LAKSHITA K J

22BEC1001

EMBEDDED C PROGRAMMING – DIGITAL ASSESSMENT 2

QUESTION

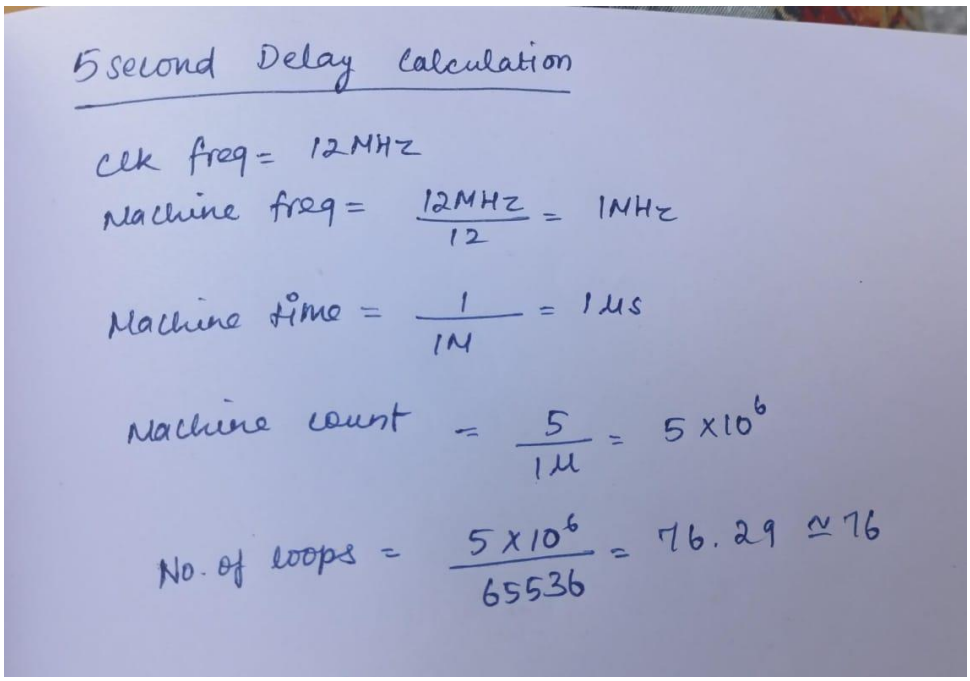
Application : Parking lot controller

Demonstrate an automatic gate opening and closing of a parking lot in a company. The parking is allotted only for those working in the company. A secret PIN is shared for them, if only the secret pin is valid the gate at the entrance will open. The current number of available lot is displayed in the LCD.

There are two gate, one at entrance and one at exit. Both the gates are attached with sensors. If a car approaches, the entrance gate, the controller will check for availability of parking lot. If it is full, then LCD will display "PARKING FULL". If parking space is available, the controller will ask for secret PIN. After verifying the secret PIN, the entry gate should open exactly for 5 secs (Use motor for gate opening, timers for timing) and then close.

If a car approaches exit gate, the gate should open for 5 secs and then close. Always LCD should display the status of available lots for parking.

TIMING CALCULATION



CODE:

```
#include<reg51.h>
#include<string.h>
//password
sbit pw1=0x80;
sbit pw2=0x81;
sbit pw3=0x82;
sbit pw4=0x83;
//LCD
sbit rs = P3^4;
sbit rw = P3^5;
sbit en = P3^6;
sfr data_lcd=0x90;
//motor for door
sfr doors=0xA0;
unsigned int slots=5;
unsigned char msg[]="PARKING FULL";
unsigned char str1[] = "Number of slots";
unsigned char str2[] = "available: ";
unsigned char pwmsg[] = "Enter Password";
unsigned char wrong[] = "Wrong Password";
void lcd_init();
void cmd_sr(unsigned char);
void data_sr(unsigned char);
void delay(){
      unsigned int i;
      for(i=0;i<10000;i++);
}

void cmd_sr(unsigned char x){
      data_lcd=x;
      rw=0;
      rs=0;
      en=1;
      delay();
      en=0;
```

