Design Project

MICROPROCESSOR PROGRMMING AND INTERFACING

(CS F241)

Problem Statement : 26

Group Number : 102

Aditya Agarwal	**2013A7PS062P**
Gyanendra Mishra	**2013A7PS126P**
Varun Wachaspati J	**2013A7PS166P**
Sumantra Sharma	**2012B5A8567P**

Birla Institute Of Technology And Science

Pilani

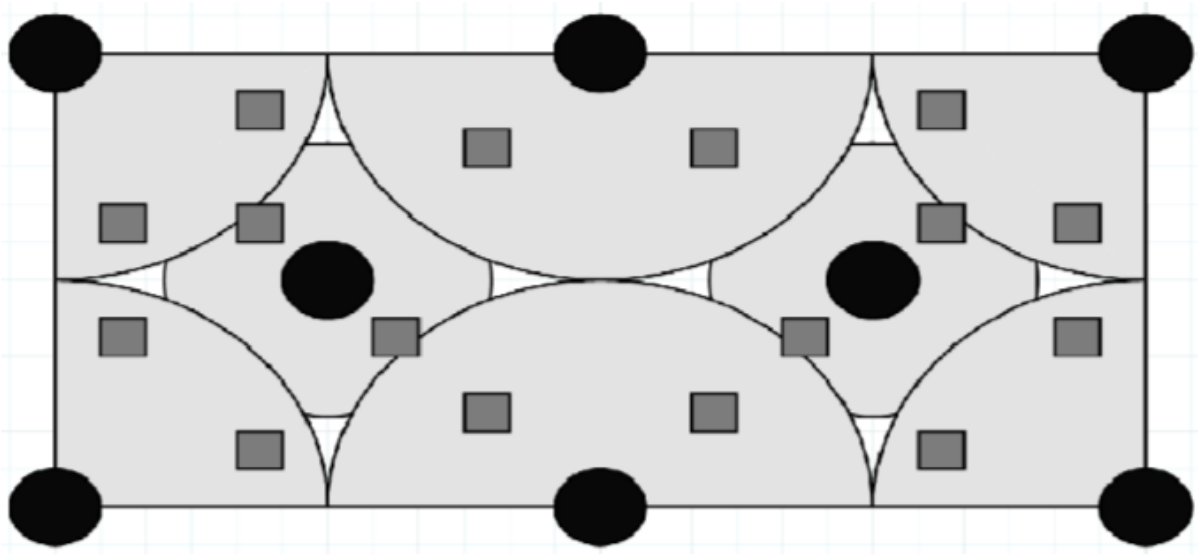
Rajasthan - 333031

Problem Statement:

System to be designed - Lawn Sprinkler System

System Description: An average sized garden has 8 sprinklers that have to be turned on and off. A series of 16 soil moisture sensors are placed at different parts of the garden. The sprinkler system works twice in a day, once at 11: 00 am and then at 6:00 pm. The sprinkler is turned on and off based on the time of the day and the soil moisture. The time for which the sprinkler remains on depends upon the difference between required soil moisture and actual soil moisture level.

Garden Specification:



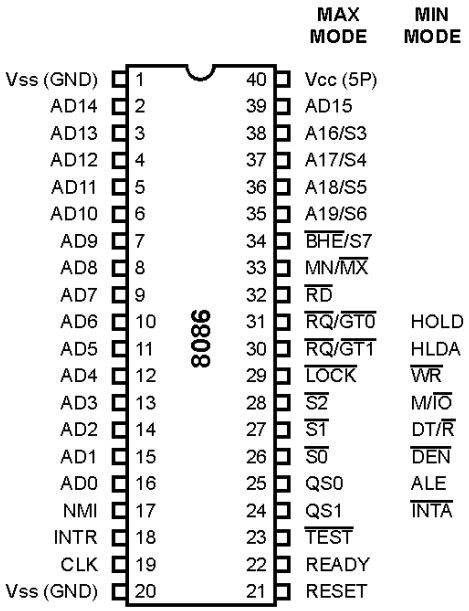
Assumptions

1. DHT22 – A simple temperature and humidity sensor is being used for checking moisture content.
2. The timer starts at 12:00 A.M. (00).
3. 10 seconds = 1 hour
4. A level of 2.5V is considered enough for water to be sprinkled.
5. After a few minutes the water goes off. It is assumed that this quantity is enough for plants to be watered.
6. LED is used instead of sprinkler to show sprinkler. A glowing LED signifies that the sprinkler is working.
7. Clock Pulse of 5 MHz is assumed.
8. It is assumed that 2 sensors work per sprinkler. If either of the sensors show moisture level lesser than required then the LED starts glowing.
9. Humidity and temperature sensors are used to quantify moisture.

