МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра **САП**

ЗВІТ

до виконання лабораторної роботи №7

На тему: ”Дослідження роботи давача температури LM35”

з курсу “Мікропроцесорні системи ”

Варіант - 4

Виконав:

Студент гр.ПП-31

Гаврилюк Н. О.

Прийняв:

Доцент кафедри САП

Головатий А.І.

ЛЬВІВ - 2025

Мета: ознайомитись з принципом роботи та зчитуванням даних давача температури LM35; закріпити навички виведення інформації на семигментний індикатор з використанням регістру зсуву 74HC595 та LCD- індикатор.

Теоретичні відомості

Серія давачів LM35 – це прецизійні інтегральні пристрої (інтегральні мікросхеми) призначені для вимірювання температури (Рис.1). Вихідна напруга давачів є лінійно-пропорційною до температури за шкалою Цельсія (до температурної шкали Цельсія). Давач LM35 має переваги над лінійними давачами температури відкаліброваних в Кельвінах, так як користувачу не потрібно віднімати велику константну напругу від вихідної, щоб отримати відповідну температуру в Цельсіях. LM35 не потребує зовнішнього калібрування і налаштування, щоб забезпечити точність ±¼ °C при кімнатній температурі і ±¾ °C на всьому температурному діапазоні від -55 °C до 150 °C.

LM35 є недорогою, надійною і достатньо точною мікросхемою (похибка вимірювання складає приблизно ±0,5 ºC). Низька вартість забезпечена завдяки налаштуванню і калібруванню на рівні напівпровідникової технології виготовлення. Низький вихідний імпеданс, лінійний виід, обов’язкове прецизійне калібрування дає змогу легко підключити давач до схем зчитування або управління. Пристрій потребує одного джерела живлення, споживає 60 мкА, мале самонагрівання менше 0,1°C в спокійному повітрі.

Пристрій LM35 вимірює температуру в діапазоні від −55°C to 150°C, а пристрій LM35C вимірює температуру в діапазоні від -40 °C до 110 °C (−10° з доведеною точністю). Давачі серії LM35 доступні в герметичному транзисторному корпусі TO, тоді як давачі LM35C, LM35CA, і LM35D є дступними в пластиковому транзисторному корпусі TO-92. LM35D давач доступний в 8-вивідному корпусі поверхневого монтажу і пластиковому корпусі TO-220.

Завдання

1. Реалізувати програму, яка виводить на LCD-індикатор значення температури з датчика LM35 та керує RGB світлодіодом. Якщо t >18°C, то світиться синій світлодіод; якщо t ≥ 25°C, то світиться зелений світлодіод; якщо t ≥ 33°C, то світиться червоний світлодіод; якщо t ≤ 18°C, то RGB світлодіод не світиться.

Код для Arduino IDE:

```
#include <LiquidCrystal.h>
```

```
#define LM35_pin A0
#define R_pin A1
#define G_pin A2
#define B_pin A3

// Підключення LCD: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int temp;

void RGB(int temp)
{
   if(temp>=33)
   {
      digitalWrite(R_pin,HIGH);
      digitalWrite(G_pin,LOW);
      digitalWrite(B_pin,LOW);
   }
   else if(temp>=25)
   {
      digitalWrite(R_pin,LOW);
      digitalWrite(G_pin,HIGH);
      digitalWrite(B_pin,LOW);
   }
   else if(temp>18)
   {
      digitalWrite(R_pin,LOW);
      digitalWrite(G_pin,LOW);
      digitalWrite(B_pin,HIGH);
```

```
    }
    else
    {
      digitalWrite(R_pin,LOW);
      digitalWrite(G_pin,LOW);
      digitalWrite(B_pin,LOW);
    }
}

void setup() {
    lcd.begin(16, 2);
    analogReference(INTERNAL); // встановлюємо опорну напругу 1.1 В
    pinMode(LM35_pin, INPUT);
    pinMode(R_pin, OUTPUT);
    pinMode(G_pin, OUTPUT);
    pinMode(B_pin, OUTPUT);
}

void loop() {
    int analogValue = analogRead(LM35_pin);
    temp = (analogValue * 1.1 * 100) / 1023; // правильне перетворення у °C
    RGB(temp);
    lcd.clear();
    lcd.print("Temp: ");
    lcd.print(temp);
    lcd.print(" C");
    delay(1000);
}
```

