HUMIDITY SENSOR USING LCD DISPLAY

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

MICROCONTROLLERS AND ITS APPLICATIONS

ECE-3003

A PROJECT REPORT

Submitted by

BALACHANDIRAN KAILASH – 16BEC0709

SAPARI A V – 16BEC0599

SUBASH J – 16BEC0739

VARUN P – 16BEC0153

B.Tech

Electronics and Communication Engineering

CERTIFICATE

Certified that the project report **"HUMIDITY SENSOR USING LCD DISPLAY"** is the bonafide work of **"BALACHANDIRAN KAILASH 16BEC0709, SAPARI AV 16BEC0599, SUBASH J 16BEC0739, VARUN P 16BEC0153"** who carried out the project work under my supervision.

<<Signature of the Supervisor>>

SIGNATURE

ACKNOWLEDGEMENT

We cordially thank our Prof Dhanabal R for his precious guidance, the Dean of SENSE for their pleaded permission and opportunity given to us for completion of our project.

Thanking You,

BALACHANDIRAN KAILASH 16BEC0709

SAPARI A V 16BEC0599

SUBASH J 16BEC0739

VARUN P 16BEC0153

MEMBERS DETAILS

Reg No.	Members	CAT-1	CAT-2	CGPA	Lab Attendance	Theory Slot, Lab Slot	Mob No.
16BEC0709	BALACHANDIRAN KAILASH	10	31	6.75	84	52, 25	9488514389
16BEC0599	SAPARI A V	05	16	6.95	84	48, 22	9698655777
16BEC0739	SUBASH J	05	25	6.59	84	60, 26	9788653582
16BEC0153	VARUN P	25	25	6.25	84	11, 10	7032727686

Email Id :

1. bchandiran.kailash2016@vitstudent.ac.in
2. j.subash2016@vitstudent.ac.in
3. p.varun2016@vitstudent.ac.in
4. saparia.v2016@vitstudent.ac.in

TABLE OF CONTENT

Serial no.	Contents	Page no.
1	Aim	6
2	Components Required	7
3	Circuit diagram	8
4	Flowchart	9
5	Pin Diagram/ Block Diagram	10
6	Instruction Set	11
7	About Components	14
8	Description	17
9	Working of the project	17
10	Code in assembly level language	18
11	C Code	21
12	Addressing modes, memory and time required	25
13	Simulation and output	29
14	Snapshots	30
15	Output	32
16	Hardware	33
17	Applications	34
18	Advantages	34
19	Result	33
20	I3, I5, I7	35
21	References	39

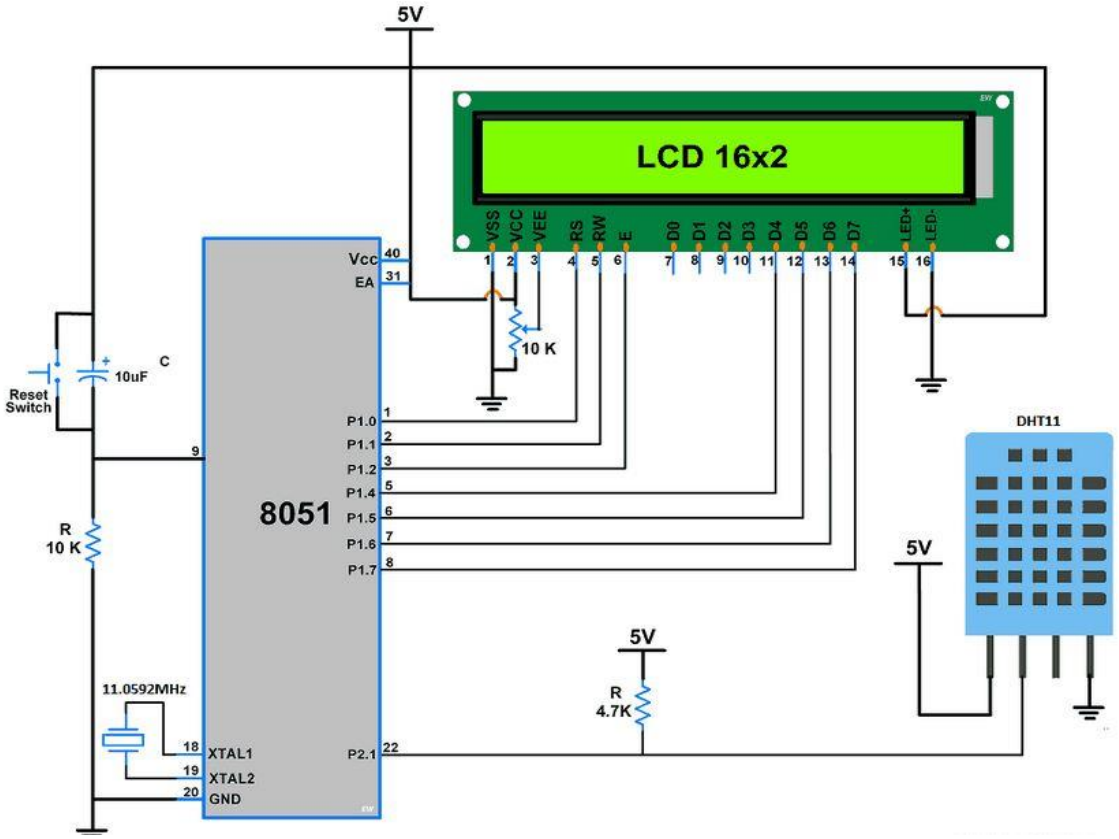
AIM of the project:

The aim of this project is to build a simple humidity sensor based on 8051 microcontroller. Humidity sensor is also called hygrometer. This circuit can sense relative humidity (RH) from 20% to 95% at an accuracy of 5%. And to display the humidity information on a 16×2 LCD display. A relay is also provided which is set to be active when the humidity crosses a certain trip point. The circuit is mains operated and it is very easy to install. DHT11 is the humidity sensor used here.

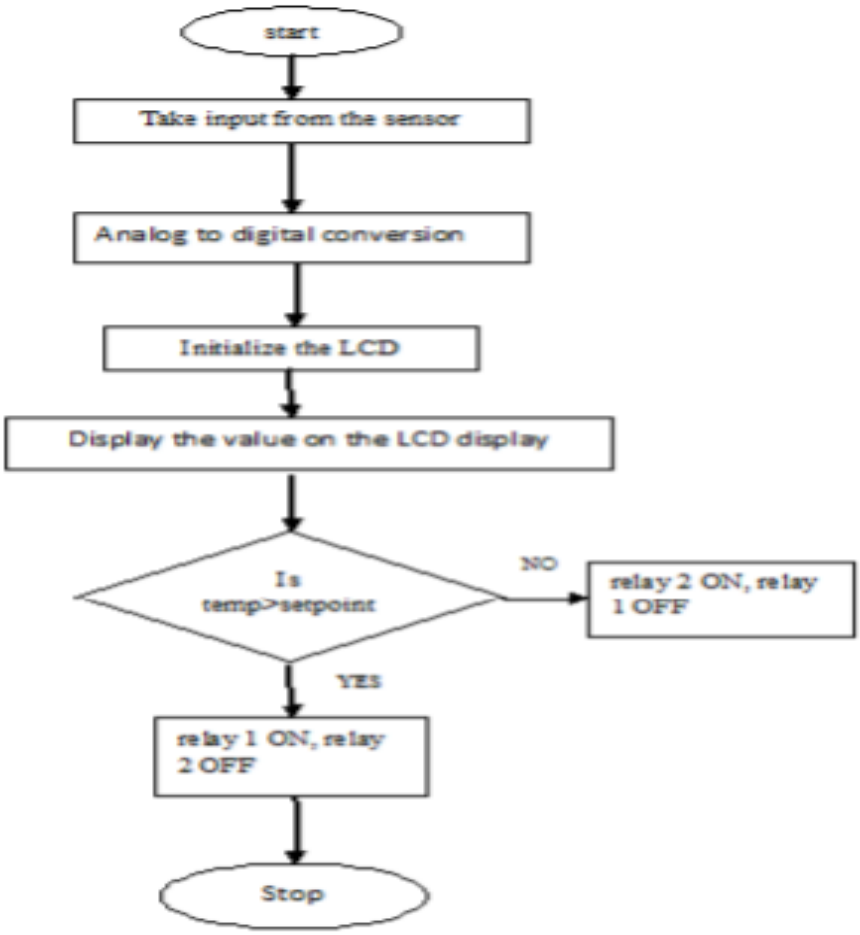
Components required:

- AT89C51 Microcontroller (or any 8051 based Microcontroller)
- 8051 Programmer (Programming Board)
- 11.0592 MHz Quartz Crystal
- 2 x 33pF Capacitor
- 2 x 10KΩ Resistor (1/4 Watt)
- 10µF Capacitor
- Push Button
- Potentiometer
- Dht11 (Humidity and temperature sensor)
- 4.7k and 10 k resistors
- 16*2 lcd display
- 10 uF capacitor
- Power supply
- Connecting wires
- Keil µVision IDE
- Programming cable
- FLIP Software (for burning code)

CIRCUIT DIAGRAM:



Flowchart :


start
Take input from the sensor
Analog to digital conversion
Initialize the LCD
Display the value on the LCD display
Is temp>setpoint
NO
relay 2 ON, relay 1 OFF
YES
relay 1 ON, relay 2 OFF
Stop

