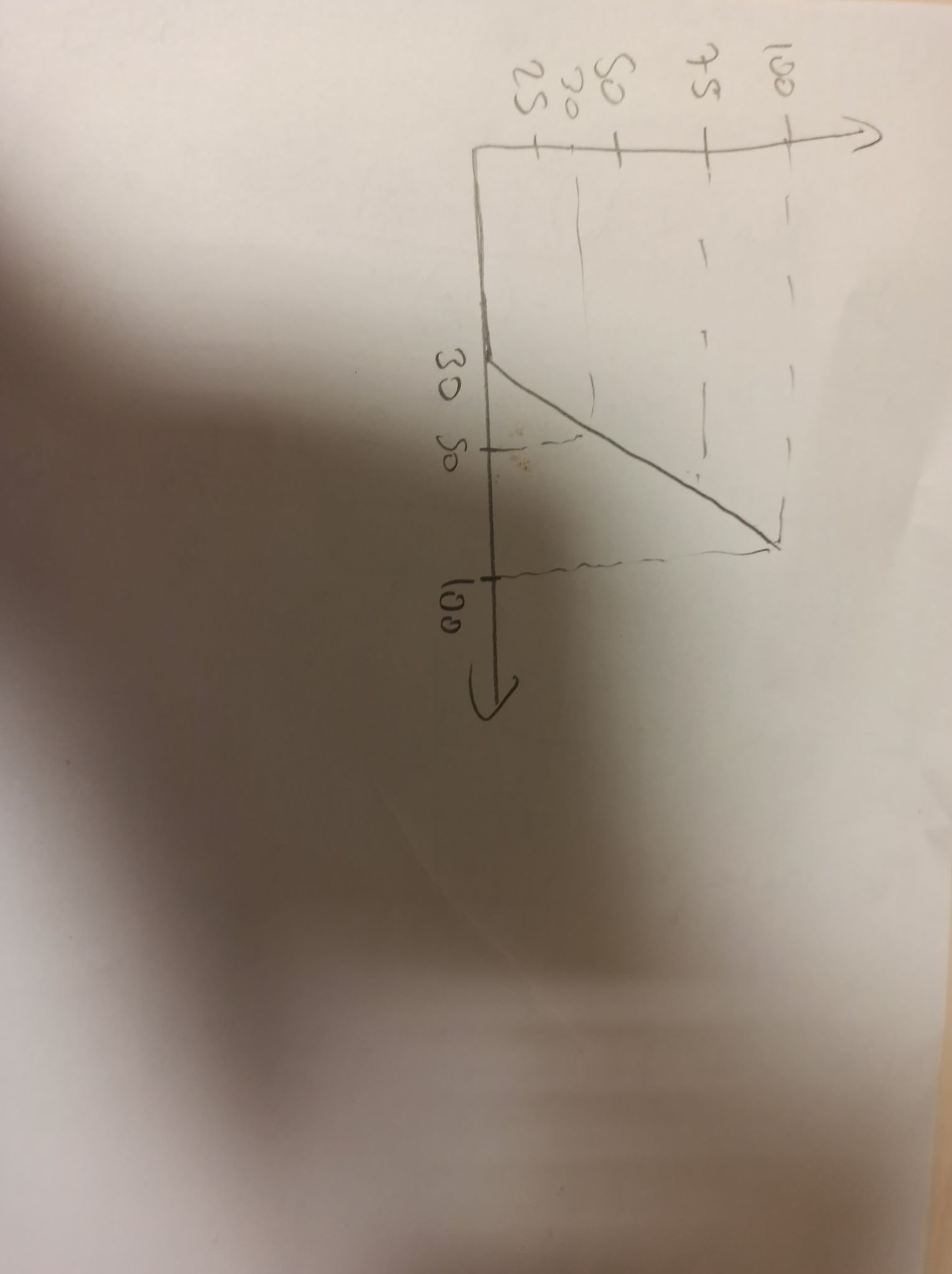
**Cng 336**

**Lab Final Prework**

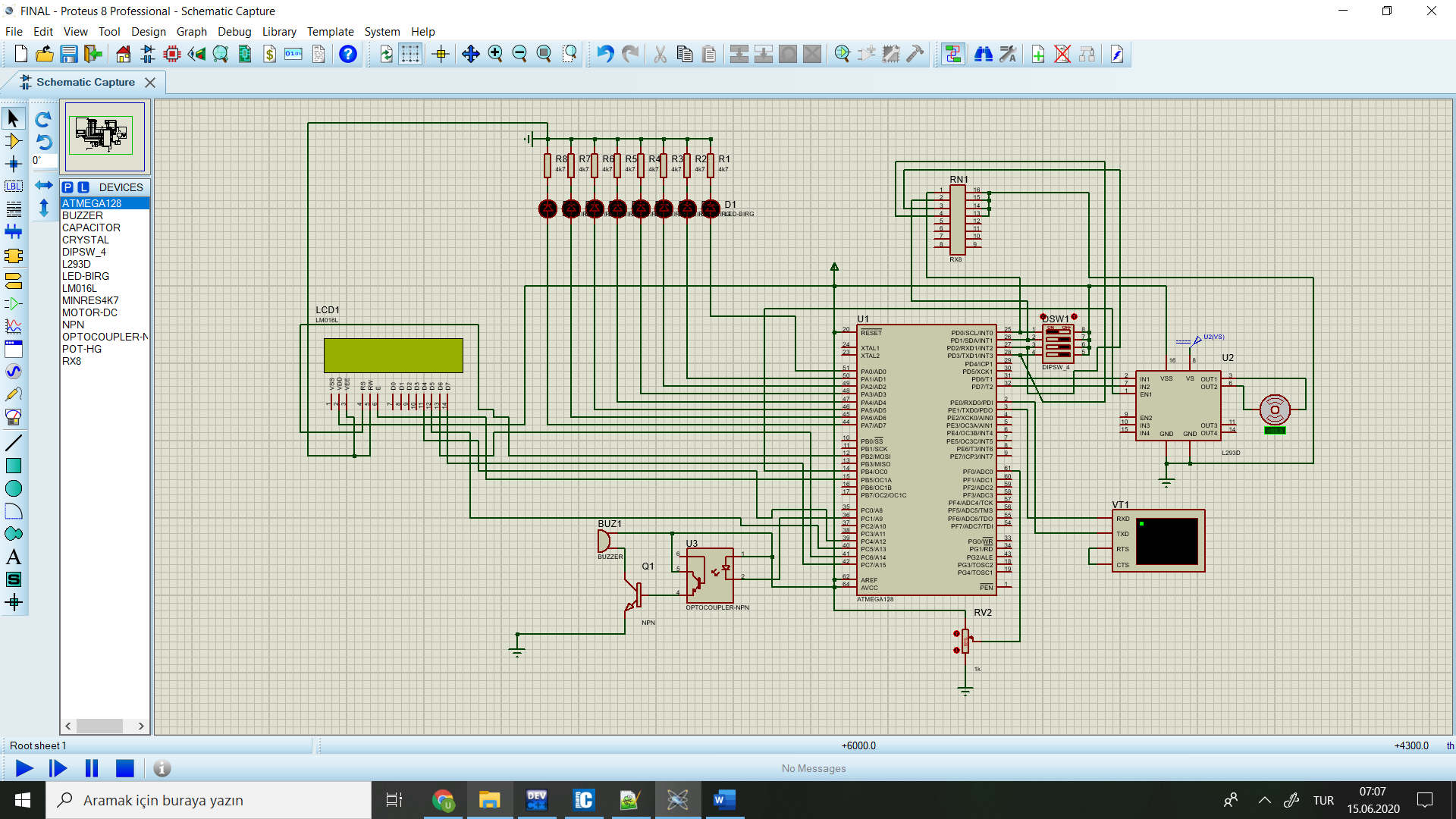
**Ulaş Cem Erten - 2256006**

**6.2.3)**



In above picture y axis shows the fan speed, x axis shows the temperature. Under 30 C, fan speed is zero. And Its working with max speed in 100 C with 100 percent fan speed.