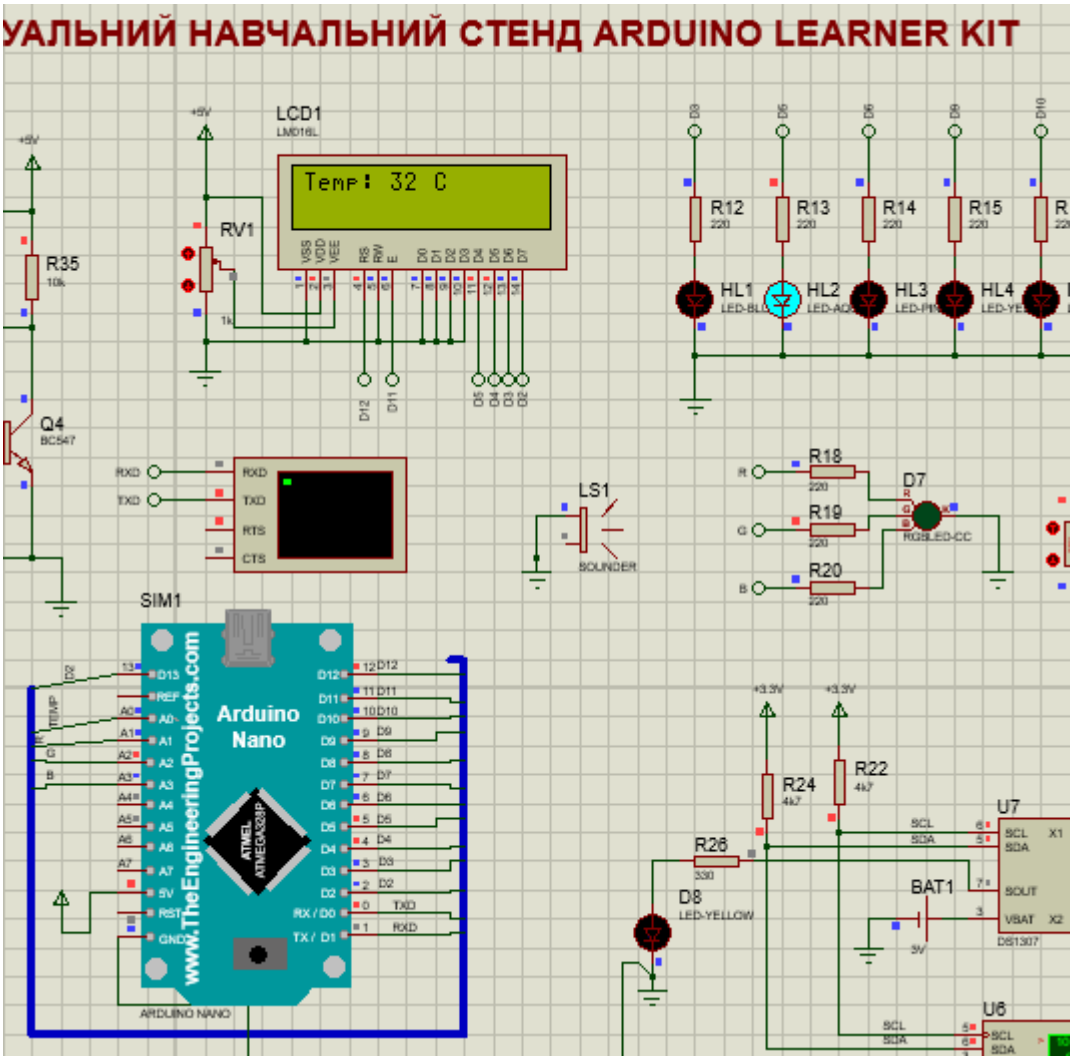


Рис. 1.Результат виконання.

Код для МК AVR:

```
#include "lcd.h"


void InitLCD()
{
_delay_ms(20);
PORTA = (1 << LCD_D4) | (1 << LCD_D5);
EnablePulseLCD();


PORTA = (1 << LCD_D5);
```

```
EnablePulseLCD();

CommandLCD(0x28);
CommandLCD(0x0C);
CommandLCD(0x06);
CommandLCD(0x01);
}

void EnablePulseLCD()
{
PORTA |= (1 << LCD_E);
_delay_ms(1);
PORTA &= ~(1 << LCD_E);
_delay_ms(1);
}

void CommandLCD(uint8_t command)
{
PORTA &= ~(1 << LCD_RS);
PORTA &= ~(1 << LCD_RW);

PORTA = (PORTA & 0x0F) | (command & 0xF0);
EnablePulseLCD();
PORTA = (PORTA & 0x0F) | (command << 4);
EnablePulseLCD();
_delay_ms(1);
}
```

```
void SendCharLCD(uint8_t ch)
{
PORTA |= (1 << LCD_RS);
PORTA &= ~(1 << LCD_RW);

PORTA |= (1 << LCD_E);
PORTA = (PORTA & 0x0F) | (ch & 0xF0);
PORTA &= ~(1 << LCD_E);

PORTA |= (1 << LCD_E);
PORTA = (PORTA & 0x0F) | (ch << 4);
PORTA &= ~(1 << LCD_E);
_delay_ms(1);
}

void SendStringLCD(char* data)
{
for (uint8_t i = 0; data[i] != '\0'; i++)
{
        if (data[i] == '\n')
        {
                GotoXY(0,1);
        }
        else
        {
                SendCharLCD(data[i]);
        }
}
```

```
}

void SendNumberLCD(uint16_t number)
{
char str[6];
itoa(number, str, 10);
SendStringLCD(str);
}

void ShiftLeftLCD(uint8_t n)
{
for (uint8_t i=0;i<n;i++)
{
        CommandLCD(0x1E);
}
}

void ClearLCD()
{
CommandLCD(0x01);
_delay_ms(2);
}

void ClearLineLCD(uint8_t line)
{
GotoXY(0, line);
for (int i = 0; i < 16; i++)
{
```

```
        SendCharLCD(' ');
}
GotoXY(0, line);
}

void GotoXY(uint8_t x, uint8_t y)
{
uint8_t address;

if (x >= 16 || y >= 2) return;

if (y == 0)
{
        address = 0x80;
}
else
{
        address = 0xC0;
}

address += x;

CommandLCD(address);
}
```

