Chapter 5

5.1 Square wave generation using port pins of 8051

To generate square wave, we have to toggle the port bit with a delay. 8051 Timers can be used for delay generation. Delay also can be generated using registers.

a. Example program for square wave generation using Timers

1. Assume that XTAL = 11.0592 MHz, write a program to generate a square wave of 2 kHz frequency on pin P1.5.

T = 1 / f = 1 / 2 kHz = 500 us the period of square wave.

1 / 2 of it for the high and low portion of the pulse is 250 us.

(65535-count+1) X 1.085 us = 250 us
65536-count = 250 us / 1.085 us = 230
Count = 65536 – 230 = 65306 which in hex is FF1AH.

```
ORG 0000H
AGAIN: SETB P1.5
ACALL T1M1DELAY
CLR P1.5
ACALL T1M1DELAY
SJMP AGAIN
T1M1DELAY:
MOV TMOD,#10H           ; Timer 0, 16-bitmode
MOV TL1,#1AH     ;  TL1=1A, low byte of  timer
MOV TH1,#0FFH           ;   TH1=FF, the high byte
SETB TR1                ;  Start timer 1
BACK: JNB TF1,BACK      ;  until timer rolls over
CLR TR1                 ;  Stop the timer 1
CLR TF1                 ;   Clear timer flag 1
RET
END
```

2. Assume that XTAL = 11.0592 MHz, write a program to generate a square wave of 5kHz frequency on pin P1.5.

T = 1 / f = 1 / 5 kHz = 200 us the period of square wave.

1 / 2 of it for the high and low portion of the pulse is 100 us.

(65535-count+1) X 1.085 us = 100 us
65536-count = 100 us / 1.085 us = 92
Count = 65536 – 92 = 65444 which in hex is FFA4H.

```
ORG 0000H
AGAIN: SETB P1.5
ACALL T1M1DELAY
CLR P1.5
ACALL T1M1DELAY
SJMP AGAIN
T1M1DELAY:
MOV TMOD,#10H            ; Timer 0, 16-bitmode
MOV TL1,#A4H    ;  TL1=1A, low byte of  timer
MOV TH1,#0FFH            ;   TH1=FF, the high byte
SETB TR1                 ;  Start timer 1
BACK: JNB TF1,BACK       ;  until timer rolls over
CLR TR1                  ;  Stop the timer 1
CLR TF1                  ;   Clear timer flag 1
RET
END
```

3. **Assume that XTAL = 11.0592 MHz, write a program to generate a square wave of 1kHz frequency on pin P2.0.**

T = 1 / f = 1 /1 kHz = 1000 us the period of square wave.

1 / 2 of it for the high and low portion of the pulse is 500 us.

(65535-count+1) X 1.085 us = 500 us
65536-count = 500 us / 1.085 us = 461
Count = 65536 – 461 = 65075 which in hex is FE33H.

```
ORG 0000H
AGAIN: SETB P2.0
ACALL T0M1DELAY
CLR P2.0
ACALL T0M1DELAY
SJMP AGAIN
T0M1DELAY:
MOV TMOD,#01h            ; Timer 0, 16-bitmode
MOV TL0,#A4H    ;  TL1=1A, low byte of  timer
MOV TH0,#0FFH            ;   TH1=FF, the high byte
SETB TR0                 ;  Start timer 1
BACK: JNB TF0,BACK       ;  until timer rolls over
CLR TR0                  ;  Stop the timer 1
CLR TF0                  ;   Clear timer flag 1
RET
END
```

4. **Assume that XTAL = 12 MHz, write a program to generate a square wave of 5kHz frequency on pin P2.0. Use Timer0 in Mode1**

T = 1 / f = 1 /5 kHz = 200 us the period of square wave.

1 / 2 of it for the high and low portion of the pulse is 100 us.