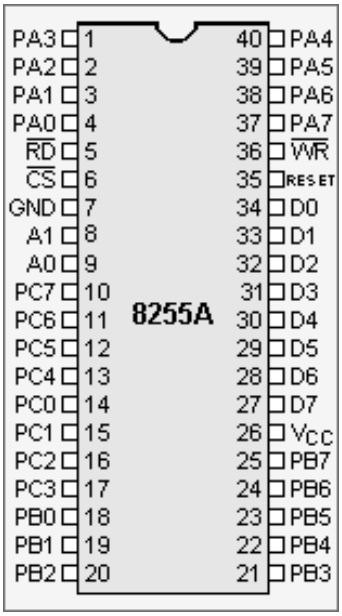
Hardware Systems Used:

Number Used

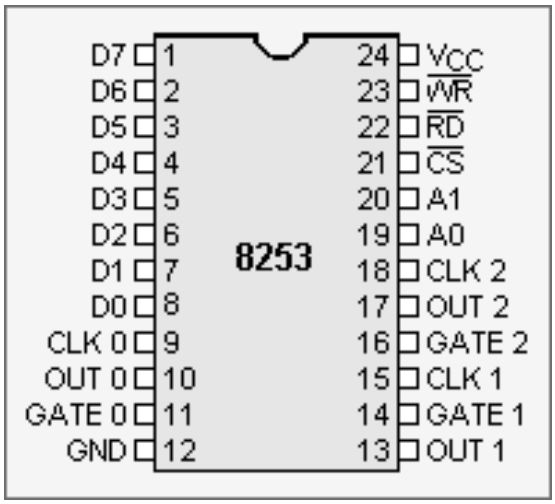
1. **8086** **1**



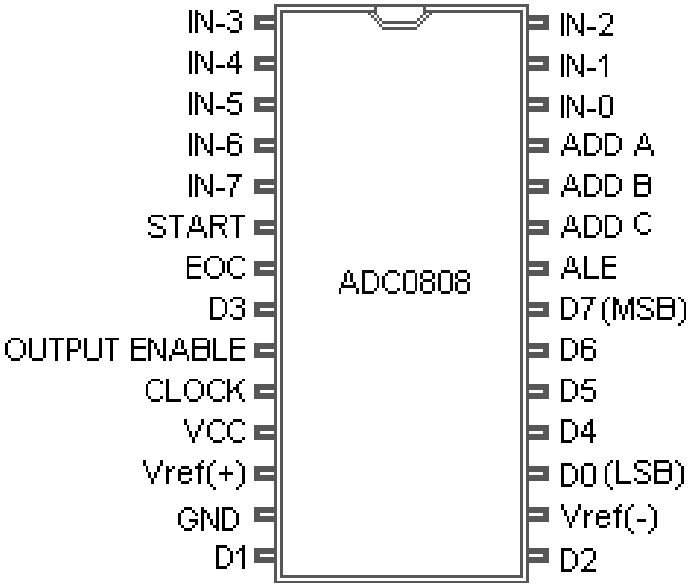
2. **8255** **3**



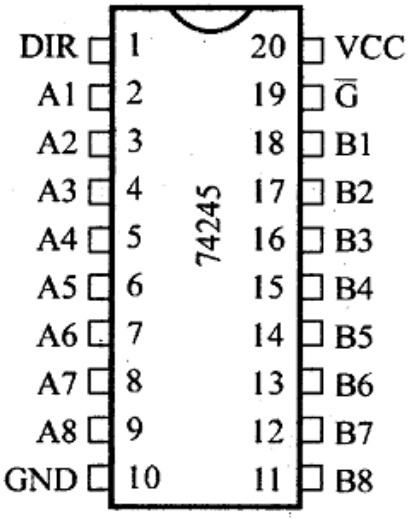
3. 8253 1



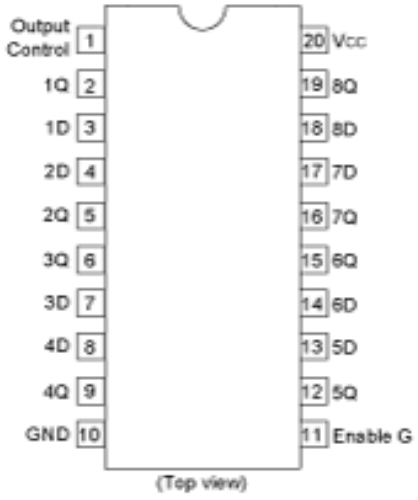
4. ADC0808 – Analog to Digital Converter 2