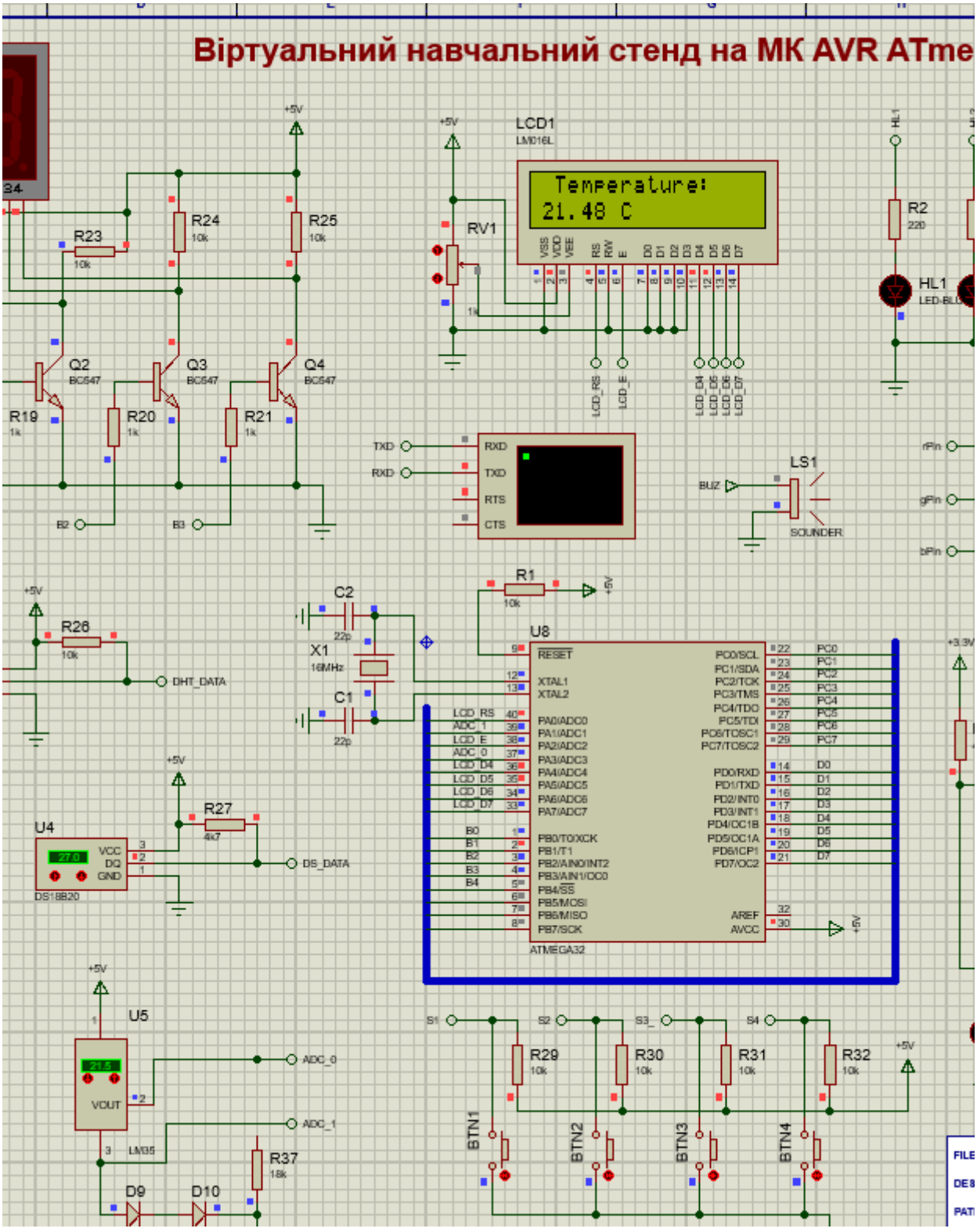


Рис. 2.Результат виконання.

2. Реалізувати програму, яка виводить на семисегментний індикатор значення температури з датчика LM35 та керує RGB світлодіодом. Якщо t >18°C, то світиться синій світлодіод; якщо t ≥ 25°C, то світиться зелений світлодіод; якщо t ≥ 33°C, то світиться червоний світлодіод; якщо t ≤ 18°C, то RGB світлодіод не світиться.

Код для Arduino IDE:

```
#include <LiquidCrystal.h>
#define LM35_pin A0
#define R_pin A1
#define G_pin A2
#define B_pin A3
int temp;

const int numeral[10] = {
  B11111100,//0
  B01100000,//1
  B11011010,//2
  B11110010,//3
  B01100110,//4
  B10110110,//5
  B10111110,//6
  B11100000,//7
  B11111110,//8
  B11110110,//9
};

// H,G,F,E,D,C,B,A
const int segmentPins[] = {13, 8, 7, 6, 5, 4, 3, 2};
const int nbrDigits = 4;

// digital 0 1 2 3
const int digitPins[nbrDigits] = {9, 10, 11, 12};

void RGB(int temp)
```

```
{
  if(temp>=33)
  {
    digitalWrite(R_pin,HIGH);
    digitalWrite(G_pin,LOW);
    digitalWrite(B_pin,LOW);
  }
  else if(temp>=25)
  {
    digitalWrite(R_pin,LOW);
    digitalWrite(G_pin,HIGH);
    digitalWrite(B_pin,LOW);
  }
  else if(temp>18)
  {
    digitalWrite(R_pin,LOW);
    digitalWrite(G_pin,LOW);
    digitalWrite(B_pin,HIGH);
  }
  else
  {
    digitalWrite(R_pin,LOW);
    digitalWrite(G_pin,LOW);
    digitalWrite(B_pin,LOW);
  }
}

void setup() {
```