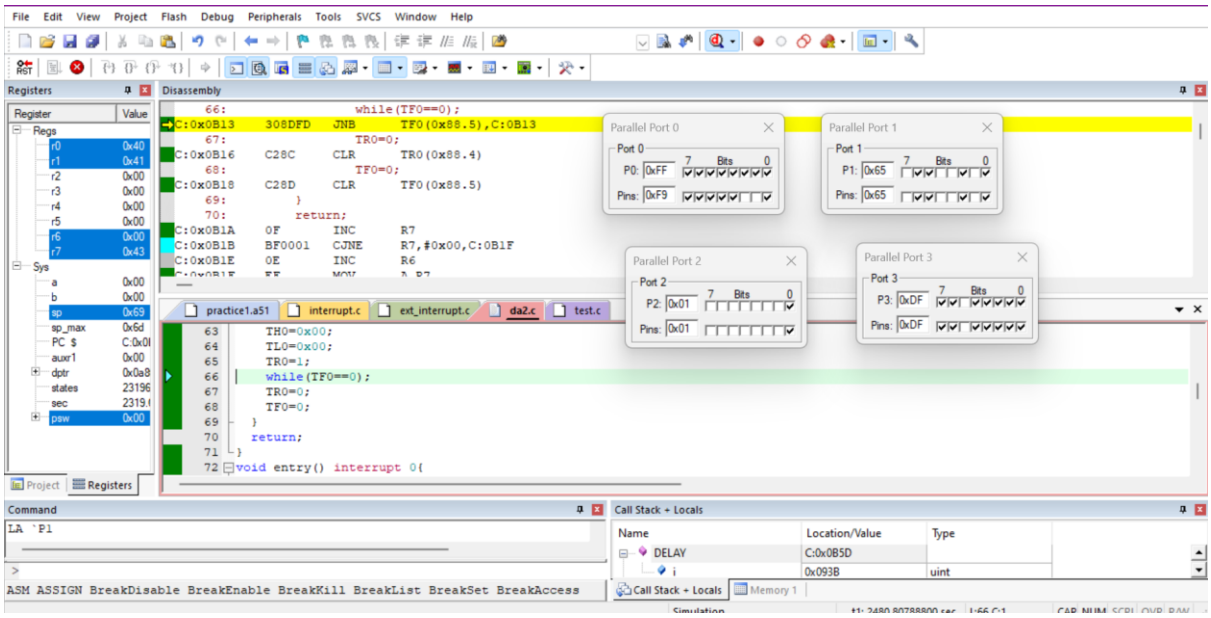
```
}

void data_sr(unsigned char x){
      data_lcd=x;
      rw=0;
      rs=1;
      en=1;
      delay();
      en=0;
}

void lcd_init(){
      cmd_sr(0x38);
      delay();
      cmd_sr(0x01);
      delay();
      cmd_sr(0x0E);
      delay();
      cmd_sr(0x06);
      delay();
      cmd_sr(0x80);
      delay();
}
void timer_5s(){
      unsigned int i;
      for(i=0;i<76;i++){
            TF0=0;
            TH0=0x00;
            TL0=0x00;
            TR0=1;
            while(TF0==0);
            TR0=0;
            TF0=0;
      }
      return;
}
```

```
void entry() interrupt 0{
      unsigned int i;
      if(slots>0){
            cmd_sr(0x01);
            for(i=0; i<strlen(pwmsg); i++){
                  data_sr(pwmsg[i]);
            }
      delay();

        if (pw1 == 1 && pw2 == 0 && pw3 == 0 && pw4 == 1) {
            slots--;

            doors = 0x01; delay();
            doors = 0x02; delay();
            doors = 0x04; delay();
            doors = 0x08; delay();

            timer_5s();

            doors = 0x04; delay();
            doors = 0x02; delay();
            doors = 0x01; delay();
        }
                        else{
                              cmd_sr(0x01);
            for(i=0; i<strlen(wrong); i++){
                  data_sr(wrong[i]);
            }
    }
      }
}
void exit() interrupt 2{
      if(slots<5){
      slots++;
      doors = 0x08; delay();
        doors = 0x40; delay();
```

```
        doors = 0x20; delay();
        doors = 0x10; delay();

        timer_5s();

        doors = 0x20; delay();
        doors = 0x40; delay();
        doors = 0x80; delay();
    }
}
void main(){
      unsigned int i;
      pw1 = pw2 = pw3 = pw4 = 1;
      lcd_init();
      TMOD=0x01;
      EX0=1;
      IT0=1;
      EX1=1;
      IT1=1;
      EA=1;
      cmd_sr(0x01);
      delay();

      while(1){
            cmd_sr(0x01);
            delay();

            if(slots > 0){
                  cmd_sr(0x80);
                  for(i = 0; i < strlen(str1); i++){
                        data_sr(str1[i]);
                  }

                  cmd_sr(0xC0);
                  for(i = 0; i < strlen(str2); i++){
                        data_sr(str2[i]);
```

```
                }


                data_sr('0' + slots);
        }
        else if(slots==0){
                cmd_sr(0x80);
                for(i = 0; i < strlen(msg); i++){
                        data_sr(msg[i]);
                }
        }


        delay();
    }
}
```

