МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра **САП**

ЗВІТ

до виконання лабораторної роботи №4

На тему: ” Робота з семисегментним індикатором”

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 семисегментного індикатора та дослідити можливості програмування його роботи у динамічному режимі; навчитися програмувати режими роботи семисегментного індикатора з використанням переривань таймера/лічильника, що входить до складу мікроконтролера Arduino; закріпити навички роботи з цифровими портами, тактовими кнопками, масивами.

Теоретичні відомості

Плата Arduino дозволяє швидко та мінімальними засобами вирішити найрізноманітніші завдання. Але там де потрібні довільні інтервали часу (періодичне опитування датчиків, високоточні ШІМ сигнали, імпульси великої тривалості) стандартні бібліотечні функції затримки не зручні. На час їх дії скетч призупиняється і керувати ним стає неможливо. У подібній ситуації краще використовувати вбудовані AVR таймери. Таймери, як і зовнішні переривання, працюють незалежно від основної програми.

У стандартних платах Arduino є три таймера Timer0, Timer1 і Timer2. Timer0 є 8 бітним таймером, це означає, що його рахунковий регістр може зберігати числа до 255. Timer0 використовується стандартними часовими функціями Arduino такими як delay() і millis(), так що краще його не використовувати у своїх проектах.

Timer1 це 16 бітний таймер з максимальним значенням 65535. Цей таймер використовує бібліотека Arduino Servo, враховуйте це якщо застосовуєте його в своїх проектах.

Timer2. 8 бітний і дуже схожий на Timer0. Він використовується в функції tone () Arduino.

Для обробки переривань у мові програмування Arduino використовується функція ***ISR ()***. У ній необхідно вказати тип переривання [1, 10]. Arduino (ATmega328P) використовує такі варіанти параметра функції *ISR ()* для роботи з таймерами:

- IMER2_COMPA_vect. переривання від Timer2 при збігу з A;
- TIMER2_COMPB_vect . переривання від Timer2 при збігу з B;
- TIMER2_OVF_vect. переривання переповнення Timer2;
- TIMER1_CAPT_vect. переривання від Timer1 (режим захоплення);
- TIMER1_COMPA_vect . переривання від Timer1 при збігу з A;
- TIMER1_COMPB_vect . переривання від Timer1 при збігу з B;
- TIMER1_OVF_vect . переривання переповнення Timer1;

- TIMER0_COMPA_vect . переривання від Timer0 при збігу з A;
- TIMER0_COMPB_vect . переривання від Timer0 при збігу з B;
- TIMER0_OVF_vect. переривання переповнення Timer0.

Для того щоб використовувати таймери в AVR є регістри налаштувань. Таймери містять безліч таких регістрів. Два з них. регістри управління таймера / лічильника містять установчі змінні й називаються TCCRxA і TCCRxB, де x. номер таймера (TCCR1A і TCCR1B). Кожен регістр містить 8 біт і кожен біт зберігає конфігураційну змінну. Найбільш важливими є три останні біта в TCCR1B: CS12, CS11 і CS10. Вони визначають тактову частоту таймера (табл. 3.2). За замовчуванням ці біти не встановлені.

TIMSK1 це регістр маски переривань Timer1. Він контролює переривання, які таймер може викликати. Установка біта 0 біту (TOIE1) вказує таймеру, що дозволено переривання коли таймер переповнюється (дораховує до максимального значення з частотою, що визначена бітами CS12, CS11, CS10). Якщо частота предільника Timer1 (табл. 3.2) встановлена у значення 001, то при тактовій частоті 16 МГц Atmega328 переривання виникне приблизно через 0,0041 секунд (65535/16МГц).

Завдання

1. Внести зміни до програми SEG5 (додаток Д). Вивести двійковий код напруги, що зчитується АЦП Arduino з потенціометру RV1.

Код Arduino Ide:

```
const int numeral[10] = {
  //ABCDEFGH
  B11111100, // 0
  B01100000, // 1
  B11011010, // 2
  B11110010, // 3
  B01100110, // 4
  B10110110, // 5
  B00111110, // 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 setup() {
  for(int i = 0; i < 8; i++) {
    pinMode(segmentPins[i], OUTPUT);
  }
  for(int i = 0; i < nbrDigits; i++) {
    pinMode(digitPins[i], OUTPUT);
  }
  pinMode(A1,INPUT);
}

void loop() {
  // int value = analogRead(0);

  int value = analogRead(A1);
  showNumber(value);
}

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) {
    digitalWrite(digitPins[digit], HIGH);
    for(int segment = 1; segment < 8; segment++) {
        boolean isBitSet = bitRead(numeral[number], segment);
        digitalWrite(segmentPins[segment], isBitSet);
    }

    digitalWrite(digitPins[digit], LOW);
}
```