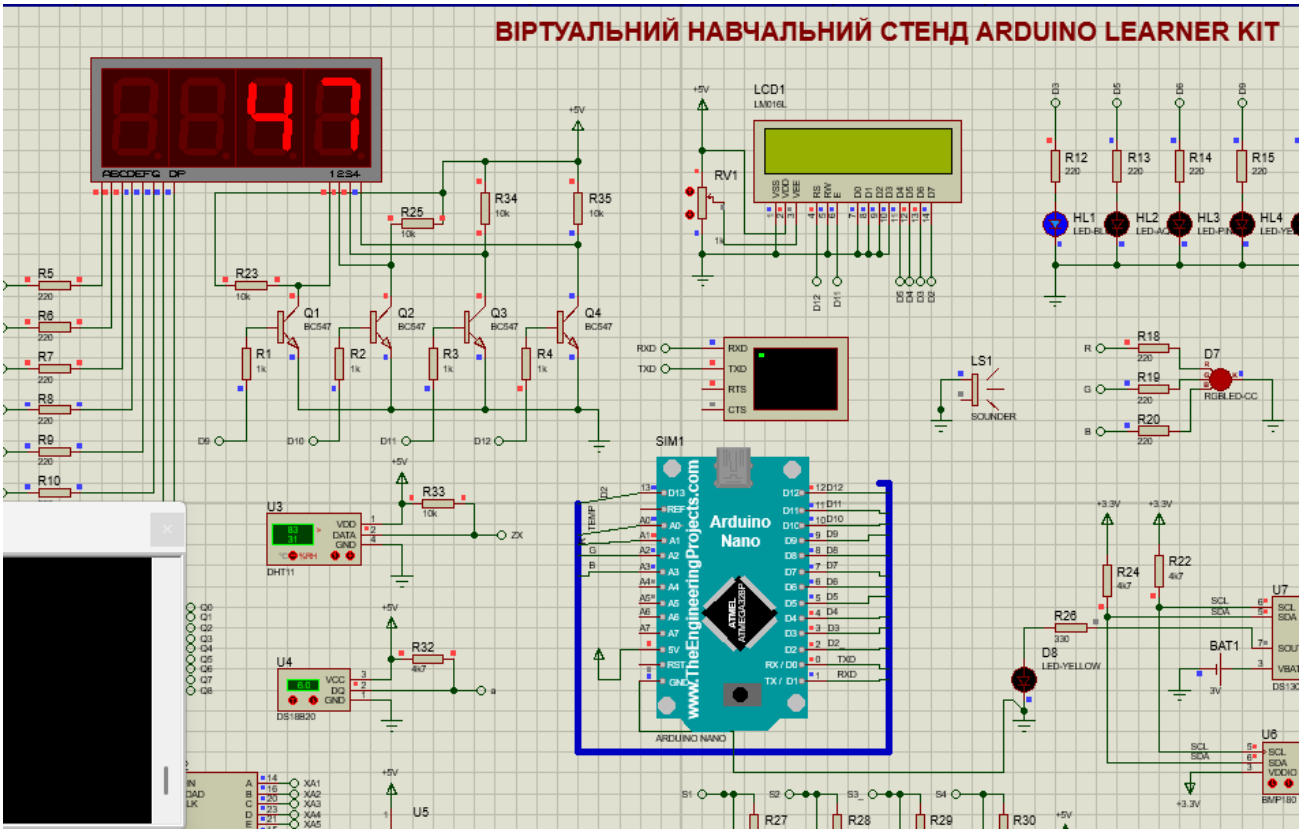
```
  for(int i = 0; i < 9; i++) {
    pinMode(segmentPins[i], OUTPUT);
  }
  for(int i = 0; i < nbrDigits; i++) {
    pinMode(digitPins[i], OUTPUT);
  }
  analogReference(INTERNAL); // встановлюємо опорну напругу 1.1 В
  pinMode(LM35_pin, INPUT);
  pinMode(R_pin, OUTPUT);
  pinMode(G_pin, OUTPUT);
  pinMode(B_pin, OUTPUT);
  Serial.begin(9600);

  pinMode(2, OUTPUT);
}

void showNumber(int number) {
  if(number == 0) {
    showDigit(0, nbrDigits - 1);
  } else {
    for(int digit = nbrDigits - 1; digit >= 0; digit--) {
      if(number > 0) {
        showDigit(number % 10, digit);
        number = number / 10;
      }
    }
  }
}
```

```
void showDigit(int number, int digit)
{
   //Serial.println(number);
   //Serial.println(digit);
   digitalWrite(digitPins[digit], HIGH);
   for(int segment = 1; segment < 8; segment++) {
      boolean isBitSet = bitRead(numeral[number], segment);
      digitalWrite(segmentPins[segment], isBitSet);
   }
   delay(5);
   digitalWrite(digitPins[digit], LOW);
}


void loop() {
   int analogValue = analogRead(LM35_pin);
   temp = (analogValue * 1.1 * 100) / 1023; // правильне перетворення у °C
   RGB(temp);
   showNumber(temp);
   Serial.println(temp);
}
```



г

Рис. 3.Результат виконання.

