Embedded System Project

Name – Ankit

Roll No – 2001029

8*8 IMAGE FILTER

THEORY :

I will do this using a linear smoothning filter or averaging filter .

Average filter is a linear image processing technique used to remove noise from an image. The following steps outline the basic theory of the average filter:

1.Define the filter window: The average filter window is a square area of pixels surrounding the current pixel. The size of the window is specified by the user.

2.Sum the pixels: Within the filter window, sum the intensity values of all pixels .

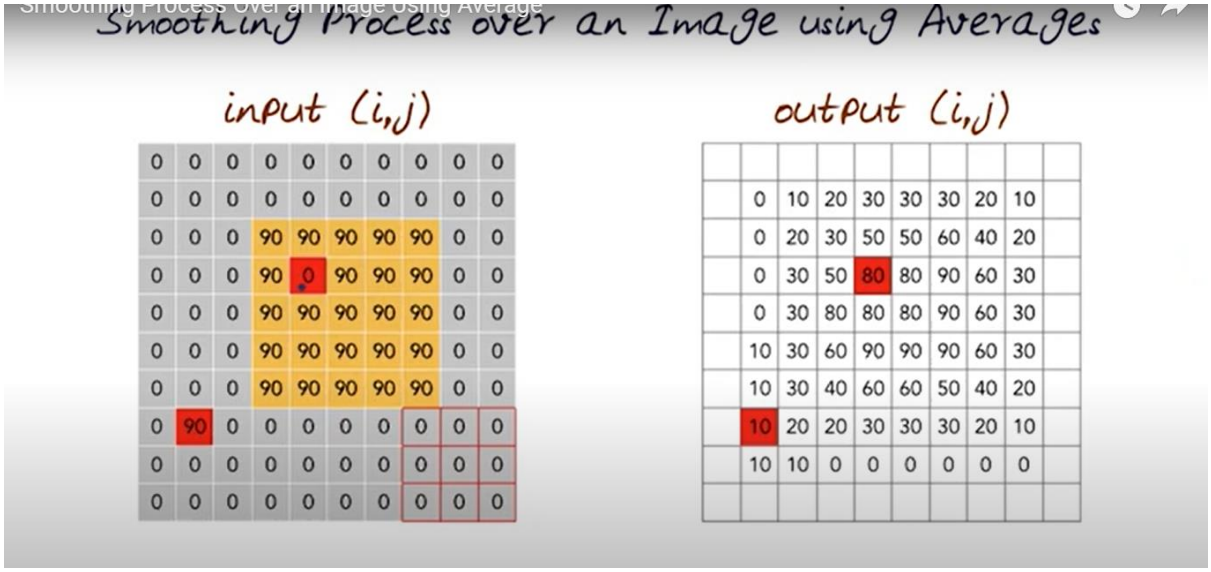
3.Select the average: The average value is the sum of values of the pixels divided by 9.

4.Replace the current pixel with the average: The intensity value of the current pixel is replaced with the average value.

5.Repeat for all pixels: The average filter operation is performed for every pixel in the image.

The average filter is effective in removing salt and pepper noise, which are random white and black pixels that appear in an image due to measurement errors. The median filter replaces the current pixel with the median value of the surrounding pixels, effectively smoothing out the noise.

Here is an example of how we do it-



PERIPHERALS :

To design a median filter using 8051 microcontroller, the following peripherals are required:

1.8051 Microcontroller

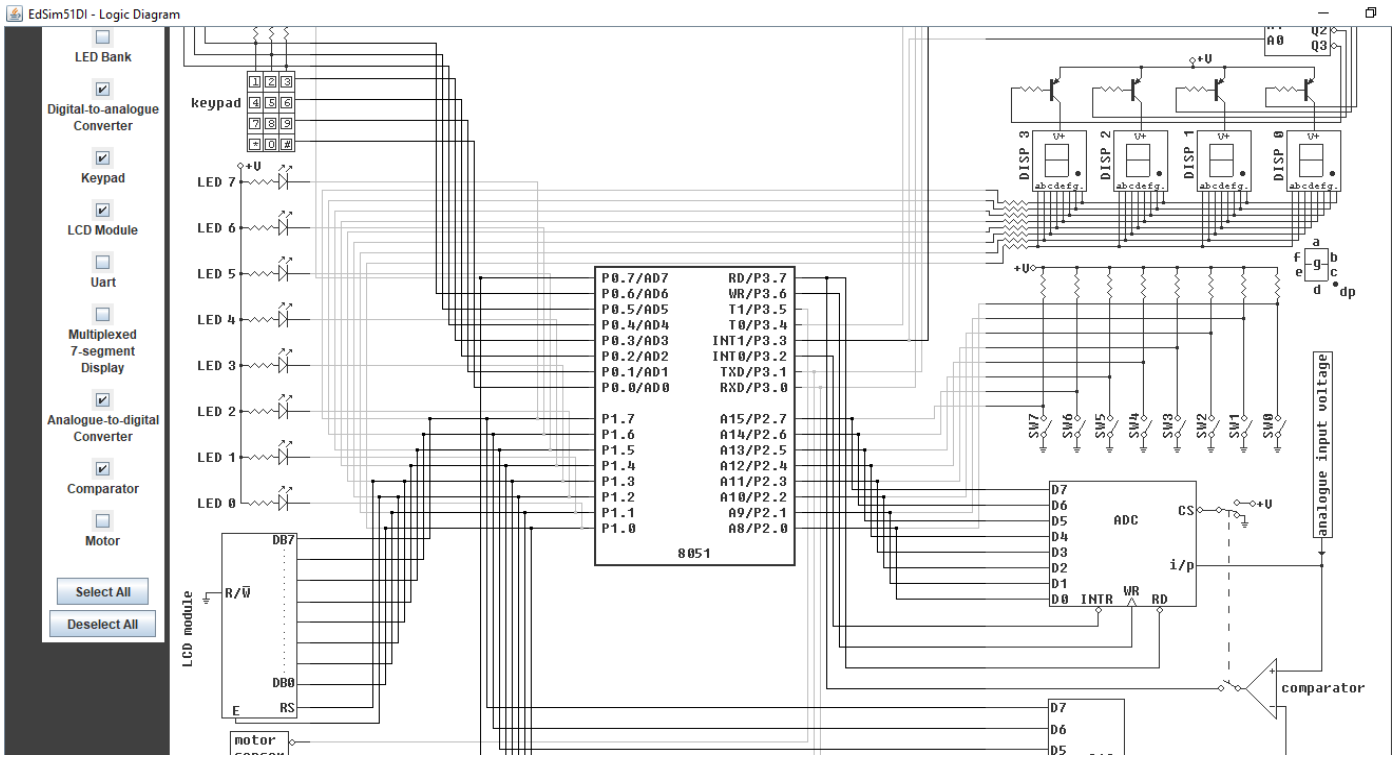
2.ADC (Analog to Digital Converter)

3.External Memory (for storing data)

4.LCD Display (for displaying filtered data)

5.Keypad (for taking inputs)

-In addition to these peripherals, the implementation of zero padding will also require additional software algorithms and programming codes to be written to properly pad the input data with zero values.



ALGORITHM:

Here's a basic algorithm for implementing a median filter in 8051 microcontroller:

1. Store the for the 8*8 image data taken serially in store it in memory address.
2. Similarly we have to define memory adress for .
3. Create two nested loops to iterate through the rows and columns of the image.
4. Within the loops, create another set of nested loops to iterate through the 3x3 neighborhood of each pixel.
5. Calculate the average value of the 3x3 neighborhood by summing up all the pixel values and dividing by 9.
6. Assign the average value to the corresponding pixel in the smoothed image array.
7. Continue iterating through all pixels in the image.
8. Output the smoothed image array.

