RED_LED //turn off RED LED
51 SETB GREEN_LED // turn GREEN LED
52 acall delay_1s
```



```
53 ACALL LCD_CMD
54 MOV DPTR,#FREE
55 ACALL LCD_DISP
56
57
58 MOV DPTR,#LINE3
59 ACALL LCD_CMD
60 MOV A,p2 //check s1 slot
61 RLC A
62 JNC NEXT1 //if slot are free print slot name on lcd
63 SETB RED_LED //turn off RED LED
64 CLR GREEN_LED // turn GREEN LED
65 DEC R4
66 INC R5
67 MOV DPTR,#R1
68 ACALL LCD_DISP
69 MOV DPTR,#R2 //check s2 slot
70 RLC A
71 JNC NEXT2 //if slot are free print slot name on lcd
72 SETB RED_LED //turn off RED LED
73 CLR GREEN_LED // turn GREEN LED
74 DEC R4
75 INC R5
76 MOV DPTR,#R2
77 ACALL LCD_DISP
78 MOV DPTR,#R3 //check s3 slot
79 RLC A
80 JNC NEXT3 //if slot are free print slot name on lcd
81 SETB RED_LED //turn off RED LED
82 CLR GREEN_LED // turn GREEN LED
83 DEC R4
84 INC R5
85 MOV DPTR,#R3
86 ACALL LCD_DISP
87 MOV DPTR,#R4 //check s4 slot
88 RLC A
89 JNC NEXT4 //if slot are free print slot name on lcd
90 SETB RED_LED //turn off RED LED
91 CLR GREEN_LED // turn GREEN LED
92 DEC R4
93 INC R5
94 MOV DPTR,#R4
95 ACALL LCD_DISP
96 MOV DPTR,#R5 //turn off RED LED
97 SETB RED_LED
98 CLR GREEN_LED
99 DEC R4
100 INC R5
101 JNC R5
```

```

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project: microproject
Source Group:
  projectpa
  projectpa
  projectpa
104 MOV D07H, #54 //if slot are free print slot name on lcd
105 ACALL LCD_DISP
106 NEXT1: MOV A, p2 //check at slot
107 RRC A
108 RRC A
109 RRC A
110 RRC A
111 RRC A
112 JNC NEXT1
113 SETB RED_LED //turn off RED LED
114 CLR GREEN_LED // turn GREEN LED
115 DEC R4
116 INC R3
117 CINE R3, #5, CONT2
118 MOV D07H, #LINE4
119 ACALL LCD_CND
120 CONT1: MOV D07H, #55 //if slot are free print slot name on lcd
121
122 ACALL LCD_DISP
123 NEXT1: MOV A, p2 //check at slot
124 RRC A
125 RRC A
126 RRC A
127 RRC A
128 RRC A
129 RRC A

```

```

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project: microproject
Source Group:
  projectpa
  projectpa
  projectpa
129 RRC A
130 JNC NEXT4
131 SETB RED_LED //turn off RED LED
132 CLR GREEN_LED // turn GREEN LED
133 DEC R4
134 INC R3
135 CINE R3, #5, CONT3
136 MOV D07H, #LINE4
137 ACALL LCD_CND
138 CONT1: MOV D07H, #56 //if slot are free print slot name on lcd
139 ACALL LCD_DISP
140 NEXT1: MOV A, p2 //check at slot
141 RRC A
142 RRC A
143 RRC A
144 RRC A
145 RRC A
146 RRC A
147 RRC A
148 JNC NEXT1
149 SETB RED_LED //turn off RED LED
150 CLR GREEN_LED // turn GREEN LED
151 DEC R4
152 INC R3
153 CINE R3, #5, CONT4
154 MOV D07H, #LINE4

```

```

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project: microproject
Source Group:
  projectpa
  projectpa
  projectpa
141 RRC A
142 RRC A
143 RRC A
144 RRC A
145 RRC A
146 RRC A
147 JNC NEXT8
148 SETB RED_LED //turn off RED LED
149 CLR GREEN_LED // turn GREEN LED
150 DEC R4
151 INC R3
152 CINE R3, #5, CONT5
153 MOV D07H, #LINE4
154 ACALL LCD_CND
155 CONT5: MOV D07H, #58 //if slot are free print slot name on lcd
156 ACALL LCD_DISP
157 NEXT5:
158 acall delay_is
159 acall delay_is //goto chak the slots
160 LAMP UP
161
162
163 commands //function to send command
164 mov p1, #
165 clr p3.4
166 setb p3.6

```

