

*University of Khartoum
Faculty of engineering
Department of Electrical and Electronic
Engineering
4th Grade project*

Microcontroller Systems Design

GSM Based Temperature Monitoring System

Project Report

Supervised by:

Dr. Sarah Omer Mahjoub

Prepared by:

Ahmed Abdelhameed Ahmed 144008

Mohammed Eltigani Alsharief 144072

Amr Abdallah Ali 144047

Almahi Alnoor Ahmed 114103

Anas Mohammed Abdelghani 144019

Objectives:

To design a system that monitors the temperature of a room and sends SMS in one of three cases:

- The temperature exceeds a certain point specified by the user (set point_high).
- The temperature decrease to a certain point also specified by the user (set point_low).
- The temperature rise rate exceeds a certain rate also specified by the user (rate threshold).

1. Functional Requirements:

The system allows the user to specify both the Temperature Threshold (high and low point) and Temperature Rate Threshold.

The resolution with which Temperature Threshold can be specified is at most 2°C or less. The resolution with which the Temperature Threshold Rate is set to be 5°C/min or less. The range in which Temperature Rate threshold exists is about [35°C - 90°C] or more.

The main Function of the project is built on how to combine individual components that have a specified job together with the atmega32 to achieve its overall goal.

The combined system should satisfy these requirements:

I. Allow the user to specify the limits:

The system shall prompt user for required data entry, and allow the user to input the data.

This is done by both the keypad and the LCD.

The LCD shows what is wanted and the user writes the input using the keypad.

The system shall allow the user to specify these limits:

- Temperature threshold (high and low point).

- Temperature rate threshold.

II. Monitor the temperature continuously:

The system shall monitor the temperature of the environment through the attached temperature sensor continuously, unless the system is reset or power is lost.

III. Send an SMS to the user when the temperature is out of range or if the rate exceeds limits:

The system shall send an SMS to the user if one of the following conditions occur after each temperature reading:

- The current temperature is lower than set point_low
- The current temperature is higher than set point_high
- The change rate higher than the rate threshold.

2. Technical Requirements:

a) Microcontroller:

The following features are needed from the microcontroller:

- 1) ADC unit for temperature measurements.
- 2) USART interface for serial communication.
- 3) Enough IO pins for keypad and sensor connections. (8 pins for the keypad, 1 pin for the sensor).
- 4) Supported by AVR-GCC.

The Atmega32 AVR microcontroller is used.

b) Temperature Sensor:

The LM35 temperature sensor is used (the temperature range of LM35 [-55°C to +155°C]).

c) GSM Module:

It should support sending SMS through the USART interface using the 3GPP TS 27.005 Communication Standard.

The SIM900A is used in the simulation and the SIM800C is used in the laboratory.

d) LCD:

It should support the 16x2 mode, and 8-bit interfacing.

The LM016L is used.

e) Keypad:

It should be a 4x4 keypad.

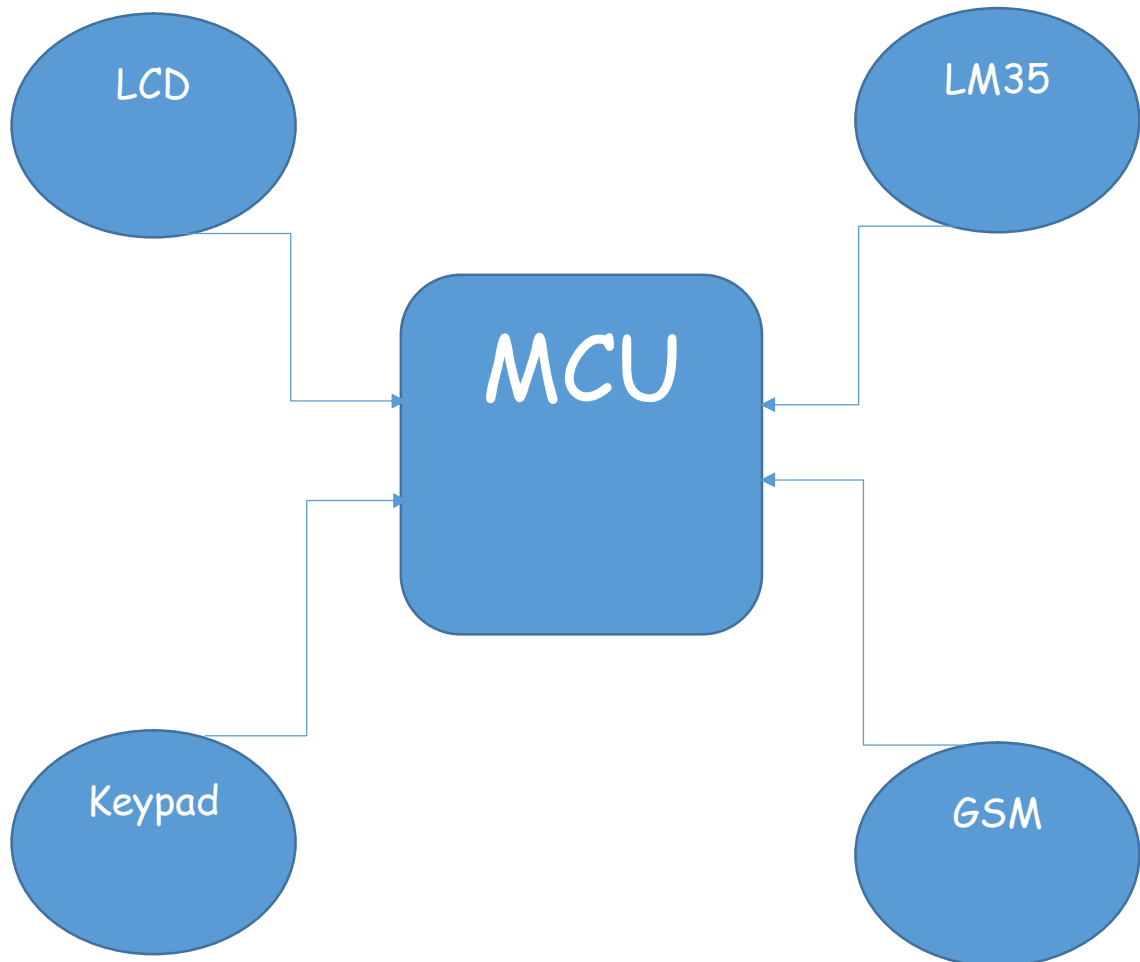
f) Power Supply.

g) Breadboard & Wires.