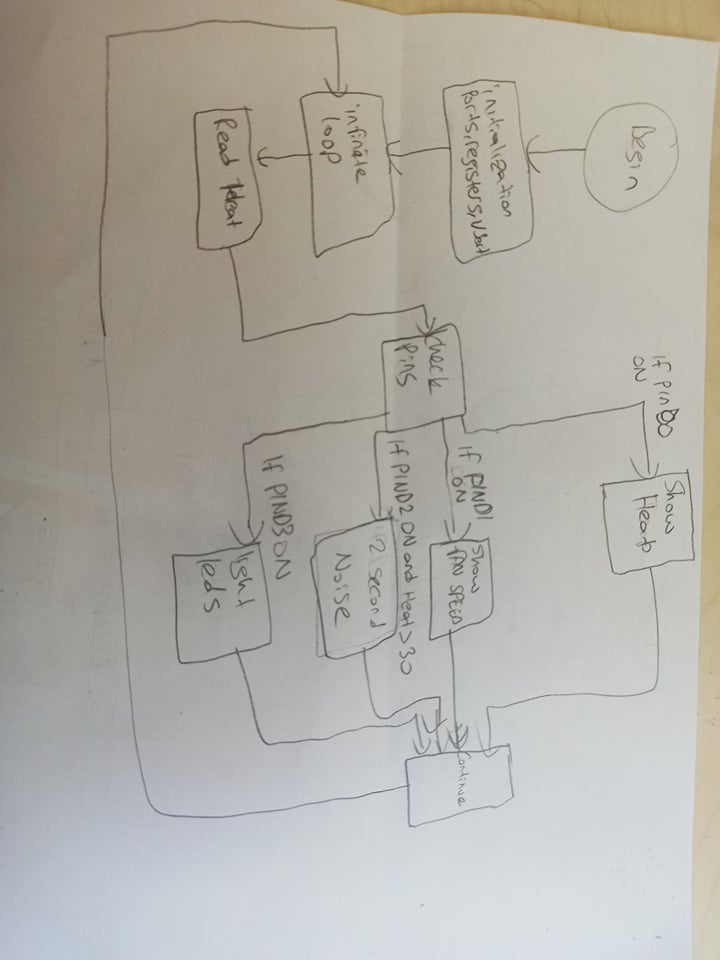
**Schematic**



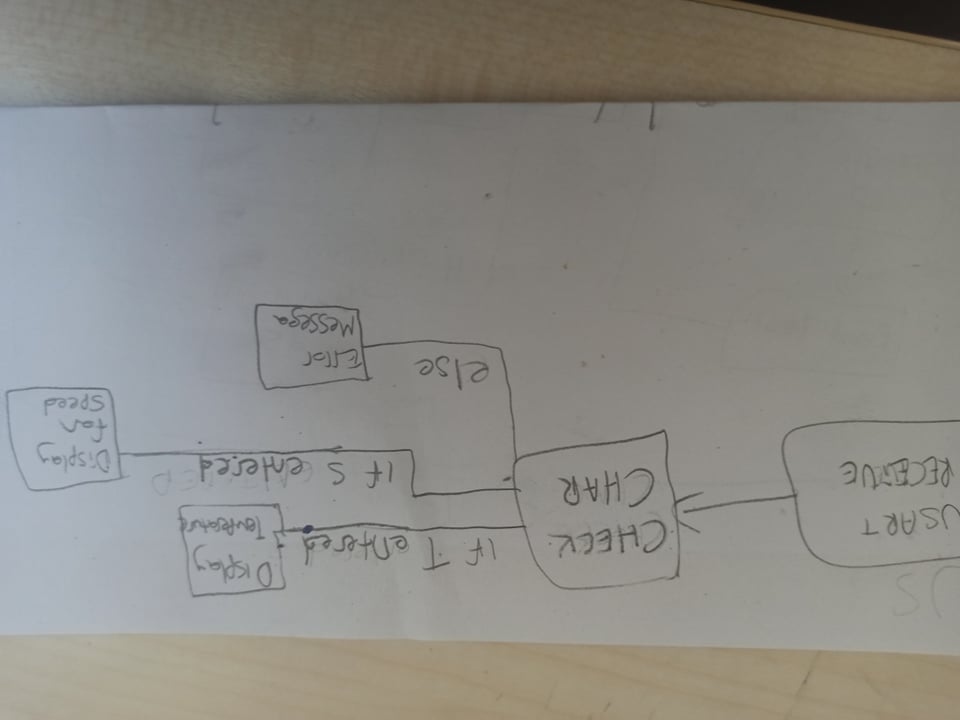
I used PE0 and PE1 to communicate with terminal. I used PD0 for LCD Line 1 select for Temperature display on LCD, PD1 for LCD Line 2 Select for Speed Display on LCD, PD3 for LEDs Display select on PORTA and PD2 for Buzzer noise select. I used PC1 for Buzzer. I used PB4 to deliver PWM signal to the motor driver. I used PB2 for LCD RS, PB5 for LCD En, PC4 to PC7 for LCD Data delivery.

**Flow Diagram**

**Main Module**

****

**Usart Module**

****

**Source Code**

sbit LCD\_RS at PORTB2\_bit;

sbit LCD\_EN at PORTB5\_bit;

sbit LCD\_D4 at PORTC4\_bit;

sbit LCD\_D5 at PORTC5\_bit;

sbit LCD\_D6 at PORTC6\_bit; // lcd init

sbit LCD\_D7 at PORTC7\_bit;

sbit LCD\_RS\_Direction at DDB2\_bit;

sbit LCD\_EN\_Direction at DDB5\_bit;

sbit LCD\_D4\_Direction at DDC4\_bit;

sbit LCD\_D5\_Direction at DDC5\_bit;

sbit LCD\_D6\_Direction at DDC6\_bit;

sbit LCD\_D7\_Direction at DDC7\_bit;

void send\_char(unsigned char my\_char){ // send char function

while (!((UCSR0A) & (1 << (UDRE0))));

UDR0 = my\_char;

}

void send\_string(unsigned char\* my\_string){ // send string function

while(\*my\_string)

send\_char(\*my\_string++);

}

unsigned char\* my\_data = 0x500;

void clear\_every\_thing(unsigned char\* my\_data, unsigned int ADDR)

{

int i = 0; // it clears everything in the given adress i used to clear usart

memset(my\_data, 0, 0x50);

my\_data = ADDR;

}

unsigned char\* int\_to\_String(unsigned int x) { //int to string convertion function

unsigned char converted[4]="";

unsigned int initial\_value = x;

char temp\_val;