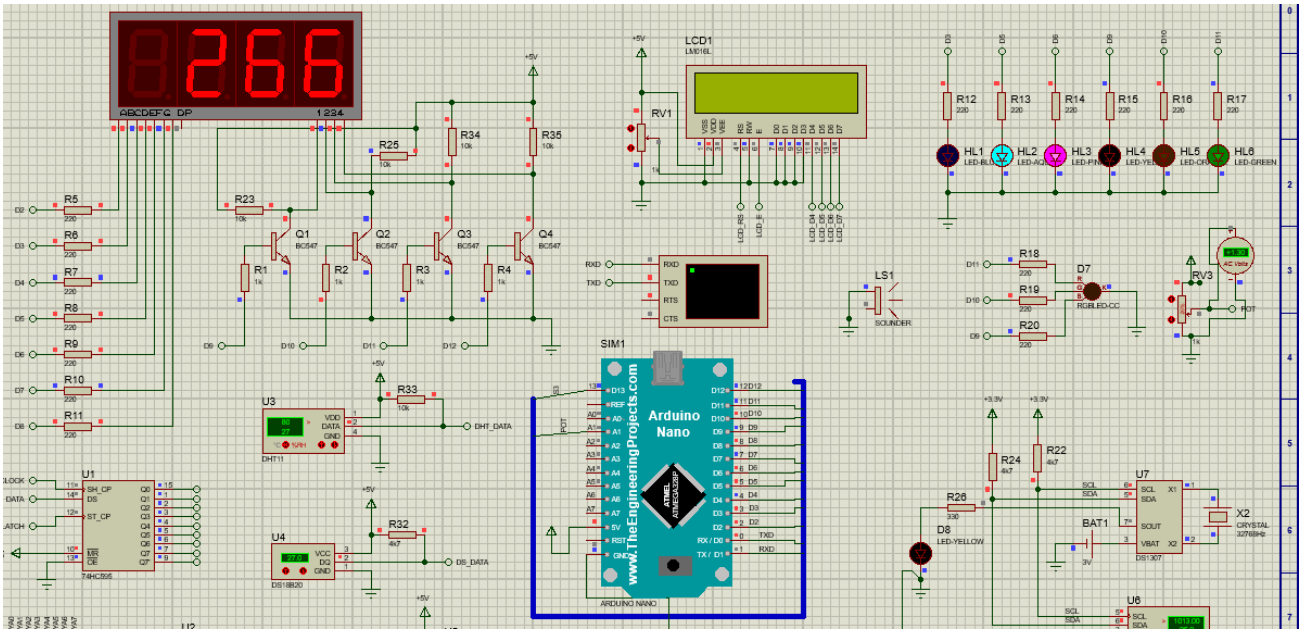

Рис. 1. Результат роботи

Код МК AVR:

```
#define F_CPU 16000000UL  // частота процесора
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

// Цифри для 7-сегментного дисплея (ABCDEFGH)
const uint8_t numeral[10] = {
    0b10111111, // 0
    0b10000110, // 1
    0b11011011, // 2
    0b11001111, // 3
    0b01100110, // 4
    0b11101101, // 5
    0b01111101, // 6
    0b10000111, // 7
    0b11111111, // 8
```

```
  0b11101111  // 9
};

// Піни сегментів H,G,F,E,D,C,B,A підключені до PD7..PD0
const uint8_t segmentPins[8] = {0,1,2,3,4,5,6,7}; // PD0..PD7

// Цифри дисплея підключені до PB0..PB3
const uint8_t digitPins[4] = {0,1,2,3}; // PB0..PB3
const uint8_t nbrDigits = 4;

// ADC канал A1 (PC1)
const uint8_t ADC_channel = 1;

void ADC_init() {
  ADMUX = (1<<REFS0) | (ADC_channel & 0x0F); // AVCC як опора, канал PC1
  ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0); // вкл ADC, преддільник 128
}

uint16_t ADC_read() {
  ADMUX = (ADMUX & 0xF0) | (ADC_channel & 0x0F); // вибір каналу
  ADCSRA |= (1<<ADSC); // початок перетворення
  while(ADCSRA & (1<<ADSC)); // чекати завершення
  return ADC;
}

void showDigit(uint8_t number, uint8_t digit) {
  PORTB = (1 << digit); // вмикаємо потрібну цифру
  for(uint8_t seg = 0; seg < 8; seg++) {
```

```
        if(numeral[number] & (1<<seg)) {
            PORTD |= (1<<seg); // встановити сегмент
        } else {
            PORTD &= ~(1<<seg); // вимкнути сегмент
        }
    }
    _delay_ms(5);
    PORTB &= ~(1 << digit); // вимикаємо цифру
}

void showNumber(uint16_t 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 /= 10;
            }
        }
    }
}

int main(void) {
    // Налаштування портів
    DDRD = 0xFF; // всі сегменти як вихід
    DDRB = 0x0F; // PB0..PB3 як вихід
```

```
    ADC_init();


    while(1) {


        uint16_t value = ADC_read(); // зчитуємо напругу з A1
        showNumber(value);
    }
}
```

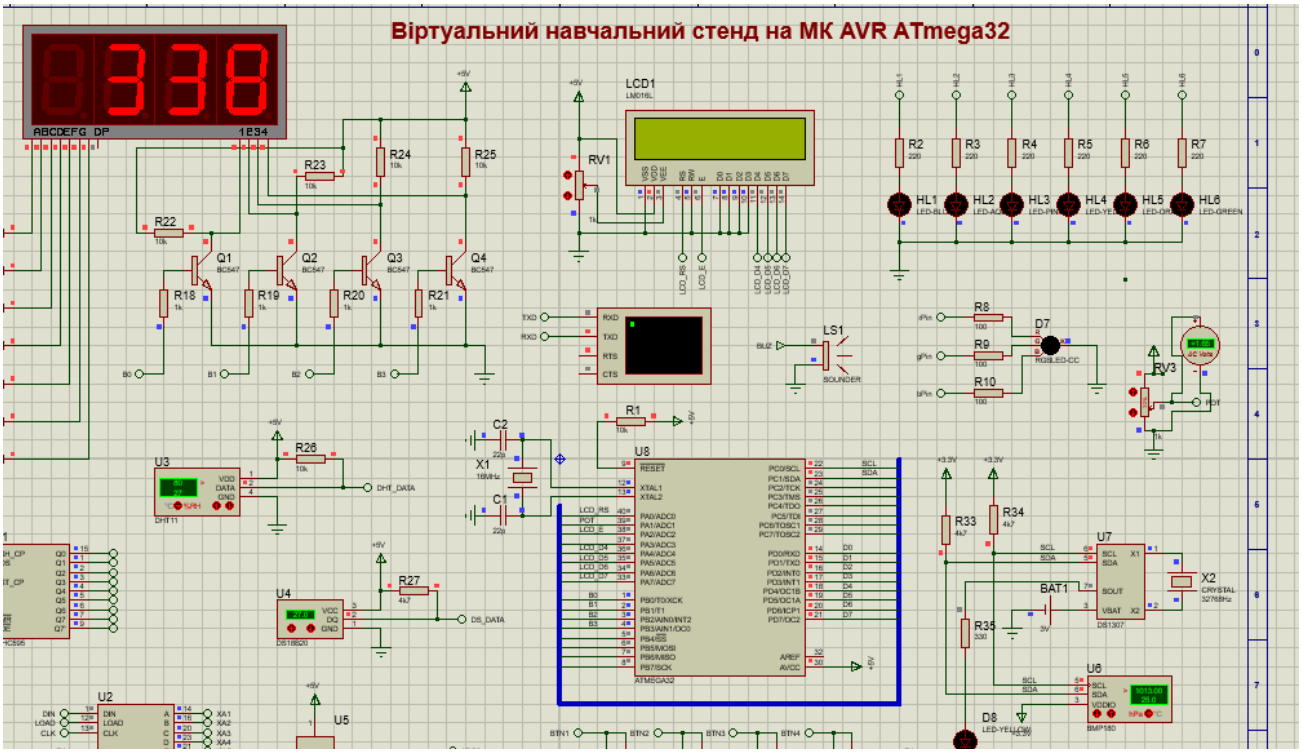

Рис. 2. Результат роботи

2. Внести зміни до програми SEG6 (додаток Д), щоб за натисненням кнопки кожен раз відображалось випадкове число.