(65535-count+1) X 1 us = 100 us
65536-count = 100 us / 1 us = 100
Count = 65536 – 100 = 65436 which in hex is FF9CH.

```
ORG 0000H
AGAIN: SETB P2.0
ACALL T0M1DELAY
CLR P2.0
ACALL T0M1DELAY
SJMP AGAIN
T0M1DELAY:
MOV TMOD,#01h              ; Timer 0, 16-bitmode
MOV TL0,#9CH     ;  TL1=1A, low byte of  timer
MOV TH0,#0FFH              ;   TH1=FF, the high byte
SETB TR0                   ;  Start timer 1
BACK: JNB TF0,BACK         ;  until timer rolls over
CLR TR0                    ;  Stop the timer 1
CLR TF0                    ;   Clear timer flag 1
RET
END
```

b. Example program for square wave generation using registers of 8051

1. **Assume that XTAL = 11.0592 MHz, write a program to generate a square wave on pin P2.0. Use 8051 registers to generate delay. Also calculate the delay generated**

Program:

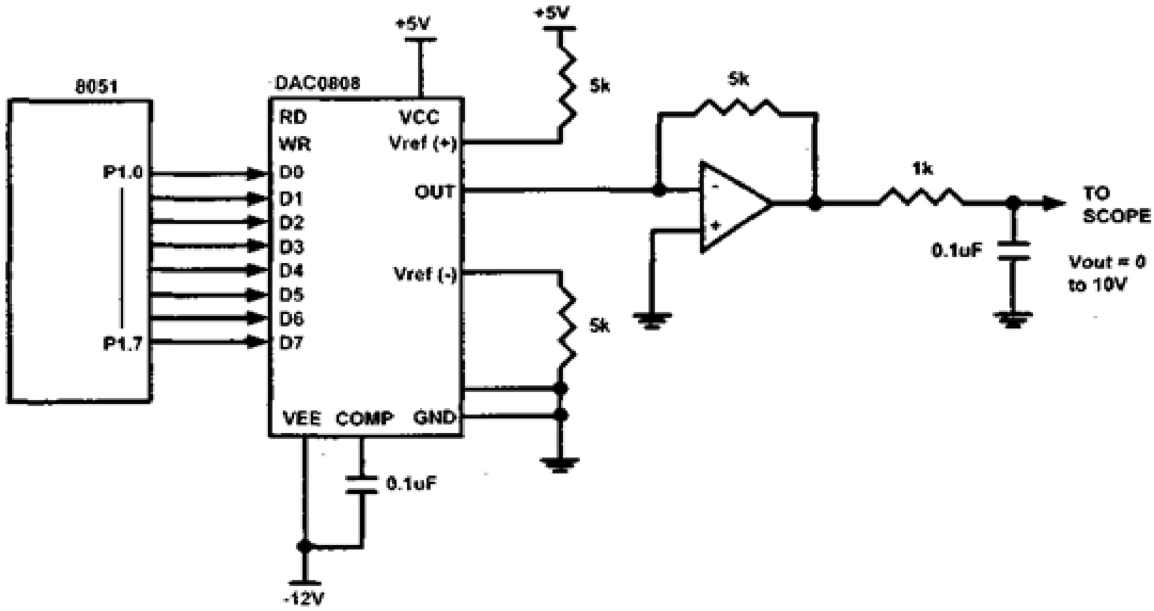
```
ORG 0000H
AGAIN: SETB P2.0
ACALL REGDELAY
CLR P2.0
ACALL REGDELAY
SJMP AGAIN
REGDELAY:
MOV R7,#05
L3: MOV R6,#255
L2: MOV R5,#255
L1: DJNZ R5,L1
DJNZ R6,L2
DJNZ R7,L3
RET
END
```

Delay generated:

First we have to find out the number of machine cycle each instruction takes to execute. If we multiply it by the time period of one machine cycle, we get the total execution time of that instruction. Adding up the execution time of all the instructions in the subroutine, gives the total delay generated.
One machine cycle = 12 clock cycles
Time for one clock cycle = (1/11.0592)uS
Time for one machine cycle = 12/11.0592 = 1.085uS

a	b	c	d	e
Instruction	no. of machine cycles	no. of times instr. Executed	total no. of machine cycles (bXc)	time taken for execution (d X 1.085uS)
MOV R7,#05	1	1	1	1.085
L3: MOV R6,#255	1	5	5	5.425
L2: MOV R5,#255	1	5X255 = 1275	1275	1379.55
L1: DJNZ R5,L1	2	5X255X255 = 325125	650250	703570.5
DJNZ R6,L2	2	5X255 = 1275	2550	2766.75
DJNZ R7,L3	2	5	10	10.85
RET	2	1	2	2.17
Total Time taken (delay generated) = 707,736.33uS = 707.74mS = 0.7S				

5.2 Square and triangular Waveform generation using DAC



DAC 0808 is an 8 bit R-2R type digital to analog converter. The analog output is the current signal at Iout. This is converted to voltage using I to V converter.

Program for Square waveform generation using DAC

```
ORG 0000H
LOOP1: MOV A, #00H
MOV P1,A
ACALL DELAY
MOV A,#0FFH
MOV P1.A
ACALL DELAY
SJMP LOOP1
DELAY:
MOV R7,#180
L3: MOV R6,#255
L2: MOV R5,#255
L1: DJNZ R5,L1
DJNZ R6,L2
DJNZ R7,L3
RET
END
```

Program for Ramp waveform generation

```
ORG 0000H
CLR A
LOOP1: MOV P1,A
INC A
SJMP LOOP1
END
```

Program for Triangular wave generation

```
ORG 0000H
CLR A
LOOP1: MOV P1,A
INC A
CJNE A, #0FFH, LOOP1
LOOP2: DEC A
MOV P1,A
CJNE A, #00H,LOOP2
SJMP LOOP1
END
```

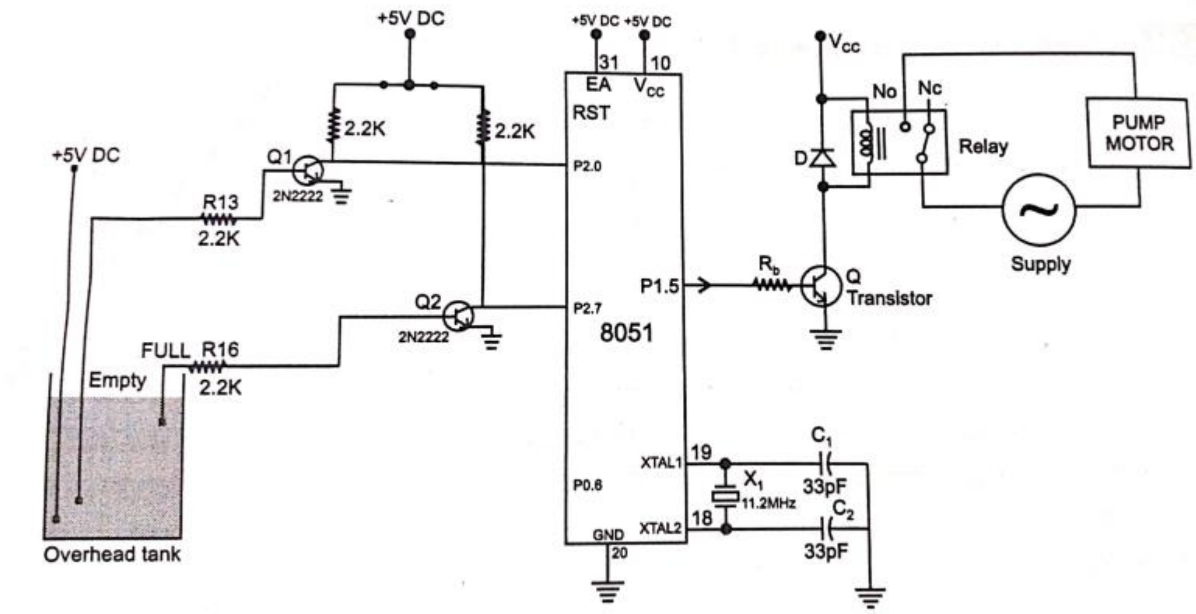
5.3 Water level controller

Water Level Controller using 8051 Microcontroller will help to automatically control the water level in a tank by switching ON/OFF a motor by sensing the water level in a tank. This system monitors the water level of the tank and automatically switches ON the motor whenever tank is empty**.**

The motor is switched OFF when the overhead tank or container is FULL.