if(initial\_value==100){ // look up table for 3 digit number and 1 digit number

return "100";

}

if(initial\_value==0){

return "0";

}

if(initial\_value==1){

return "1";

}

if(initial\_value==2){

return "2";

}

if(initial\_value==3){

return "3";

}

if(initial\_value==4){

return "4";

}

if(initial\_value==5){

return "5";

}

if(initial\_value==6){

return "6";

}

if(initial\_value==7){

return "7";

}

if(initial\_value==8){

return "8";

}

if(initial\_value==9){

return "9";

}

if(x==0)

{

\*converted = '0';

\*(converted + 1) = '\0';

return "";

}

else

{

unsigned int i = 0;

while(x)

{

unsigned char d = x % 10;

x /= 10;

if(d>9)

converted[i++]=(d + 55);// to find its ascii if it is a letter

else

converted[i++]=(d + 48);

}

converted[i] = '\0';

temp\_val = converted[0]; //swap operation because now it is in opposite order

converted[0] = converted[1];

converted[1] = temp\_val;

return converted;

}

}

unsigned char waves = 0;

void start\_noise() iv 0x001E{ // create square wave on portb 4 for making noise if timer 0 compare handle

PORTB.B4 = waves ^ 1;

}

unsigned int sum\_twos = 0;

void stop\_noise() iv 0x0014 // for stopping noise when timer2 overflow

{++sum\_twos; // increment until it becomes 2 sec

}

unsigned char temp;

int my\_flag;

unsigned char text\_for\_usart[0x50];

unsigned char temp\_for\_usart[4];

unsigned int unit\_select;

unsigned int fan\_speed = 0;

unsigned int heat\_value = 0;

void usart() iv 0x0024{

unsigned int enter = 0;

temp = UDR0;

if(temp == '\n' || temp == '\r') // check if enter is pressed

enter = 1;

else

{if((unsigned int) my\_data < 0x530) {

\*my\_data++ = temp;

\*my\_data = '\0';}

}

UDR0 = temp;

if(enter)