3. Design of project:

The system consists of 5 parts: a temperature sensor (LM35), a 4x4 keypad, an atmega32 MCU, an LCD and a GSM module in addition to a breadboard.

Basic system architecture is as follows:



1. Hardware:

a) Configuration:

1. Atmega32:

* Internal clock frequency of 8MHz.

* Disabled interrupts.

* Input pins:

PA0: for temperature sensor (ADC).

PB [3:0]: for keypad columns.

* Output pins:

PB [7:4]: for keypad rows.

PC [7:0]: for LCD data lines.

PD [7:5]: for LCD control pins.

PD1: for GSM receiving pin (TXD).

2. Atmega32 USART module:

Baud rate of 9600 bps.

8 bit characters.

1 stop bit.

No parity.

3. Atmega32 ADC module:

Channel selection: single ended channel 0

Prescaler: clk/128.

Left adjusted output.

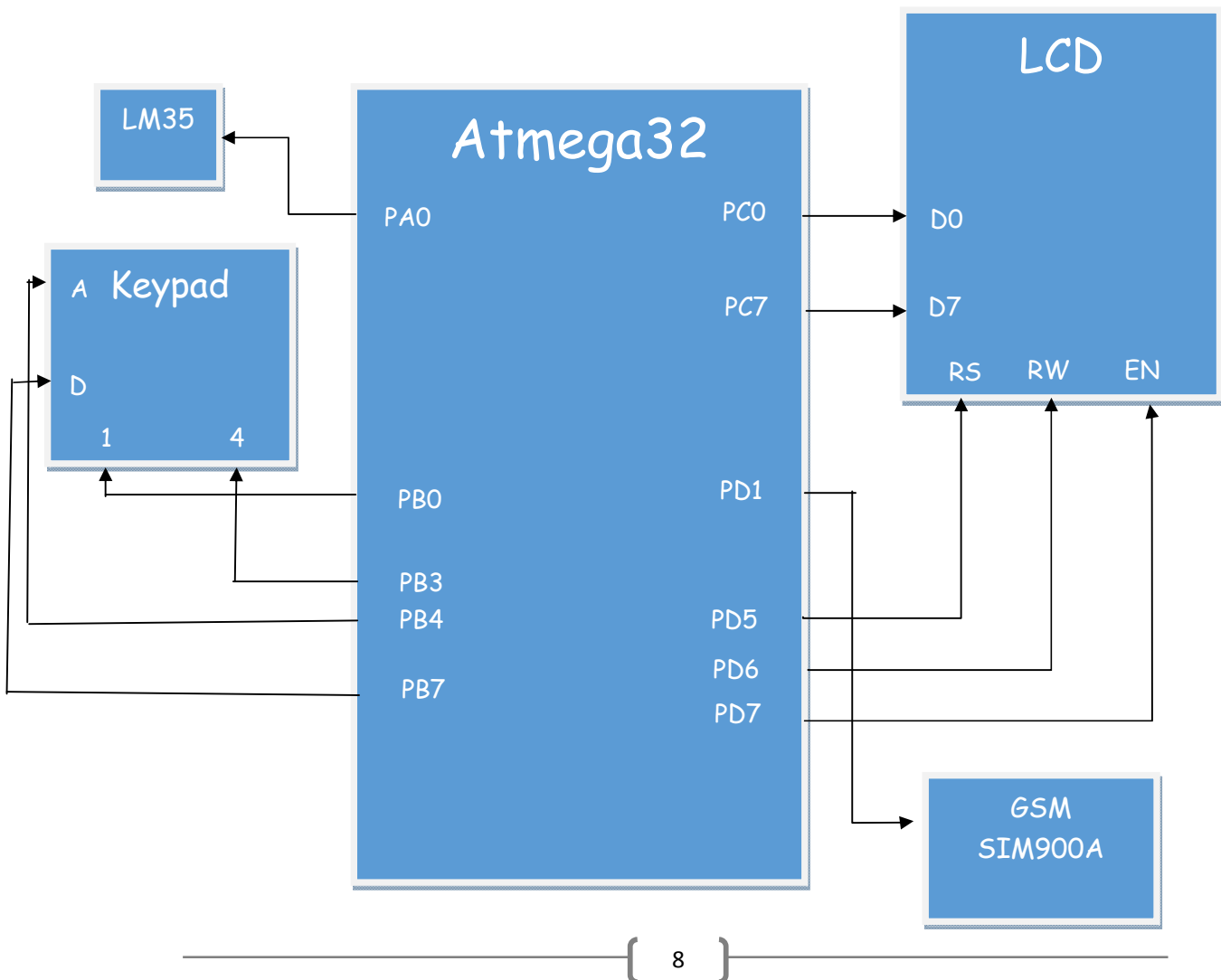
Internal voltage reference of 2.56 V.

4. LCD:

16x2 mode.

8 bit data wires.

b) Connection diagram:



2. Software:

On reset, the first step is to get the temperature and change rate limits from the user using the LCD to prompt the user and the keypad to take input from the user.