KEIL SIMULATION AND EXPLANATION

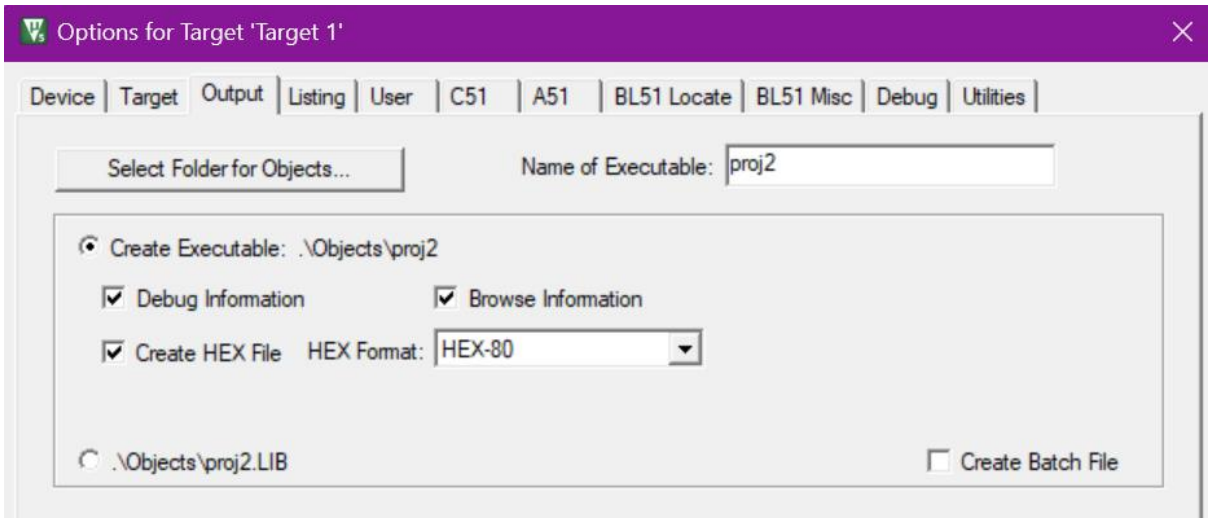


This program is designed for an 8051 microcontroller to manage a simple parking system with limited slots, LCD display, and motor-controlled entry/exit. Initially, five parking slots are available. The system uses an LCD to display either the number of available slots or a "PARKING FULL" message when all slots are occupied. Entry is controlled using an external interrupt (INT0), where the user must input a predefined 4-bit password (1 0 0 1). If the password is correct, the system decreases the slot count, opens the gate using motor control signals, waits for 5 seconds (simulated using Timer0), and then closes the gate. If the password is incorrect, a "Wrong Password" message is shown on the LCD. Exit is triggered using a second external interrupt (INT1), which increases the available slot count and similarly operates the gate. The