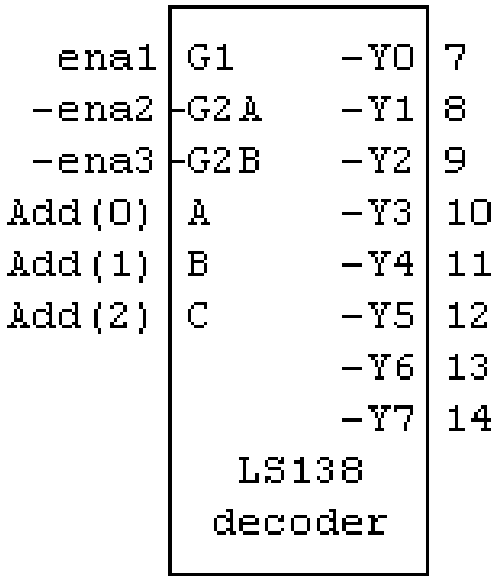
5. SN74LS245 – Octal Bidirectional Buffer 2



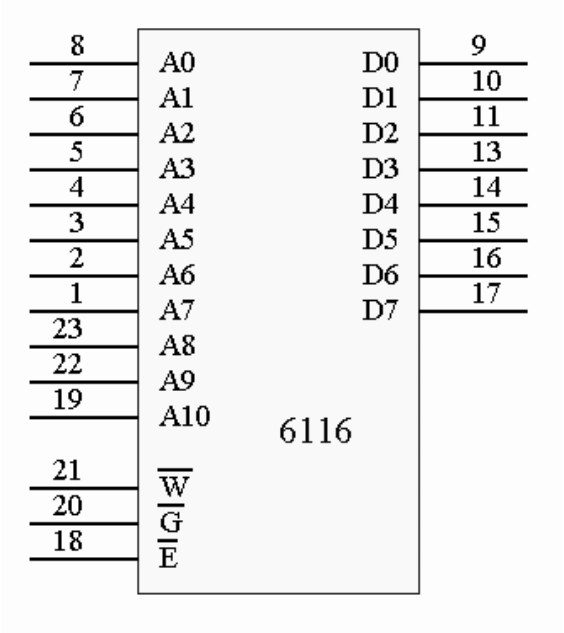
6. SN74LS373 – Octal Latch 3



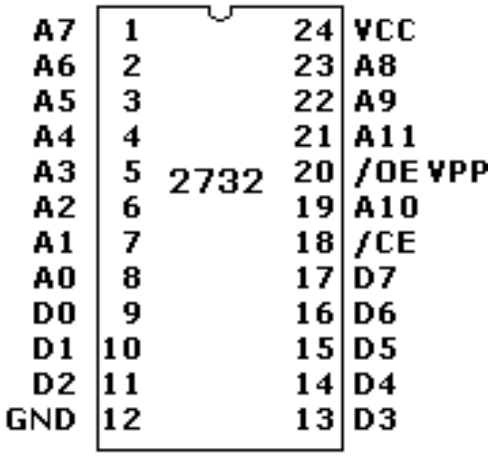
7. SN74LS138 – 3:8 Decoder 1



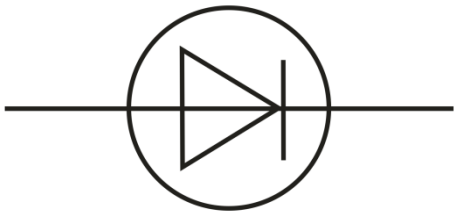
8. 6116 – SRAM 2



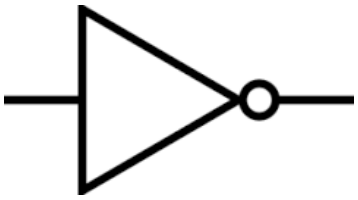
9. <u>2732 – EPROM</u> **2**



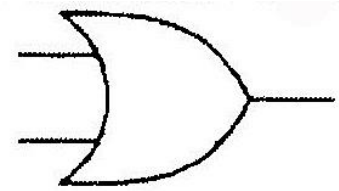
10. <u>LED – Blue</u> **8**



11. <u>NOT Gate</u> **3**



12. <u>OR Gate</u> **6**



13. SW-SPDT-MOM – Switches **1**

Hardware Table

Chip Number	Chip Type	Use
8086	Intel Microprocessor	The main microprocessor that runs the entire system
6116	2x8 Kb SRAM	Static Random Access Memory used to store memory
2732	4x8 Kb EPROM	Erasable Programmable Read-Only Memory can be erased and reused
8255	Programmable Peripheral Interface	3 8255s are used to connect to sprinklers, control the ADCs an update the hr register
8253	Programmable Interval Timers	8253 does the counting function in the design
SN74LS245	Octal Bi-directional Buffer	It is used for generation of the dual character data bus