```

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project: microproject
Source Group:
  projectpa
  projectpa
  projectpa
146 RRC A
147 JNC NEXT8
148 SETB RED_LED //turn off RED LED
149 CLR GREEN_LED // turn GREEN LED
150 DEC R4
151 INC R3
152 CINE R3, #5, CONT5
153 MOV D07H, #LINE4
154 ACALL LCD_CND
155 CONT5: MOV D07H, #58 //if slot are free print slot name on lcd
156 ACALL LCD_DISP
157 NEXT5:
158 acall delay_is
159 acall delay_is //goto chak the slots
160 LAMP UP
161
162
163 commands //function to send command
164 mov p1, #
165 clr p3.4
166 setb p3.6
167 acall delay
168 clr p3.4
169 acall delay
170 acall delay
171 ret

```

```

Flash Debug Peripherals Tools SVCS Window Help
Target 1
projectparkinglot.asm
188 clr p3.6
189 acall delay
190 acall delay
191 ret
192
193 write: //function to send data
194 mov p1, #
195 setb p3.4
196 setb p3.6
197 acall delay
198 clr p3.6
199 acall delay
200 acall delay
201 ret
202
203
204 LCD_DISP: CLR A //function to send display string
205 MOV A, #A+D07H
206 JZ EXIT1
207 INC D07H
208 ACALL write
209 SJMP LCD_DISP
210 EXIT1: RET
211
212
213 LCD_CND: CLR A //function to send commandes

```

```

212
213 LCD_CMD: CLR A           //function to send commandes
214 MOV A, R0+DPTR
215 DJNZ EXIT2
216 INC DPTR
217 ACALL command
218 SJMP LCD_CMD
219 EXIT2: RET
220
221
222
223 delay:
224 mov r0, #1ch
225 rep:
226 djnz r0, rep
227 ret
228
229 delay_1s:                //function for 1s delay
230 mov r3, #00h
231 dlist:
232 mov r2, #0ffh
233 dls:
234 mov r1, #0ffh
235 dls1:
236 djnz r1, dls
237 djnz r2, dls

```

SUCCESSFUL SIMULATION OF CODE:

```

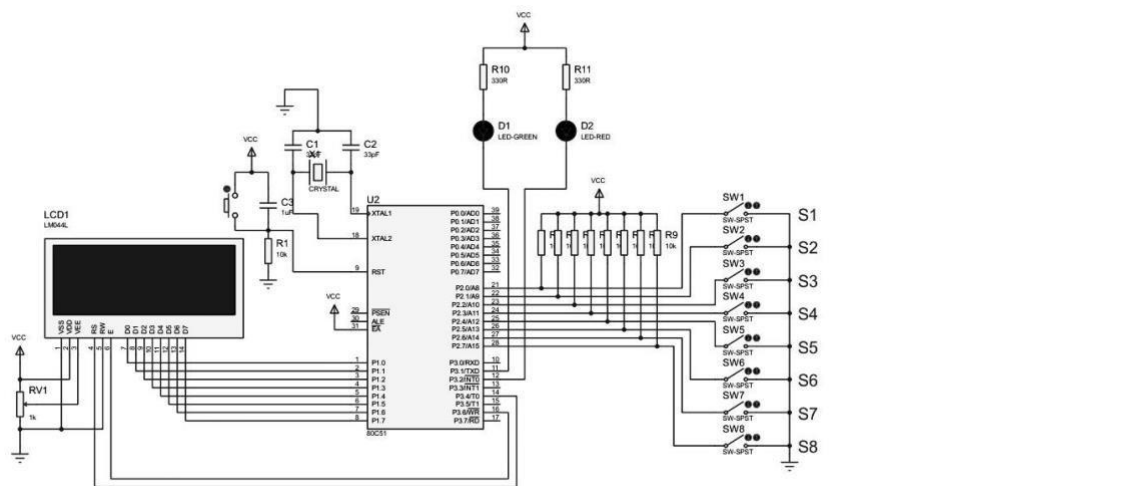
241 TEXT1: DB " PARKING SYSTEM",0
242 TEXT2: DB " NO FREE SLOTS",0
243
244 FREE: DB " Free slots",0
245 S1: DB "S1 ",0
246 S2: DB "S2 ",0
247 S3: DB "S3 ",0
248 S4: DB "S4 ",0
249 S5: DB "S5 ",0
250 S6: DB "S6 ",0
251 S7: DB "S7 ",0
252 S8: DB "S8 ",0
253
254
255 INIT_COMMANDS: DB 38h,06h,0ch,01H,80H,0
256 LINE1: DB 01H,06H,06H,80H,0
257 LINE2: DB 0C0H,0
258 LINE3: DB 094H,0
259 LINE4: DB 0D4H,0
260
261 CLEAR: DB 01H,0
262 PARKED_VEHICAL: DB " parked vehicles:",0
263
264 end
265

```

".\Objects\microproject 18BEC0976"...
 18BEC0976 - 0 Error(s), 3 Warning(s).

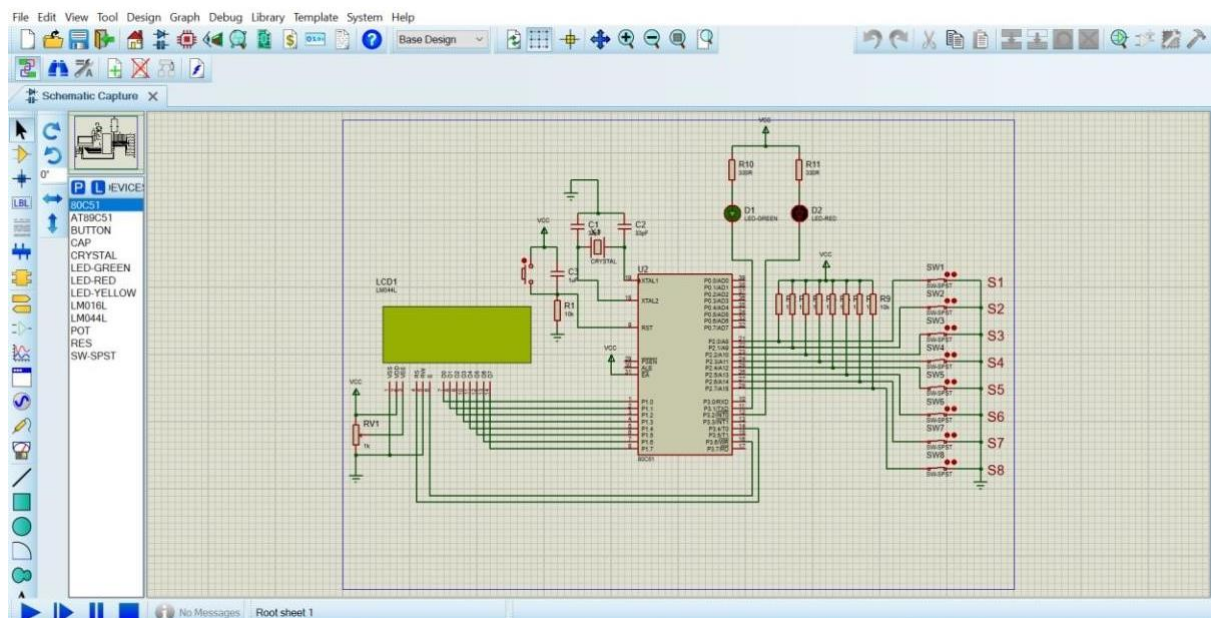
(ii) IMPLEMENTATION USING PROTEUS

CIRCUIT DIAGRAM:

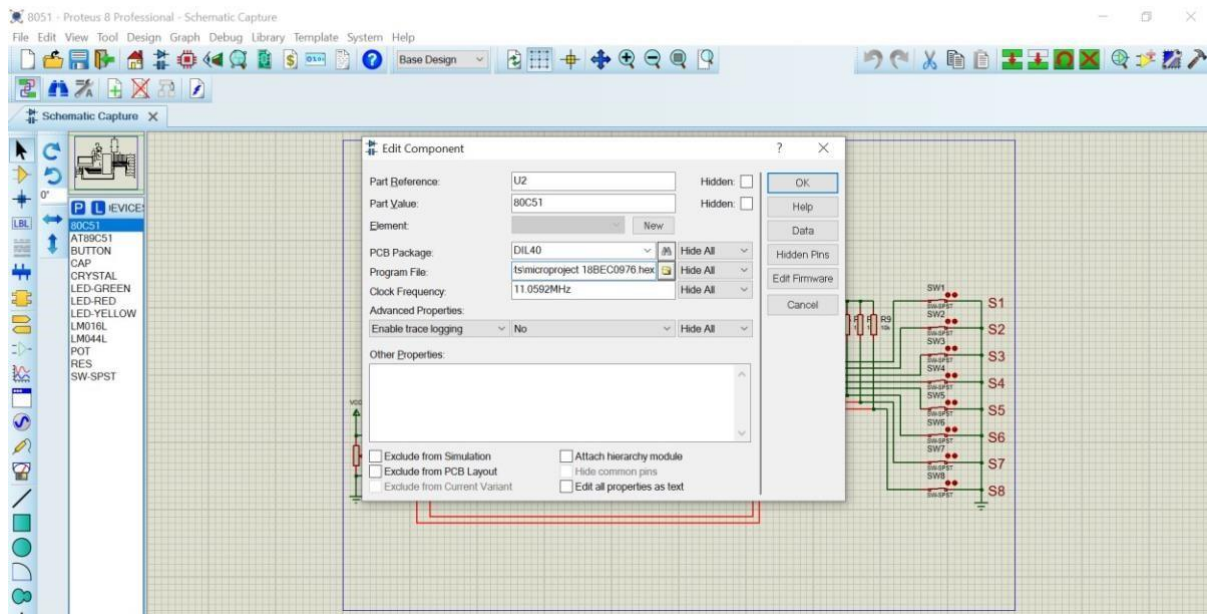


SIMULATIONS AND RESULTS of Proteus:

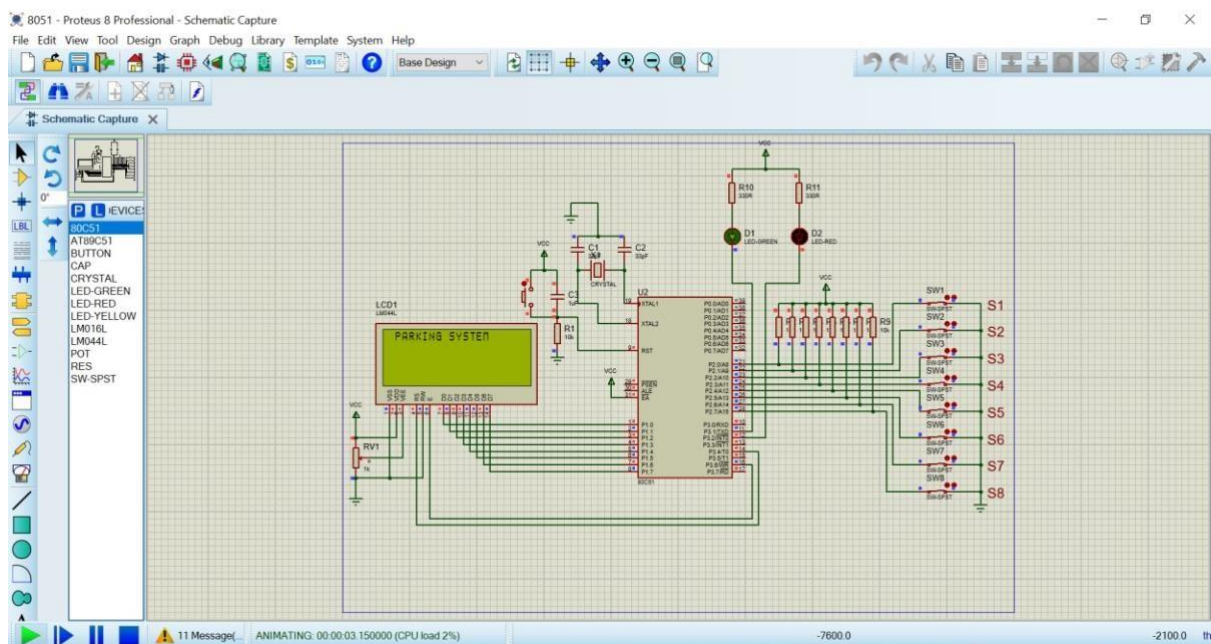
Circuit:



Importing HEX file to MC8051:



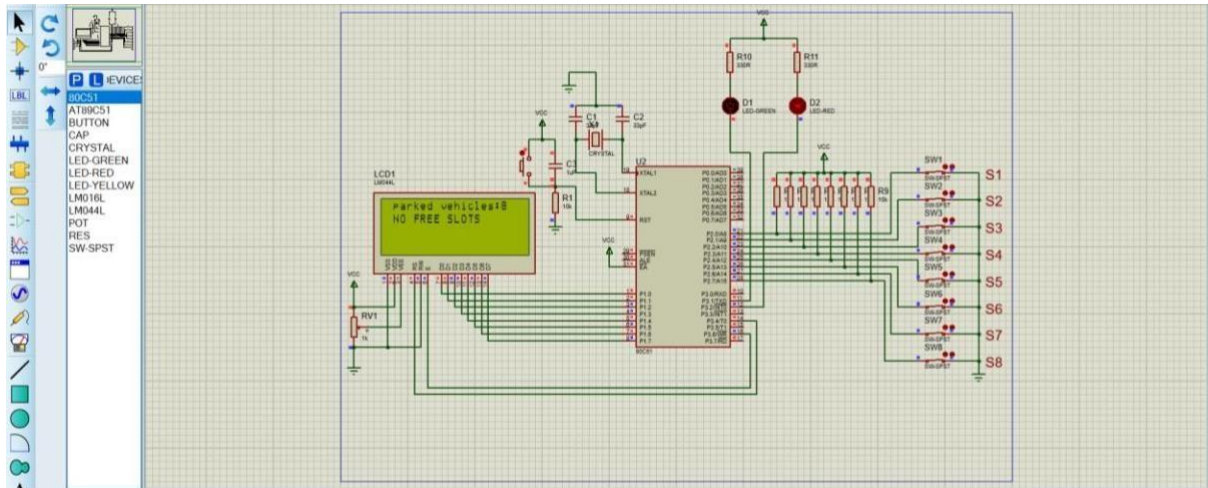
OUTPUT RESULTS:



CASE 1:

WHEN ALL SLOTS ARE FULL:

- Parked vehicles: 8
- Free slots: NO FREE SLOTS



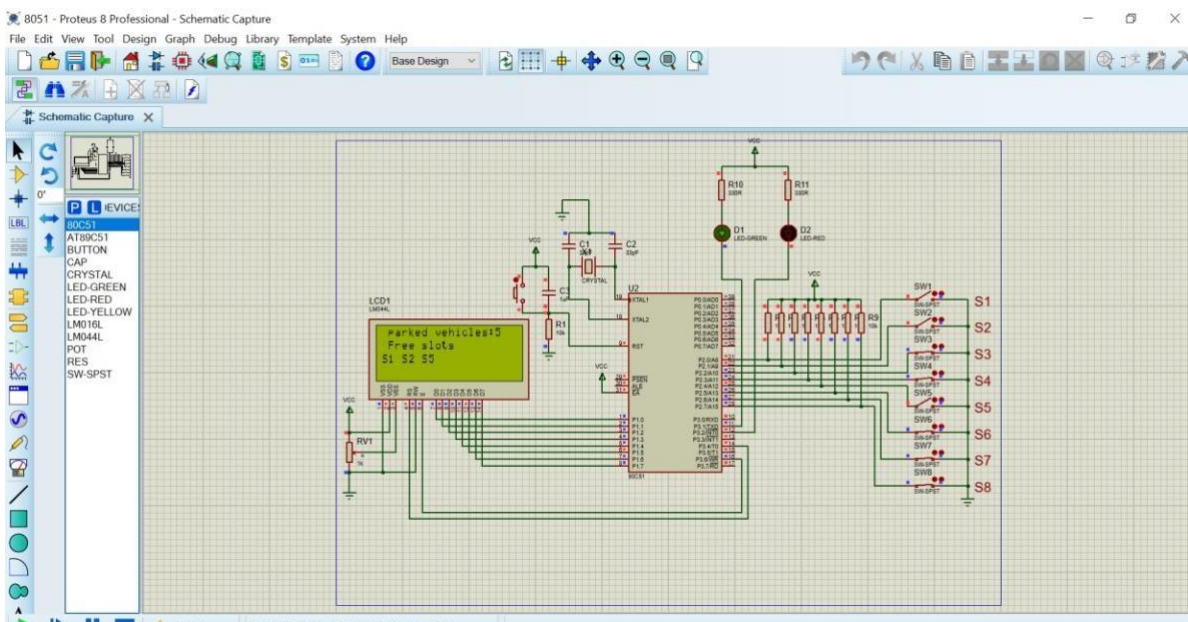
CASE 2: WHEN SOME SLOTS ARE FREE: •

Parked vehicles: 5 • Free slots:

FREE SLOTS

S1 S2 S5

(NOTE: S1 S2 S5 these are the free slots available)



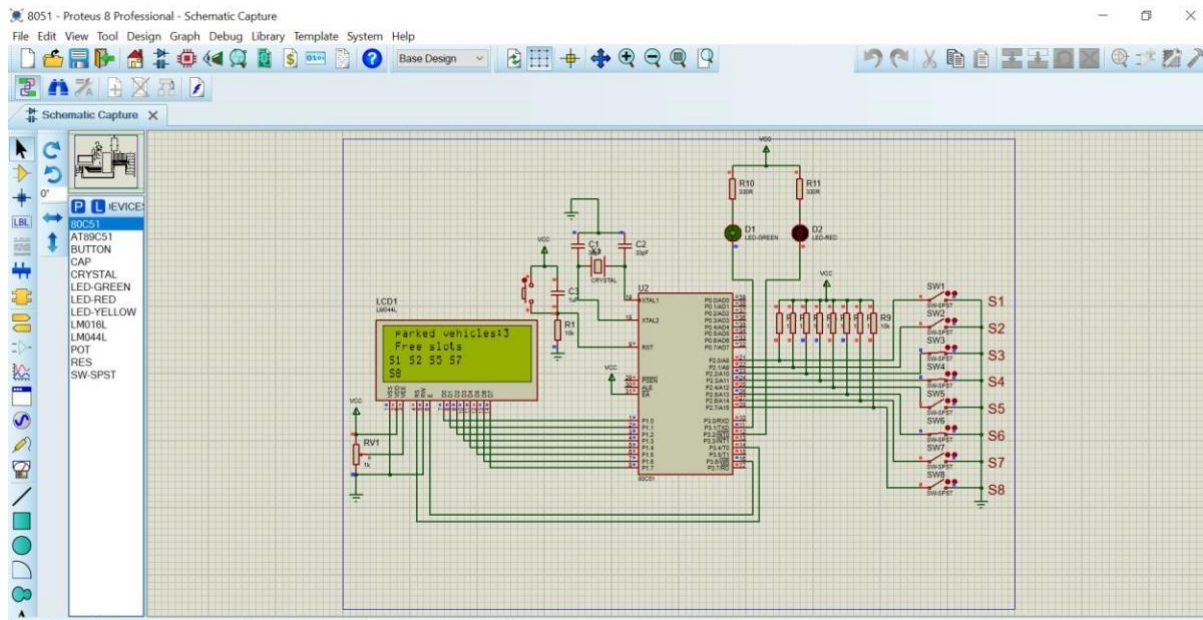
CASE 3: WHEN SOME SLOTS ARE FREE: •

Parked vehicles: 3

- Free slots:
FREE SLOTS

S1 S2 S5 S7 S8

(NOTE: S1 S2 S5 these are the free slots available)



CASE 4: WHEN ALL SLOTS ARE FREE: •

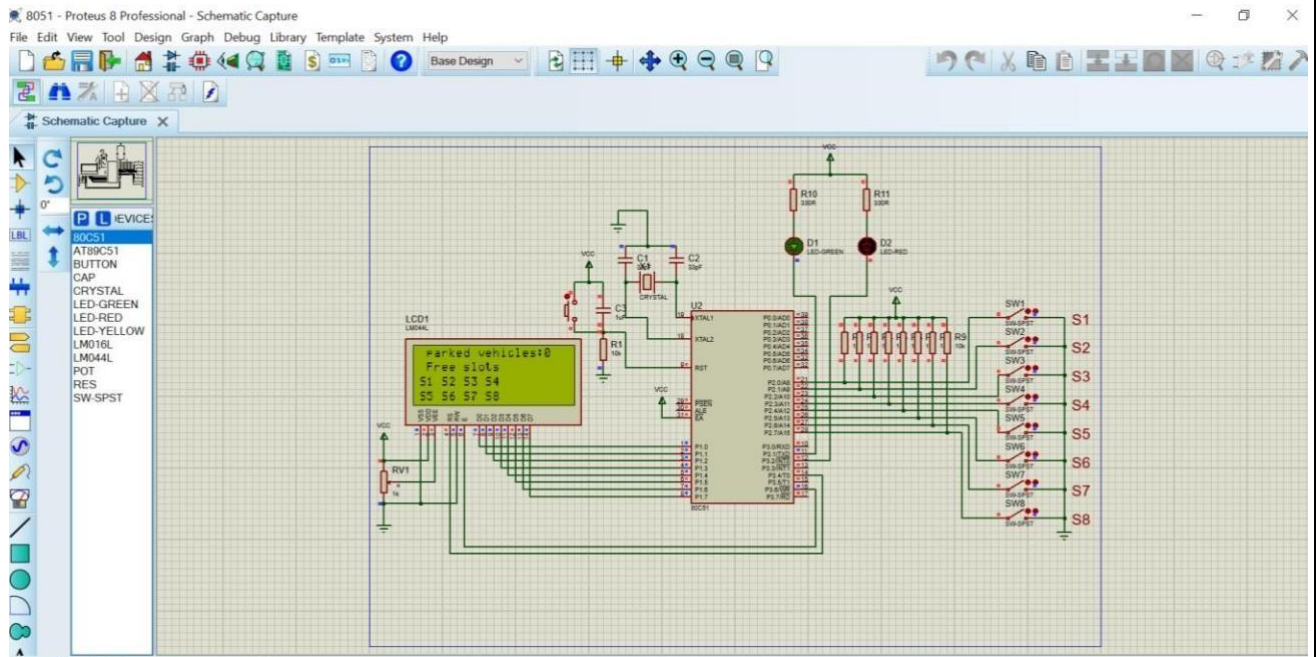
Parked vehicles: 0 • Free slots:

FREE SLOTS

S1 S2 S3 S4

S5 S6 S7 S8