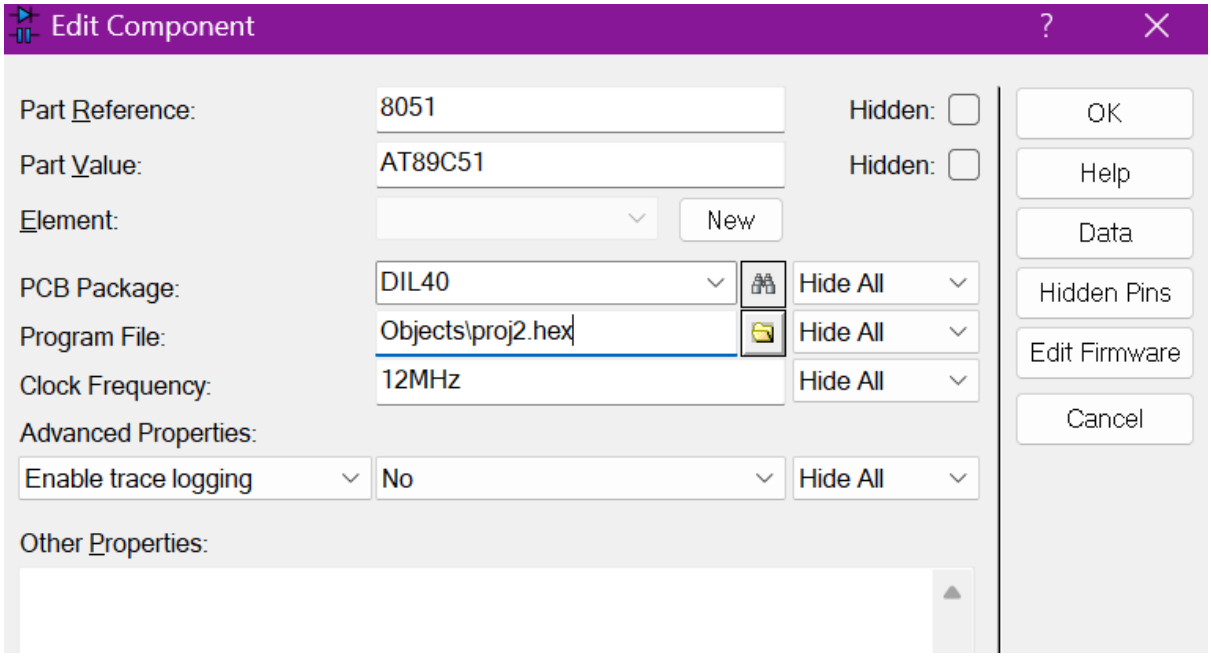
LCD is initialized in 8-bit mode and continuously updates in the main loop to reflect current parking status.

SETTING IN KEIL AND PROTEUS

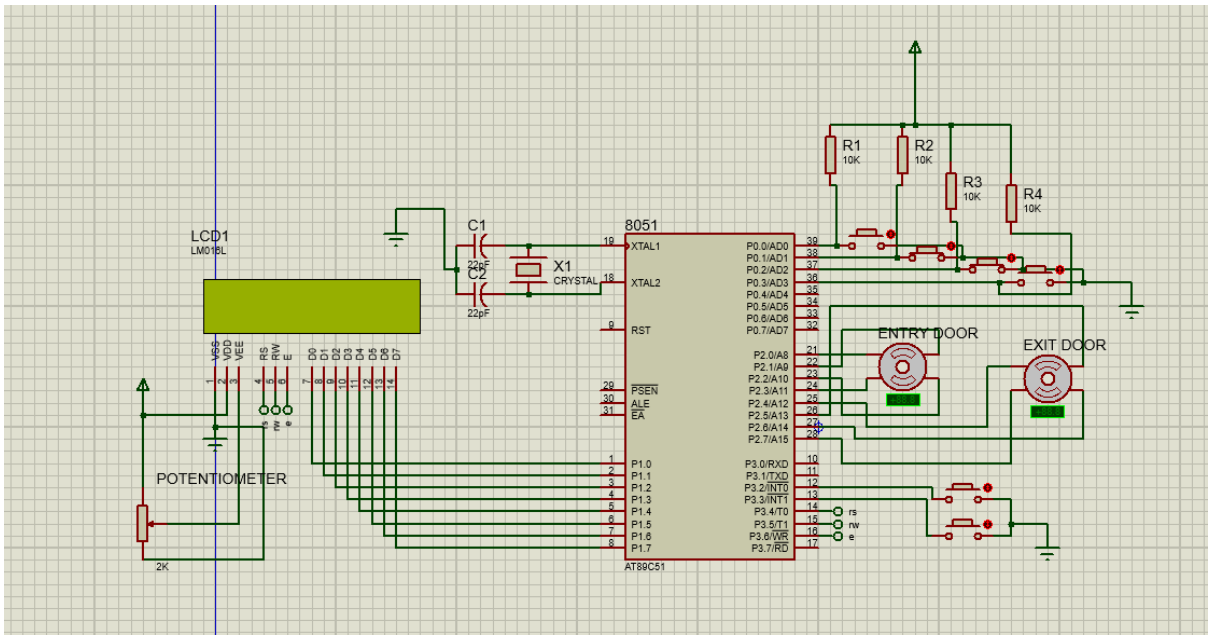
Once the code is typed, go to **Project → Options for Target → Output** tab. Click on creae hex file and build, rebuild and translate again.