Water sensing can be done by using a set of 2 wires, which are placed at different levels in tank. DC supply probe is placed at the base of the tank.

Circuit Diagram



Three wires are inserted to the tank as water level sensors. Of these, two wires which are dipped into the tank will indicate the two different water levels: EMPTY and FULL. The third wire is for positive voltage. Based on the outputs of these wires, microcontroller controls the motor.

Initially when the tank is empty, the transistors will be OFF and P2.0 and P2.7 will be at logic 1. Motor is made On by making P1,5 high.. As the water level rises, the base of the EMPTY level transistor becomes high and P2.0 goes to logic 0 and then as it further rises, the full level transistor will be ON and P2.7 will be at logic 0. Now the motor automatically stops. Again, the motor runs when water level in the tank becomes LOW.

Algorithm for Water Level Controller Circuit

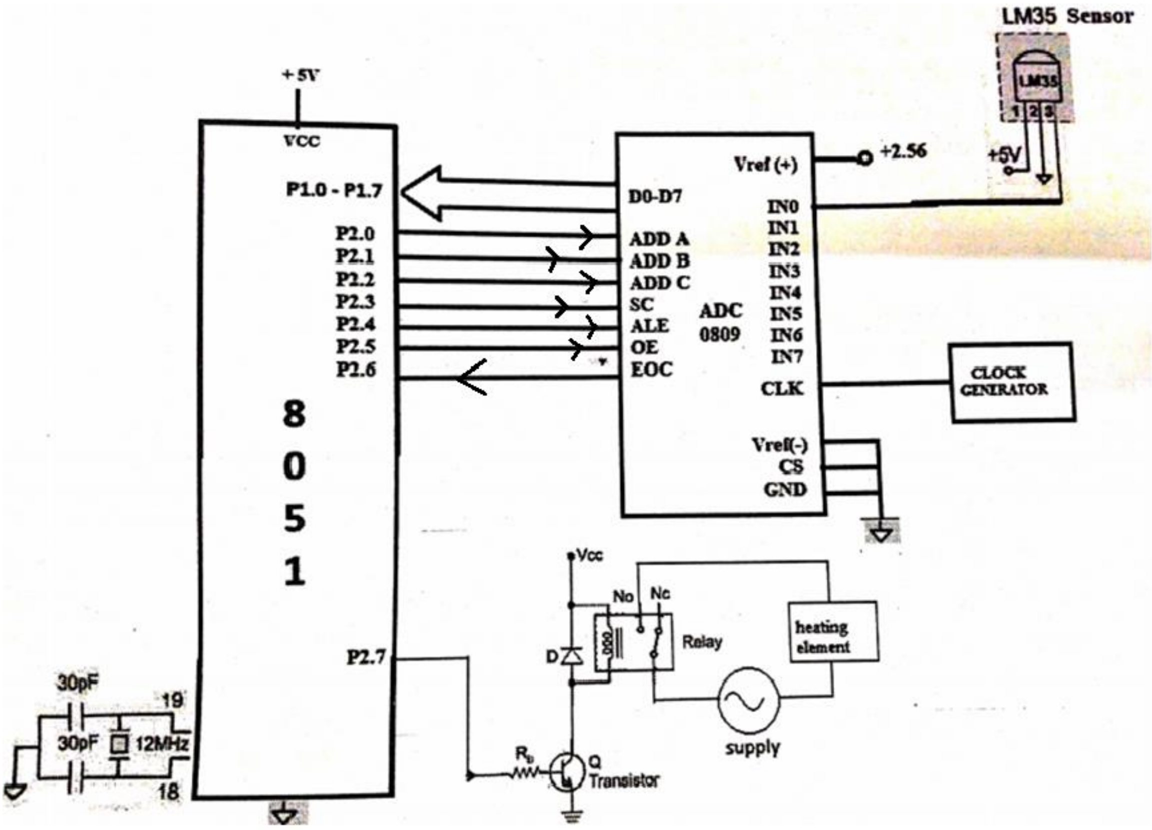
- First configure the controller pins P2.0 and P2.7 as inputs and P2.7 as output.
- Continuously check the water level input pins P2.0 and P2.7.
- If all the pins are high, then make P1.5 pin HIGH to run the motor automatically.
- If the level is low i.e. if P2.0 is 0, continue to run the motor.
- If P2.7 becomes LOW, then the water level in the tank is FULL.
- Now, make the P1.5 pin as LOW to turn off the motor automatically.