PIN DIAGRAM

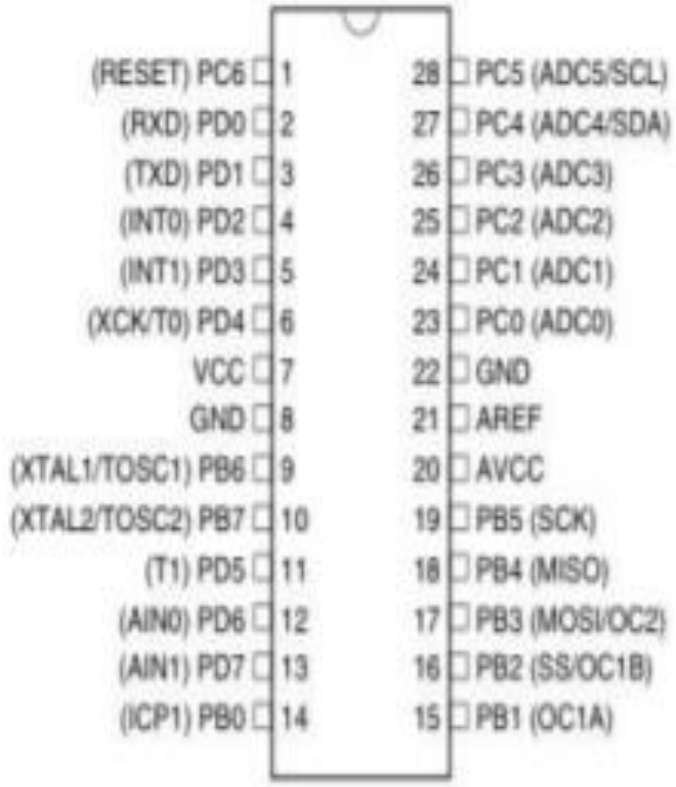
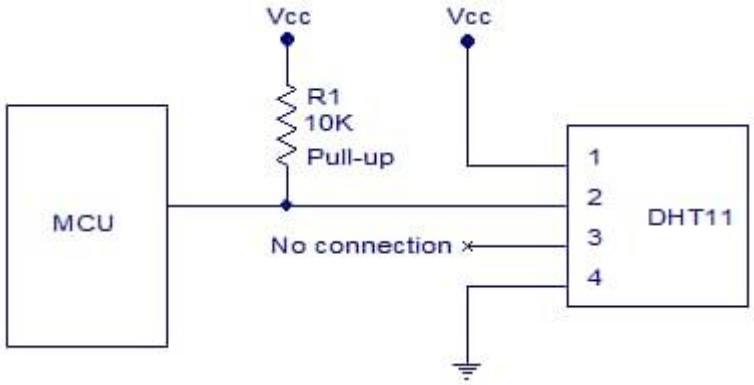


Figure 5. Pin diagram of ATMega328

BLOCK DIAGRAM



8051 INSTRUCTION SET

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
MOV	A, #Data	A ← Data	Immediate	2	1
	A, Rn	A ← Rn	Register	1	1
	A, Direct	A ← (Direct)	Direct	2	1
	A, @Ri	A ← @Ri	Indirect	1	1
	Rn, #Data	Rn ← data	Immediate	2	1
	Rn, A	Rn ← A	Register	1	1
	Rn, Direct	Rn ← (Direct)	Direct	2	2
	Direct, A	(Direct) ← A	Direct	2	1
	Direct, Rn	(Direct) ← Rn	Direct	2	2
	Direct1, Direct2	(Direct1) ← (Direct2)	Direct	3	2
	Direct, @Ri	(Direct) ← @Ri	Indirect	2	2
	Direct, #Data	(Direct) ← #Data	Direct	3	2
	@Ri, A	@Ri ← A	Indirect	1	1
	@Ri, Direct	@Ri ← Direct	Indirect	2	2
	@Ri, #Data	@Ri ← #Data	Indirect	2	1
	DPTR, #Data16	DPTR ← #Data16	Immediate	3	2
MOVC	A, @A+DPTR	A ← Code Pointed by A+DPTR	Indexed	1	2
	A, @A+PC	A ← Code Pointed by A+PC	Indexed	1	2
	A, @Ri	A ← Code Pointed by Ri (8-bit Address)	Indirect	1	2
MOVX	A, @DPTR	A ← External Data Pointed by DPTR	Indirect	1	2
	@Ri, A	@Ri ← A (External Data 8-bit Addr)	Indirect	1	2
	@DPTR, A	@DPTR ← A (External Data 16-bit Addr)	Indirect	1	2
PUSH	Direct	Stack Pointer SP ← (Direct)	Direct	2	2
POP	Direct	(Direct) ← Stack Pointer SP	Direct	2	2
XCH	Rn	Exchange ACC with Rn	Register	1	1
	Direct	Exchange ACC with Direct Byte	Direct	2	1
	@Ri	Exchange ACC with Indirect RAM	Indirect	1	1
XCHD	A, @Ri	Exchange ACC with Lower Order Indirect RAM	Indirect	1	1

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ADD	A, #Data	A ← A + Data	Immediate	2	1
	A, Rn	A ← A + Rn	Register	1	1
	A, Direct	A ← A + (Direct)	Direct	2	1
	A, @Ri	A ← A + @Ri	Indirect	1	1
ADDC	A, #Data	A ← A + Data + C	Immediate	2	1
	A, Rn	A ← A + Rn + C	Register	1	1
	A, Direct	A ← A + (Direct) + C	Direct	2	1
	A, @Ri	A ← A + @Ri + C	Indirect	1	1
SUBB	A, #Data	A ← A – Data – C	Immediate	2	1
	A, Rn	A ← A – Rn – C	Register	1	1
	A, Direct	A ← A – (Direct) – C	Direct	2	1
	A, @Ri	A ← A – @Ri – C	Indirect	1	1
MUL	AB	Multiply A with B (A ← Lower Byte of A*B and B ← Higher Byte of A*B)	--	1	4
DIV	AB	Divide A by B (A ← Quotient and B ← Remainder)	--	1	4
DEC	A	A ← A – 1	Register	1	1
	Rn	Rn ← Rn – 1	Register	1	1
	Direct	(Direct) ← (Direct) – 1	Direct	2	1
	@Ri	@Ri ← @Ri – 1	Indirect	1	1
INC	A	A ← A + 1	Register	1	1
	Rn	Rn ← Rn + 1	Register	1	1
	Direct	(Direct) ← (Direct) + 1	Direct	2	1
	@Ri	@Ri ← @Ri + 1	Indirect	1	1
	DPTR	DPTR ← DPTR + 1	Register	1	2
DA	A	Decimal Adjust Accumulator	--	1	1

Mnemonic	Instruction	Description	Addressing Mode	# of Bytes	# of Cycles
ANL	A, #Data	A ← A AND Data	Immediate	2	1
	A, Rn	A ← A AND Rn	Register	1	1
	A, Direct	A ← A AND (Direct)	Direct	2	1
	A, @Ri	A ← A AND @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) AND A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) AND #Data	Direct	3	2
ORL	A, #Data	A ← A OR Data	Immediate	2	1
	A, Rn	A ← A OR Rn	Register	1	1
	A, Direct	A ← A OR (Direct)	Direct	2	1
	A, @Ri	A ← A OR @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) OR A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) OR #Data	Direct	3	2
XRL	A, #Data	A ← A XRL Data	Immediate	2	1
	A, Rn	A ← A XRL Rn	Register	1	1
	A, Direct	A ← A XRL (Direct)	Direct	2	1
	A, @Ri	A ← A XRL @Ri	Indirect	1	1
	Direct, A	(Direct) ← (Direct) XRL A	Direct	2	1
	Direct, #Data	(Direct) ← (Direct) XRL #Data	Direct	3	2
CLR	A	A← 00H	--	1	1
CPL	A	A ← A	--	1	1
RL	A	Rotate ACC Left	--	1	1
RLC	A	Rotate ACC Left through Carry	--	1	1
RR	A	Rotate ACC Right	--	1	1
RRC	A	Rotate ACC Right through Carry	--	1	1
SWAP	A	Swap Nibbles within ACC	--	1	1

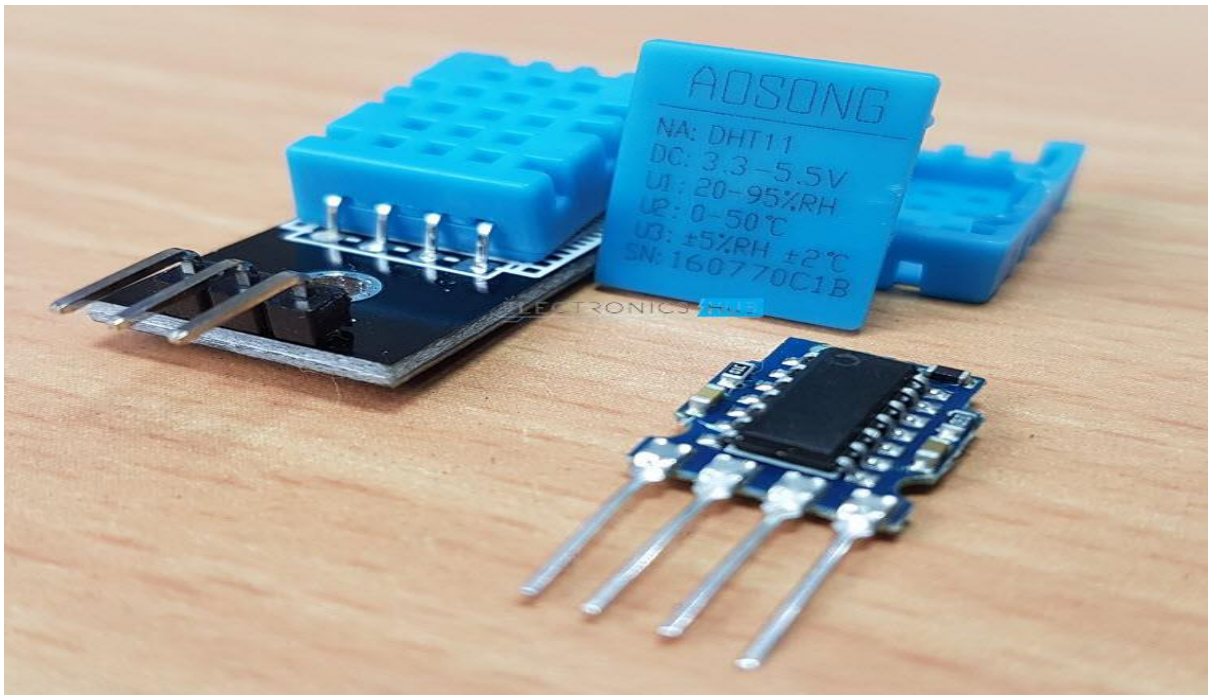
Mnemonic	Instruction	Description	# of Bytes	# of Cycles
CLR	C	C ← 0 (C = Carry Bit)	1	1
	Bit	Bit ← 0 (Bit = Direct Bit)	2	1
SET	C	C ← 1	1	1
	Bit	Bit ← 1	2	1
CPL	C	C ← $\overline{C}$	1	1
	Bit	Bit ← $\overline{Bit}$	2	1
ANL	C, /Bit	C ← C. $\overline{Bit}$ (AND)	2	1
	C, Bit	C ← C . Bit (AND)	2	1
ORL	C, /Bit	C ← C + $\overline{Bit}$ (OR)	2	1
	C, Bit	C ← C + Bit (OR)	2	1
MOV	C, Bit	C ← Bit	2	1
	Bit, C	Bit ← C	2	2
JC	rel	Jump is Carry (C) is Set	2	2
JNC	rel	Jump is Carry (C) is Not Set	2	2
JB	Bit, rel	Jump is Direct Bit is Set	3	2
JNB	Bit, rel	Jump is Direct Bit is Not Set	3	2
JBC	Bit, rel	Jump is Direct Bit is Set and Clear Bit	3	2