Код Arduino Ide:

```
#define button A0


// segment pin definitions
#define SegA 2
#define SegB 3
```

```
#define SegC 4
#define SegD 5
#define SegE 6
#define SegF 7
#define SegG 8

// common pins of the four digits
#define Dig1 9
#define Dig2 10
#define Dig3 11
#define Dig4 12

// variable declarations
byte current_digit;
int count = 0;

void setup() {
  pinMode(button, INPUT_PULLUP);
  pinMode(SegA, OUTPUT);
  pinMode(SegB, OUTPUT);
  pinMode(SegC, OUTPUT);
  pinMode(SegD, OUTPUT);
  pinMode(SegE, OUTPUT);
  pinMode(SegF, OUTPUT);
  pinMode(SegG, OUTPUT);
  pinMode(Dig1, OUTPUT);
  pinMode(Dig2, OUTPUT);
  pinMode(Dig3, OUTPUT);
```

```
  pinMode(Dig4, OUTPUT);

  disp_off(); // turn off display

  // Timer1 overflow interrupt configuration
  TCCR1A = 0;
  TCCR1B = 1;     // prescaler = 1
  TCNT1 = 0;      // reset timer
  TIMSK1 = 1;     // enable overflow interrupt
}

ISR(TIMER1_OVF_vect) {
  disp_off();
  switch (current_digit) {
    case 1:
      disp(count / 1000);
      digitalWrite(Dig1, HIGH);
      break;

    case 2:
      disp((count / 100) % 10);
      digitalWrite(Dig2, HIGH);
      break;

    case 3:
      disp((count / 10) % 10);
      digitalWrite(Dig3, HIGH);
      break;
```

```
    case 4:
      disp(count % 10);
      digitalWrite(Dig4, HIGH);
      break;
  }
  current_digit = (current_digit % 4) + 1;
}

void loop() {
  if (digitalRead(button) == 0) {
    count=random(0, 9999);


  }
}

void disp(byte number) {
  switch (number) {
    case 0:
      digitalWrite(SegA, HIGH); digitalWrite(SegB, HIGH); digitalWrite(SegC,
HIGH);
      digitalWrite(SegD, HIGH); digitalWrite(SegE, HIGH); digitalWrite(SegF,
HIGH);
      digitalWrite(SegG, LOW);
      break;


    case 1:
      digitalWrite(SegA, LOW); digitalWrite(SegB, HIGH); digitalWrite(SegC,
HIGH);
```

```
    digitalWrite(SegD, LOW); digitalWrite(SegE, LOW); digitalWrite(SegF, LOW);
    digitalWrite(SegG, LOW);
    break;

  case 2:
    digitalWrite(SegA, HIGH); digitalWrite(SegB, HIGH); digitalWrite(SegC,
LOW);
    digitalWrite(SegD, HIGH); digitalWrite(SegE, HIGH); digitalWrite(SegF, LOW);
    digitalWrite(SegG, HIGH);
    break;

  case 3:
    digitalWrite(SegA, HIGH); digitalWrite(SegB, HIGH); digitalWrite(SegC,
HIGH);
    digitalWrite(SegD, HIGH); digitalWrite(SegE, LOW); digitalWrite(SegF, LOW);
    digitalWrite(SegG, HIGH);
    break;

  case 4:
    digitalWrite(SegA, LOW); digitalWrite(SegB, HIGH); digitalWrite(SegC,
HIGH);
    digitalWrite(SegD, LOW); digitalWrite(SegE, LOW); digitalWrite(SegF, HIGH);
    digitalWrite(SegG, HIGH);
    break;

  case 5:
    digitalWrite(SegA, HIGH); digitalWrite(SegB, LOW); digitalWrite(SegC,
HIGH);
    digitalWrite(SegD, HIGH); digitalWrite(SegE, LOW); digitalWrite(SegF, HIGH);
```

```
    digitalWrite(SegG, HIGH);
    break;

  case 6:
    digitalWrite(SegA, HIGH); digitalWrite(SegB, LOW); digitalWrite(SegC, HIGH);
    digitalWrite(SegD, HIGH); digitalWrite(SegE, HIGH); digitalWrite(SegF, HIGH);
    digitalWrite(SegG, HIGH);
    break;

  case 7:
    digitalWrite(SegA, HIGH); digitalWrite(SegB, HIGH); digitalWrite(SegC, HIGH);
    digitalWrite(SegD, LOW); digitalWrite(SegE, LOW); digitalWrite(SegF, LOW);
    digitalWrite(SegG, LOW);
    break;

  case 8:
    digitalWrite(SegA, HIGH); digitalWrite(SegB, HIGH); digitalWrite(SegC, HIGH);
    digitalWrite(SegD, HIGH); digitalWrite(SegE, HIGH); digitalWrite(SegF, HIGH);
    digitalWrite(SegG, HIGH);
    break;

  case 9:
    digitalWrite(SegA, HIGH); digitalWrite(SegB, HIGH); digitalWrite(SegC, HIGH);
    digitalWrite(SegD, HIGH); digitalWrite(SegE, LOW); digitalWrite(SegF, HIGH);
```

```
    digitalWrite(SegG, HIGH);
    break;
  }
}

void disp_off() {
  digitalWrite(Dig1, LOW);
  digitalWrite(Dig2, LOW);
  digitalWrite(Dig3, LOW);
  digitalWrite(Dig4, LOW);
}
```