PROGRAM

```
ORG 0000H
SETB P2.0                 ;make input
SETB P2.7                 ;make input
CLR P1.5                  ; initially motor OFF
```

```
UP: JNB P2.0,L1          ; CHECK IF LOW LEVEL SENSOR IS OUT OF WATER
SETB P1.5                ; MOTOR ON
SJMP UP
L1: JB P2.7,L2           ; CHECK IF FULL LEVEL SENSOR IS IN WATER
CLR P1.5                 ; MOTOR OFF
L2: SJMP UP
END
```

5.4 Temperature controller using ADC(0808/09).



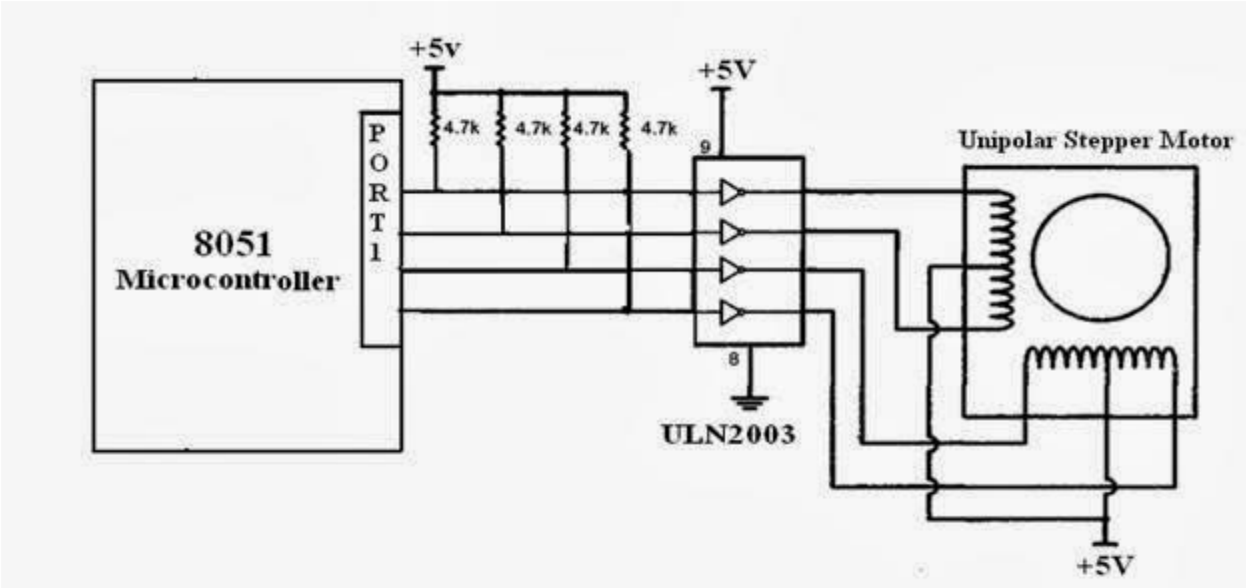
- Here LM35 is used as the temperature sensor. LM35 is an integrated analog temperature sensor whose electrical output is proportional to Degree Centigrade.
- It produces an output of 10mV for each degree Celsius change in temperature.
- This voltage output is connected to an Analog to Digital converter 0809.
- Since the V_{ref} of ADC is 2.56V, this 10mV change at the input, will produce a change of one count at the output. (256 counts $\rightarrow$ 2.56V, therefore 1 count $\rightarrow$ 10mV)
- Suppose we want to switch on the heater if temperature is below 10^{o}Celsius and switch it off , if the temperature goes above 30^{o}C.
- For 10^{o}C, output of LM35 will be 100mV, and the output count of ADC will be 10D
- For 30^{o}C, output of LM35 will be 300mV, and the output count of ADC will be 30D