Mnemonic	Instruction	Description	# of Bytes	# of Cycles
ACALL	ADDR11	Absolute Subroutine Call PC + 2 → (SP); ADDR11 → PC	2	2
LCALL	ADDR16	Long Subroutine Call PC + 3 → (SP); ADDR16 → PC	3	2
RET	--	Return from Subroutine (SP) → PC	1	2
RETI	--	Return from Interrupt	1	2
AJMP	ADDR11	Absolute Jump ADDR11 → PC	2	2
LJMP	ADDR16	Long Jump ADDR16 → PC	3	2
SJMP	rel	Short Jump PC + 2 + rel → PC	2	2
JMP	@A + DPTR	A + DPTR → PC	1	2
JZ	rel	If A=0, Jump to PC + rel	2	2
JNZ	rel	If A ≠ 0, Jump to PC + rel		
CJNE	A, Direct, rel	Compare (Direct) with A. Jump to PC + rel if not equal	3	2
	A, #Data, rel	Compare #Data with A. Jump to PC + rel if not equal	3	2
	Rn, #Data, rel	Compare #Data with Rn. Jump to PC + rel if not equal	3	2
	@Ri, #Data, rel	Compare #Data with @Ri. Jump to PC + rel if not equal	3	2
DJNZ	Rn, rel	Decrement Rn. Jump to PC + rel if not zero	2	2
	Direct, rel	Decrement (Direct). Jump to PC + rel if not zero	3	2
NOP		No Operation	1	1

About components:

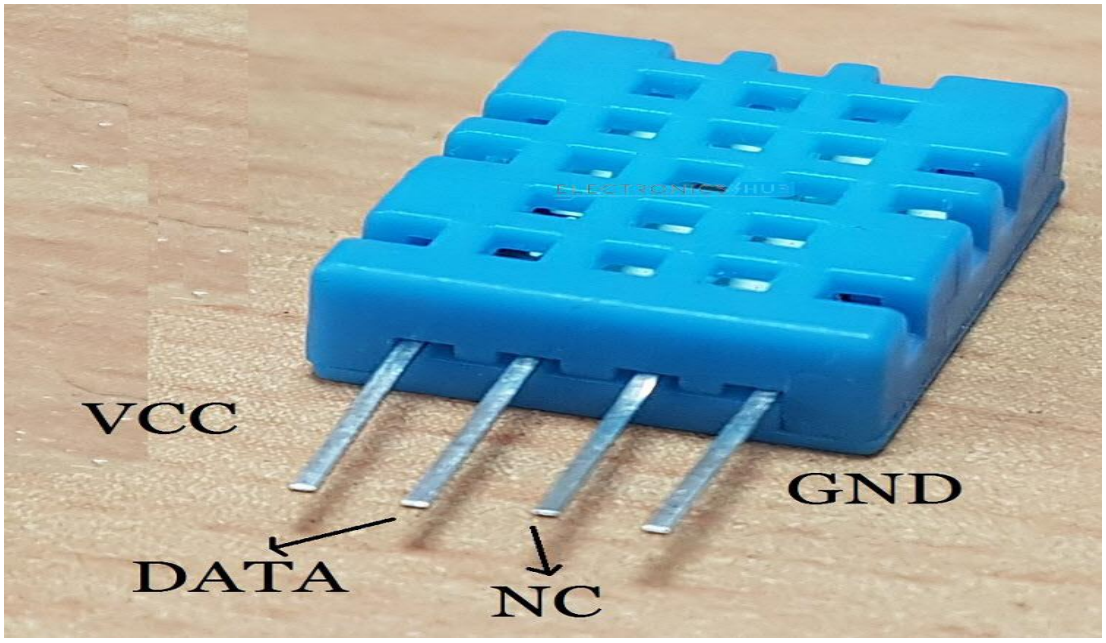
DHT11 Temperature and Humidity Sensor

DHT11 is a part of DHTXX series of Humidity sensors. The other sensor in this series is DHT22. Both these sensors are Relative Humidity (RH) Sensor. As a result, they will measure both the humidity and temperature. Although DHT11 Humidity Sensors are cheap and slow, they are very popular among hobbyists and beginners.



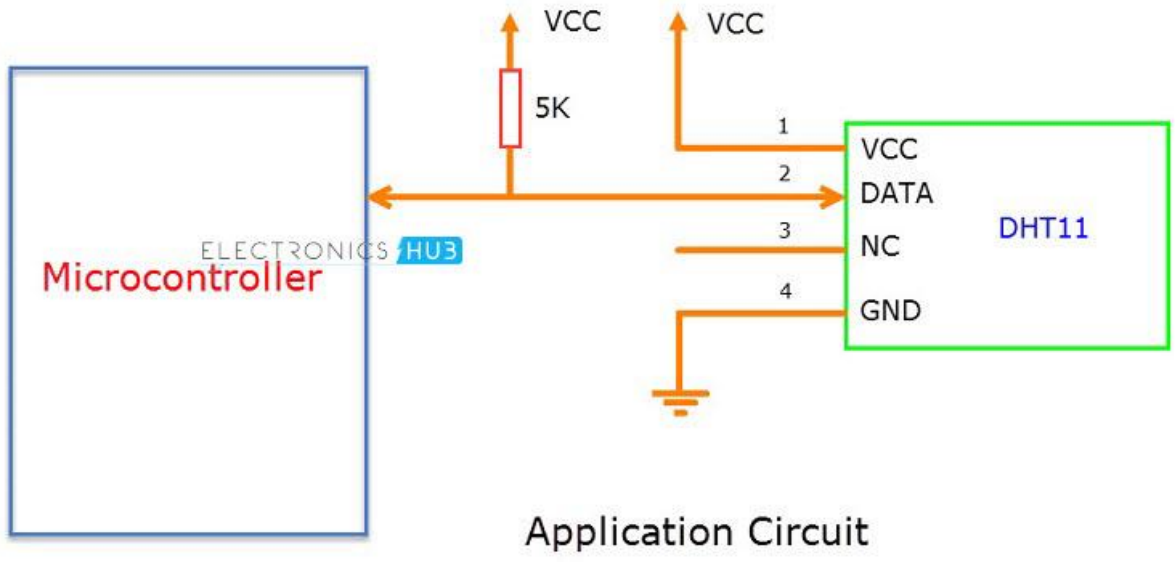
The DHT11 Humidity and Temperature Sensor consists of 3 main components. A resistive type humidity sensor, an NTC (negative temperature coefficient) thermistor (to measure the temperature) and an 8-bit microcontroller, which converts the analog signals from both the sensors and sends out single digital signal.

This digital signal can be read by any microcontroller or microprocessor for further analysis.



DHT11 Humidity Sensor consists of 4 pins: VCC, Data Out, Not Connected (NC) and GND. The range of voltage for VCC pin is 3.5V to 5.5V. A 5V supply would do fine. The data from the Data Out pin is a serial digital data.

The following image shows a typical application circuit for DHT11 Humidity and Temperature Sensor. DHT11 Sensor can measure a humidity value in the range of 20 – 90% of Relative Humidity (RH) and a temperature in the range of 0 – 50^{0}C. The sampling period of the sensor is 1 second i.e.



Application Circuit

All the DHT11 Sensors are accurately calibrated in the laboratory and the results are stored in the memory. A single wire communication can be established between any microcontroller like Arduino or 8051 and the DHT11 Sensor.

(2) **8051 Microcontroller**- The AT89C51 is a lowpower, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density non-volatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pin out. The on-chip Flash allows the program memory to be reprogrammed insystem or by a conventional non-volatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highlyflexible and cost-effective solution to many embedded control applications.

The AT89C51 provides the following standard features: 4K bytes of Flash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, five vector two-level interrupt architecture, a full duplex serial port, and onchip oscillator and clock circuitry. In addition, the AT89C51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power-down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Description:

DHT11 is a single wire digital humidity and temperature sensor, which provides humidity and temperature values serially.

It can measure relative humidity in percentage (20 to 90% RH) and temperature in degree Celsius in the range of 0 to 50°C.

It has 4 pins of which 2 pins are used for supply, 1 is not used and the last one is used for data.

The data is the only pin used for communication. Pulses of different TON and TOFF are decoded as logic 1 or logic 0 or start pulse or end of the frame.

WORKING OF THE PROJECT

Working of Humidity Sensor-

The MCU (microcontroller unit) first sends a low signal of width 18mS to the DHT11. After this signal, the MCU pulls up the communication line and waits for the response from DHT11. It make take up to 2. to 40uS. Then the DHT11 pulls down the communication line and keeps it low for 80uS. Then DHT11 pulls up the line and keeps it high for 80uS. Then the DHT pulls down the line for 50uS and the next high pulse will be the first bit of the data. The data is send in bursts of 8 bits. Each high pulse of the burst indicates a data signal. The 50uS low signals between the data bits are just spacers. The logic of the data bit is identified by measuring the width of it. A 26 to 28uS wide pulse indicates a "LOW" and 70uS wide pulse indicates a "HIGH". In simple words, an pulse narrower than 50uS can be taken as a "LOW" and wider than 50us can be taken as a "HIGH". The first 8 bits of the data burst represents the integral value of the relative humidity, second 8 bits represent the decimal value of the relative humidity, third 8 bits represent the integral value of the temperature data, and the last 8 bits represent the decimal value of the temperature data, For DHT11 the decimal values are always zero and we are measuring the relative humidity

only in this project. So we need to just concern about the first 8 bits of data, that is the integral part of the relative humidity data.

Programming Steps

- First, initialize the LCD16x2_4bit.h library.
- Define pin no. to interface DHT11 sensor, in our program we define P2.1 (Pin no.22)
- Send the start pulse to the DHT11 sensor by making low to high on data pin.
- Receive the response pulse from the DHT11 sensor.
- After receiving the response, receive 40-bit data serially from DHT11 sensor.
- Display this received data on LCD16x2 along with error indication.