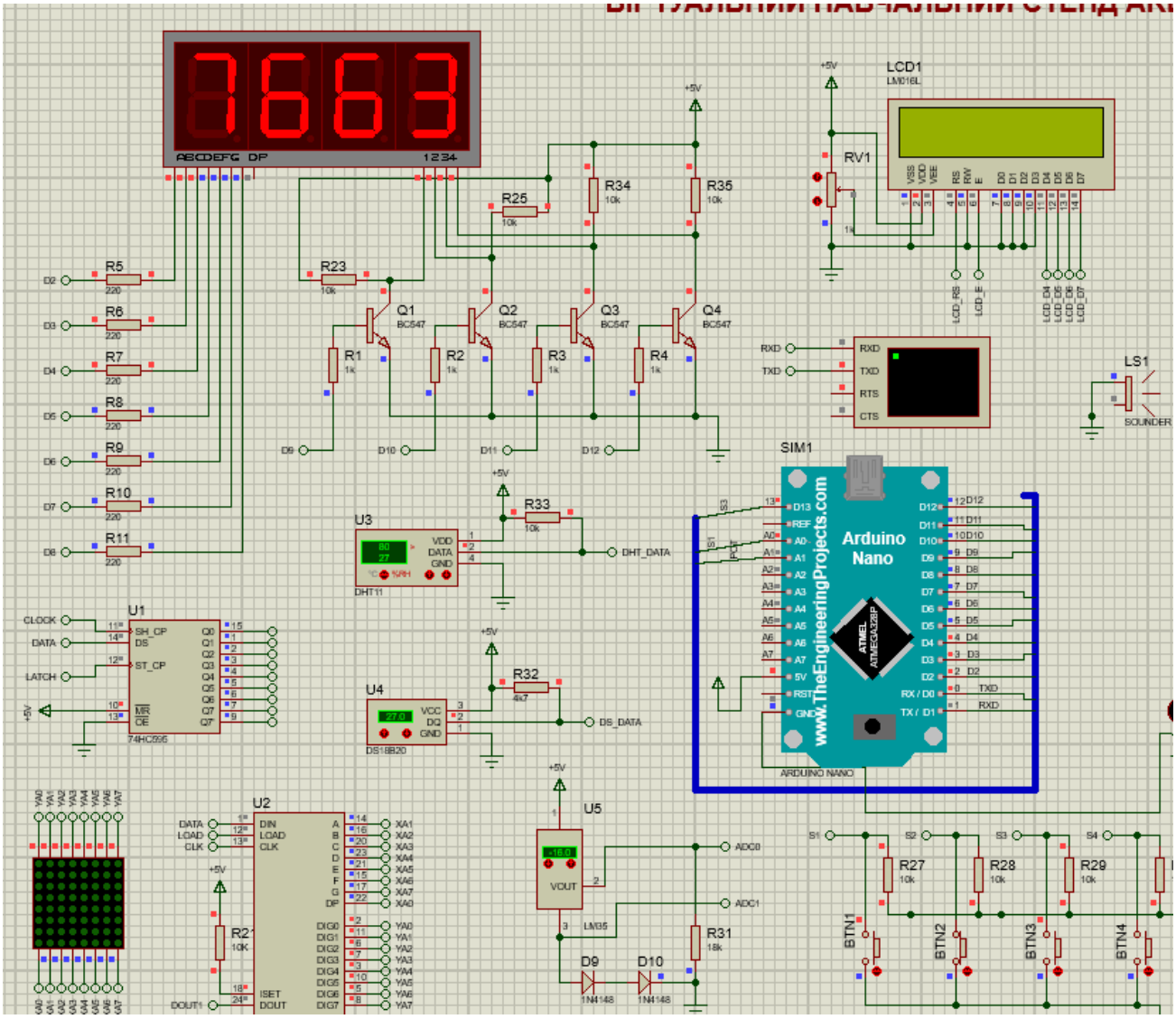

Рис. 3. Результат роботи

Код МК AVR:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>

#define BUTTON   PA0

#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

volatile uint16_t count = 1234;
volatile uint8_t current_digit = 1;

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);
DDRC &= ~(1<<BUTTON);
PORTC |= (1<<BUTTON);
DDRA &= ~(1 << PA1);
PORTA |= (1 << PA1);

TCCR1A = 0;
TCCR1B = (1<<CS10);
TIMSK |= (1<<TOIE1);
sei();

while (1) {
        if (!(PINA & (1<<PA1))) {
                count = rand() % 10000;
                _delay_ms(200);
        }
}
}
```

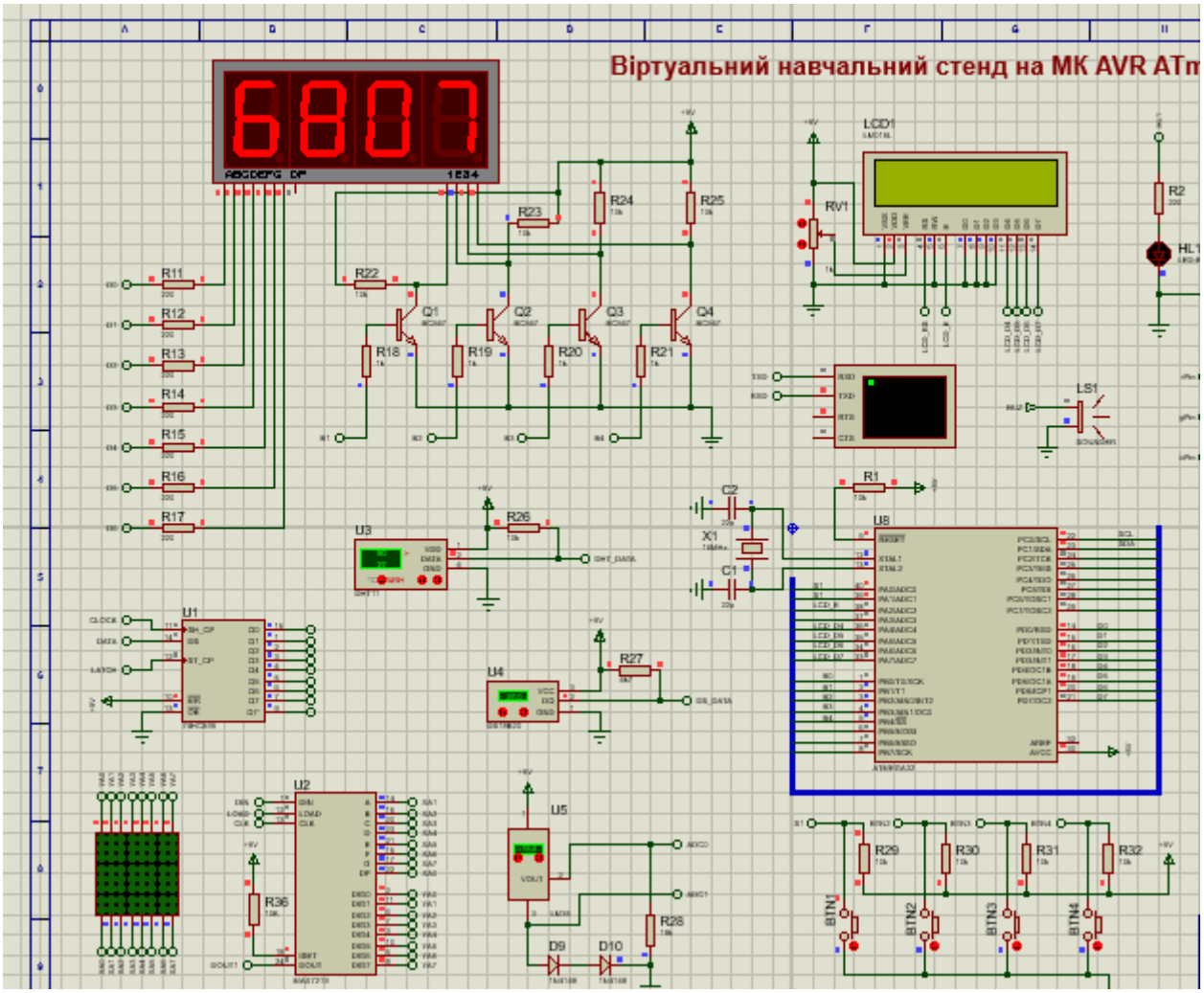

Рис. 4. Результат роботи

3. Реалізувати програму з застосуванням тактових кнопок S1-S4. Кожна кнопка встановлює число від 0 до 9 у відповідній позиції індикатора.

Код Arduino Ide:

```
const int numeral[10] = {
  B11111100,//0
  B01100000,//1
  B11011010,//2
  B11110010,//3
  B01100110,//4
  B10110110,//5
  B10111110,//6
  B11100000,//7
```

```
  B11111110,//8
  B11110110,//9
};
unsigned long currentTime;
// H,G,F,E,D,C,B,A
const int segmentPins[] = {13,8, 7, 6, 5, 4, 3,13};
const int nbrDigits = 4;

//SW1-Sw4
const byte swPins[]={A0,A1,A2,A3};

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

//pres sw1-sw4
volatile bool buttonsPres[]={false,false,false,false};

volatile unsigned long lastInteraptTime[]={0,0,0,0};
volatile int count[]={0,0,0,0};
 int a=0;

void setup() {
  for(int i=0;i<7;i++)
  {
    pinMode(segmentPins[i],OUTPUT);
  }
  for(int i=0;i<4;i++)
  {
```

```
  pinMode(swPins[i],INPUT_PULLUP);
 }
 pinMode(2, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(2), buttonISR, CHANGE);

 for(int i=0;i<4;i++)
 {
  pinMode(digitPins[i],OUTPUT);
 }
 Serial.begin(9600);
}

void buttonISR() // для A0–A3 на Arduino Uno
{



  for(int i = 0; i < 4; i++)
  {
     if(digitalRead(swPins[i]) == LOW && buttonsPres[i] == false && currentTime
- lastInteraptTime[i] >= 50)
     {

        count[i]++;
        if(count[i] == 10)
        {
          count[i] = 0;
        }
        buttonsPres[i] = true;
        lastInteraptTime[i] = currentTime;
```

```
        }

        if(digitalRead(swPins[i]) == HIGH && buttonsPres[i] == true )
        {
            buttonsPres[i] = false;
            lastInteraptTime[i] = currentTime;
        }
    }

}

void loop()
{
  currentTime = millis();
  showNumber();
}


void showNumber()
{

  for(int i=0;i<4;i++)
  {
    showDigit(count[i],i);
  }
}

void showDigit(int number, int 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);
}
```

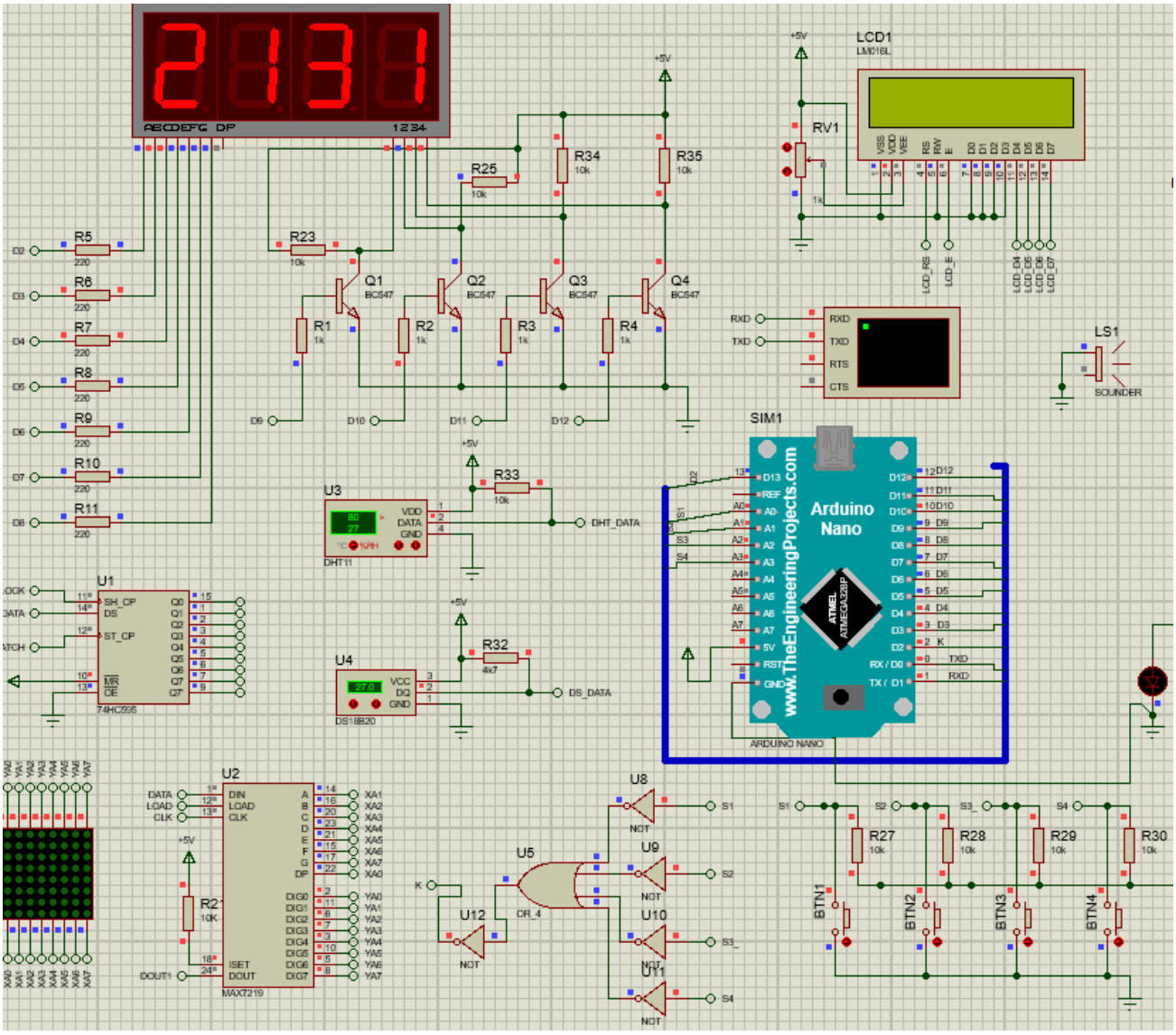

Рис. 5. Результат роботи

Код МК AVR:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdbool.h>

const uint8_t numeral[10] = {
0b11111100,  //0
0b01100000, //1
0b11011010, //2
0b11110010, //3
0b01100110, //4
0b10110110, //5
0b10111110, //6
0b11100000, //7
0b11111110, //8
0b11110110  //9
};

unsigned long currentTime;

const int segmentPins[] = {7, 6, 5, 4, 3, 2, 1, 0};
const int nbrDigits = 4;


const uint8_t  swPins[]={0,1,2,3};
```

```
const int digitPins[] = {3, 4, 5, 6};


volatile bool buttonsPres[]={false,false,false,false};
volatile uint32_t millis_count = 0;

volatile unsigned long lastInteraptTime[]={0,0,0,0};
volatile int count[]={4,5,6,7};

void timer0_init(void)
{
TCCR0 = (1 << WGM01) | (1 << CS01) | (1 << CS00);
OCR0 = 249;
TIMSK |= (1 << OCIE0);
sei();
}

ISR(TIMER0_COMP_vect)
{
millis_count++;
}

uint32_t millis(void)
{
uint32_t ms;
cli();
```

```
ms = millis_count;
sei();
return ms;
}

int main()
{
timer0_init();
for(int i=0; i<8; i++)
{
	DDRC |= (1 << segmentPins[i]);
}

for(int i=0; i<4; i++)
{
	DDRD |= (1 << digitPins[i]);
}
DDRB |= (1 << 0);
for(int i=0; i<4; i++)
{
	DDRA &= ~(1 << swPins[i]);
	PORTA |= (1 << swPins[i]);
}


PORTB &= ~(1 << 0);
while(1)
{
```

```
        currentTime = millis();
        but();
        showNumber();
}
}
void but() {
for(int i = 0; i < 4; i++)
{
        if(!(PINA & (1 << swPins[i])) && (currentTime - lastInteraptTime[i] >= 50)
&& buttonsPres[i] == false)
        {
                count[i]++;
                if(count[i] == 10)
                count[i] = 0;

                buttonsPres[i] = true;
                lastInteraptTime[i] = currentTime;
        }

        if((PINA & (1 << swPins[i])) && buttonsPres[i] == true && (currentTime -
lastInteraptTime[i] >= 50))
        {
                buttonsPres[i] = false;
                lastInteraptTime[i] = currentTime;
        }
}
}
```

```
void showNumber()
{

for(int i=0;i<4;i++)
{
        showDigit(count[i],i);
}
}

void showDigit(uint8_t number, uint8_t digit)
{
PORTD |= (1 << digitPins[digit]);

for(uint8_t seg = 0; seg < 8; seg++)
{
        if (numeral[number] & (1 << seg))
                PORTC |= (1 << (7 - seg));
        else
                PORTC &= ~(1 << (7 - seg));
}

_delay_ms(5);
PORTD &= ~(1 << digitPins[digit]);
}
```