SN74LS373	Octal Latch	Used to generate the address bus
SN74LS138	3:8 Decoder	Controlling the CS signals of both the 8255s to turn them on by coordinating
ADC0808	Analog to Digital Converter	It works like a multiplexer with 3 select signals in Address A, B and C ports to convert to digital signals

Memory Organization:

The system uses two 2KB of SRAM (2x8Kb) 6116 chips and two 4KB (4x8Kb) of EPROM 2732 chips. They are organized into odd and even bank to facilitate both byte and word size data transfers.

Random Access Memory:

Starting Address: 02000h

Ending Address: 02FFFh

Read Only Memory:

Starting Address: 00000h

Ending Address: 01FFFh

I/O organization:

Three 8255(Programmable Peripheral Interface) are used to communicate with other input and output devices. It is organized in the following manner.

8255(1):

Port	Port Address	Mode	Input/Output	Connected to
A	00H	0	Input	ADC-1
B	02H	0	Input	ADC-2
C lower	04H	0	Output	LEDs
C upper	04H	0	Output	LEDs
Control Register	06H			

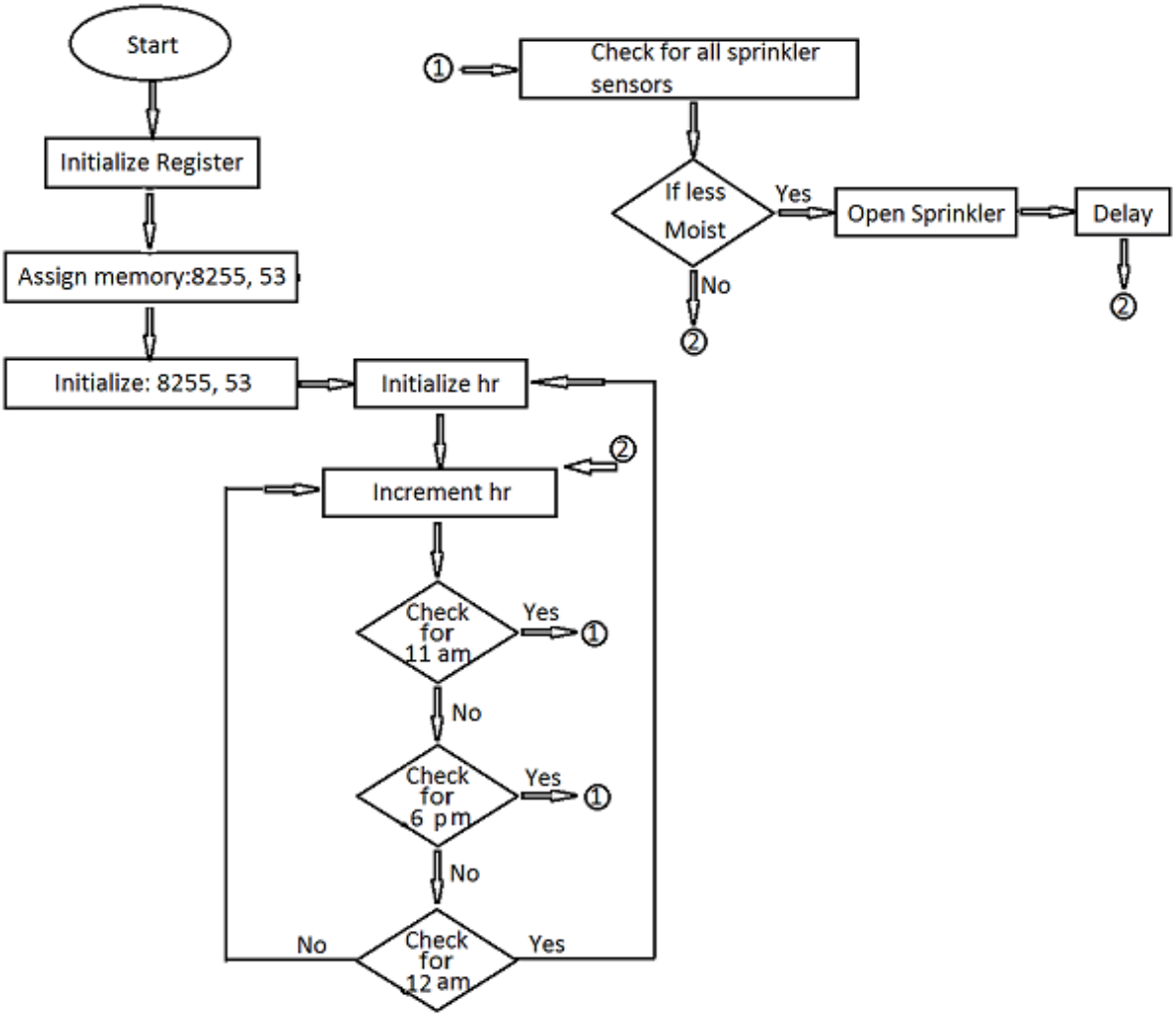
8255(2):