Program to control temperature between 10^{o}C and 30^{o}C (For ADC interfacing refer Chapter 4)

```
ADC_A BIT  P2.0
ADC_B BIT P2.1
ADC_C  BIT P2.2
ADC_START BIT P2.3
ADC_ALE BIT P2.4
ADC_OE BIT P2.5
ADC_EOC BIT P2.6
HEATER BIT P2.7
ORG 0000H
MOV P1,#OFFH                    ;P1 AS INPUT
SETB ADC_EOC                    ; EOC AS INPUT
UP: CLR ADC_ALE
CLR ADC_START
CLR ADC_A                       ;
CLR ADC_B                       ;SELECT CHANNEL 0
CLR ADC_C                       ;
SETB ADC_ALE                    ;ALE PIN HIGH
SETB ADC_START                  ;START PIN HIGH
CALL DELAY_SMALL
CLR ADC_ALE                     ;ALE PIN LOW
CLR ADC_START                   ;START PIN LOW
AGAIN: JB ADC_EOC, AGAIN        ; EOC PIN SHOULD BE LOW AS SOON
                                START SIGNAL IS GIVEN
AGAIN1: JNB ADC_EOC, AGAIN1 ; WAIT FOR EOC TO BECOME HIGH
SETB ADC_OE
MOV A,P1                        ; READ DIGITAL DATA FROM P1
CLR ADC_OE
CJNE A,#10,L1
L1: JC HEATERON
CJNE A,#30,L3
L3: JNC HEATEROFF
HEATERON: SETB HEATER                 ; ON HEATER IF TEMP < 10
SJMP UP
HEATEROFF: CLR HEATER                 ; OFF THE HEATER IF TEMP >30
SJMP UP
END
```

5.5 Stepper motor control for clock wise, anti-clock wise rotation

The stepper motor discussed here has a total of 6 leads, 4 leads representing the four stator windings, 2 commons for the center tapped leads.
As the sequence of power is applied as per the table given below to each stator winding, the rotor will rotate in clockwise direction. To rotate in anticlockwise, the sequence has to be given in reverse pattern.



P1.3	P1.2	P1.1	P1.0
A	C	B	D
1	0	0	1
1	1	0	0
0	1	1	0
0	0	1	0

Program for continuous rotation (clock wise)

```
ORG 0000H
MOV P1, #09H
ACALL DELAY
MOV P1, #0CH
ACALL DELAY
MOV P1, #06H
ACALL DELAY
MOV P1, #03H
ACALL DELAY
SJMP MAIN
DELAY:
MOV R7,#4
WAIT2:  MOV R6,#0FFH
WAIT1:  MOV R5,#0FFH
WAIT:   DJNZ R5,WAIT
DJNZ R6,WAIT1
DJNZ R7,WAIT2
RET
END
```