{my\_data = 0x500;

my\_flag=1; // flag for adding the unit % or C

if(!(strcmp(my\_data,"T"))){ // if equals will return 0

unit\_select=1; //choose % or C

strcpy(text\_for\_usart, "TEMPERATURE: "); // basic copy string

strcpy(temp\_for\_usart,int\_to\_String(heat\_value));

strcpy(text\_for\_usart + strlen("TEMPERATURE: "), temp\_for\_usart);

}

else if(!(strcmp(my\_data,"S"))){

unit\_select=2;//same as above

strcpy(text\_for\_usart, "FAN SPEED: ");

strcpy(temp\_for\_usart,int\_to\_String(fan\_speed));

strcpy(text\_for\_usart + strlen("FAN SPEED: "), temp\_for\_usart);

}

else{

send\_string("Wrong input entered, Please "); // another input

my\_flag=0; // flag will choose wrong input message

}

if(my\_flag==1){

if(unit\_select==1)

strcpy(text\_for\_usart + strlen("TEMPERATURE: ") + strlen(temp\_for\_usart), "C\r"); //explained above

if(unit\_select==2)

strcpy(text\_for\_usart + strlen("FAN SPEED: ") + strlen(temp\_for\_usart), "%\r"); //explained above

send\_string(text\_for\_usart);

}

clear\_every\_thing(my\_data, 0x500); // clear the data so last data value it wont be seen while asking input

send\_string("Enter S for % fan speed and T for temperature:");

}

enter = 0;

}

void active\_switches();

unsigned char len\_temperature;

unsigned char\* lcd\_up = 0x300;

unsigned char\* lcd\_down = 0x320;

unsigned char buzzer\_1 = 0;

unsigned char buzzer\_2 = 0;

unsigned char motor = 0;

unsigned int second\_adc = 0;

unsigned char my\_led = 0;

unsigned int first\_adc = 0;

void all\_calculations() iv 0x002A //adc function

{

second\_adc = ADCL;

second\_adc += (ADCH & 0x03) << 8; // hence value max 10 bit we should take first 2 bits of high and shift it 8 bits left

if(first\_adc != second\_adc) // if them changed

{first\_adc = second\_adc; //store the

heat\_value = (second\_adc \* 0.098); // heat = 5/1023/50 mV \* (10 bit value)

if(heat\_value < 30) // if new heat value <30

{

buzzer\_2 = 0; // buzzer will not sound

fan\_speed = 0; //fan speed will be 0

if(motor) // if motor already active

{

((TIMSK) &= ~(1 << (OCIE0))); // enable interrupt for timer0

waves = 0; // make waves variable 0 so when next time interrupt it will create waves on portb4

PORTB.B4 = 0; // but no need because heat < 30

motor = 0; // motor shouldnt work under 30 C

}

if(buzzer\_1) //this condition is for high to low temp which means buzzer1 is already set but now we are under 30 C

{((TIMSK) &= ~(1 << (TOIE2))); // enable timer2 interrupt

TCNT2 = 0; //load 0 to timer 2

buzzer\_1 = 0; // now buzzer1 should be 0

}

}

else //if heat > 30

{

fan\_speed = 100\*(heat\_value-30)/70; // calculate fan speed

if(!motor) //if low to high

TIMSK |= 1 << (OCIE0); // start timer 0 to make noise

motor = 1; //now motor is active

}

OCR0 = (fan\_speed \* 255) / 100; // load needed value to ocr0

strcpy(lcd\_up, "TEMPERATURE:");

strcpy(temp\_for\_usart,int\_to\_String(heat\_value));

strcpy(lcd\_up + 12, temp\_for\_usart);

len\_temperature = strlen(temp\_for\_usart);

lcd\_up[12 + len\_temperature] = 'C';

memset(lcd\_up + 13 + len\_temperature, ' ', 11 - len\_temperature);

strcpy(lcd\_down, "FAN SPEED:");

strcpy(temp\_for\_usart,int\_to\_String(fan\_speed));

strcpy(lcd\_down + 10, temp\_for\_usart);

len\_temperature = strlen(temp\_for\_usart);

lcd\_down[10 + strlen(temp\_for\_usart)] = '%';

memset(lcd\_down + 11 + len\_temperature, ' ', 11 - len\_temperature);

my\_led = ((1 << (second\_adc / 127)) - 1) & 0xFF; // led values on port a 5V 100 -> 0.625 V for 12.5 for each led

}

active\_switches();

}

void active\_switches(){ // to check which switches are active

if(PIND.B0==1) //display temperature on first line

Lcd\_Out(1, 1, lcd\_up);

else

{unsigned char temp[0x10]; // else display empty lines

memset(temp, ' ', 0x10);

Lcd\_Out(1, 1, temp);}

if(PIND.B1==1) //same as above

Lcd\_Out(2, 1, lcd\_down);

else

{unsigned char temp[0x10];

memset(temp, ' ', 0x10);

Lcd\_Out(2, 1, temp);}

if(PIND.B2==0) //if buzzer noise switch not active

{

buzzer\_2 = 0; // make buzzer2 0 for future

if(buzzer\_1) // even if it should have been make noise

{((TIMSK) &= ~(1 << (TOIE2))); // disable timer2

TCNT2 = 0; //load 0 to timer 2

((PORTC) |= (1 << (1))); //portc1 is 1

buzzer\_1 = 0;}} //make buzzer1 0 for future

else if(!buzzer\_1 && heat\_value > 30 && !buzzer\_2) //so prev buzzers are 0 and heat > 30