Port	Port Address	Mode	Input/Output	Connected to
A	08h	0	Input	Nothing
B	0ah	0	Input	Nothing
C lower	0ch	0	Output	ADD A, ADD B, ADD C of bothADC
C upper	0ch	0	Input	START, ALE, EOC(1), EOC(2)
Control Register	0eh			

8255(3):

Port	Port Address	Mode	Input/Output	Connected to
A	10h	0	Input	Clock Output 0 of 8253
B	12h	0	Input	
C lower	14h	0	Output	
C upper	14h	0	Output	
Control Register	16h			

Flow Chart


Start
Initialize Register
Assign memory:8255, 53
Initialize: 8255, 53
Initialize hr
Increment hr
Check for 11 am
Yes
No
Check for 6 pm
Yes
No
Check for 12 am
No
Yes
1
Check for all sprinkler sensors
If less Moist
Yes
No
Open Sprinkler
Delay
2

ALGORITHM

In the beginning of the program various variables are set and memory is allocated.
This is followed by initializing the various components included like 8255 and 8253.
The timer is set in rate generator mode and it outputs a low signal after every 10 seconds (assumed to be 1hr).
We check after every 10 seconds (1hrs) how many hours have passed. After the timer reaches 110 seconds (11hrs) or 1800 seconds (18hrs) the remainder of the program is executed. After 2400 seconds (24hrs) the hour counter is reset.
At 11 am and 6pm we check the value returned by different sensors and compare it to a standard moisture value. If the moisture value is lower than standard value then the sprinklers are turned on. The sprinklers are in common anode configuration. After receiving a low signal they light up. After all sprinklers are checked we jump to the `final` block which calls the delay function so that the water sprinklers run for some time.
This is followed by checking whether the sprinklers are on or off. If all are off then we check the hour counter again and go back to the time checking loop. If even one of the sprinklers is still running we check back for values and continue sprinkling till values are normal.