With the limits specified, the monitor loop begins, and it consists of the following steps:

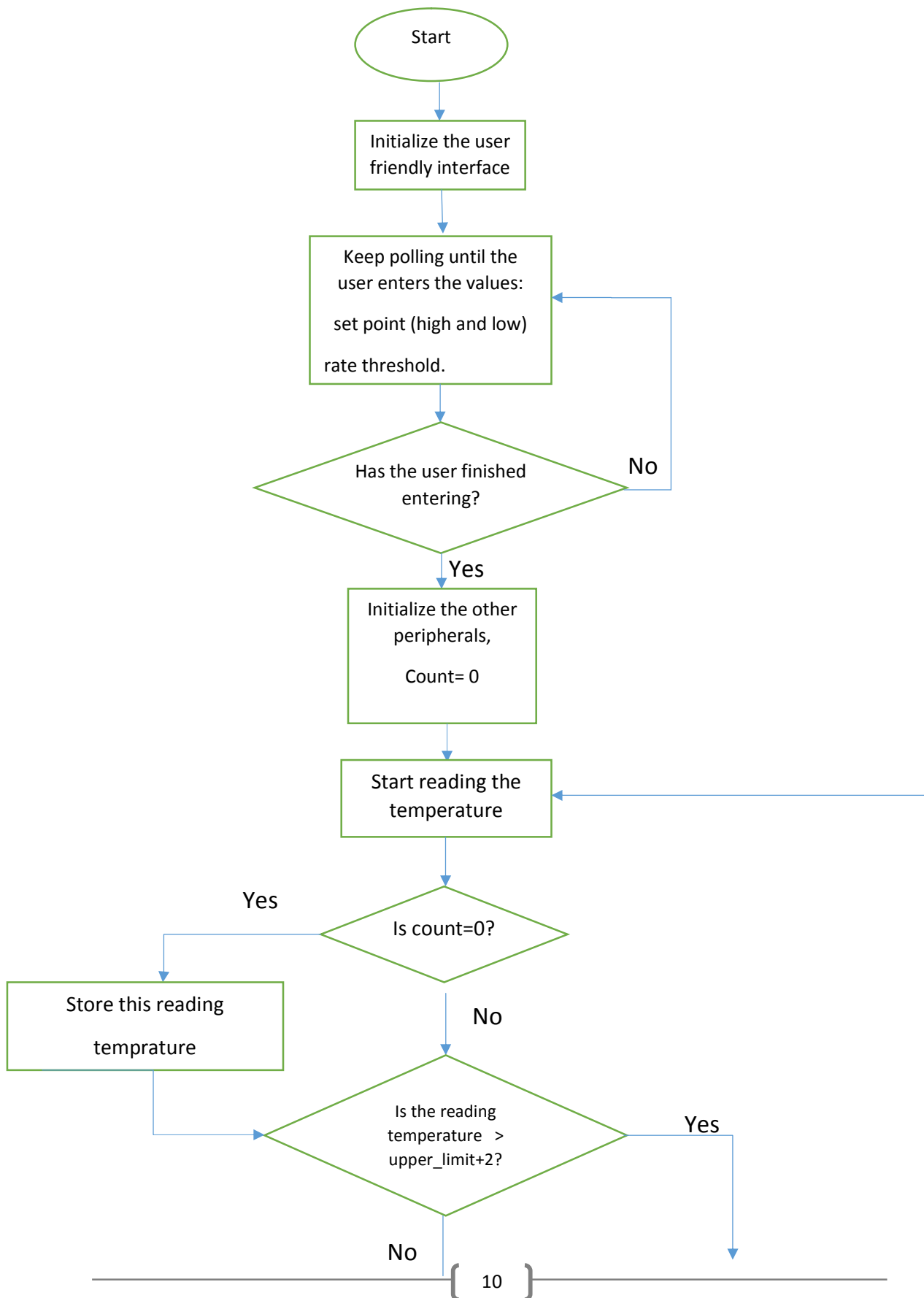
1. Initialize peripherals.
2. Take the temperature reading from the sensor using the ADC.
3. If the temperature is not within the limits, send an SMS to the user.
4. If this is the first reading, store this reading , go back to step 2 and wait for the next reading.
5. After each minute:
 - if the absolute value of the difference between the stored and current temperature is larger than the rate limit, send an SMS to the user. ($|previous - current| > ratelimit$).
 - change the stored temperature to the current temperature.