ASSEMBLY CODE

```
RS EQU P2.7
RW EQU P2.6
E  EQU P2.5
ORG 000H
MOV DPTR,#LUT
SETB P3.5
CLR P2.0
MOV TMOD,#00100001B
MOV TL1,#00D
ACALL DINT
ACALL TEXT1
MAIN: MOV R1,#8D
     SETB P3.5
     CLR P3.5
     ACALL DELAY1
     SETB P3.5
HERE:JB P3.5,HERE
HERE1:JNB P3.5,HERE1
HERE2:JB P3.5,HERE2
LOOP:JNB P3.5,LOOP
    RL A
    MOV R0,A
    SETB TR1
HERE4:JB P3.5,HERE4
     CLR TR1
     MOV A,TL1
     SUBB A,#50D
     MOV A,R0
     JB PSW.7, NEXT
     SETB ACC.0
     SJMP ESC
NEXT:CLR ACC.0
ESC: MOV TL1,#00D
    CLR PSW.7
    DJNZ R1,LOOP
    ACALL DINT
    ACALL TEXT1
    ACALL LINE2
```

```
    ACALL TEXT2
    ACALL HMDTY
    ACALL CHECK
    ACALL DELAY2
    LJMP MAIN


DELAY1: MOV TH0,#0B9H
       MOV TL0,#0B0H
       SETB TR0
HERE5: JNB TF0,HERE5
      CLR TR0
      CLR TF0
      RET

DELAY2:MOV R1,#112D
 BACK:ACALL DELAY1
      DJNZ R1,BACK
      RET

 CHECK:MOV A,R0
      MOV B,#65D
      SUBB A,B
      JB PSW.7,NEXT1
      ACALL TEXT3
      SETB P2.0
      SJMP ESC1
 NEXT1:ACALL TEXT4
       CLR P2.0
 ESC1:CLR PSW.7
      RET

 CMD: MOV P0,A
   CLR RS
   CLR RW
   SETB E
   CLR E
   ACALL DELAY
   RET

DISPLAY:MOV P0,A
   SETB RS
   CLR RW
   SETB E
   CLR E
   ACALL DELAY
   RET

HMDTY:MOV A,R0
     MOV B,#10D
     DIV AB
     MOV R2,B
     MOV B,#10D
     DIV AB
     ACALL ASCII
     ACALL DISPLAY
     MOV A,B
     ACALL ASCII
     ACALL DISPLAY
```

```
        MOV A,R2
        ACALL ASCII
        ACALL DISPLAY
        MOV A,#"%"
        ACALL DISPLAY
        RET


TEXT1: MOV A,#"H"
       ACALL DISPLAY
       MOV A,#"y"
       ACALL DISPLAY
       MOV A,#"g"
       ACALL DISPLAY
       MOV A,#"r"
       ACALL DISPLAY
       MOV A,#"o"
       ACALL DISPLAY
       MOV A,#"m"
       ACALL DISPLAY
       MOV A,#"e"
       ACALL DISPLAY
       MOV A,#"t"
       ACALL DISPLAY
       MOV A,#"e"
       ACALL DISPLAY
       MOV A,#"r"
       ACALL DISPLAY
       RET

TEXT2: MOV A,#"R"
       ACALL DISPLAY
       MOV A,#"H"
       ACALL DISPLAY
       MOV A,#" "
       ACALL DISPLAY
       MOV A,#"="
       ACALL DISPLAY
       MOV A,#" "
       ACALL DISPLAY
       RET
 TEXT3: MOV A,#" "
       ACALL DISPLAY
       MOV A,#" "
       ACALL DISPLAY
       MOV A,#"O"
       ACALL DISPLAY
       MOV A,#"N"
       ACALL DISPLAY
       RET

 TEXT4:MOV A,#" "
       ACALL DISPLAY
       MOV A,#"O"
       ACALL DISPLAY
       MOV A,#"F"
       ACALL DISPLAY
       MOV A,#"F"
       ACALL DISPLAY
       RET
```

```
 DINT:MOV A,#0CH
    ACALL CMD
    MOV A,#01H
    ACALL CMD
    MOV A,#06H
    ACALL CMD
    MOV A,#83H
    ACALL CMD
    MOV A,#3CH
    ACALL CMD
    RET

LINE2:MOV A,#0C0H
      ACALL CMD
      RET


DELAY: CLR E
    CLR RS
    SETB RW
    MOV P0,#0FFH
    SETB E
    MOV A,P0
    JB ACC.7,DELAY
    CLR E
    CLR RW
    RET

ASCII: MOVC A,@A+DPTR
       RET

LUT: DB  48D
     DB  49D
     DB  50D
     DB  51D
     DB  52D
     DB  53D
     DB  54D
     DB  55D
     DB  56D
     DB  57D
    END
```

C CODE

```
#include<reg51.h>

#include<stdio.h>

#include<string.h>

#include <stdlib.h>

#include "LCD16x2_4bit.h"


sbit DHT11=P2^1;                    /* Connect DHT11 output Pin to P2.1 Pin */
```

```
int I_RH,D_RH,I_Temp,D_Temp,CheckSum;

void timer_delay20ms()                  /* Timer0 delay function */
{
        TMOD = 0x01;
        TH0 = 0xB8;                     /* Load higher 8-bit in TH0 */
        TL0 = 0x0C;                     /* Load lower 8-bit in TL0 */
        TR0 = 1;                        /* Start timer0 */
        while(TF0 == 0);                /* Wait until timer0 flag set */
        TR0 = 0;                        /* Stop timer0 */
        TF0 = 0;                /* Clear timer0 flag */
}

void timer_delay30us()          /* Timer0 delay function */
{
        TMOD = 0x01;                    /* Timer0 mode1 (16-bit timer mode) */
        TH0 = 0xFF;                     /* Load higher 8-bit in TH0 */
        TL0 = 0xF1;                     /* Load lower 8-bit in TL0 */
        TR0 = 1;                        /* Start timer0 */
        while(TF0 == 0);                /* Wait until timer0 flag set */
        TR0 = 0;                        /* Stop timer0 */
        TF0 = 0;                /* Clear timer0 flag */
}

void Request()                  /* Microcontroller send request */
{
        DHT11 = 0;                      /* set to low pin */
        timer_delay20ms();              /* wait for 20ms */
        DHT11 = 1;                      /* set to high pin */
}

void Response()                         /* Receive response from DHT11 */
```

```
{
        while(DHT11==1);
        while(DHT11==0);
        while(DHT11==1);
}

int Receive_data()              /* Receive data */
{
        int q,c=0;
        for (q=0; q<8; q++)
        {
                while(DHT11==0);/* check received bit 0 or 1 */
                timer_delay30us();
                if(DHT11 == 1)  /* If high pulse is greater than 30ms */
                c = (c<<1)|(0x01);/* Then its logic HIGH */
                else            /* otherwise its logic LOW */
                c = (c<<1);
                while(DHT11==1);
        }
        return c;
}void main()
{
        unsigned char dat[20];
        LCD_Init();             /* initialize LCD */

        while(1)
        {
                Request();      /* send start pulse */
                Response();     /* receive response */

                I_RH=Receive_data();    /* store first eight bit in I_RH */
                D_RH=Receive_data();    /* store next eight bit in D_RH */
```

```
                else
                {
                    sprintf(dat,"Hum = %d.%d",I_RH,D_RH);

                }
                delay(100);
            }
}
```

ADDRESSING MODES, MEMORY AND TIME REQUIRED

S.No.	Label	Mnemonic	Operands	Addressing mode	Memory required(BYTES)	Time taken(CYS)
1.		ORG	0000H			
2.		MOV	DPTR	Direct bit	2	1
3.		SETB	P3.5	Absolute	2	1
4.		CLR	P2.0	Immediate	2	1
5.		MOV	TMOD	Direct bit	2	1
6.		MOV	TL1	Direct bit	2	1
7.		MOV	TMOD	Direct bit	2	1
8.		MOV	TL1	Direct bit	2	1
9.		ACALL	DINT	Absolute	2	2
10.		ACALL	TEXT1	Absolute	2	2
11.	MAIN	MOV	R1	Direct	3	2
12.		SETB	P3.5	Direct bit	2	1
13.		CLR	P3.5	Absolute	2	2
14.		ACALL	DELAY1	Absolute	2	2
15.		SETB	P3.5	Absolute	2	2
16	HERE	JB	P3.5	Absolute	2	2
17.	HERE1	JNB	P3.5	Absolute	2	2
18.	HERE2	JB	P3.5	Absolute	2	2
19.	LOOP	JNB	P3.5	Absolute	2	2
20.		RL	A	Absolute	2	2
21.		MOV	R0,A	Absolute	2	2
22.		SETB	TR1	Absolute	2	2
23.	HERE 4	JB	P3.5,HERE4	Absolute	2	2
24.		CLR	TR1	Immediate	2	1
25.		MOV	A,TL1	Absolute	2	2
26.		SUBB	A,#50D	Absolute	2	2
27.		MOV	A,R0	Absolute	2	2
28.		JB	PSW.7,NEXT	Absolute	2	2
29.		SETB	ACC.0	Absolute	2	2
30.		SJMP	ESC	Absolute	2	1
31.	NEXT	CLR	ACC.0	Immediate	2	2
32.	ESC	MOV	TL1	Absolute	2	2
33.		CLR	PSW.7	Immediate	2	1

34.		DJNZ	R1,LOOP	Absolute	2	2
35.		ACALL	DINT	Absolute	2	2
36.		ACALL	TEXT1	Absolute	2	2
37.		ACALL	LINE2	Absolute	2	2
38.		ACALL	TEXT2	Absolute	2	2
39.		ACALL	HMDTY	Absolute	2	1
40.		ACALL	CHECK	Absolute	2	2
41.		ACALL	DELAY2	Absolute	2	2
42.		LJMP	MAIN	Indirect	2	2
43.		SETB	P1.2	Direct bit	2	1
44.	DELAY1	MOV	TH0	Absolute	2	2
45.		MOV	TL0	Absolute	2	2
46.		SETB	TR0	Absolute	2	2
47.	HERE5	JNB	TF0,HERE5	Absolute	2	2
48.		CLR	TR0	Absolute	2	2
49.		CLR	TF0	Absolute	2	2
50.		RET				
51.	DELAY2:	MOV	R1,#112D	Absolute	2	2
52.	BACK	ACALL	DELAY1	Absolute	2	2
53.		DJNZ	R7,BACK	Absolute	2	2
54.		RET				
55.	CHECK	MOV	A,R0	Absolute	2	2
56.		MOV	B,#65D	Absolute	2	2
57.		SUBB	A,B	Absolute	2	2
58.		ACALL	TEXT3	Absolute	2	2
59.		SETB	P2.0	Absolute	2	2
60.		SJMP	ESC1	Absolute	2	2
61.	NEXT1	ACALL	TEXT4	Absolute	2	2
62.		CLR	P2.0	Immediate	2	1
63.	ESC1	CLR	PSW.7	Immediate	2	1
64.		RET				
65.	CMD	MOV	P0,A	Absolute	2	2
66.		CLR	RS	Absolute	2	2
67.		CLR	RW	A	2	1
68.		SETB	E	Absolute	2	2
69.		CLR	E	Absolute	2	2
70.		ACALL	DEALY	Absolute	2	2
71.		RET				
72.	DISPLAY	MOV	P0,A	Absolute	2	2
73.		SETB	RS	Absolute	2	2
74.		CLR	RW	Immediate	2	1
75.		SETB	E	Absolute	2	2