Code

```
#make_BIN#

; set loading address, .bin file will be loaded to this address:
#LOAD_SEGMENT=ffffh#
#LOAD_OFFSET=0000h#

; set entry point:
#CS=0000h# ; same as loading segment
#IP=0000h# ; same as loading offset

; set segment registers
#DS=0000h#; same as loading segment
#ES=0000h# ; same as loading segment

; set stack
#SS=0000h# ; same as loading segment
#SP=FFFEh# ; set to top of loading segment

; set general registers
#AX=0000h#
#BX=0000h#
```

```
#CX=0000h#

#DX=0000h#

#SI=0000h#

#DI=0000h#

#BP=0000h#


; add your code here


        jmp     st1 ;jump to the main function.

        db     1021 dup(0)  ;fill up unadressed memory.

        hr db 00h ; hour count

        spr db 0ffh ; sprinkler values

        std_value db 80h ; standard value for moisture. above this means that we
;need water to be turned on/off.

        run db 2



;main program

        st1:      cli


; intialize ds, es,ss to start of RAM

         mov       ax,0200h

         mov       ds,ax

         mov       es,ax
```

```
        mov     ss,ax
        mov     sp,0FFFEH

; 8255(1) is for storing values from adcs and connecting to sprinklers
; 8255(2) is for controlling the two ADCs
; 8255(3) if for updating the hr register using port A

; assigning memory location to 8255(1)
      creg1 equ 06h
      porta1 equ 00h
      portb1 equ 02h
      portc1 equ 04h

; assigning memory location to 8255(3)
      creg2 equ 16h
      porta2 equ 10h
      portb2 equ 12h
      portc2 equ 14h

; assigning memory location to 8255(2)
      creg3 equ 0eh
      porta3 equ 08h
      portb3 equ 0ah
```

```
        portc3 equu 0ch


; assigning memory location to 8253
        cnt0 equu 18h
        cnt1 equu 1ah
        cnt2 equu 1ch
        cre2 equu 1eh


; initialise timer - 8253


mov al,00010101b
out cre2, al  ;counter0 sent to rate generator mode.
mov al,10
out cnt0, al  ;10 seconds is our equivalent of one hour.



; initialise 8255(1)
mov al,92h
out creg1,al
mov porta1, 00h
mov portb1, 00h
mov al, 00h
out portc1, al
```

```
; initialise 8255(3)
mov al,92h
out creg2,al

; initialise 8255(2)
mov al,9Ah
out creg3,al

;to check for time

time: lea si,hr
      mov [si],0

X1:   in al,porta2
      cmp al,0  ;checks if the timing is 0. our program starts at 12.00 am.
      je X2
      jmp x1

X2:
      inc [si]
      cmp [si],11
```

```
    jmp spr1 ;after 11 hours pass. our sprinkler is switched on.


    cmp [si],18
    je spr1 ; at 18.00 hrs sprinkler is turned on again.
    cmp [si],24
    je time ; at 24.00 hrs sprinkler is reset.
    jmp X1


spr1:


    mov al,30h ; sensors of sprinkler 1 selected.
    call conversion ;conversion function checks whether adc conversion is
;complete or not.
    mov al,0f8h ; sensor of sprinkler 2 selected.
    out 0ch,al  ; signals that end of covnversion is reahed at both adcs.


    compare1: in al, porta1  ; output of adc1 sent to port a of 8255(1)
        cmp al, std_value ;to check whether 8th sensors output returned from
;ADC(1) is lower than standard value. 8255(1) port A.
        jge make_bit_high1  ;it is higher. so switch off/dont change.


        in al, 02h
        cmp al, std_value  ;checks second sensor for the same sprinkler which
;happens to be connected to ADC(2). 8255(1) port B.
```

```
        jge make_bit_high1



    make_bit_low1: and spr, 0feh ;turns on the LED (sprinkler). common anode.
            jmp spr2

    make_bit_high1: or spr,01h ; switch off the sprinkler/led.
            jmp spr2

spr2: mov al,31h
    call conversion
    mov al,0fch
    out 0ch,al

    compare2:in al,00h
        cmp al, std_value
        jge make_bit_high2

        in al, 02h
        cmp al, std_value
        jge make_bit_high2
```

```
    make_bit_low2: and spr,0fdh
                   jmp spr3

    make_bit_high2: or spr,02h
                    jmp spr3

spr3: mov al,32h
    call conversion
    mov al,0fah
    out 0ch,al

    compare3:in al,00h
        cmp al, std_value
        jge make_bit_high3

        in al, 02h
        cmp al, std_value
        jge make_bit_high3

    make_bit_low3: and spr,0fbh
                   jmp spr4

    make_bit_high3: or spr,04h
```

```
                jmp spr4
spr4: mov al,33h
    call conversion
    mov al,0feh
    out 0ch,al

    compare4:in al,00h
        cmp al, std_value
        jge make_bit_high4

        in al, 02h
        cmp al, std_value
        jge make_bit_high4

    make_bit_low4: and spr,0f7h
            jmp spr5

    make_bit_high4: or spr,08h
             jmp spr5

spr5: mov al,34h
    call conversion
    mov al,0f9h
```

```
    out 0ch,al
    compare5:in al,00h
        cmp al, std_value
        jge make_bit_high5

        in al, 02h
        cmp al, std_value
        jge make_bit_high5

    make_bit_low5: and spr,0efh
            jmp spr6

    make_bit_high5: or spr,10h
             jmp spr6

spr6: mov al,35h
    call conversion
    mov al,0fdh
    out 0ch,al

    compare6:in al,00h
        cmp al, std_value
        jge make_bit_high6
```