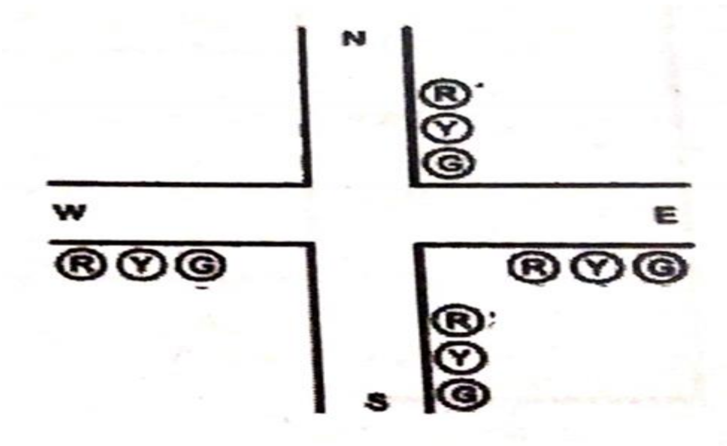
OR

```
ORG 0000H
LJMP MAIN
MAIN:  MOV A, #99H
AGAIN: MOV P1,A
ACALL DELAY
RR A
SJMP AGAIN
DELAY:
MOV R7,#4
WAIT2:  MOV R6,#0FFH
WAIT1:  MOV R5,#0FFH
WAIT:   DJNZ R5,WAIT
DJNZ R6,WAIT1
DJNZ R7,WAIT2
RET
END
```

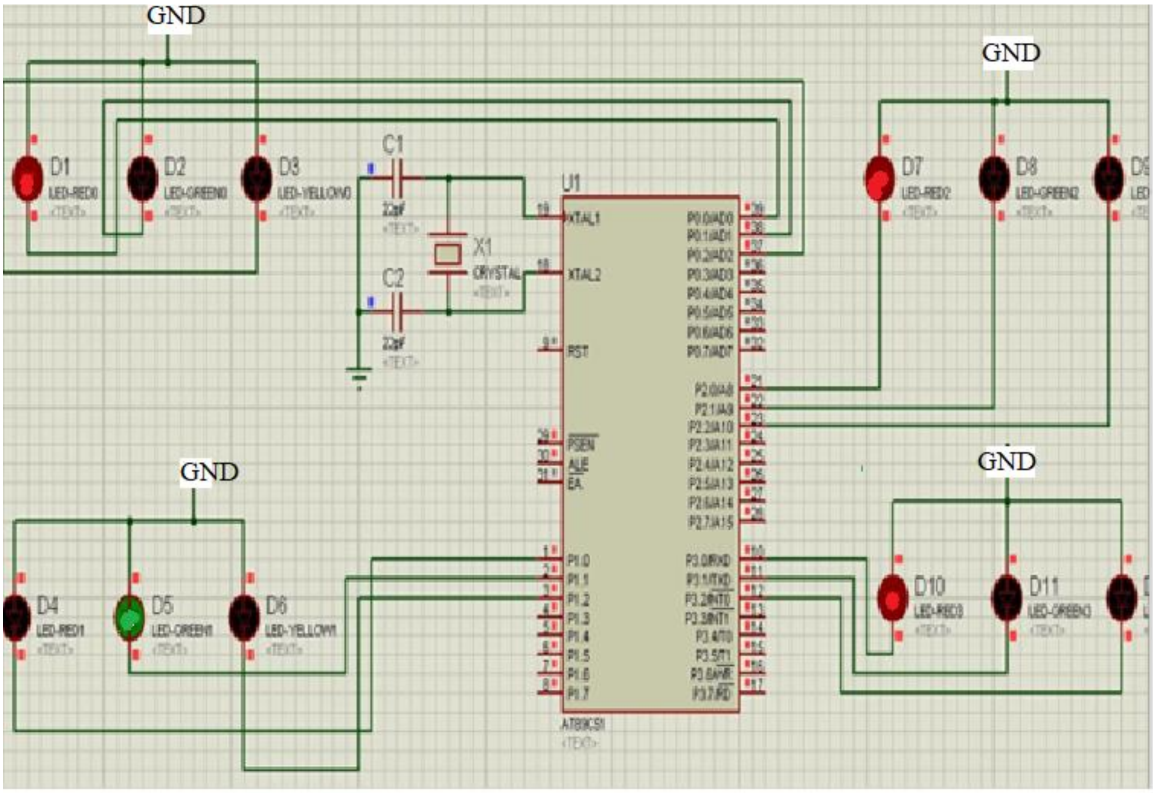
Program to rotate anti clockwise

```
ORG 0000H
LJMP MAIN
MAIN:  MOV A, #99H
AGAIN: MOV P1,A
ACALL DELAY
RL A
SJMP AGAIN
DELAY:
MOV R7,#4
WAIT2:  MOV R6,#0FFH
WAIT1:  MOV R5,#0FFH
WAIT:   DJNZ R5,WAIT
DJNZ R6,WAIT1
DJNZ R7,WAIT2
RET
END
```