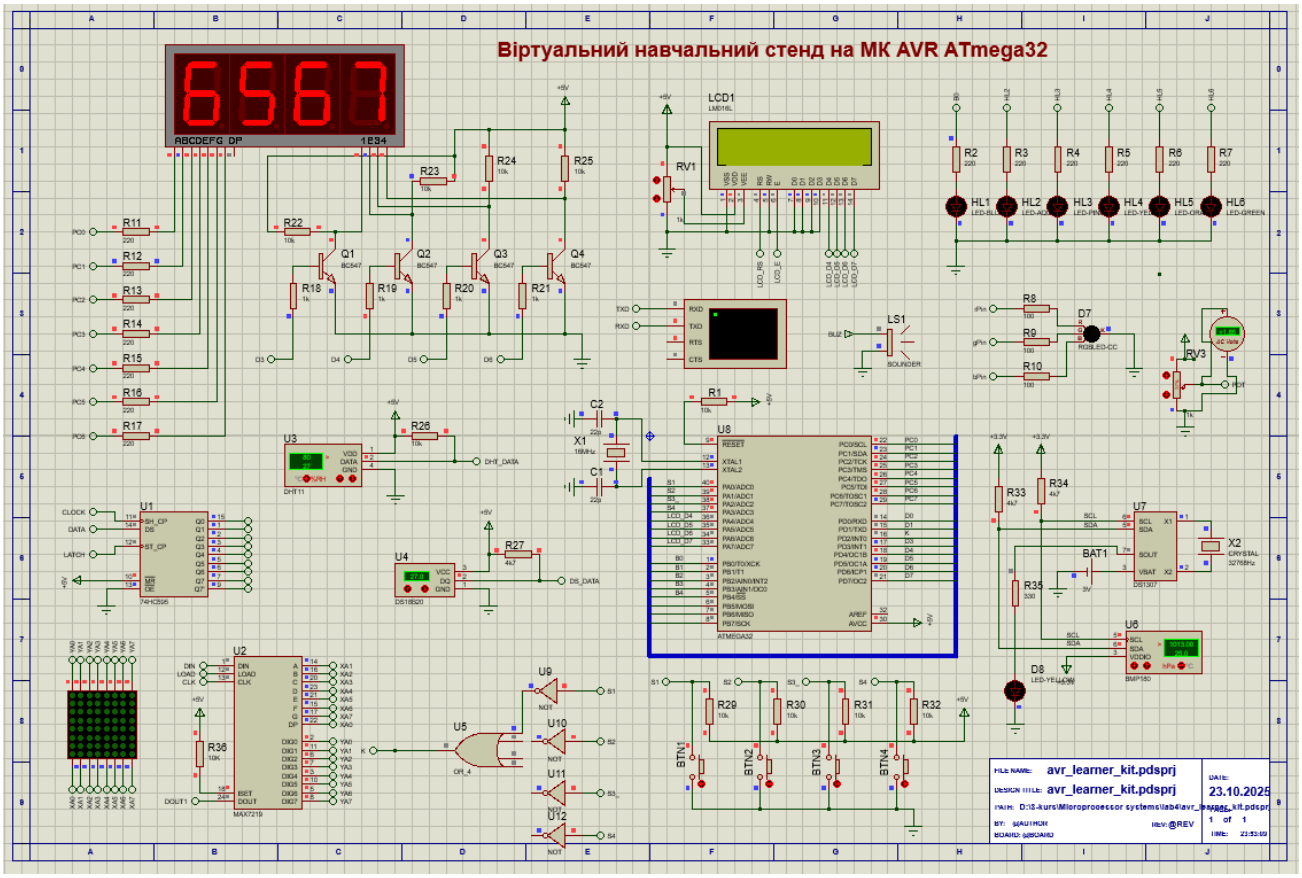

Рис. 6. Результат роботи

4. Реалізувати програму з застосуванням тактових кнопок S1-S4. Кнопки S3, S4 реалізують інкремент / декремент у старшій позиції індикатора в діапазоні від 0 до 99. Кнопки S1, S2 виконують аналогічну дію над числом у молодшій позиції індикатора.