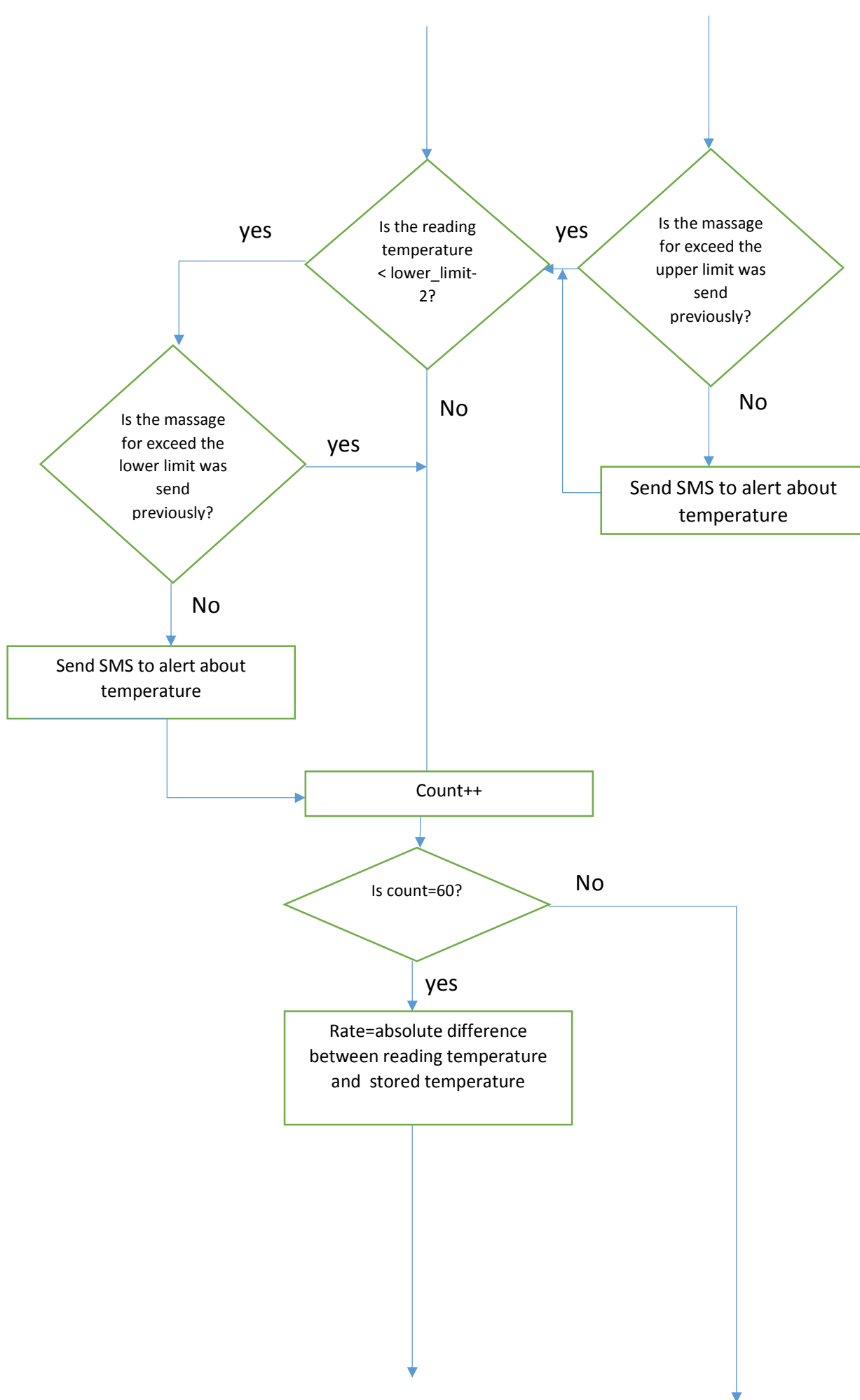
SMS Algorithm:

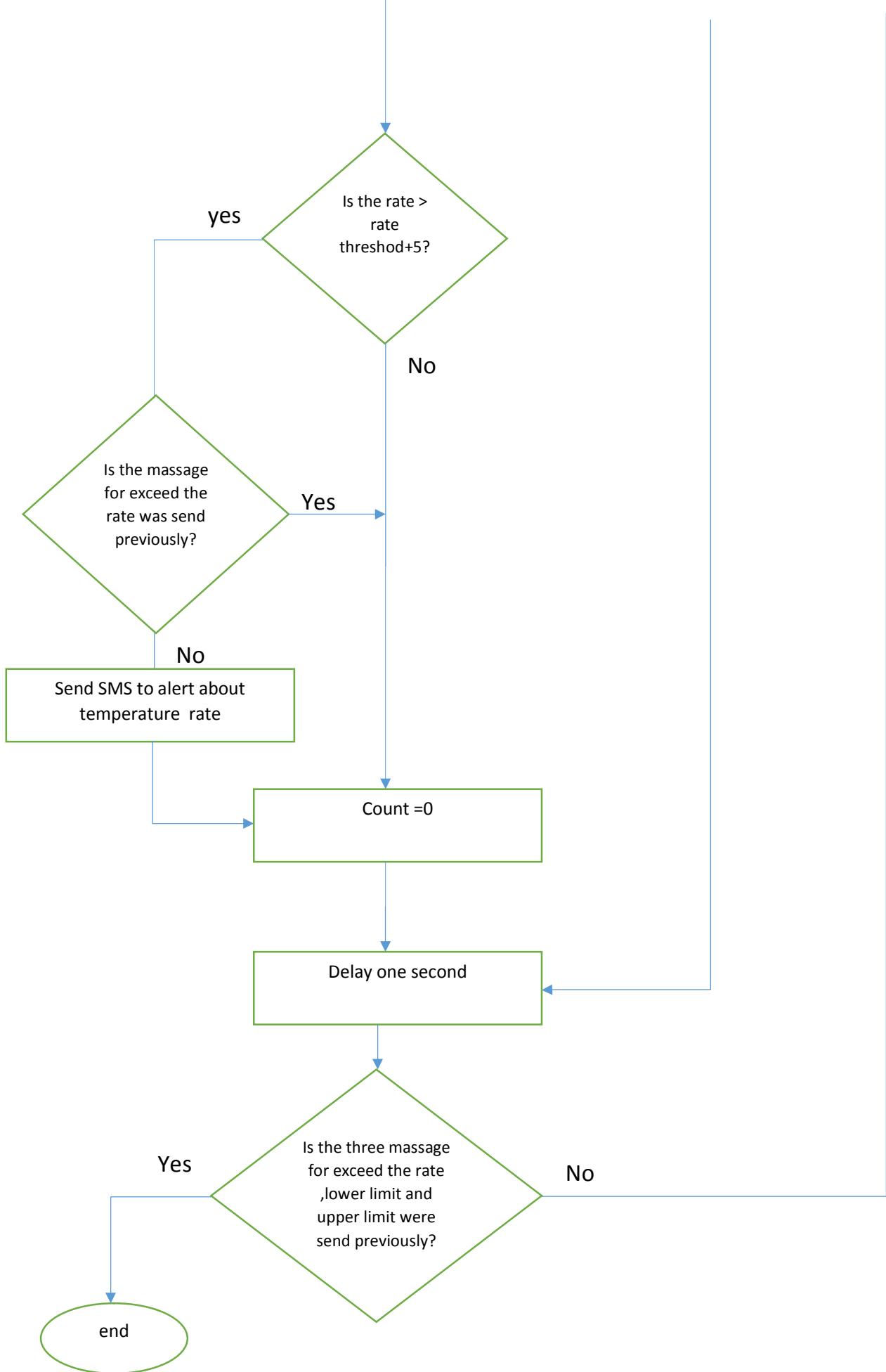
We send the SMS using the USART interface to the SIM900A by the following steps:

1. Send “AT\r” using the USART interface to the SIM900A to check if GSM module is ready or not. SIM900A send “ok” if it is ready.
2. Define the mode of the message by sending “AT+CMGF=1\r” to the SIM900A(SMS text mode).
3. Send “AT+CMGF= “{Phone Number}” ” to the SIM900A, followed by a carriage return ('\r').
4. Wait for response containing the character '>'.
5. Send the SMS text serially, followed by (Ctrl+Z).
6. Wait for confirmation from the SIM900A.

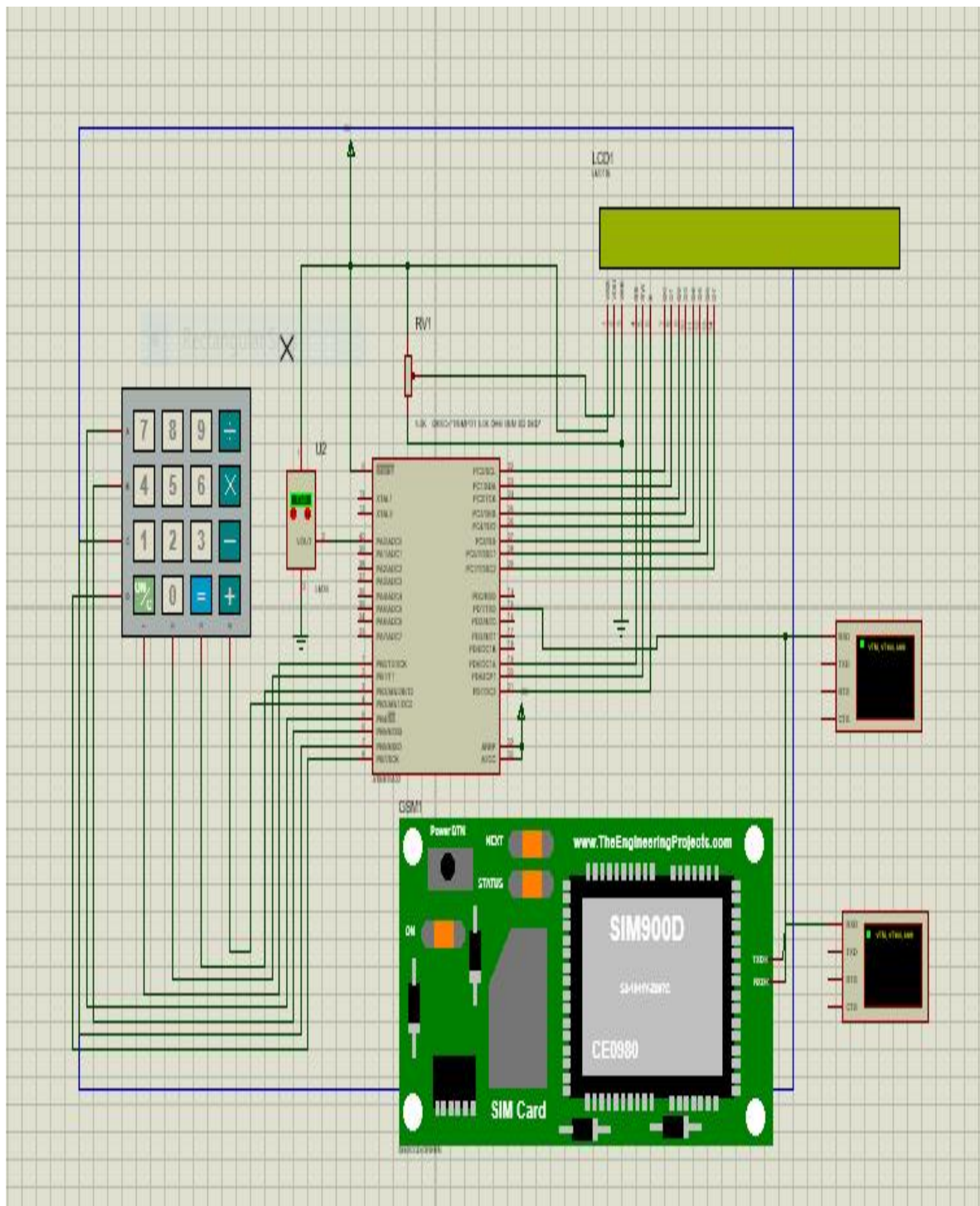
4. Dataflow diagram:







5.The implementation:



The steps to connect and implement the AVR connections are:

1. Connect the Atmega32 to the breadboard after the code burned to the chip (in lab) or upload the .hex file (in PROTEUS) and adjust the frequency (in our case it is 8 MHZ)
2. Connect one pin of the LM35 to PA0/ADC0 pin of the Atmega32 that represents channel zero of the ADC peripheral, and connect the other two pins of LM35 to 5 volts DC source and ground.
3. Connect the eight pins of the 4x4 keypad to PORTB of the Atmega32 with the following sequence:
 - PB0/T0/XCK is connected with pin 1 of the keypad.
 - PB1/T1 is connected with pin 2 of the keypad.
 - PB2/AIN0/INT2 is connected with pin 3 of the keypad.
 - PB3/AIN1/OC0 is connected with pin 4 of the keypad.
 - PB4/SS is connected with pin A of the keypad.
 - PB5/MOSI is connected with pin B of the keypad.
 - PB6/MISO is connected with pin C of the keypad.
 - PB7/SCK is connected with pin D of the keypad.
4. Connect the GSM module to the Atmega32, the connection has two possibilities:
 - If the GSM is TTL compatible, then the serial connection could be achieved by directly connecting the TXD and RXD pins of the GSM to RXD and TXD of the Atmega32 respectively.
 - If the GSM is not TTL compatible there is another connection called RS232 that can connect the GSM indirectly to Atmega32 chip by an intermediate chip called MAX232, this chip has the ability to match between the RS232 logic and the TTL logic.
 - In our case the GSM chip is called SIM900D is both TTL compatible and RS232 compatible, so we can use the TTL compatible.
 - In case of PROTEUS the .hex file of the GSM should also be uploaded.

5. Connect the LCD (LM018L) to PORTC and the connection is as follow:

- D7 of the LCD is connected with PC7/TOSC2
- D6 of the LCD is connected with PC6/TOSC1
- D5 of the LCD is connected with PC5/TDI
- D4 of the LCD is connected with PC4/TD0
- D3 of the LCD is connected with PC3/TMS
- D2 of the LCD is connected with PC2/TCK
- D1 of the LCD is connected with PC1/SDA
- D0 of the LCD is connected with PC0/SCL
- E of the LCD is connected with PD4/OC1B
- RW of the LCD is connected with PD3/INT1
- RS of the LCD is connected with PD2/INT0

6. Connect the LCD to the 10K potentiometer and the connection is as follows:

- Connect the VEE to the variable leg of the potentiometer
- Connect one of the pins of the potentiometer to the ground and the VSS of the LCD to the ground too.
- Connect the other pin of the potentiometer to the 5 volts DC source and the VDD of the LCD also.

7. In case of PROTEUS connect the VIRTUAL TERMINAL tool to the RXD and TXD of the connection between the Atmega32 and the GSM, this done by connecting the RX pin of the virtual terminal to TXD line AND the RX pin of the other virtual terminal to the RXD line.

8. After the configurations then power on

9. There are many scenarios:

- If the temperature is out of range, then there will be a message that will be sent to the phone .

number to alert about the temperature is out of range.

- If the rate at which we change the temperature exceed a certain limit this will alert also.

6. Test and Experiments:

I. Unit testing (component testing):

- Atmega32 testing:

A simple blinking LED program using all ports (A, B, C and D). LEDs are connected to all 8 pins for all ports, at a time, all ports are off.

We change Atmega32 many times, and then we find one that gives us 2 ports are on and the others are off.

- Keypad testing:

A 4x4 keypad, continuity can be tested between any circuit connections, row and column. Testing leads are connected at the start & end terminal of the specific path, after placing the leads on their position the Multimeter will produce a continuous beep sound indicating that track is continuous and there is no breakage in the track.

- LCD test:

A simple program that prints 'A' character continuously was made to test the LCD (we change the LCD many times to find one that acts correctly).

- LM35 sensor:

The sensor was not found in the lab, so we use potentiometer instead of it.

- GSM module :

GSM module was not found in the lab, so we just output the message to the LCD

II. Integration testing (whole system):

- The LCD, keypad, and the potentiometer (instead of sensor) were all connected to the atmega32.

Supplying power to the atmega32, the prompt appeared on the LCD while waiting for the user to use the keypad to enter the set point, and rate threshold values. We entered random values and then we change the value of the potentiometer until the message appeared in the LCD.

7.Problems:

Simulation Problems:

*First: the message was not sent correctly by USART (sum of the characters was sent and the others were ignored). This problem was solved by increasing the delay after sending each character.

We find that the delay has a large effect in all programs.

*Second: GSM module did not respond correctly and sent unfamiliar characters. This problem was solved by changing the frequency of Atmega32 to 8 MHz.

Lab Problems:

*There is a lack of components (Every two or three groups share the same components).

*The components found turned out to be malfunctioning.

*The temperature sensor and the GSM module were unavailable so we used the potentiometer and LCD instead of them, respectively.

*Some of the pins of the Atmega32 were broken.

8.Code:

```
#define F_CPU 8000000UL

#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define key_prt PORTB
#define key_ddr DDRB
#define key_pin PINB

unsigned char keypad[4][4]=
{{'7','8','9','#'},{'4','5','6','#'},{'1','2','3','#'},{'*','0','#','#' }};
int setpoint_H;
int setpoint_L;

int Temperature1;
int Temperature2;
int rate;
int rate_threshold;
unsigned int count=0;

int num1=0;
int num2=0;
int num3=0;

void ADC_init(){
    ADCSRA = 0x87;
    ADMUX = 0xe0;
    DDRA = 0x00;

    return;
}

void usart_init(){
    UCSRC = (1<<URSEL) | (1<<UCSZ1) | (1<<UCSZ0);
    UCSRB = (1<<TXEN) | (1<<RXEN);
    UBRR1 = 0x33;
}

unsigned char pressedKey(void)
{
    unsigned char colloc,rowloc;
    DDRB=0xff;
    key_ddr=0xf0;
    key_prt=0xff;
    while(1){
        do
        {
            key_prt &=0x0f;
            colloc=(key_pin & 0x0f);
```

```

    } while(colloc!=0x0f);

    do{

        do
        {
            _delay_ms(20);
            colloc=(key_pin& 0x0f);

            } while (colloc==0x0f);
            _delay_ms(20);
            colloc=(key_pin& 0x0f);

        }while (colloc==0x0f);

        while(1){

            key_prt =0xef;
            colloc=(key_pin & 0x0f);
            if(colloc != 0x0f){
                rowloc=0;
                break;
            }
            key_prt=0xdf;
            colloc=(key_pin & 0x0f);

            if(colloc != 0x0f){
                rowloc=1;
                break;
            }
            key_prt=0xbf;
            colloc=(key_pin & 0x0f);

            if(colloc != 0x0f){
                rowloc=2;
                break;
            }
            key_prt=0x7f;
            colloc=(key_pin & 0x0f);
            rowloc=3;
            break;
        }

        if(colloc == 0x0e)
            return (keypad[rowloc][0]);
        else if(colloc == 0x0d)
            return (keypad[rowloc][1]);
        else if(colloc == 0x0b)
            return(keypad[rowloc][2]);
        else
            return (keypad[rowloc][3]);
    }
}

void LCD_command(unsigned char temp){

    PORTC = temp; //screen lines connected to port c
    PORTD = 0b00010000; //E=1,RS=0,WR=0
    _delay_us(2); //High to low pulse to latch command
    PORTD = 0b00000000;