5.6 Traffic light controller



Interfacing Diagram



LANE Direction	8051 LINES	TRAFFIC LIGHT
NORTH	P1.0(NR)	RED
	P1.1(NY)	YELLOW
	P1.2(NG)	GREEN
SOUTH	P0.0 (SR)	RED
	P0.1 (SY)	YELLOW
	P0.2 (SG)	GREEN
EAST	P2.0 (ER)	RED
	P2.1(EY)	YELLOW
	P2.2 (EG)	GREEN
WEST	P3.1(WR)	RED
	P3.2(WY)	YELLOW
	P3.3(WG)	GREEN

Process

1. Allow traffic from East to West and West to East for 20 seconds
2. Give a transition period of 5 seconds (Yellow bulb ON)
3. Allow traffic from North to South and South to North for 20 seconds
4. Give a transition period of 5 seconds (Yellow bulb ON)

Program:

```
NR BIT P1.0
NY BIT P1.1
```

```
NG BIT P1.2
SR BIT P0.0
SY BIT P0.1
SG BIT P0.2
ER BIT P2.0
EY BIT P2.1
EG BIT P2.2
WR BIT P3.1
WY BIT P3.2
WG BIT P3.3
MOV P0,#00              ;ALL LEDS OFF
MOV P1,#00                ;ALL LEDS OFF
MOV P2,#00                ;ALL LEDS OFF
MOV P3,#00                ;ALL LEDS OFF
UP: SETB NR               ;NORTH RED ON
SETB SR                   SOUTH RED ON
SETB EG                   ;EAST GREEN ON
SETB WG                   ;WEST GREEN ON
ACALL DELAY               ;20SEC DELAY
CLR EG                    ;EAST GREEN OFF
CLR WG                    ;WEST GREEN OFF
SETB EY                   ;EAST YELLOW ON
SETB WY                   ;WEST YELLOW ON
ACALL DELAY1              ; 5 SEC DELAY
CLR EY                    ; EAST YELLOW OFF
CLR WY                    ; WESY YELLOW OFF
SETB ER                   ;EAST RED ON
SETB WR                   ;WEST RED ON
CLR NR                    ; NORTH RED OFF
CLR SR                    ;SOUTH RED OFF
SETB NG                   ; NORTH GREEN ON
SETB SG                   ; SOUTH GREEN ON
ACALL DELAY               ; WAIT FOR 20 SEC
CLR NG                    ; NORTH GREEN OFF
CLR SG                    ; SOUTH GREEN OFF
SETB NY                   ; NORTH YELLOW ON
SETB SY                   ; SOUTH YELLOW ON
ACALL DELAY1              ; WAIT FOR 5 SEC
CLR NY                    ; NORTH YELLOW OFF
CLR SY                    ; SOUTH YELLOW OFF
CLR ER                    ;EAST RED OFF
CLR WR                    ; WEST RED OFF
SJMP UP                   ; REPEAT THE OPERATION
DELAY: MOV R7,#140        ; 20 SEC DELAY
L3: MOV R6,#255
L2: MOV R5,#255
L1: DJNZ R5,L1
DJNZ R6,L2
DJNZ R7,L3
RET
```

```
DELAY1: MOV R7,#35   ;5 SEC DELAY
L3: MOV R6,#255
L2: MOV R5,#255
L1: DJNZ R5,L1
DJNZ R6,L2
DJNZ R7,L3
RET
END
```