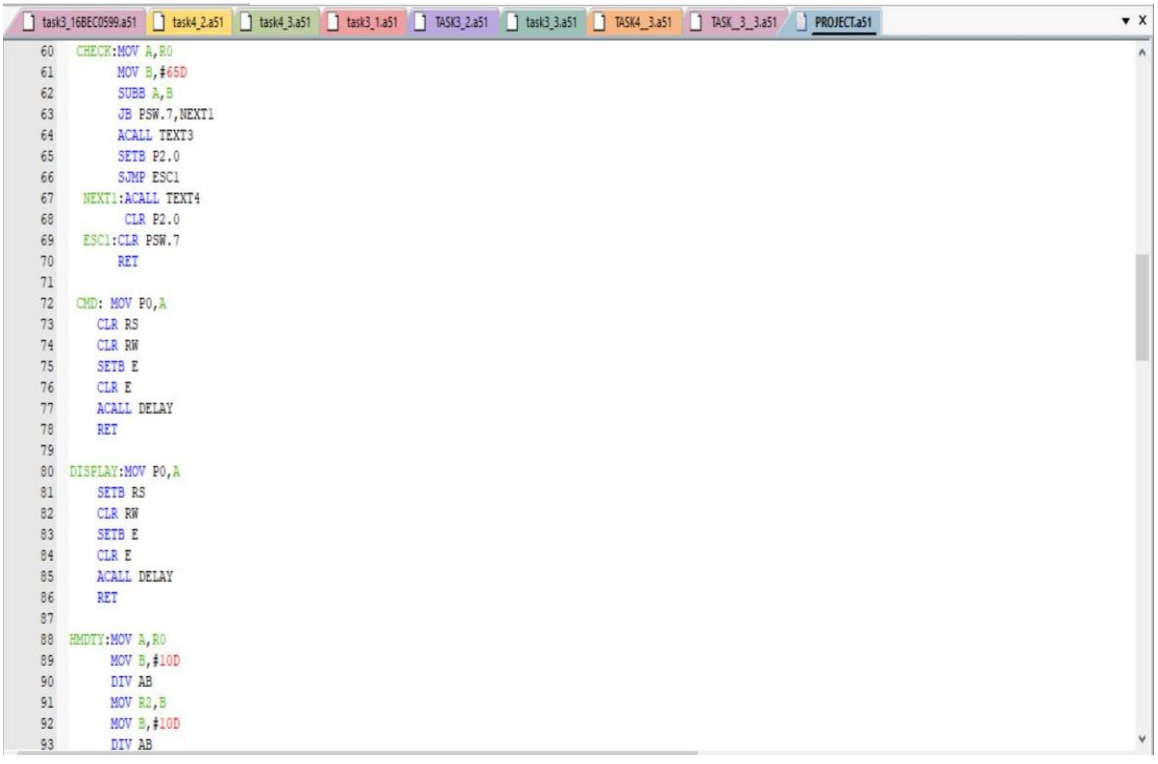
76.		CLR	E	Immediate	2	1
77.		ACALL	DELAY	Absolute	2	2
78.		RET				
79.	HMDTY	MOV	A,R0	Absolute	2	2
80.		MOV	B,#10D	Absolute	2	2
81.		DIV	AB	Absolute	2	2
82.		ACALL	ASCII	Absolute	2	2
83.		ACALL	DISPLAY	Absolute	2	2
84.		MOV	A,R2	Absolute	2	2
85.		ACALL	ASCII	Absolute	2	2
86.		ACALL	DISPLAY	Absolute	2	2
87.		MOV	A,#”%”	Absolute	2	2
88.		ACALL	DISPLAY	Absolute	2	2
89.		RET				
90.		MOV	A,#”H”	Absolute	2	2
91.		ACALL	DISPLAY	Absolute	2	2
92.		MOV	A,#”y”	Absolute	2	2
93.		ACALL	DISPLAY	Absolute	2	2
94.		MOV	A,#”g”	Absolute	2	2
95.		ACALL	DISPLAY	Absolute	2	2
96.		MOV	A,#”r”	Absolute	2	2
97.		ACALL	DISPLAY	Absolute	2	2
98.		MOV	A,#”o”	Absolute	2	2
99.		ACALL	DISPLAY	Absolute	2	2
100.		MOV	A,#”m”	Absolute	2	2
101.		ACALL	DISPLAY	Absolute	2	2
102,		MOV	A,#”e”	Absolute	2	2
103.		ACALL	DISPLAY	Absolute	2	2
104.		MOV	A,#”t”	Absolute	2	2
105.		ACALL	DISPLAY	Absolute	2	2
106.		MOV	A,#”e”	Absolute	2	2
107.		ACALL	DISPLAY	Absolute	2	2
108.		MOV	A,#”r”	Absolute	2	2
109.		ACALL	DISPLAY	Absolute	2	2
110.		RET				
111.	TEXT2	MOV	A,#” ”	Absolute	2	2
112.		ACALL	DISPLAY	Absolute	2	2
113.		MOV	A,#” H”	Absolute	2	2
114.		ACALL	DISPLAY	Absolute	2	2
115.		MOV	A,#” ”	Absolute	2	2
116.		ACALL	DISPLAY	Absolute	2	2
117.		MOV	A,#” =”	Absolute	2	2
118.		ACALL	DISPLAY	Absolute	2	2
119.		MOV	A,#” ”	Absolute	2	2
120.		ACALL	DISPLAY	Absolute	2	2
121.		RET				

122.	TEXT3	MOV	A,#" "	Absolute	2	2
123.		ACALL	DISPLAY	Absolute	2	2
124.		MOV	A,#" "	Absolute	2	2
125.		ACALL	DISPLAY	Absolute	2	2
126.		MOV	A,#"O "	Absolute	2	2
127.		ACALL	DISPLAY	Absolute	2	2
128.		MOV	A,#"N"	Absolute	2	2
129.		ACALL	DISPLAY	Absolute	2	2
130.		RET				
131.	TEXT4	MOV	A,#" "	Absolute	2	2
132.		ACALL	DISPLAY	Absolute	2	2
133.		MOV	A,#"O"	Absolute	2	2
134.		ACALL	DISPLAY	Absolute	2	2
135.		MOV	A,#"F"	Absolute	2	2
136.		ACALL	DISPLAY	Absolute	2	2
137.		MOV	A,#"F"	Absolute	2	2
138.		ACALL	DISPLAY	Absolute	2	2
139.		RET				
140.	DINT	MOV	A,#0CH	Absolute	2	2
141.		ACALL	CMD	Absolute	2	2
142.		MOV	A,#01H	Absolute	2	2
143.		ACALL	CMD	Absolute	2	2
144.		MOV	A,#06H	Absolute	2	2
145.		ACALL	CMD	Absolute	2	2
146.		MOV	A,#83H	Absolute	2	2
147.		ACALL	CMD	Absolute	2	2
148.		MOV	A,#3CH	Absolute	2	2
149.		ACALL	CMD	Absolute	2	2
150.		RET				
151.	DELAY	CLR	E	Immediate	2	1
152.		CLR	RS	Immediate	2	1
153.		SETB	RW	Absolute	2	2
154.		MOV	P0,#0FFH	Absolute	2	2
155.		SETB	E	Absolute	2	2
156.		MOV	A,P0	Absolute	2	2
157.		JB	ACC.7,DELAY	Absolute	2	2
158.		CLR	E	Immediate	2	1
159.		CLR	RW	Immediate	2	1
160.		RET				
161.	ASCII	MOVC	A,@A+DPTR	Direct	2	1
162.		RET				
163.	LUT	DB	48D	Absolute	2	2
164.		DB	49D		2	2

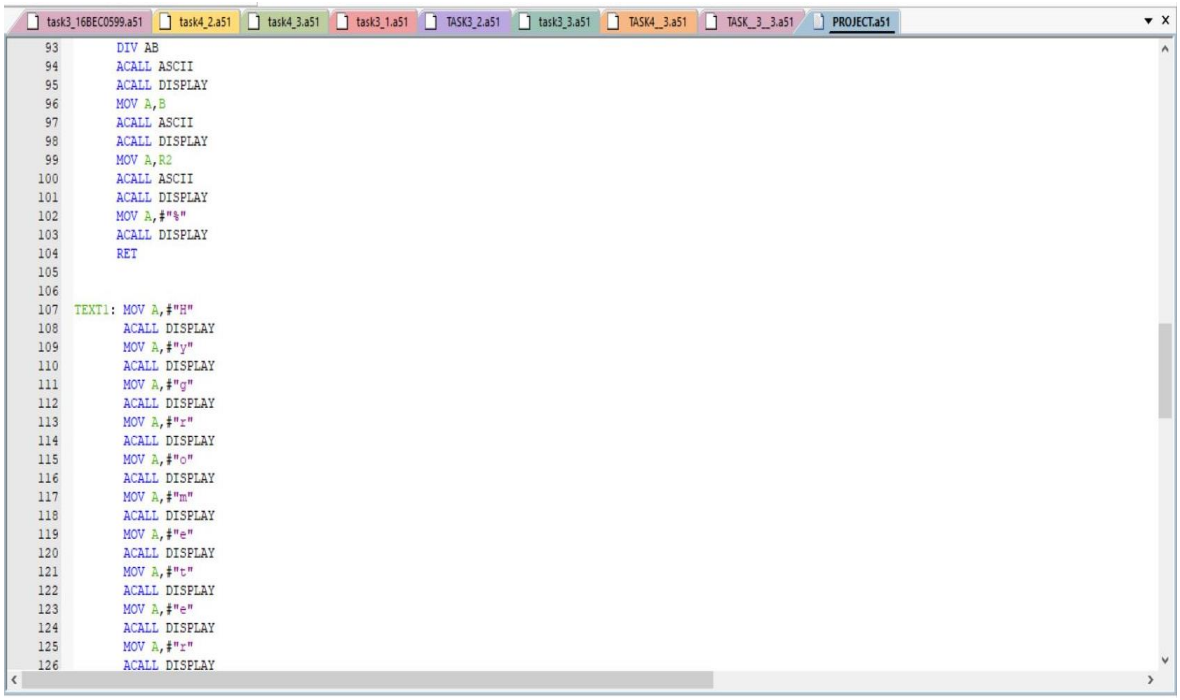
165.		DB	50D		2	2
166.		DB	51D		2	2
167.		DB	52D		2	2
168.		DB	53D		2	1
169.		DB	54D		2	2
170.		DB	55D		2	2
171.		DB	56D		2	1
172.		DB	57D		2	2
173.		END				