Код Arduino Ide:

```
const int numeral[10] = {
  B11111100,//0
  B01100000,//1
  B11011010,//2
  B11110010,//3
  B01100110,//4
  B10110110,//5
  B10111110,//6
  B11100000,//7
```

```
  B11111110,//8
  B11110110,//9
};
unsigned long currentTime;
// H,G,F,E,D,C,B,A
const int segmentPins[] = {13,8, 7, 6, 5, 4, 3,13};
const int nbrDigits = 4;

//SW1-Sw4
const byte swPins[]={A0,A1,A2,A3};

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

//pres sw1-sw4
volatile bool buttonsPres[]={false,false,false,false};

volatile unsigned long lastInteraptTime[]={0,0,0,0};
volatile int count[]={0,0,0,0};
 int a=0;

void setup() {
  for(int i=0;i<7;i++)
  {
    pinMode(segmentPins[i],OUTPUT);
  }
  for(int i=0;i<4;i++)
  {
```

```
  pinMode(swPins[i],INPUT_PULLUP);
 }
 pinMode(2, INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(2), buttonISR, CHANGE);

 for(int i=0;i<4;i++)
 {
  pinMode(digitPins[i],OUTPUT);
 }
 Serial.begin(9600);
}

void buttonISR() // для A0–A3 на Arduino Uno
{



    if(count[0]*10+count[1]>99)
 {
  count[0]=0;
  count[1]=0;
 }
 if(count[2]*10+count[3]>99)
 {
  count[2]=0;
  count[3]=0;
 }
      if(digitalRead(swPins[0]) == LOW && buttonsPres[0] == false &&
currentTime - lastInteraptTime[0] >= 50)
      {
```

```
        count[1]++;
        if(count[1]==10)
        {
          count[0]++;
          count[1] = 0;
        }
        buttonsPres[0] = true;
        lastInteraptTime[0] = currentTime;
      }

      if(digitalRead(swPins[0]) == HIGH && buttonsPres[0] == true && currentTime
- lastInteraptTime[0] >= 50)
      {
        buttonsPres[0] = false;
        lastInteraptTime[0] = currentTime;
      }



          if(digitalRead(swPins[1]) == LOW && buttonsPres[1] == false &&
currentTime - lastInteraptTime[1] >= 50)
      {

        if(count[1]==0)
      {
       if(count[0]>0)
        {

        count[0]--;
        count[1]=9;
```

```
  }
}
else
{

  count[1]--;
}
    buttonsPres[1] = true;
    lastInteraptTime[1] = currentTime;
}

if(digitalRead(swPins[1]) == HIGH && buttonsPres[1] == true && currentTime - lastInteraptTime[1] >= 50)
{
    buttonsPres[1] = false;
    lastInteraptTime[1] = currentTime;
}

if(digitalRead(swPins[2]) == LOW && buttonsPres[2] == false && currentTime - lastInteraptTime[2] >= 50)
{
count[3]++;
if(count[3]==10)
{
  count[2]++;
  count[3] = 0;
}
    buttonsPres[2] = true;
    lastInteraptTime[2] = currentTime;
```

```
    }


    if(digitalRead(swPins[2]) == HIGH && buttonsPres[2] == true && currentTime
- lastInteraptTime[2] >= 50)
    {
       buttonsPres[2] = false;
       lastInteraptTime[2] = currentTime;
    }


          if(digitalRead(swPins[3]) == LOW && buttonsPres[3] == false &&
currentTime - lastInteraptTime[3] >= 50)
    {
    if(count[3]==0)
    {
     if(count[2]>0)
     {

       count[2]--;
       count[3]=9;
     }
    }
    else
    {

     count[3]--;
    }
       buttonsPres[3] = true;
       lastInteraptTime[3] = currentTime;
    }
```

```
      if(digitalRead(swPins[3]) == HIGH && buttonsPres[3] == true && currentTime
- lastInteraptTime[3] >= 50)
      {
         buttonsPres[3] = false;
         lastInteraptTime[3] = currentTime;
      }
}

void loop()
{
  currentTime = millis();
  /*Serial.print("Значення count: ");
  for(int i=0;i<4;i++)
  {
     Serial.print(count[i]);
  }*/
  showNumber();
}


void showNumber()
{

  for(int i=0;i<4;i++)
  {
    showDigit(count[i],i);
  }
}
```

```
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);
}
```