Код для МК AVR:

```
#include "lcd.h"
#include <avr/interrupt.h>
#define R_PIN PB5
#define G_PIN PB6
#define B_PIN PB7
#define SegA PD0
#define SegB PD1
#define SegC PD2
#define SegD PD3
#define SegE PD4
#define SegF PD5
#define SegG PD6
#define Dig1 PB1
```

```
#define Dig2 PB2
#define Dig3 PB3
#define Dig4 PB4

float adc_volt0;
float adc_volt1;
volatile uint16_t count = 1234;
volatile uint8_t current_digit = 1;

void InitADC();
uint16_t ReadADC(uint8_t channel);
void RGB(int temp)
{
if(temp >= 33) {
        PORTB |=  (1<<R_PIN);
        PORTB &= ~(1<<G_PIN);
        PORTB &= ~(1<<B_PIN);
}
else if(temp >= 25) {
        PORTB &= ~(1<<R_PIN);
        PORTB |=  (1<<G_PIN);
        PORTB &= ~(1<<B_PIN);
}
else if(temp > 18) {
        PORTB &= ~(1<<R_PIN);
        PORTB &= ~(1<<G_PIN);
        PORTB |=  (1<<B_PIN);
}
```

```
else {
        PORTB &= ~(1<<R_PIN);
        PORTB &= ~(1<<G_PIN);
        PORTB &= ~(1<<B_PIN);
}
}

void disp_off() {
PORTB &= ~((1<<Dig1) | (1<<Dig2) | (1<<Dig3) | (1<<Dig4));
}

void disp(uint8_t number) {
PORTD &= ~((1<<SegA)|(1<<SegB)|(1<<SegC)|(1<<SegD)|(1<<SegE)|(1<<SegF)|(1<<SegG));
switch(number) {
        case 0: PORTD |= (1<<SegA)|(1<<SegB)|(1<<SegC)|(1<<SegD)|(1<<SegE)|(1<<SegF); break;
        case 1: PORTD |= (1<<SegB)|(1<<SegC); break;
        case 2: PORTD |= (1<<SegA)|(1<<SegB)|(1<<SegD)|(1<<SegE)|(1<<SegG); break;
        case 3: PORTD |= (1<<SegA)|(1<<SegB)|(1<<SegC)|(1<<SegD)|(1<<SegG); break;
        case 4: PORTD |= (1<<SegB)|(1<<SegC)|(1<<SegF)|(1<<SegG); break;
        case 5: PORTD |= (1<<SegA)|(1<<SegC)|(1<<SegD)|(1<<SegF)|(1<<SegG); break;
        case 6: PORTD |= (1<<SegA)|(1<<SegC)|(1<<SegD)|(1<<SegE)|(1<<SegF)|(1<<SegG); break;
        case 7: PORTD |= (1<<SegA)|(1<<SegB)|(1<<SegC); break;
        case 8: PORTD |= (1<<SegA)|(1<<SegB)|(1<<SegC)|(1<<SegD)|(1<<SegE)|(1<<SegF)|(1<<SegG); break;
```

```
        case 9: PORTD |=
(1<<SegA)|(1<<SegB)|(1<<SegC)|(1<<SegD)|(1<<SegF)|(1<<SegG); break;
}
}

ISR(TIMER1_OVF_vect) {
disp_off();
switch(current_digit) {
        case 1: disp(count / 1000); PORTB |= (1<<Dig1); break;
        case 2: disp((count/100) % 10); PORTB |= (1<<Dig2); break;
        case 3: disp((count/10) % 10); PORTB |= (1<<Dig3); break;
        case 4: disp(count % 10); PORTB |= (1<<Dig4); break;
}
current_digit = (current_digit % 4) + 1;
}




int main(void)
{
DDRD
|=(1<<SegA)|(1<<SegB)|(1<<SegC)|(1<<SegD)|(1<<SegE)|(1<<SegF)|(1<<SegG);
DDRB |= (1<<Dig1)|(1<<Dig2)|(1<<Dig3)|(1<<Dig4);
DDRB |= (1<<R_PIN)|(1<<G_PIN)|(1<<B_PIN);
DDRA &= ~((1<<PA0) | (1<<PA1) | (1<<PA3));
TCCR1A = 0;
TCCR1B = (1<<CS10);
```

```
TIMSK |= (1<<TOIE1);
sei();

InitADC();
while (1)
{
        adc_volt0 = (ReadADC(0) * 5.0) / 1024.0; // ADC0
        adc_volt1 = (ReadADC(1) * 5.0) / 1024.0; // ADC1
        count = (adc_volt0 - adc_volt1) * 100;
        RGB(count);

}
}

void InitADC()
{
ADMUX |= (1 << REFS0);
ADCSRA |= (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1);
}


uint16_t ReadADC(uint8_t channel)
{
ADMUX = (ADMUX & 0xF0) | (channel & 0x0F);
ADCSRA |= (1 << ADSC);
while (ADCSRA & (1 << ADSC));
return ADC;
}
```