CODE-

Sample input - 1213472986259611128946128669785432 19

```
    CLR SM0
```

```
    SETB SM1  // Serial port mode selection
    SETB REN    //Enable serial reception



    MOV A, PCON   //Power control register
    SETB ACC.7    //Bit will be se if data coming out of bit 7 of acc.
    MOV PCON, A

    MOV TMOD, #20H //Timer mode
    MOV TH1, #253   //Setting the baud rate
    MOV TL1, #253    //To start timer
    SETB TR1    //Timer 1 start or stop bit
mainloop1:
    SETB REN //Enabling serial reception
    MOV R1,#20H//Starting from 20H address
again:
//Storing input in memory adress
    JNB RI, $
    CLR RI
    MOV A, SBUF
    CJNE A, #0DH, skip
    MOV @R1,#00H
    call start
```

```
    JMP mainloop1
skip:
        SUBB A,#30H
    MOV @R1,A
    INC R1
    JMP again
finish:
    JMP $


start:
        MOV R1,#27H //Starting with first non edge pixal
        MOV R4,#05H //iterate it for 4 counts or 5 pixals
        MOV R0,#57H // First storing adsress for   ouput at is 57H location which is first       non edge case.
loop1:
        DJNZ R4, mainloop // Iterate row counter till it do 5 iteration


        MOV R4,#05H
        MOV A,R1
        ADD A,#02H // Jump it by 2 places to skip the two edge pixals
        MOV R1,A


        MOV A,R0 // Putting ouput at locations stating from 57
        ADD A,#02H
```

```
        MOV R0,A
        JMP loop1
mainloop:
//If it reaches 3F location then stop the loop
        CJNE R1,#3FH,loop2
        JMP endloop
loop2:
//Adding all the neighbouring pixals and then summing it
        MOV R2,1
        MOV A,#00H

        ADD A,@R1
        MOV R3,A
        MOV A,R2
        ADD A,#06H
        MOV R1,A
        MOV A,R3

        ADD A,@R1
        MOV R3,A
        MOV A,R2
        SUBB A,#06H
        MOV R1,A
        MOV A,R3
```

```
ADD A,@R1
MOV R3,A
MOV A,R2
ADD A,#05H
MOV R1,A
MOV A,R3

ADD A,@R1
MOV R3,A
MOV A,R2
SUBB A,#05H
MOV R1,A
MOV A,R3

ADD A,@R1
MOV R3,A
MOV A,R2
ADD A,#07H
MOV R1,A
MOV A,R3

ADD A,@R1
MOV R3,A
```

```
MOV A,R2
SUBB A,#07H
MOV R1,A
MOV A,R3

ADD A,@R1
MOV R3,A
MOV A,R2
ADD A,#01H
MOV R1,A
MOV A,R3

ADD A,@R1
MOV R3,A
MOV A,R2
SUBB A,#01H
MOV R1,A
MOV A,R3
 //Dividing it by 9 to find average
ADD A,@R1
MOV R1,2
MOV B,#09H
DIV AB
```

```
        MOV @R0,A //Moving output to a location starting from  57H (stored by R0)
        INC R1
        INC R0 // We are just increamenting next place to store
        JMP loop1
endloop:
// Copying all edge cases to ouput  serial port , output serial port is starting from 50H
        MOV 50H,20H
        MOV 51H,21H
        MOV 52H,22H
        MOV 53H,23H
        MOV 54H,24H
        MOV 55H,25H
        MOV 56H,26H
        MOV 5BH,2BH
        MOV 5CH,2CH
        MOV 61H,31H
        MOV 62H,32H
        MOV 67H,37H
        MOV 68H,38H
        MOV 6DH,3DH
        MOV 6EH,3EH
        MOV 6FH,3FH
```

```
        MOV 70H,40H
        MOV 71H,41H
        MOV 72H,42H
        MOV 73H,43H
        MOV 74H,#00H
        CALL writedata
        JMP finish

// Serially giving output data to output port
writedata:
        MOV R0,#50H
        MOV R4,#07H
writeloop:
        CJNE @R0,#00H,writesub
        JMP writeend
writesub:
        DJNZ R4, writesub1
        MOV A,#0AH
        MOV R4,#06H
        CALL sendCharacter
writesub1:
        MOV A,@R0
        ADD A,#30H
        CALL sendCharacter
```

```
        MOV A,#' '
        CALL sendCharacter
        INC R0
        JMP writeloop
writeend:
        RET

sendCharacter:
        MOV SBUF, A
        JNB TI, $
        CLR TI
        RET
```