SIMULATION AND OUTPUT

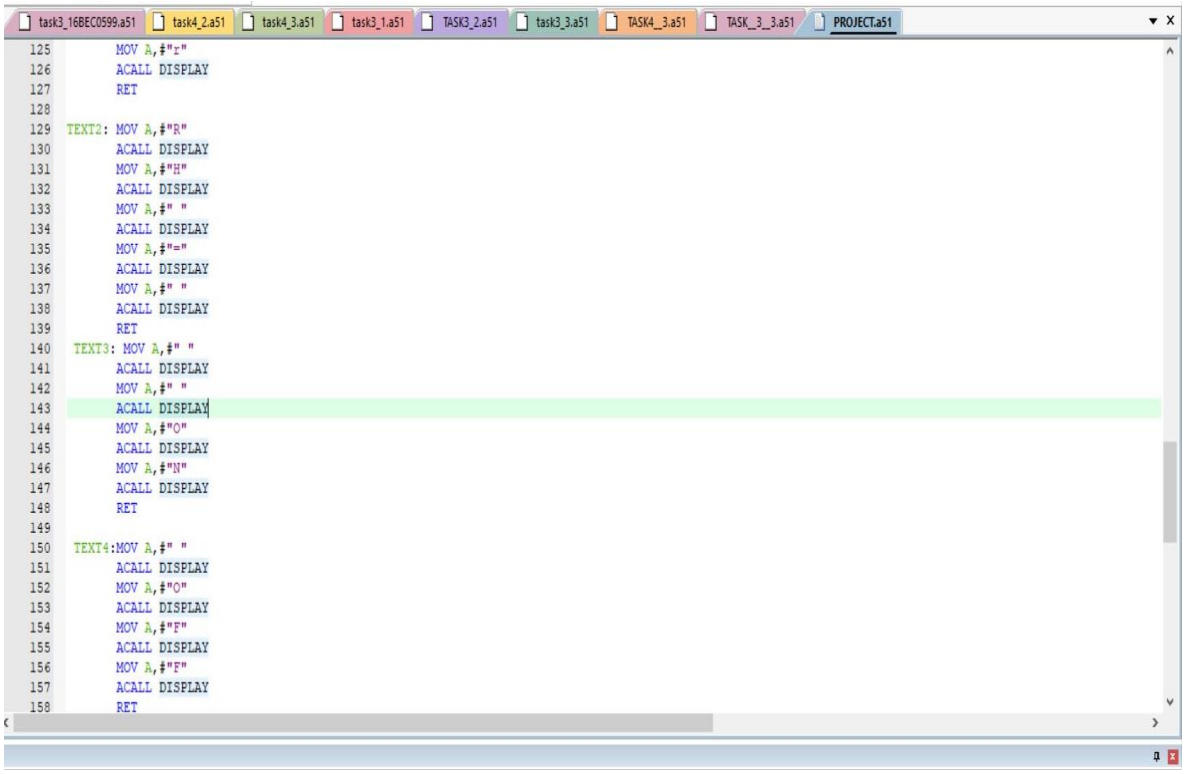
```
ESC: MOV TL1,#00D
     CLR PSW.7
     DJNZ R1,LOOP
     ACALL DINT
     ACALL TEXT1
     ACALL LINE2
     ACALL TEXT2
     ACALL HMDTY
     ACALL CHECK
     ACALL DELAY2
     LJMP MAIN


DELAY1: MOV TH0,#0B9H
        MOV TL0,#0B0H
        SETB TR0
HERE5: JNB TF0,HERE5
       CLR TR0
       CLR TF0
       RET

DELAY2:MOV R1,#112D
  BACK:ACALL DELAY1
       DJNZ R1,BACK
       RET

 CHECK:MOV A,R0
       MOV B,#65D
       SUBB A,B
       JB PSW.7,NEXT1
       ACALL TEXT3
       SETB P2.0
       SJMP ESC1
```



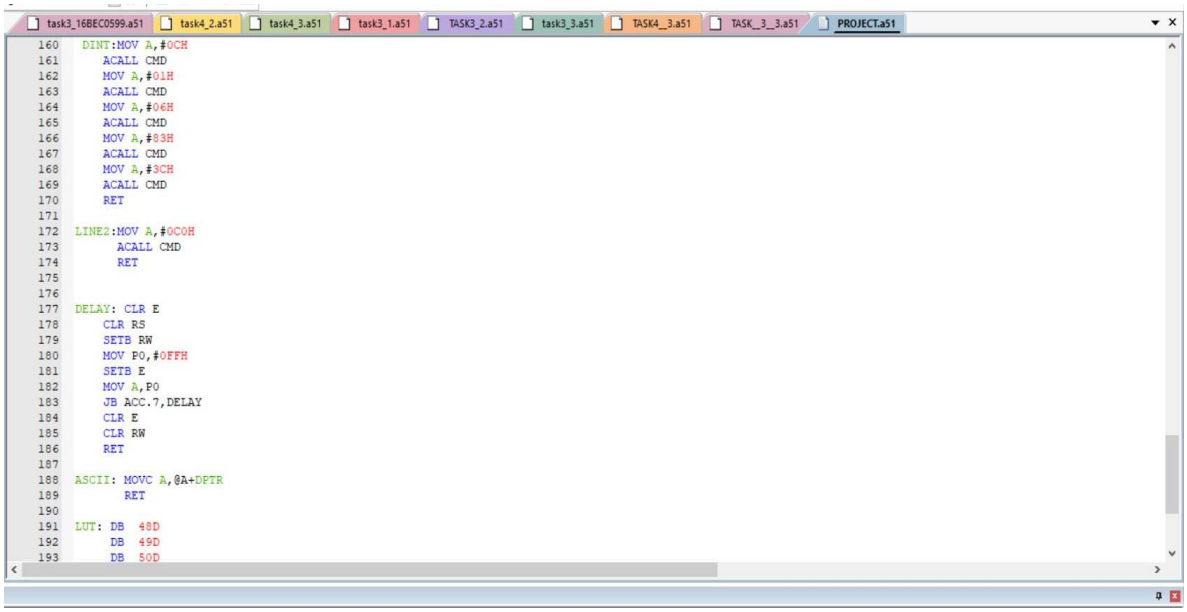
```
 CHECK:MOV A,R0
       MOV B,#65D
       SUBB A,B
       JB PSW.7,NEXT1
       ACALL TEXT3
       SETB P2.0
       SJMP ESC1
 NEXT1:ACALL TEXT4
        CLR P2.0
 ESC1:CLR PSW.7
      RET

CMD: MOV P0,A
   CLR RS
   CLR RW
   SETB E
   CLR E
   ACALL DELAY
   RET

DISPLAY:MOV P0,A
   SETB RS
   CLR RW
   SETB E
   CLR E
   ACALL DELAY
   RET

HMDTY:MOV A,R0
      MOV B,#10D
      DIV AB
      MOV R2,B
      MOV B,#10D
      DIV AB
```



```
93        DIV AB
94        ACALL ASCII
95        ACALL DISPLAY
96        MOV A,B
97        ACALL ASCII
98        ACALL DISPLAY
99        MOV A,R2
100       ACALL ASCII
101       ACALL DISPLAY
102       MOV A,#"%"
103       ACALL DISPLAY
104       RET
105
106
107   TEXT1: MOV A,#"H"
108          ACALL DISPLAY
109          MOV A,#"y"
110          ACALL DISPLAY
111          MOV A,#"g"
112          ACALL DISPLAY
113          MOV A,#"z"
114          ACALL DISPLAY
115          MOV A,#"o"
116          ACALL DISPLAY
117          MOV A,#"m"
118          ACALL DISPLAY
119          MOV A,#"e"
120          ACALL DISPLAY
121          MOV A,#"t"
122          ACALL DISPLAY
123          MOV A,#"e"
124          ACALL DISPLAY
125          MOV A,#"z"
126          ACALL DISPLAY
```