{((TIMSK) |= (1 << (TOIE2))); //enable timer2 overflow interrupt

((PORTC) &= ~(1 << (1)));

buzzer\_1 = 1; //made noise become 1

buzzer\_2 = 1;}

if(PIND.B3==0) //if led display pin inactive

PORTA = 0; // no display

else

PORTA = my\_led; //else display 1 bit for each 12.5 C

}

void restart\_timer(){

sum\_twos = 0; //make it 0

TIMSK &= ~(1 << TOIE2); // Timer2 overflow interrupt enable

TCNT2 = 0; //timer2 is again 0

PORTC |= 1 << 1; // make PORTC.1 to 1 again

buzzer\_1 = 0; // buzzer\_1 will become 0 so it will not start

}

void initialize\_ports\_and\_registers(){

DDRA = 0xFF; // A for led outputs

DDRC = 0x02; // C1 connection for buzzer

DDRD = 0xF0; // D7 and D6 for motor connection output

UCSR0C |= (1 << UCSZ01)|(1 << UCSZ00); // UCR0C init

UCSR0B |= (1 << RXEN0)| (1 << RXCIE0) | (1 <<TXEN0) ; // UCSR0B init

DDRB |= 1 << 4; //portb4 will be connected L293D(motor entegre) EN1

PORTD &= ~(1 << (7)); //portd4 will be connected L293D(motor entegre) IN2

PORTD |= 1 << 6; //portd4 will be connected L293D(motor entegre) IN1

PORTC |= 1 << 1; // portc1 will be connected to buzzer

Lcd\_Init(); // lcd init

Lcd\_Cmd(\_LCD\_CLEAR); // clear

Lcd\_Cmd(\_LCD\_CURSOR\_OFF); // if i dont write this it is seen like cloudy

UBRR0L = 0x40; // baud rate = fclock/(16\*UBRR) -1 for async mode, normal speed

UBRR0H = 0; // UBRR = 10^7/(9600\*16)-1

TCCR0 |= (1 << CS02) | (1 << CS00) | (1 << WGM00) | (1 << COM01); //prescale 1024 normal mode

OCR0 = fan\_speed \* 2.55;

TCCR2 |= 1 << CS22; //presecaler 128, normal mode

ADCSRA = 0xEF; // prescelar 64 ,no ADIF and ADPS0, free run mode

}

void enable\_interrupt(){//enable interrupt

SREG\_I\_bit = 1;

}

void main() {

initialize\_ports\_and\_registers(); //call function

enable\_interrupt(); // call function

send\_string("Enter S for % fan speed and T for temperature:"); // send this to Terminal for 1 time

while(1) {

//just wait

if(sum\_twos >= 0x92) { // if it reaches 2 second restart

restart\_timer();