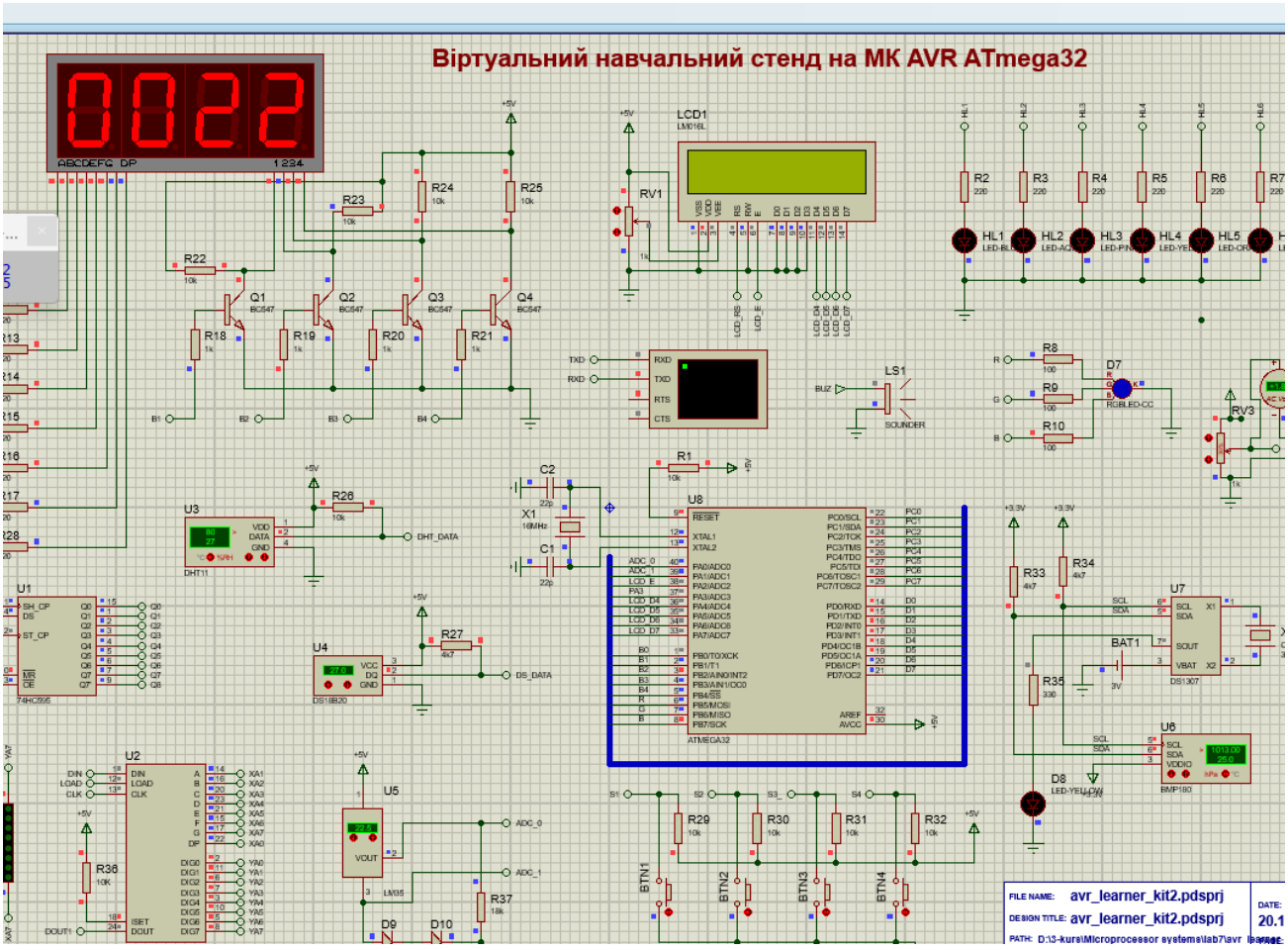


Рис. 4.Результат виконання.

3. Реалізувати програму, яка виводить на семисегментний індикатор з використанням регістру зсуву 74HC595 значення температури з датчика LM35 та керує RGB світлодіодом. Якщо t >18°C, то світиться синій світлодіод; якщо t ≥ 25°C, то світиться зелений світлодіод; якщо t ≥ 33°C, то світиться червоний світлодіод; якщо t ≤ 18°C, то RGB світлодіод не світиться..

Код для Arduino IDE:

```
#include <LiquidCrystal.h>

#define LM35_pin A0

#define R_pin A1

#define G_pin A2

#define B_pin A3


#define button A0

#define clockPin 7   // SH_CP
```

```
#define dataPin 6    // SER
#define latchPin 8   // ST_CP

byte current_digit;
int temp;

void disp(byte number);
void disp_off();

void RGB(int temp)
{
   if(temp>=33)
   {
      digitalWrite(R_pin,HIGH);
      digitalWrite(G_pin,LOW);
      digitalWrite(B_pin,LOW);
   }
   else if(temp>=25)
   {
      digitalWrite(R_pin,LOW);
      digitalWrite(G_pin,HIGH);
      digitalWrite(B_pin,LOW);
   }
   else if(temp>18)
   {
      digitalWrite(R_pin,LOW);
      digitalWrite(G_pin,LOW);
      digitalWrite(B_pin,HIGH);
```

```
    }
    else
    {
       digitalWrite(R_pin,LOW);
       digitalWrite(G_pin,LOW);
       digitalWrite(B_pin,LOW);
    }
}

void setup() {
    pinMode(button, INPUT_PULLUP);
    pinMode(5, OUTPUT);
    pinMode(4, OUTPUT);
    pinMode(3, OUTPUT);
    pinMode(2, OUTPUT);
    pinMode(clockPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(latchPin, OUTPUT);
    analogReference(INTERNAL); // встановлюємо опорну напругу 1.1 В
    pinMode(LM35_pin, INPUT);
    pinMode(R_pin, OUTPUT);
    pinMode(G_pin, OUTPUT);
    pinMode(B_pin, OUTPUT);
    Serial.begin(9600);
    disp_off();

    // Timer1 module overflow interrupt configuration
    TCCR1A = 0;
```

```
    TCCR1B = 1; // prescaler = 1
    TCNT1 = 0;
    TIMSK1 = 1; // enable Timer1 overflow interrupt
}

ISR(TIMER1_OVF_vect) {
    disp_off();

    switch (current_digit) {
        case 1:
            disp(temp / 1000);
            digitalWrite(5, HIGH);
            break;
        case 2:
            disp((temp / 100) % 10);
            digitalWrite(4, HIGH);
            break;
        case 3:
            disp((temp / 10) % 10);
            digitalWrite(3, HIGH);
            break;
        case 4:
            disp(temp % 10);
            digitalWrite(2, HIGH);
            break;
    }

    current_digit = (current_digit % 4) + 1;
```

```
}

void loop() {
  int analogValue = analogRead(LM35_pin);
  temp = analogValue * 1.1 * 100.0 / 1023.0; // °C
  Serial.println(temp);
  RGB(temp);
}

void disp(byte number) {
  byte pattern;
  switch (number) {
    case 0: pattern = 0xFC; break;
    case 1: pattern = 0x60; break;
    case 2: pattern = 0xDA; break;
    case 3: pattern = 0xF2; break;
    case 4: pattern = 0x66; break;
    case 5: pattern = 0xB6; break;
    case 6: pattern = 0xBE; break;
    case 7: pattern = 0xE0; break;
    case 8: pattern = 0xFE; break;
    case 9: pattern = 0xF6; break;
    default: pattern = 0x00; break;
  }

  digitalWrite(latchPin, LOW);              // зупиняємо оновлення виходів
  shiftOut(dataPin, clockPin, MSBFIRST, pattern); // передаємо дані
  digitalWrite(latchPin, HIGH);             // оновлюємо виходи
```

```
}

void disp_off() {
    digitalWrite(5, LOW);
    digitalWrite(4, LOW);
    digitalWrite(3, LOW);
    digitalWrite(2, LOW);
}
```