```

```

        _delay_us(100);

        return;
    }

    void LCD_data(unsigned char temp){

        PORTC = temp;
        PORTD = 0b00010100;//RS=1
        _delay_us(2);
        PORTD = 0b00000100;//H TO L PULSE
        _delay_us(100);

        return;
    }

    void LCD_init(){

        DDRC = 0xff;
        DDRD = 0xff;
        PORTD = 0x00;//OUTPUT 00X00
        _delay_ms(2);
        LCD_command(0x38);//8 BIT ,5*7 MATRIX
        _delay_ms(2);
        LCD_command(0x0f);//display on cursor blinking
        _delay_ms(2);
        LCD_command(0x01);// clear lcd screen
        _delay_ms(2);
        LCD_command(0x06);// INCREMENT CURSOR TO RIGHT

        return;
    }

    void clearLCD(){

        LCD_command(0x01);// clear lcd screen
        _delay_ms(2);
        LCD_command(0x02);//return home 0,0
        _delay_ms(2);
        LCD_command(0x0f);//display on cursor blinking
        _delay_ms(2);
        LCD_command(0x06);//INCREMENT CURSOR TO RIGHT AFTER ENTERING A CHAR
        _delay_ms(2);

        return;
    }

    void gotoxy(unsigned char x, unsigned char y){    //X Column,starts from 1 ,Y row
    starts from 1

        unsigned char firstCharAdr[] = {0x80, 0xC0};    //80,C0 beginning of 1st and
    2nd rows respectively
        LCD_command(firstCharAdr[y-1] + x - 1); //0,0
        _delay_ms(2);
        LCD_command(0x0f);//blinking cursor
        _delay_ms(2);
        return;
    }

    void LCD_string(char* str){

```

```

        int len = strlen(str), i;
        for(i = 0 ; i < len ; i++){
            LCD_data(str[i]); //display string on lcd
            _delay_ms(2);
        }
        return;
    }
    void usart_transmit(char data){
        while (UCSRA & (1<<UDRE)==0); //ucsra empty
        UDR = data;
        _delay_us(500);
    }

    void sendData(char* str){
        int len = strlen(str);
        for (int i = 0; i < len; i++){
            usart_transmit(str[i]); //send string to usart
            _delay_ms(20);
        }
    }

    void check_gsm(){
        sendData("AT\r");
        _delay_ms(50);
    }

    void text_mode(){
        sendData("AT+CMGF=1\r"); //sms text mode
        _delay_us(500);
    }

    void message(char* sms,int Temperature){

        sprintf(sms, "current Temperature is %dC ", Temperature); // I added a
        space after %d
        sendData(sms);
        LCD_string(sms); //print in LCD
        usart_transmit(13); //enter
        _delay_ms(50);
    }

    void rate_message(char* msg, int rate){
        sprintf(msg, "Temperature rise rate is %dC/min ",rate); // I added a space
        after %d
        sendData(msg);
        LCD_string(msg);
        usart_transmit(13);
        _delay_ms(50);
    }

    void send_cmgs(){ //message for threshold
        sendData("AT+CMGS=\"+249128884414\"\r");
        _delay_us(500);
    }

    void send_SMS(){
        char sms[50];

```

```

        text_mode();
        send_cmgs();
        _delay_us(1000);
        message(sms, Temperature1);
        usart_transmit(26); // ctrl_Z
        _delay_us(1000);
    }

void rate_SMS()
{
    char msg [50]; //rate message holder
    text_mode();
    send_cmgs();
    rate_message(msg, rate);

    usart_transmit(26); //ctrl z
    _delay_us(1000);
}

int main(){

    LCD_init();
    _delay_ms(100);

    char* setpoint_prompt_H = "Enter high setpoint: ";
    char* setpoint_prompt_L = "Enter Low setpoint: ";
    char* rate_threshold_prompt = "Enter rate: ";

    char tmp1 ;
    char tmp2;

    int i = 0;
    int j = 0;
    int k=0;

    char str_setpoint_H[4];          //store the value of setpoint from keypad
    char str_setpoint_L[4];
    char str_rate_threshold[4];      //rate message

    //=====

    LCD_string(setpoint_prompt_H);
    gotoxy(1,2);
    LCD_command(0x0f);
    _delay_ms(500);

    while((tmp1 = pressedKey()) != '#'){

        if(tmp1 != '*'){
            str_setpoint_H[i] = tmp1;
            i++; // no of setpoin entries
            LCD_data(tmp1);
        }
        else if(tmp1 == '*' && i > 0){
            i--;
            LCD_command(0x10); //cursor to left
            _delay_ms(20);
            LCD_data(0);
            _delay_ms(20);
            LCD_command(0x10);
        }
    }
}