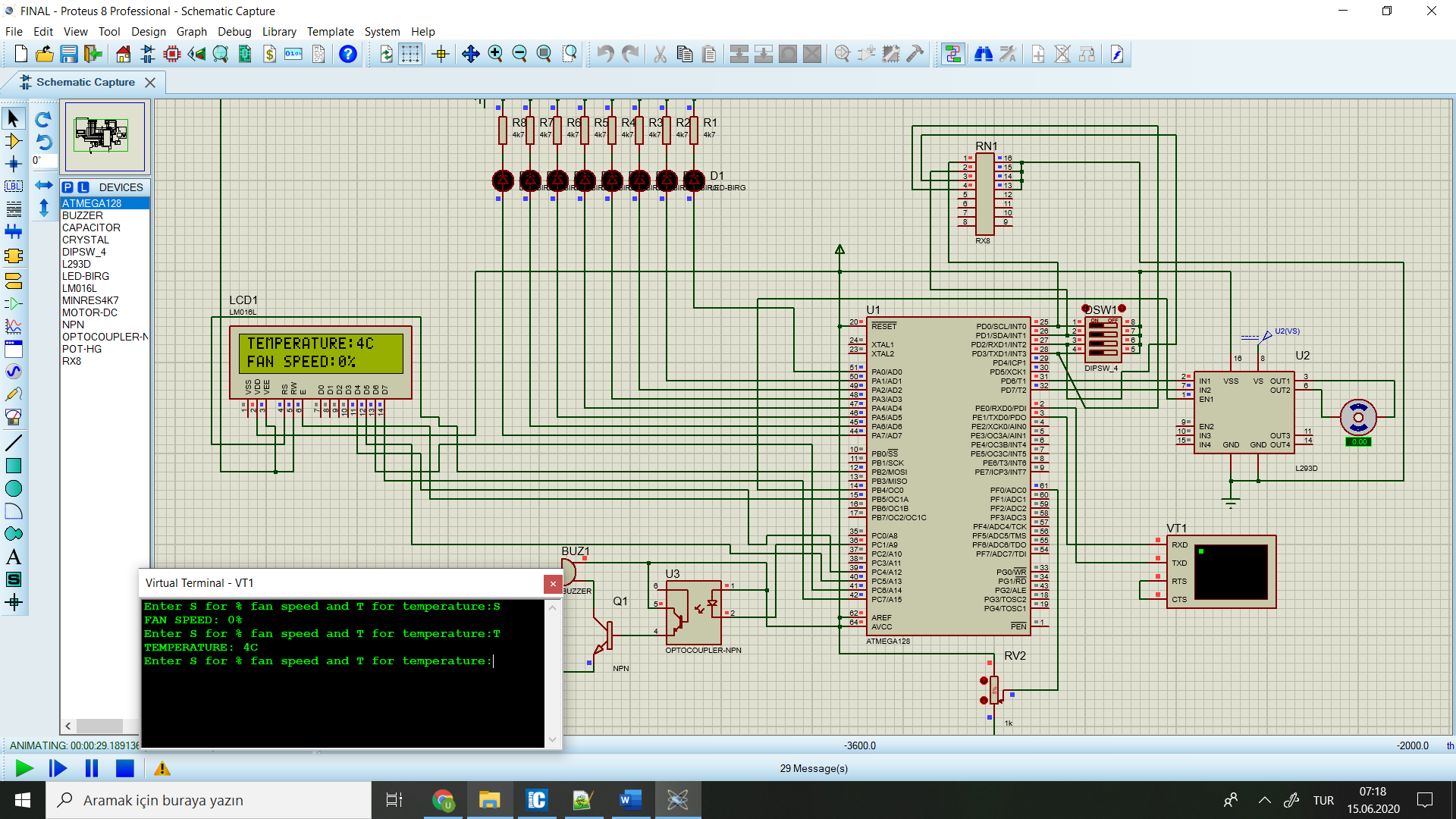
}

}

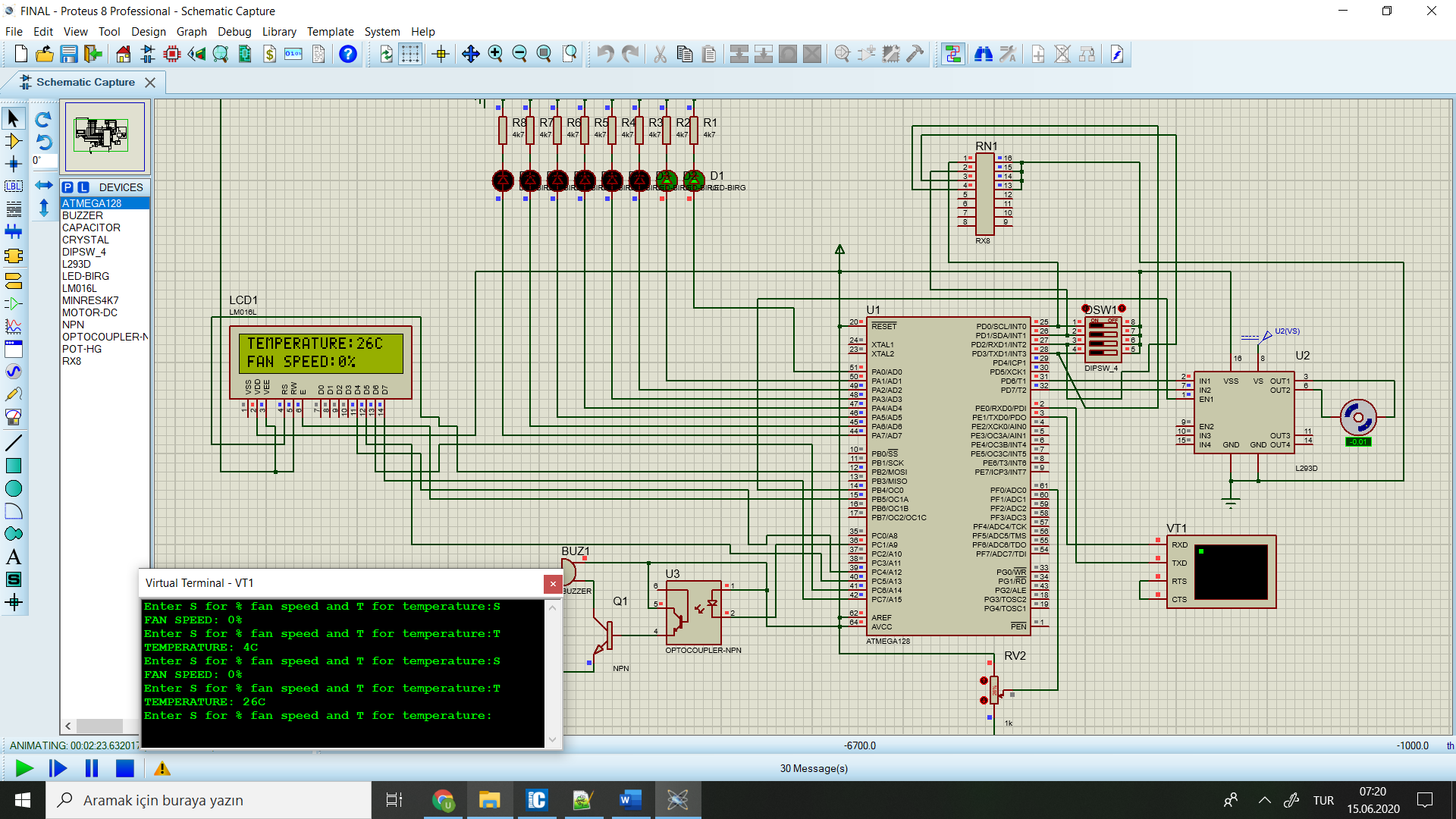
}