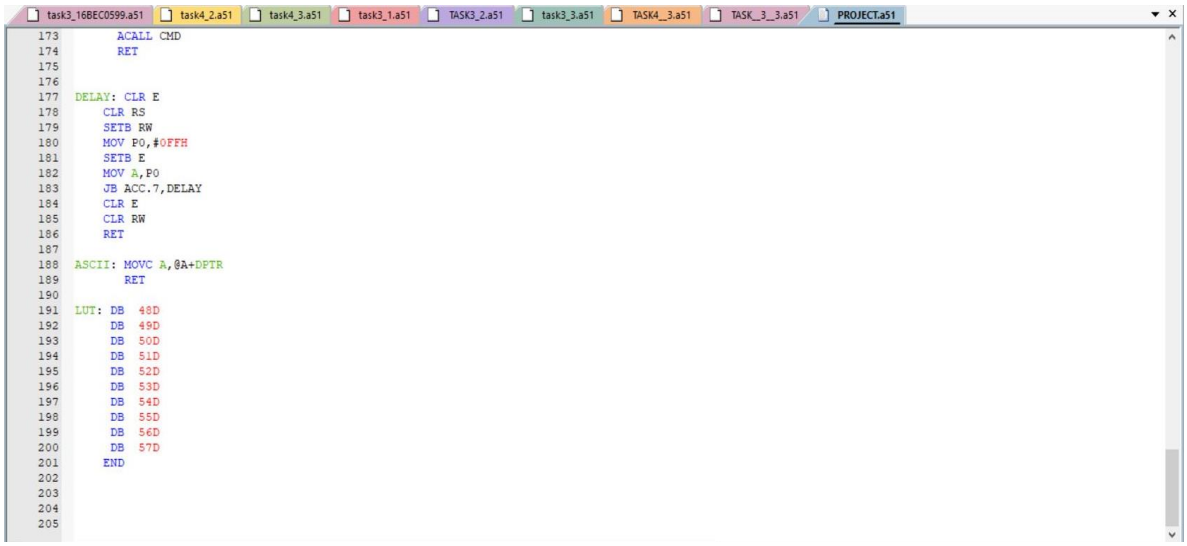


```
125       MOV A,#"r"
126       ACALL DISPLAY
127       RET
128
129   TEXT2: MOV A,#"R"
130          ACALL DISPLAY
131          MOV A,#"H"
132          ACALL DISPLAY
133          MOV A,#" "
134          ACALL DISPLAY
135          MOV A,#"="
136          ACALL DISPLAY
137          MOV A,#" "
138          ACALL DISPLAY
139          RET
140    TEXT3: MOV A,#" "
141          ACALL DISPLAY
142          MOV A,#" "
143          ACALL DISPLAY
144          MOV A,#"O"
145          ACALL DISPLAY
146          MOV A,#"N"
147          ACALL DISPLAY
148          RET
149
150    TEXT4:MOV A,#" "
151          ACALL DISPLAY
152          MOV A,#"O"
153          ACALL DISPLAY
154          MOV A,#"F"
155          ACALL DISPLAY
156          MOV A,#"F"
157          ACALL DISPLAY
158          RET
```



```
 160   DINT:MOV A,#0CH
 161      ACALL CMD
 162      MOV A,#01H
 163      ACALL CMD
 164      MOV A,#06H
 165      ACALL CMD
 166      MOV A,#83H
 167      ACALL CMD
 168      MOV A,#3CH
 169      ACALL CMD
 170      RET
 171
 172   LINE2:MOV A,#0C0H
 173         ACALL CMD
 174         RET
 175
 176
 177   DELAY: CLR E
 178      CLR RS
 179      SETB RW
 180      MOV P0,#0FFH
 181      SETB E
 182      MOV A,P0
 183      JB ACC.7,DELAY
 184      CLR E
 185      CLR RW
 186      RET
 187
 188   ASCII: MOVC A,@A+DPTR
 189         RET
 190
 191   LUT: DB  48D
 192        DB  49D
 193        DB  50D
```

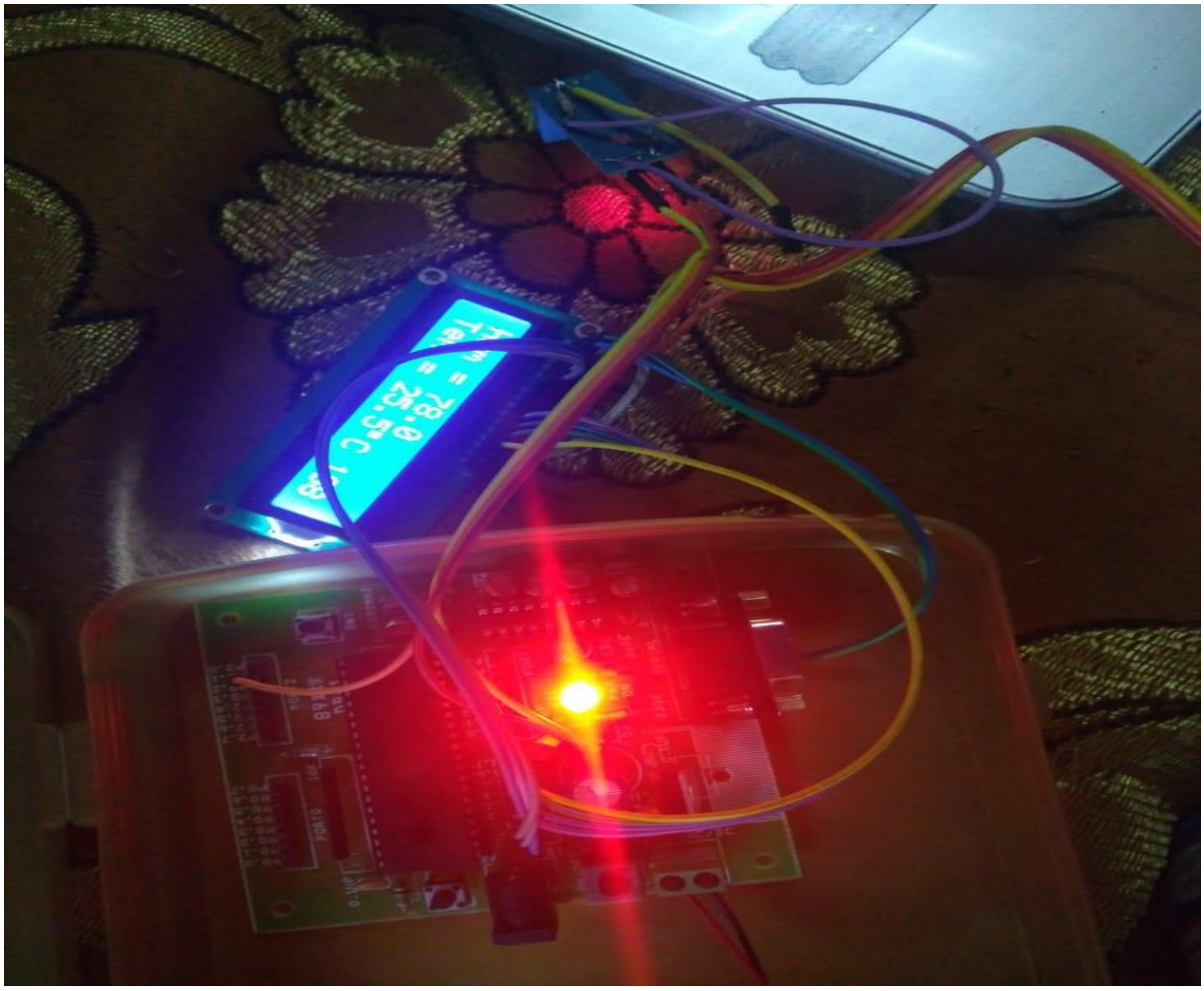
```
 173         ACALL CMD
 174         RET
 175
 176
 177   DELAY: CLR E
 178      CLR RS
 179      SETB RW
 180      MOV P0,#0FFH
 181      SETB E
 182      MOV A,P0
 183      JB ACC.7,DELAY
 184      CLR E
 185      CLR RW
 186      RET
 187
 188   ASCII: MOVC A,@A+DPTR
 189         RET
 190
 191   LUT: DB  48D
 192        DB  49D
 193        DB  50D
 194        DB  51D
 195        DB  52D
 196        DB  53D
 197        DB  54D
 198        DB  55D
 199        DB  56D
 200        DB  57D
 201      END
 202
 203
 204
 205
```



```
 173         ACALL CMD
 174         RET
 175
 176
 177   DELAY: CLR E
 178      CLR RS
 179      SETB RW
 180      MOV P0,#0FFH
 181      SETB E
 182      MOV A,P0
 183      JB ACC.7,DELAY
 184      CLR E
 185      CLR RW
 186      RET
 187
 188   ASCII: MOVC A,@A+DPTR
 189         RET
 190
 191   LUT: DB  48D
 192        DB  49D
 193        DB  50D
 194        DB  51D
 195        DB  52D
 196        DB  53D
 197        DB  54D
 198        DB  55D
 199        DB  56D
 200        DB  57D
 201      END
 202
 203
 204
 205
```

RESULTS

- An assembly level code for the humidity sensor using LCD display has been written and executed in Keil software.
- A hardware model of humidity sensor using LCD display has also been implemented.



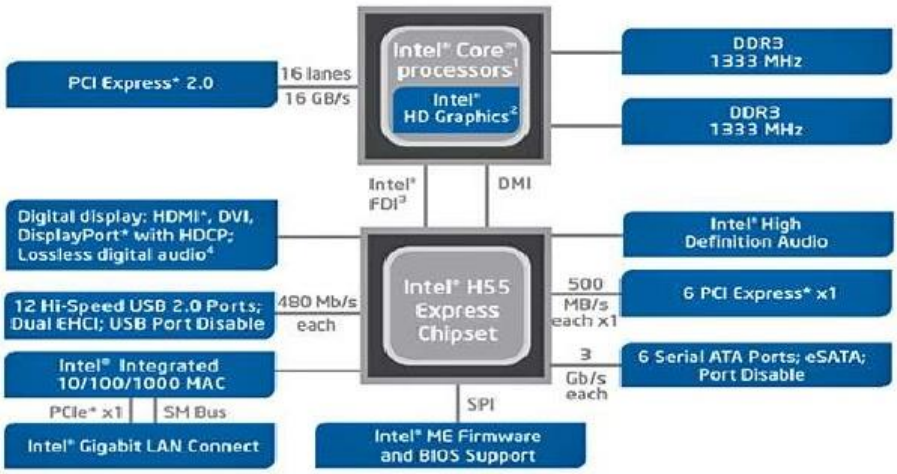
APPLICATIONS

- DHT11 Relative Humidity and Temperature Sensor can be used in many applications like:
- HVAC (Heating, Ventilation and Air Conditioning) Systems
- Weather Stations
- Medical Equipment for measuring humidity
- Home Automation Systems
- Automotive and other weather control applications.

Advantages:

- Low production cost.
- This system is applicable for both the indoor and outdoor environment.
- Setting the destination is very easy.
- It is dynamic system.
- Less space.
- Low power consumption.
- Low design time.