In proteus, open a new schematic, save it in the same folder as that of the hex file and make the connections as required. Click on the 8051, in the program file field, browse and select the HEX file generated by Keil.



PROTEUS SIMULATION

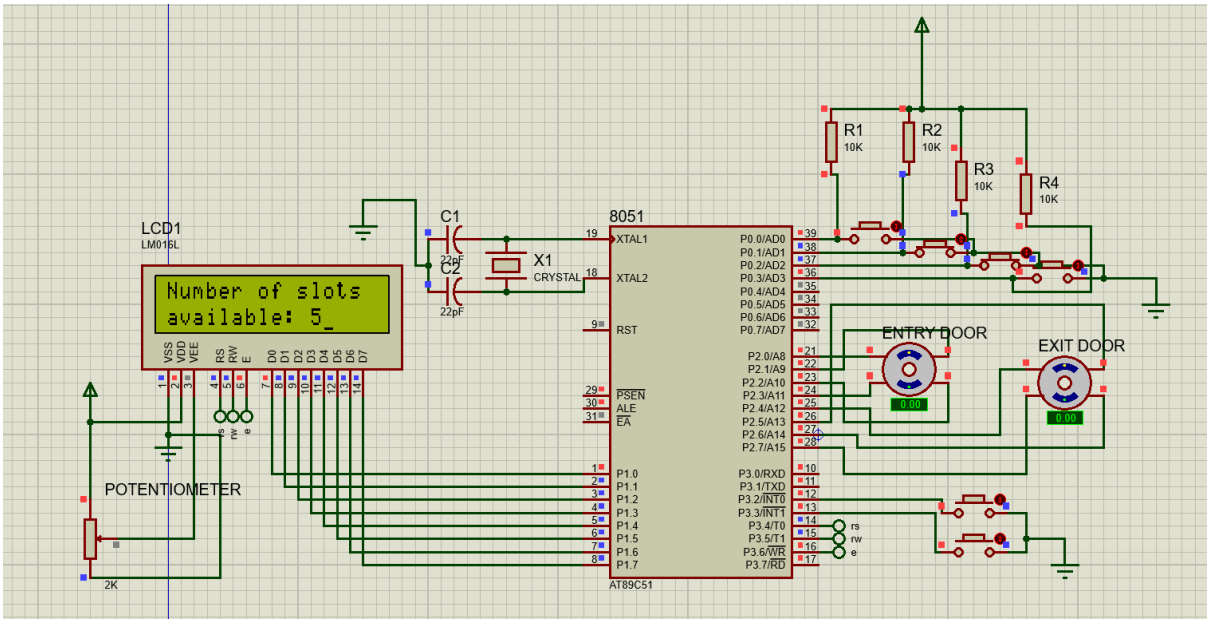


Components used:

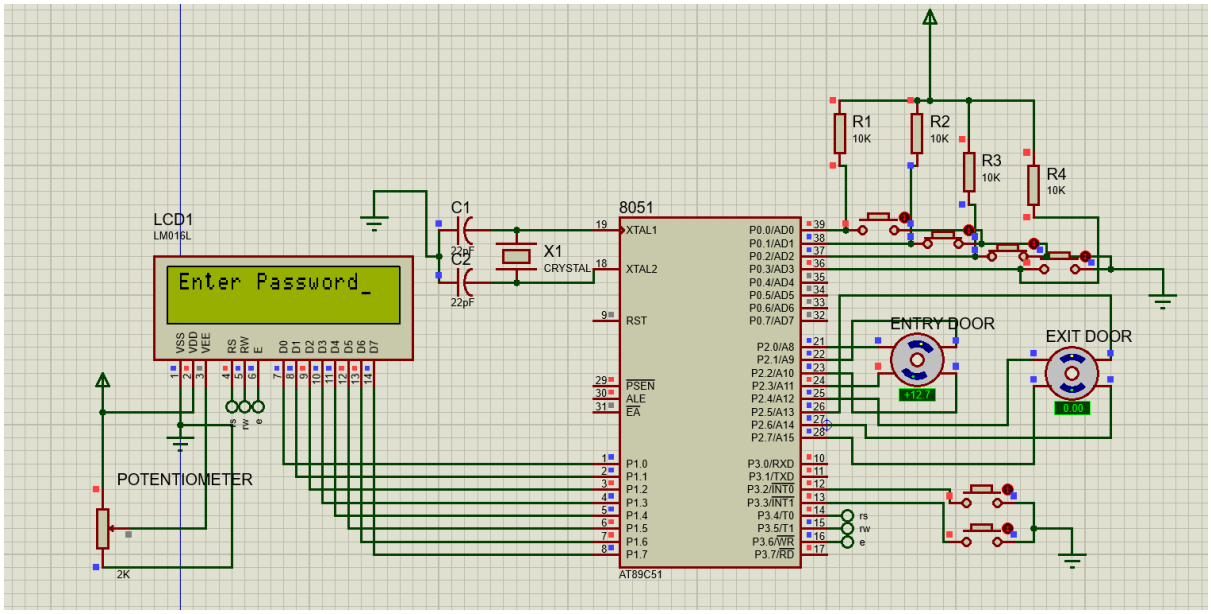
- 8051 microcontroller
- LCD – RS pin to P3.4, RW to pin 3.5, enable pin to P3.6, data line to port P1, VSS to ground, VDD to voltage source, VEE to a 2K potentiometer
- Entry and exit door motors – entry connected to P2.0 – P2.3, exit to P2.4 – P2.7
- External interrupts are accessed via push buttons at P3.2 and P3.3
- Password is entered via push buttons at P0.0 to P0.3. 10K pull-up resistors are connected to this port

Working:

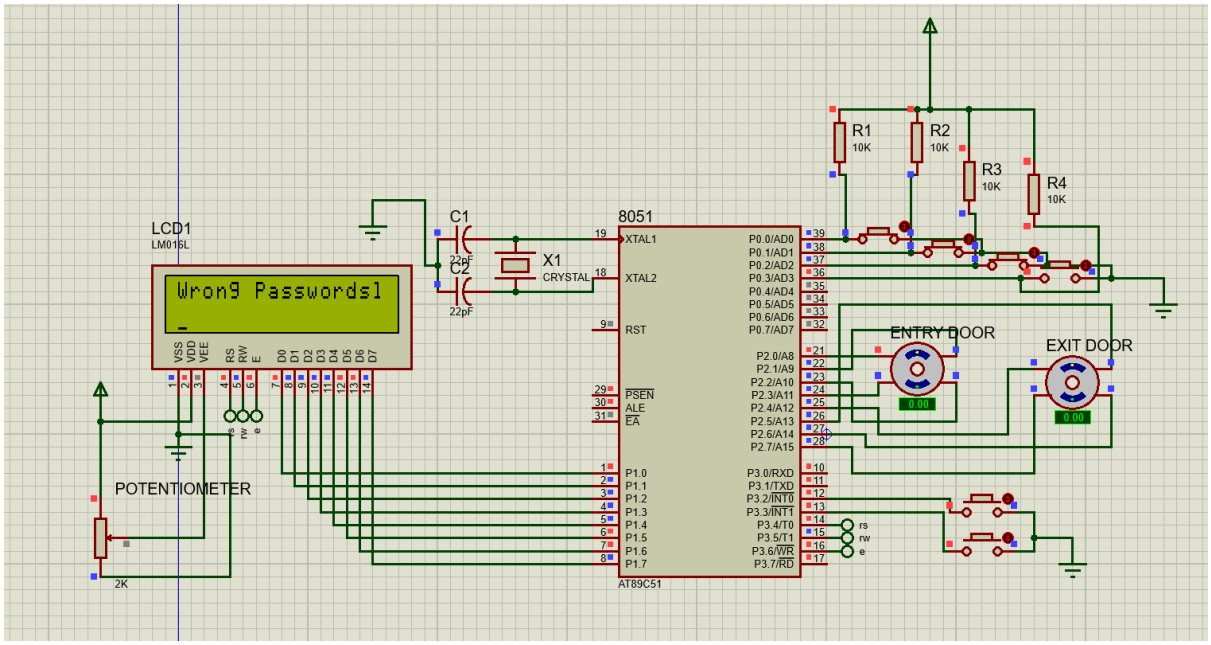
1. Initially the number of slots available is 5.