**Demonstration**

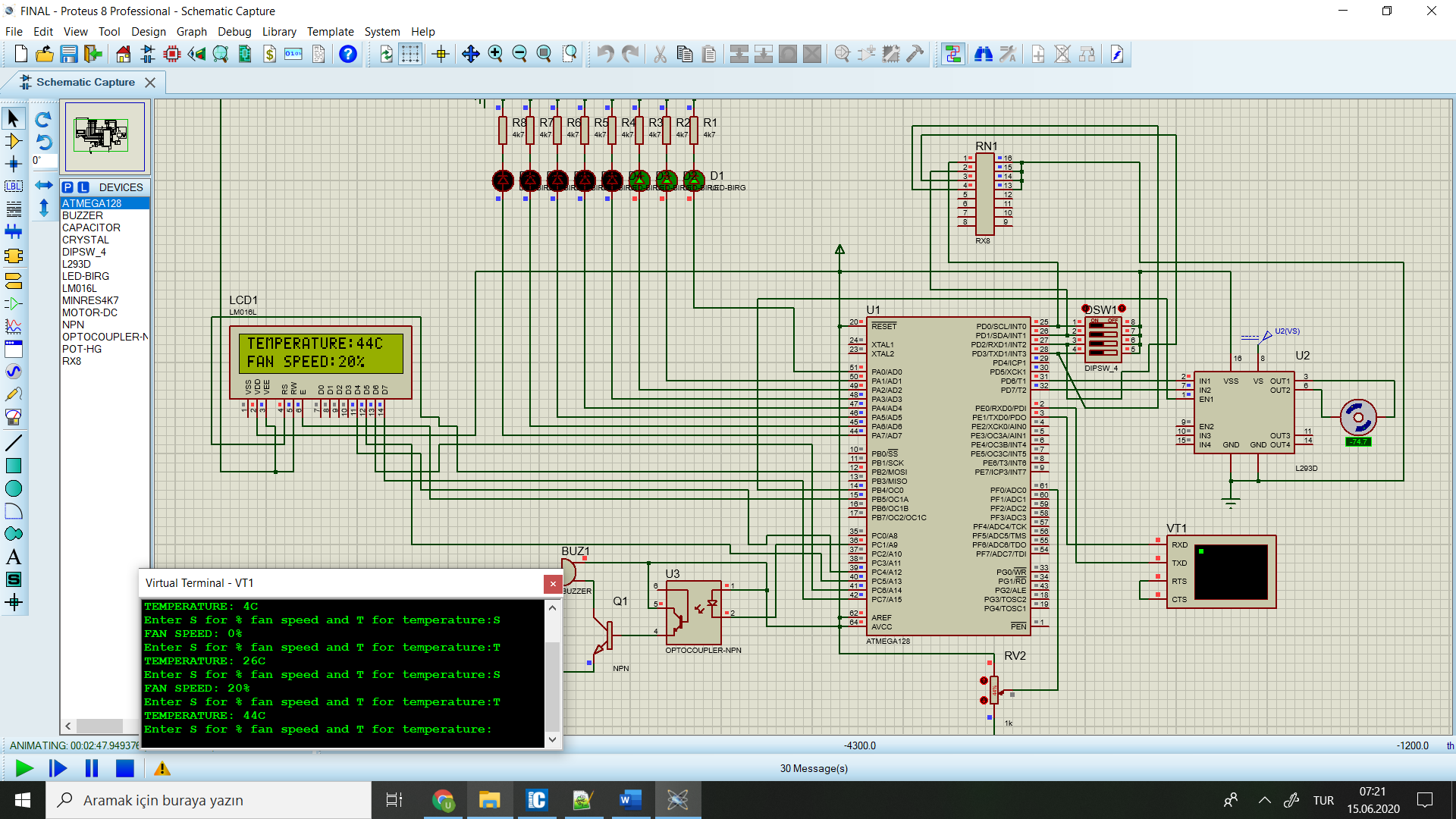
4C



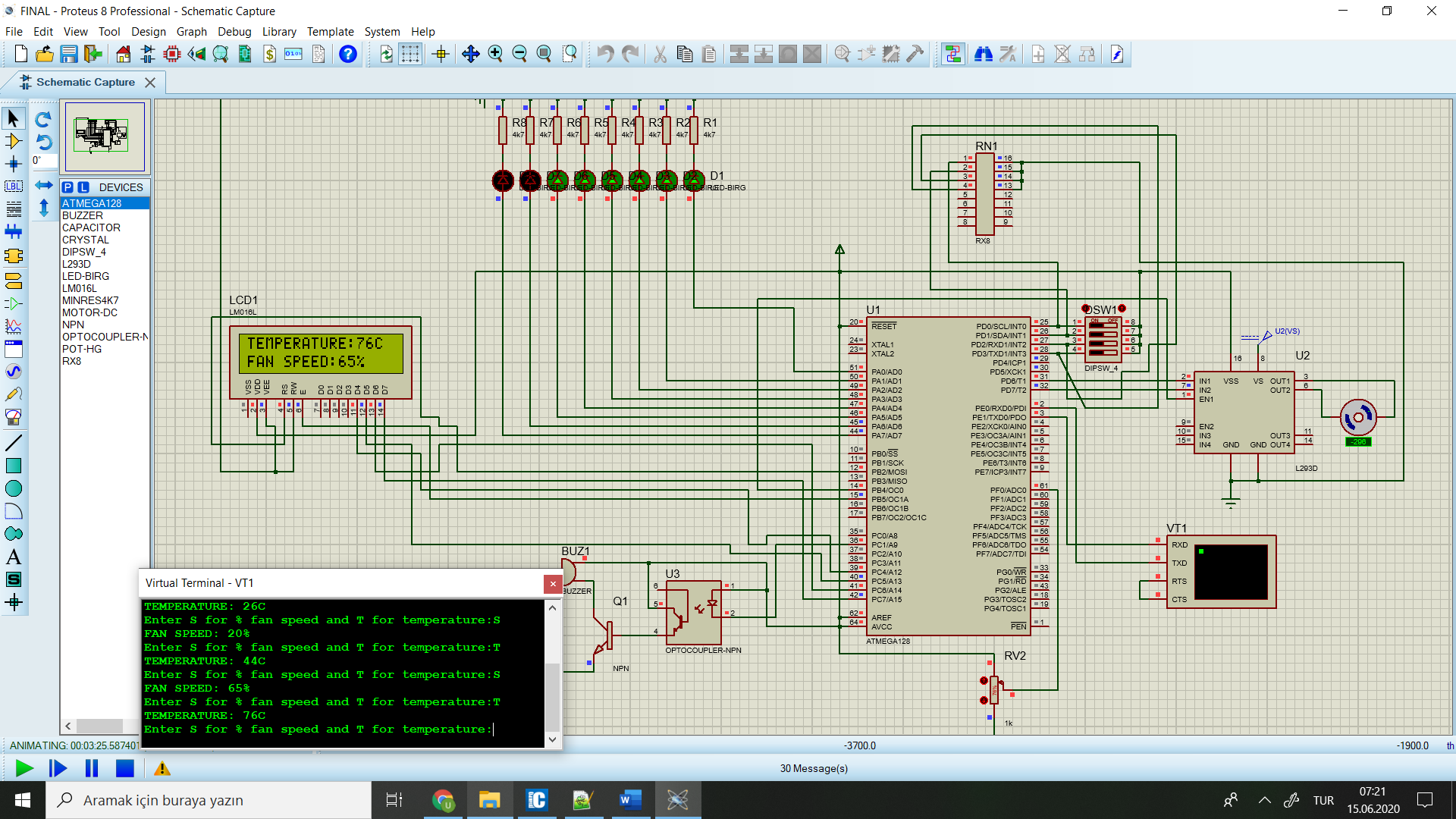
26 C