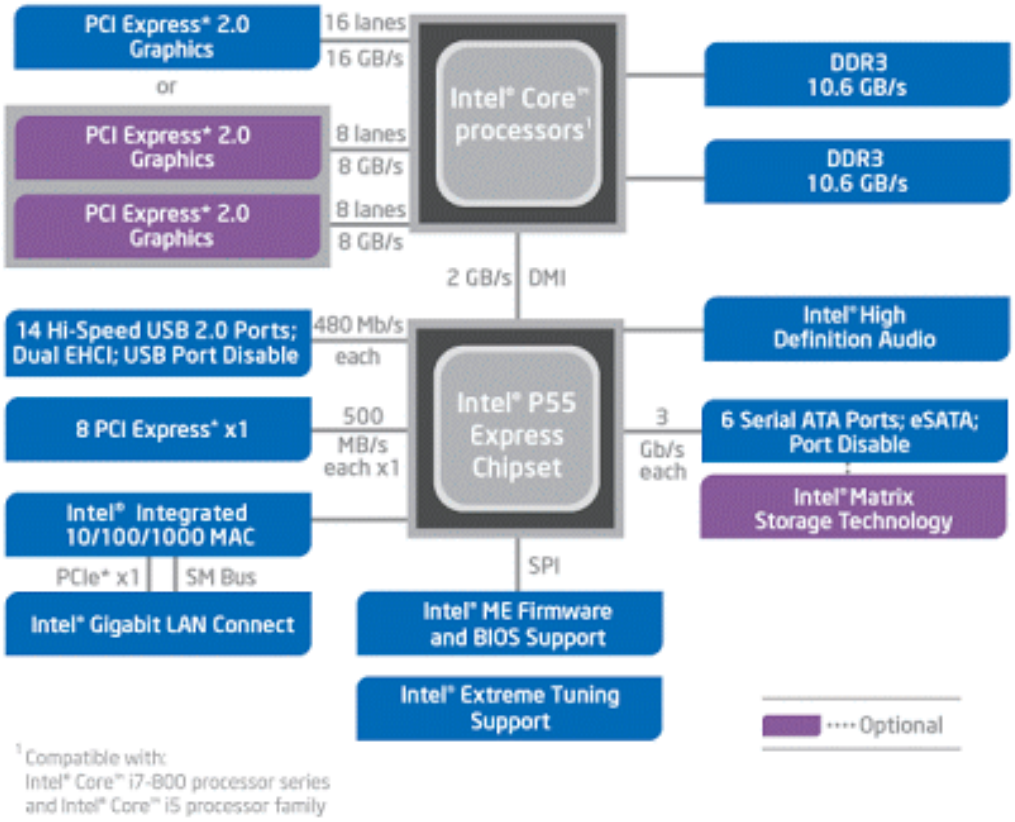
I3 BLOCK DIAGRAM :



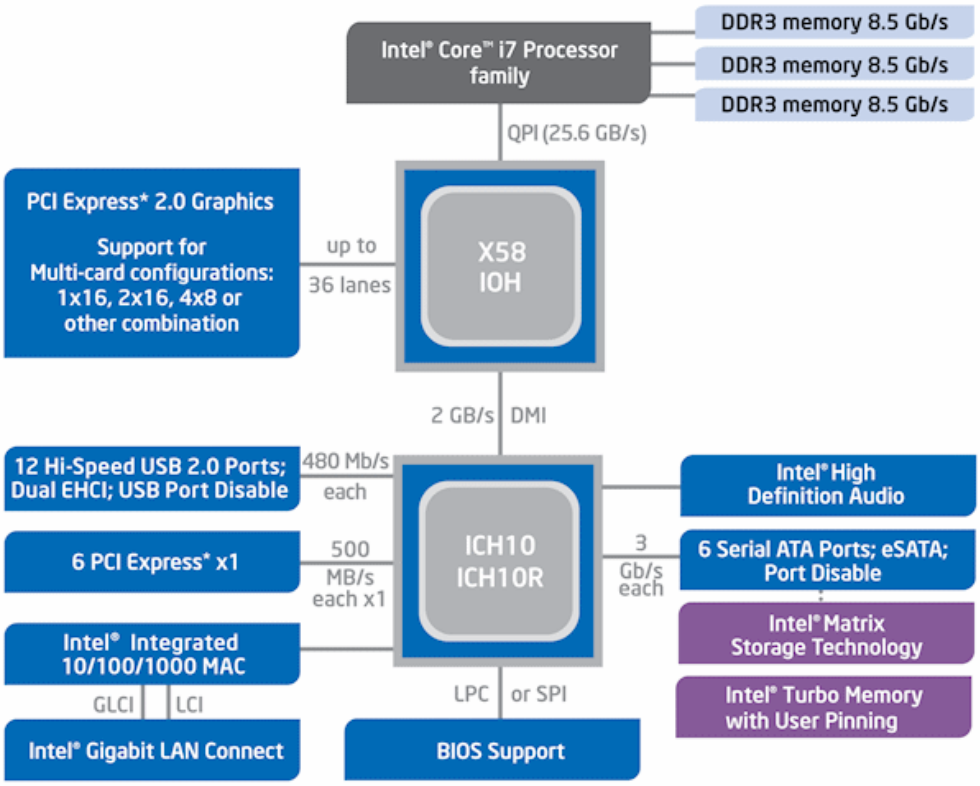
6

Intel core processor Arrandale 8-9-2010

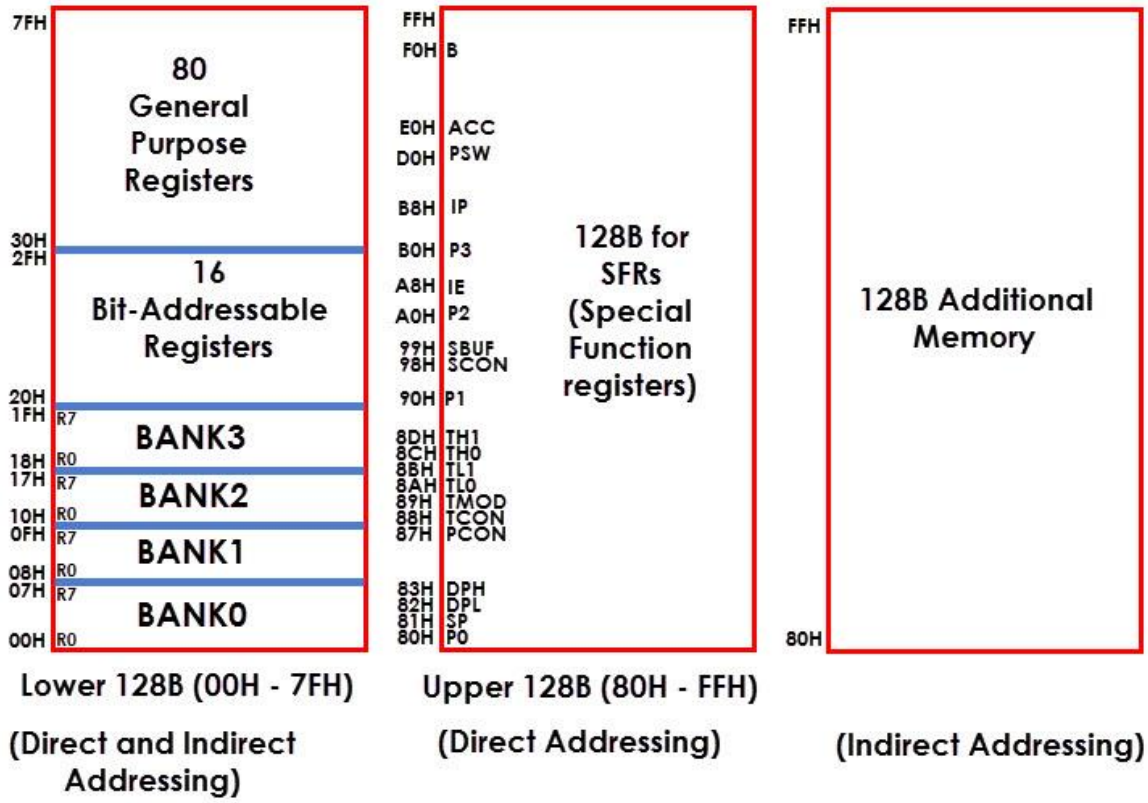
Intel I5


PCI Express* 2.0 Graphics
16 lanes
16 GB/s
or
PCI Express* 2.0 Graphics
8 lanes
8 GB/s
PCI Express* 2.0 Graphics
8 lanes
8 GB/s
Intel® Core™ processors[1]
DDR3 10.6 GB/s
DDR3 10.6 GB/s
2 GB/s
DMI
14 Hi-Speed USB 2.0 Ports; Dual EHCI; USB Port Disable
480 Mb/s each
Intel® High Definition Audio
8 PCI Express* x1
500 MB/s each x1
Intel® P55 Express Chipset
3 Gb/s each
6 Serial ATA Ports; eSATA; Port Disable
Intel® Matrix Storage Technology
Intel® Integrated 10/100/1000 MAC
PCIe* x1
SM Bus
SPI
Intel® Gigabit LAN Connect
Intel® ME Firmware and BIOS Support
Intel® Extreme Tuning Support
Optional
[1] Compatible with:
Intel® Core™ i7-800 processor series
and Intel® Core™ i5 processor family

Intel I7


Intel® Core™ i7 Processor family
DDR3 memory 8.5 Gb/s
DDR3 memory 8.5 Gb/s
DDR3 memory 8.5 Gb/s
QPI (25.6 GB/s)
PCI Express* 2.0 Graphics
Support for Multi-card configurations: 1x16, 2x16, 4x8 or other combination
up to
36 lanes
X58 IOH
2 GB/s
DMI
12 Hi-Speed USB 2.0 Ports; Dual EHCI; USB Port Disable
480 Mb/s each
6 PCI Express* x1
500 MB/s each x1
Intel® Integrated 10/100/1000 MAC
GLCI
LCI
Intel® Gigabit LAN Connect
ICH10 ICH10R
LPC
or SPI
BIOS Support
Intel® High Definition Audio
3 Gb/s each
6 Serial ATA Ports; eSATA; Port Disable
Intel® Matrix Storage Technology
Intel® Turbo Memory with User Pinning

Register organization of 8051



REFERENCES

[1]. Dusyant Pande, Jeetender Singh Chauhan and Nitin Parihar, The Real Time Hardware Design to Automatically Monitor Light and Temperature, International Journal of Innovative Research In Science, Engineering and Technology, Vol. 2, Issue 5, May 2013. [2]. A. Goswami, T.Bezboruah and K. C. Sarma, Design of an Embedded System for Monitoring and Controlling Temperature and Light, International Journal of Electronic Engineering Research, Vol.1, No.1, pp. 27-36, 2009.

[3]. R. A. Eigenberg, J. A. Nainaber, T. M. BrownBrandl and G. L. Hahn, Development of Rugged Environmental Monitoring Units for Temperature and Humidity, American Society of Agricultural Engineers,Vol.18(4), 493-496, 2002.

[4]. Ajit Pal, Microcontrollers: Principles and Applications, PHI Learning Pvt. Ltd., 2011

[5]. Muhammed Ali Mazidi, Janice Gillispie Mazidi and Rolin D. McKinlay, The 8051 Microcontroller and Embedded Systems using Assembly and C, Prentice Hall, Second Edition, 2006.

[6]. M Ramu, CH. Rajendra prasad, Cost Effective Atomization of Indian Agricultural System using 8051 Microcontroller,International Journal of Advanced Research in Computer and Communication Engineering, Vol.2, Issue 7, July 2013.