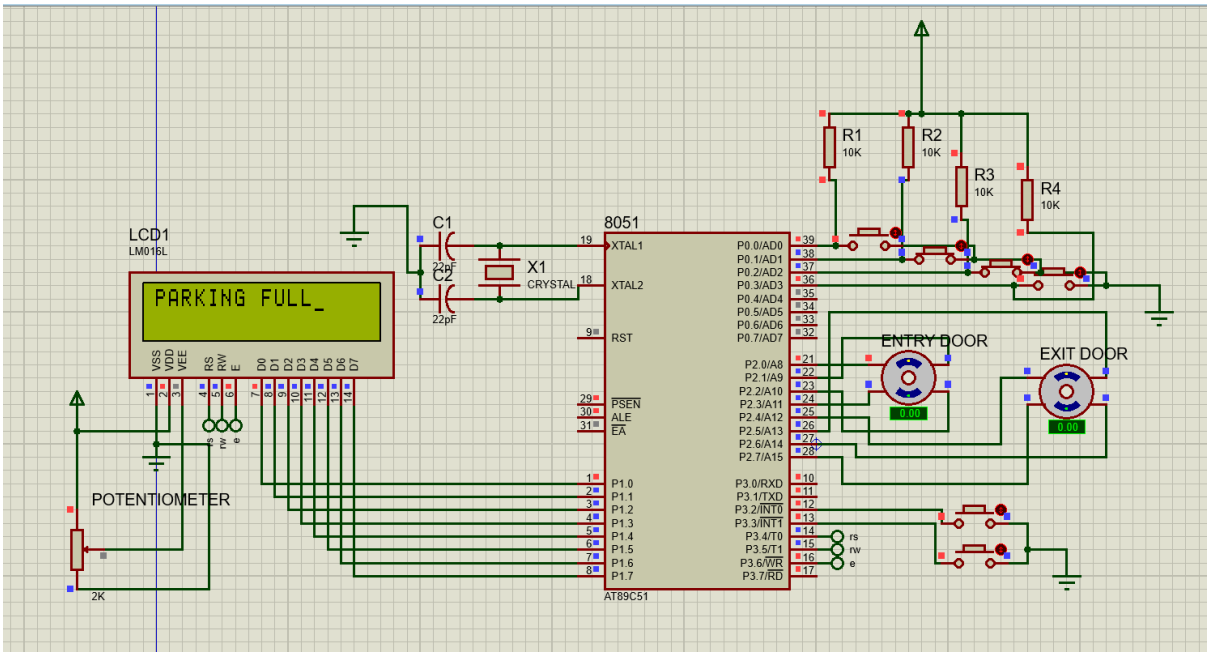
2. When a car comes to the entry gate, the interrupt P3.2 is triggered and we are asked to enter the password. If the password is correct, the entry gate opens (motor moves clockwise), waits for 5 seconds and closes. Number of slots gets decremented.



3. If password is wrong, message is displayed on LCD



4. When car approaches exit gate, P3.3 gets an interrupt. The exit gate opens, waits for 5s and then closes, number of slots available gets incremented.
5. Once all slots are full, LCD displays that message.


LCD1
LM016L
PARKING FULL_
VSS
VDD
VEE
RS
RW
E
D0
D1
D2
D3
D4
D5
D6
D7
C1
22pF
C2
22pF
X1
CRYSTAL
8051
XTAL1
XTAL2
RST
PSEN
ALE
EA
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
AT89C51
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
R1
10K
R2
10K
R3
10K
R4
10K
ENTRY DOOR
EXIT DOOR
POTENTIOMETER
2K
rs
rw
e