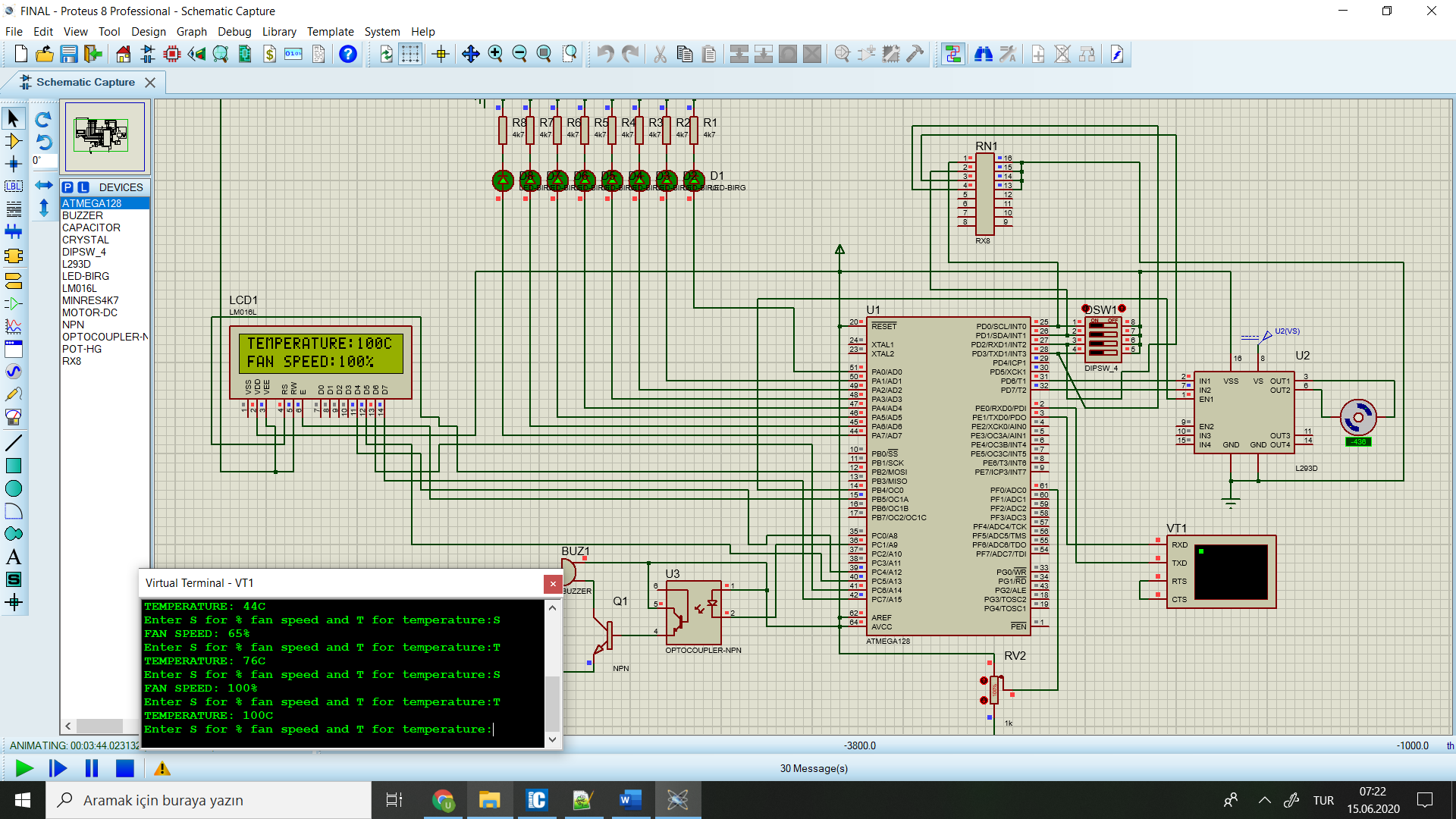
44 C



76C



100 C (Highest)



Capture for wrong input in terminal