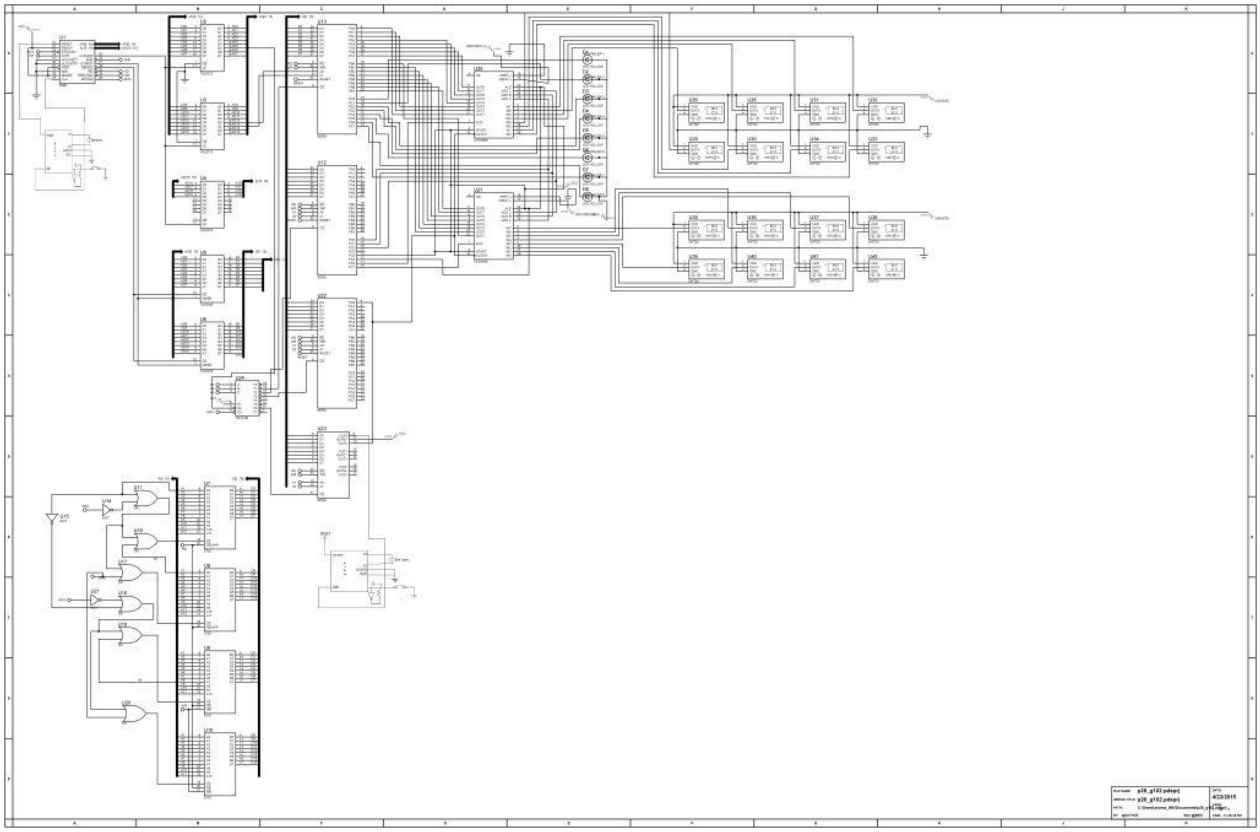
```
        in al, 02h
        cmp al, std_value
        jge make_bit_high6

    make_bit_low6: and spr,0dfh
            jmp spr7

    make_bit_high6: or spr,20h
             jmp spr7

spr7: mov al,36h
    call conversion
    mov al,0fbh
    out 0ch,al

    compare7:in al,00h
        cmp al, std_value
        jge make_bit_high7

        in al, 02h
        cmp al, std_value
        jge make_bit_high7
```

```
    make_bit_low7: and spr,0bfh
                   jmp spr8

    make_bit_high7: or spr,40h
                    jmp spr8

spr8: mov al,37h
    call conversion
    mov al,0ffh
    out 0ch,al

    compare8:in al,00h
          cmp al, std_value
          jge make_bit_high8

          in al, 02h
          cmp al, std_value
          jge make_bit_high8

    make_bit_low8: and spr,7fh
                   jmp final
```