```

```

        _delay_ms(20);
        LCD_command(0x0f);
        _delay_ms(20);
    }
}

clearLCD();
LCD_string(setpoint_prompt_L);
gotoxy(1,2);
LCD_command(0x0f);
_delay_ms(500);

while((tmp1 = pressedKey()) != '#'){

    if(tmp1 != '*'){
        str_setpoint_L[j] = tmp1;
        j++; // no of setpoin entries
        LCD_data(tmp1);
    }
    else if(tmp1 == '*' && i > 0){
        j--;
        LCD_command(0x10); //cursor to left
        _delay_ms(20);
        LCD_data(0);
        _delay_ms(20);
        LCD_command(0x10);
        _delay_ms(20);
        LCD_command(0x0f);
        _delay_ms(20);
    }
}

_delay_ms(15);
clearLCD();
_delay_ms(500);
LCD_string(rate_threshold_prompt);
gotoxy(1,2);
LCD_command(0x0f);

while((tmp2 = pressedKey()) != '#'){

    if(tmp2 != '*'){
        str_rate_threshold[k] = tmp2;
        k++;
        LCD_data(tmp2);
    }
    else if(tmp2 == '*' && j > 0){
        k--;
        LCD_command(0x10);
        _delay_ms(20);
        LCD_data(0);
        _delay_ms(20);
        LCD_command(0x10);
        _delay_ms(20);
        LCD_command(0x0f);
        _delay_ms(20);
    }
}

setpoint_H = atoi(str_setpoint_H); //convert to int

```

```

setpoint_L = atoi(str_setpoint_L);    //convert to int
rate_threshold = atoi(str_rate_threshold);
_delay_ms(20);

//=====
LCD_init();
_delay_ms(20);
usart_init();
_delay_ms(20);
ADC_init();

while(1){

    ADCSRA |= (1<<ADSC); //start conversion
    while ((ADCSRA & (1<<ADIF))==0);
    Temperature1 = ADCH;

    if(count==0)
        Temperature2=Temperature1;
    count++;

    if (Temperature1 >= setpoint_H +2)
    {
        if (num1==0 )
        {
            clearLCD();
            check_gsm();
            send_SMS();
            num1=1;
        }

    }

    if (Temperature1 <= setpoint_L -2)
    {
        if (num2==0 )
        {
            clearLCD();
            check_gsm();
            send_SMS();
            num2=1;
        }

    }

    //===== Temperature rate
    if(count==60){
        rate =abs( Temperature1 - Temperature2);
        if(rate >= rate_threshold +5)
        {
            if (num3==0)
            {

```



```

        clearLCD();
        check_gsm();
        rate_SMS();
        num3=1;
    }

    }
    count=0;
}
_delay_ms(1000);
ADCSRA |= (1<<ADIF);
if (num1==1 && num2==1 && num3==1)
{
    break;
}
}
return 0;
}

```

Components and Billing Outline(BO):


1-ATmega32 microcontroller:

		
AVR AVR ATmega Microcontroller IC 8-Bit 16MHz 32KB (16K x 16) FLASH 40-PDIP		
Price Break	Unit Price	Extended Price
1	5.62000	\$5.62

2-Keypad:

ebay

Hot Membrane Switch Keypad for Arduino 12 Key Keyboard 3*4 AVR..



Brand new: lowest price ⓘ

\$0.99

3-LCD:

Arducam 1602 16x2 LCD Display Module Based on HD44780 Controller Character White on Blue with Backlight for Arduino

Amazon's **Choice** for "16x2 lcd"

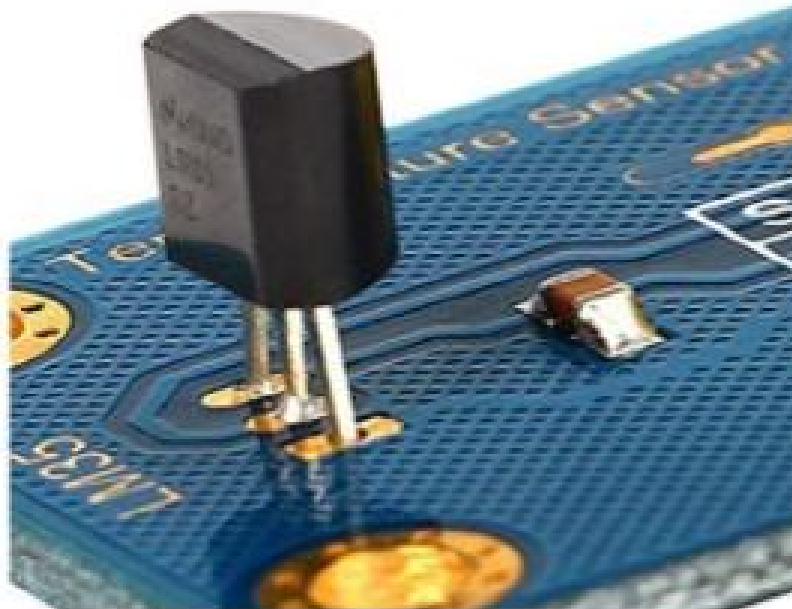


\$ **5**⁹⁹

4-LM35 Temperature Sensor:

ALSR

LM35 Temperature Sensor



\$ 16⁶⁶

5-Potentiometer:



\$ 5⁶³ ~~\$7.42~~ Save \$1.79 (24%)

Quantity: 1

6-GSM:



4 Frequency Development Board
GSM GPRS Module Wireless Data
Transmission SIM900

\$16.74

Total Bill:

$\$(5.62+.99+5.99+16.66+5.63+16.74)=\51.63