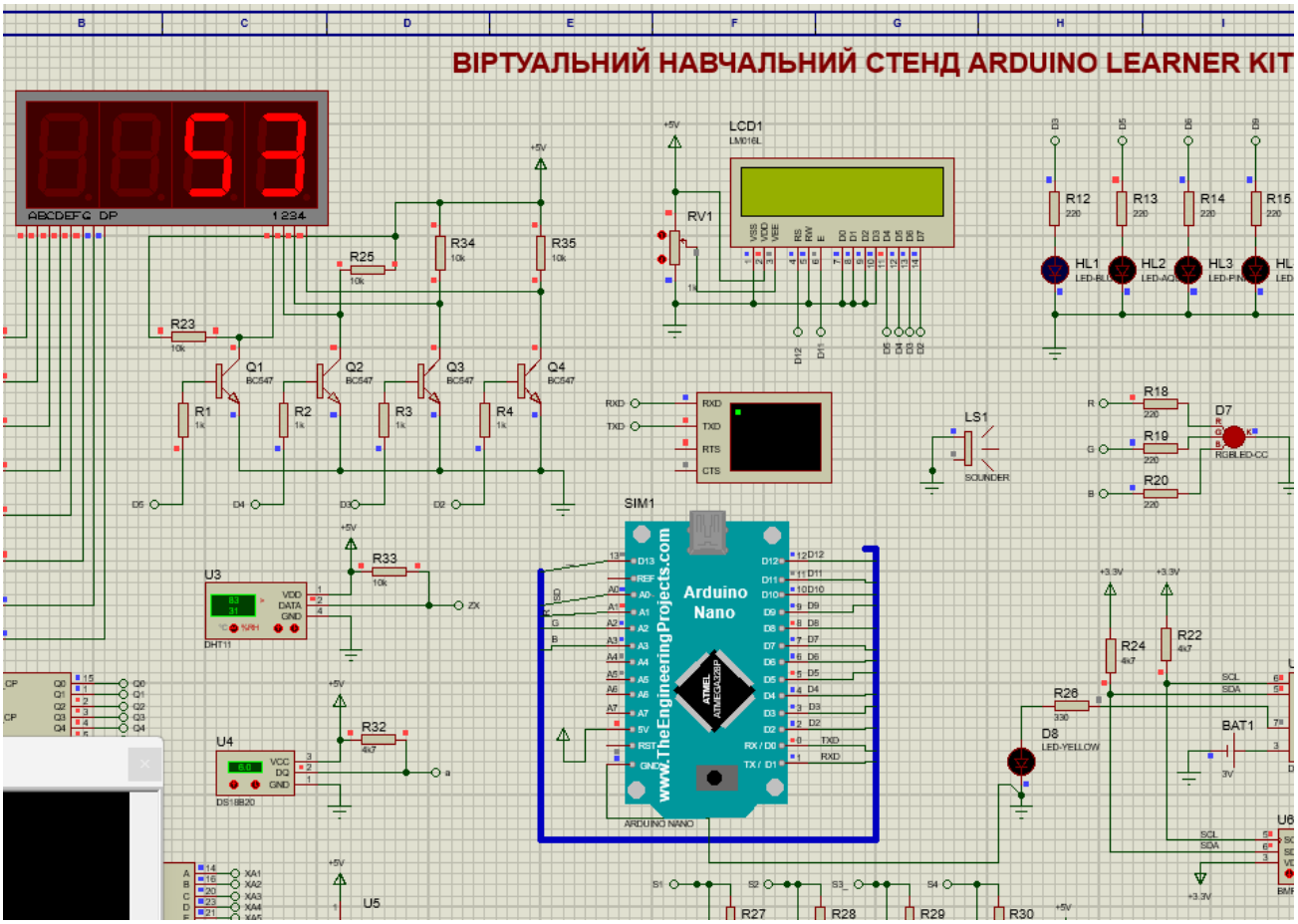


Рис. 5.Результат виконання.

Код для МК AVR:

```
#include "lcd.h"
#include <avr/interrupt.h>
#define R_PIN PB5
```

```
#define G_PIN PB6
#define B_PIN PB7
#define Dig1 PB1
#define Dig2 PB2
#define Dig3 PB3
#define Dig4 PB4
unsigned char clockPin = 7;
unsigned char dataPin  = 6;
unsigned char latchPin = 5;
unsigned char digitPins[4] = {5,4,3,2};
uint8_t digits[10] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6};
float adc_volt0;
float adc_volt1;
volatile uint16_t count = 1234;
volatile int current_digit = 1;

void InitADC();
uint16_t ReadADC(uint8_t channel);

void shiftOutByte(uint8_t data) {
for(int i=7;i>=0;i--) {
        if(data & (1<<i)) PORTD |=  (1<<dataPin);
        else              PORTD &= ~(1<<dataPin);
        PORTD |=  (1<<clockPin);
        PORTD &= ~(1<<clockPin);
}
}
```

```
void RGB(int temp)
{
if(temp >= 33) {
        PORTB |=  (1<<R_PIN);
        PORTB &= ~(1<<G_PIN);
        PORTB &= ~(1<<B_PIN);
}
else if(temp >= 25) {
        PORTB &= ~(1<<R_PIN);
        PORTB |=  (1<<G_PIN);
        PORTB &= ~(1<<B_PIN);
}
else if(temp > 18) {
        PORTB &= ~(1<<R_PIN);
        PORTB &= ~(1<<G_PIN);
        PORTB |=  (1<<B_PIN);
}
else {
        PORTB &= ~(1<<R_PIN);
        PORTB &= ~(1<<G_PIN);
        PORTB &= ~(1<<B_PIN);
}
}

void disp_digit(unsigned char number) {

for(int i=0;i<4;i++) PORTB &= ~(1<<digitPins[i]);
```

```
PORTD &= ~(1<<latchPin);
shiftOutByte(digits[number]);
PORTD |= (1<<latchPin);


PORTB |= (1<<digitPins[current_digit-1]);


}


ISR(TIMER1_OVF_vect) {
unsigned char d;
switch(current_digit) {
        case 1: d = (count / 1000) % 10; break;
        case 2: d = (count / 100)  % 10; break;
        case 3: d = (count / 10)   % 10; break;
        case 4: d = count % 10; break;
}
disp_digit(d);
current_digit = (current_digit % 4) + 1;
}


int main(void)
{
DDRD |= (1<<dataPin)|(1<<clockPin)|(1<<latchPin);
```

```
DDRB |= (1<<digitPins[0])|(1<<digitPins[1])|(1<<digitPins[2])|(1<<digitPins[3]);
DDRB |= (1<<R_PIN)|(1<<G_PIN)|(1<<B_PIN);
DDRA &= ~((1<<PA0) | (1<<PA1) | (1<<PA3));
TCCR1A = 0;
TCCR1B = (1<<CS10);
TIMSK |= (1<<TOIE1);
sei();

InitADC();
while (1)
{
        adc_volt0 = (ReadADC(0) * 5.0) / 1024.0; // ADC0
        adc_volt1 = (ReadADC(1) * 5.0) / 1024.0; // ADC1
        count = (adc_volt0 - adc_volt1) * 100;
        RGB(count);

}
}

void InitADC()
{
ADMUX |= (1 << REFS0);
ADCSRA |= (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1);
}


uint16_t ReadADC(uint8_t channel)
{
```

```
ADMUX = (ADMUX & 0xF0) | (channel & 0x0F);
ADCSRA |= (1 << ADSC);
while (ADCSRA & (1 << ADSC));
return ADC;
}
```