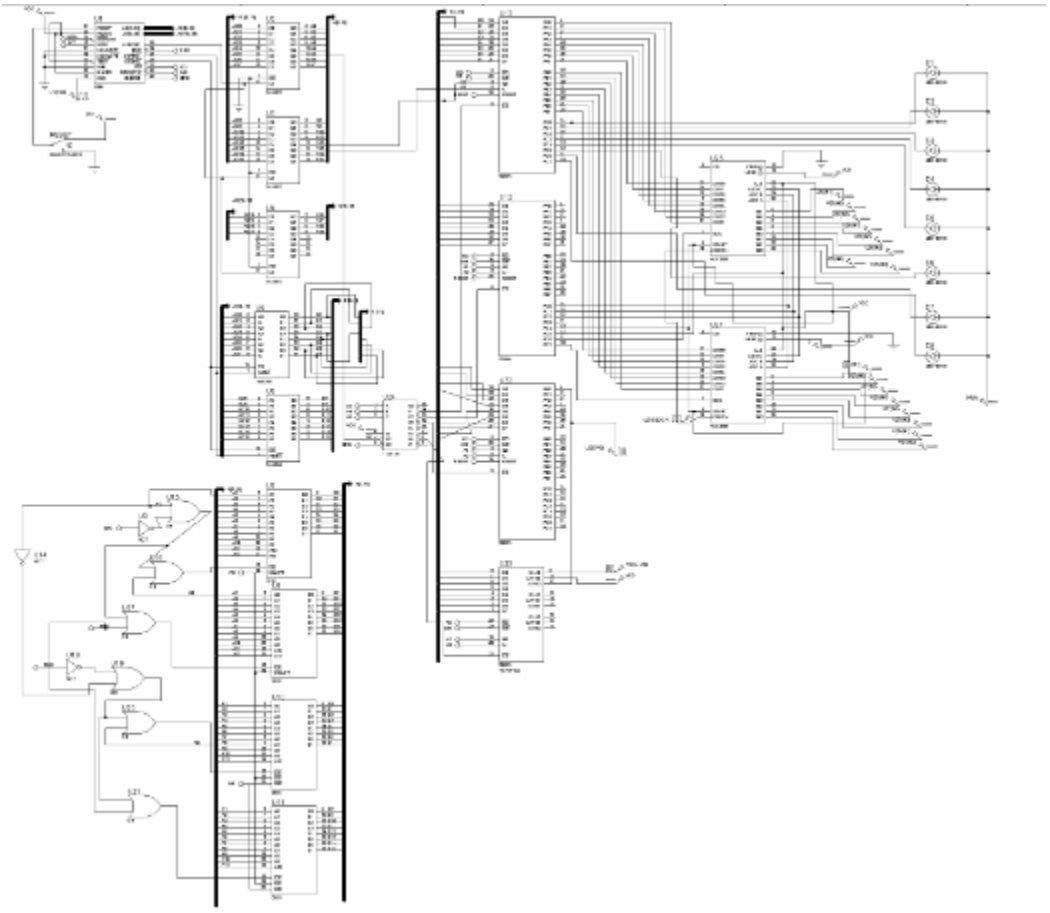
```
    make_bit_high8: or spr,80h
                    jmp final

final: mov al, spr
       out 04h, al

       ; to give a delay. so that plants are watered.
       call delay

       ; check if all sprinklers are off
       cmp spr, 0ffh

       ;if all off, go back to checking hr
       jz x1

       ;if some sprinkler is still on, check sensor values again
       jmp spr1

conversion proc near
          ;this procedure is used to check whether conversion from analoag to
;digital is complete or not.
          out 0ch,al
          back: in al,0ch
```

```
            and al,11000000b
            cmp al,11000000b
            jne back
            pop dx
            pop cx
            popf

        RET
conversion endp

delay proc near ; a simple delay loop.
    mov cx,7d00h
x0:   nop
    loop x0
delay endp

HLT         ; halt!

                                    ; END OF CODE
```

Circuit Design



With Sensors



With VCC.