(NOTE: S1 S2 S1 S2 S3 S4 S5 S6 S7 S8, these are the free slots available)



From above results, we can see functional parking system that works on the basis of 8051 microcontroller using ASM language.

Here the switches will be replaced by sensors in real like which will detect the presence of the car and based on that, the cars will be detected and hence the Smart parking system is implemented successfully.

10.CONCLUSION:

The system can be used at all places starting from domestic to the industrial sectors. The simplicity in the usage of circuit helps it to be used by a large number of people, because people with less knowledge of hardware can also use it without facing any problem. This Automated car parking system enables the parking of vehicles and thus reduces the time taken to check the space to be used by displaying the spot where the space for parking is available on an LCD display by using IR sensors (represented as switches) at the entrance. This can be expanded in the sense of security. Using metal detectors and CCTV cameras security of the parking area can be enhanced, we can add the Pick and Place facility to park the cars automatically.

FUTURE SCOPE:

- We can monitor some parameters like temperature, fire and at the same time control them. Our project monitors only the Parking slots. If we must detect fire or smoke in the parking area then we can connect these sensors to our project and at the output side we can connect water sprinkler to control the fire.
- The data can be stored using cloud technologies like AWS, Fire Store, Azure, etc and retrieve it using an app that can be developed in mobiles so that the data displayed will be given directly to the user.

11. REFERENCES:

<https://www.keil.com/rtx51/>

http://fadhilalakwa.weebly.com/uploads/5/3/6/4/5364958/_the_8051_microcontroller_and_embedded_systems_using_assembly_and_c-2nd-ed_by_mazidi.pdf

<https://export.arxiv.org/ftp/arxiv/papers/1912/1912.01697.pdf>

<https://youtu.be/xIYWZTu4L0c>

https://www.technoarete.org/common_abstract/pdf/IJERCSE/v4/i6/Ext_03_295.pdf

https://www.researchgate.net/publication/321027384_Modern_car_parking_system_using_Micro_controller_and_smart_intelligent_application_system_Techniques

<http://blogs.cornell.edu/armapp/2013/03/27/revenue-management-in-carparking-industry/>

<http://arxiv-export-lb.library.cornell.edu/pdf/1708.07932>