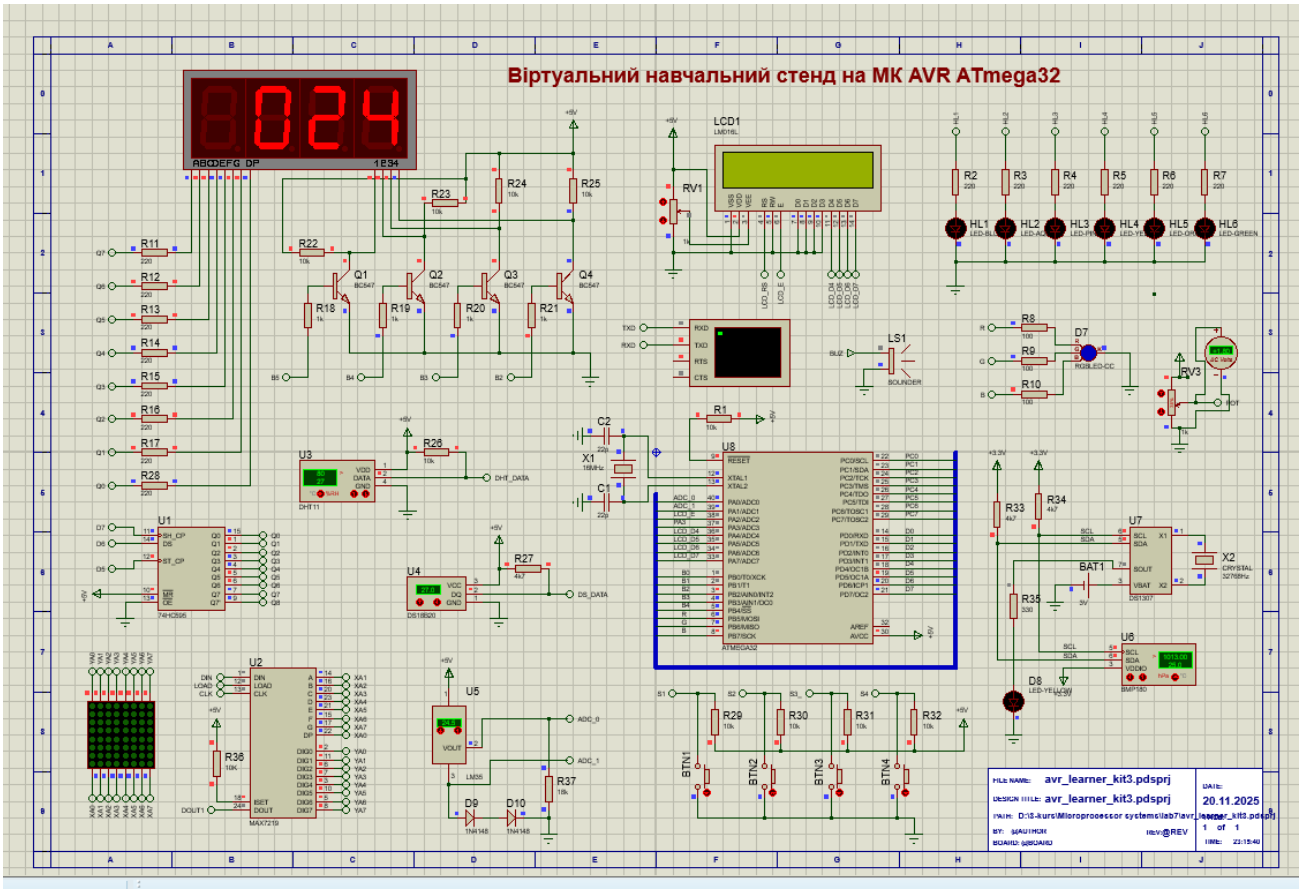


Рис. 6.Результат виконання.

Висновок: У ході виконання лабораторної роботи я ознайомився з принципом роботи та зчитуванням даних давача температури LM35; закріпив навички виведення інформації на семигментний індикатор з використанням регістру зсуву 74HC595 та LCD- індикатор.