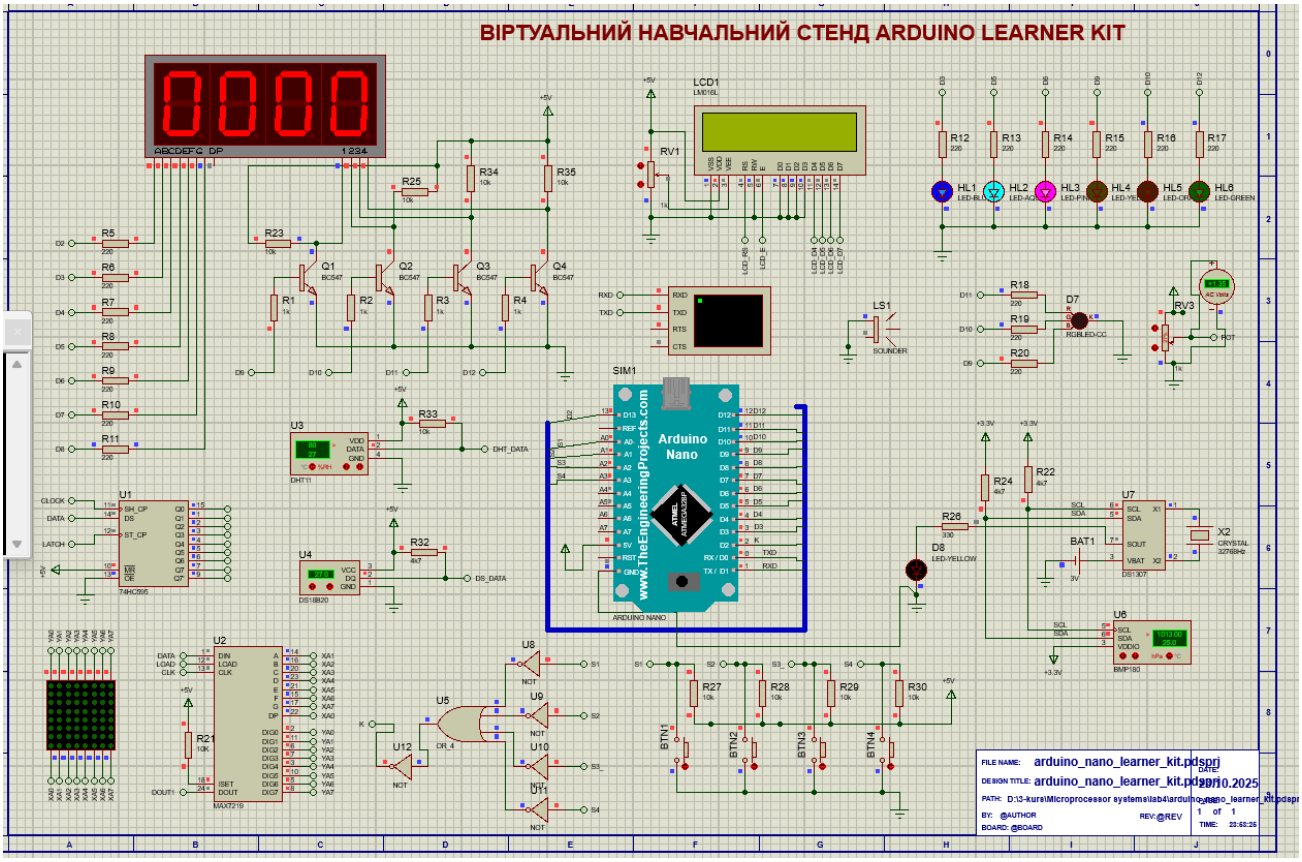

Рис. 7. Результат роботи

Код МК AVR:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdbool.h>

const uint8_t numeral[10] = {
0b11111100,  //0
0b01100000, //1
0b11011010, //2
0b11110010, //3
0b01100110, //4
0b10110110, //5
0b10111110, //6
0b11100000, //7
0b11111110, //8
0b11110110  //9
};

unsigned long currentTime;

const int segmentPins[] = {7, 6, 5, 4, 3, 2, 1, 0};
const int nbrDigits = 4;
```

```
const uint8_t  swPins[]={0,1,2,3};


const int digitPins[] = {3, 4, 5, 6};


volatile bool buttonsPres[]={false,false,false,false};
volatile uint32_t millis_count = 0;

volatile unsigned long lastInteraptTime[]={0,0,0,0};
volatile int count[]={4,5,9,8};

void timer0_init(void)
{
TCCR0 = (1 << WGM01) | (1 << CS01) | (1 << CS00);
OCR0 = 249;
TIMSK |= (1 << OCIE0);
sei();
}

ISR(TIMER0_COMP_vect)
{
millis_count++;
}

uint32_t millis(void)
{
uint32_t ms;
```

```
cli();
ms = millis_count;
sei();
return ms;
}

int main()
{
timer0_init();
for(int i=0; i<8; i++)
{
	DDRC |= (1 << segmentPins[i]);
}

for(int i=0; i<4; i++)
{
	DDRD |= (1 << digitPins[i]);
}
DDRB |= (1 << 0);
for(int i=0; i<4; i++)
{
	DDRA &= ~(1 << swPins[i]);
	PORTA |= (1 << swPins[i]);
}


PORTB &= ~(1 << 0);
while(1)
```

```
{
        currentTime = millis();
        but();
        showNumber();
}
}
void but() {

if(count[0]*10+count[1]>=100)
{
        count[0]=0;
        count[1]=0;
}
if(count[2]*10+count[3]>=100)
{
        count[2]=0;
        count[3]=0;
}

        if(!(PINA & (1 << swPins[0])) && (currentTime - lastInteraptTime[0] >= 50)
&& buttonsPres[0] == false)
        {
                count[1]++;
                if(count[1]==10)
                {
                        count[0]++;
                        count[1] = 0;
                }
```

```
            buttonsPres[0] = true;
            lastInteraptTime[0] = currentTime;
        }

        if((PINA & (1 << swPins[0])) && buttonsPres[0] == true && (currentTime - lastInteraptTime[0] >= 50))
        {
            buttonsPres[0] = false;
            lastInteraptTime[0] = currentTime;
        }


            if(!(PINA & (1 << swPins[1])) && (currentTime - lastInteraptTime[1] >= 50) && buttonsPres[1] == false)
            {

                if(count[1]==0)
                {
                    if(count[0]>0)
                    {

                        count[0]--;
                        count[1]=9;
                    }
                }
                else
                {
```

```
                count[1]--;
            }
            buttonsPres[1] = true;
            lastInteraptTime[1] = currentTime;
        }

        if((PINA & (1 << swPins[1])) && buttonsPres[1] == true &&
(currentTime - lastInteraptTime[1] >= 50))
        {
            buttonsPres[1] = false;
            lastInteraptTime[1] = currentTime;
        }

        if(!(PINA & (1 << swPins[2])) && (currentTime - lastInteraptTime[2]
>= 50) && buttonsPres[2] == false)
        {
            count[3]++;
            if(count[3]==10)
            {
                count[2]++;
                count[3] = 0;
            }


            buttonsPres[2] = true;
            lastInteraptTime[2] = currentTime;
        }
```

```
		if((PINA & (1 << swPins[2])) && buttonsPres[2] == true &&
(currentTime - lastInteraptTime[2] >= 50))
		{
			buttonsPres[2] = false;
			lastInteraptTime[2] = currentTime;
		}

		if(!(PINA & (1 << swPins[3])) && (currentTime - lastInteraptTime[3]
>= 50) && buttonsPres[3] == false)
		{
			if(count[3]==0)
			{
				if(count[2]>0)
				{

					count[2]--;
					count[3]=9;
				}
			}
			else
			{

				count[3]--;
			}

			buttonsPres[3] = true;
			lastInteraptTime[3] = currentTime;
		}
```

```
		if((PINA & (1 << swPins[3])) && buttonsPres[3] == true &&
(currentTime - lastInteraptTime[3] >= 50))
		{
			buttonsPres[3] = false;
			lastInteraptTime[3] = currentTime;
		}
}



void showNumber()
{

for(int i=0;i<4;i++)
{
	showDigit(count[i],i);
}
}

void showDigit(uint8_t number, uint8_t digit)
{
PORTD |= (1 << digitPins[digit]);

for(uint8_t seg = 0; seg < 8; seg++)
{
	if (numeral[number] & (1 << seg))
		PORTC |= (1 << (7 - seg));
	else
		PORTC &= ~(1 << (7 - seg));
}
```

```
_delay_ms(5);
PORTD &= ~(1 << digitPins[digit]);
}
```

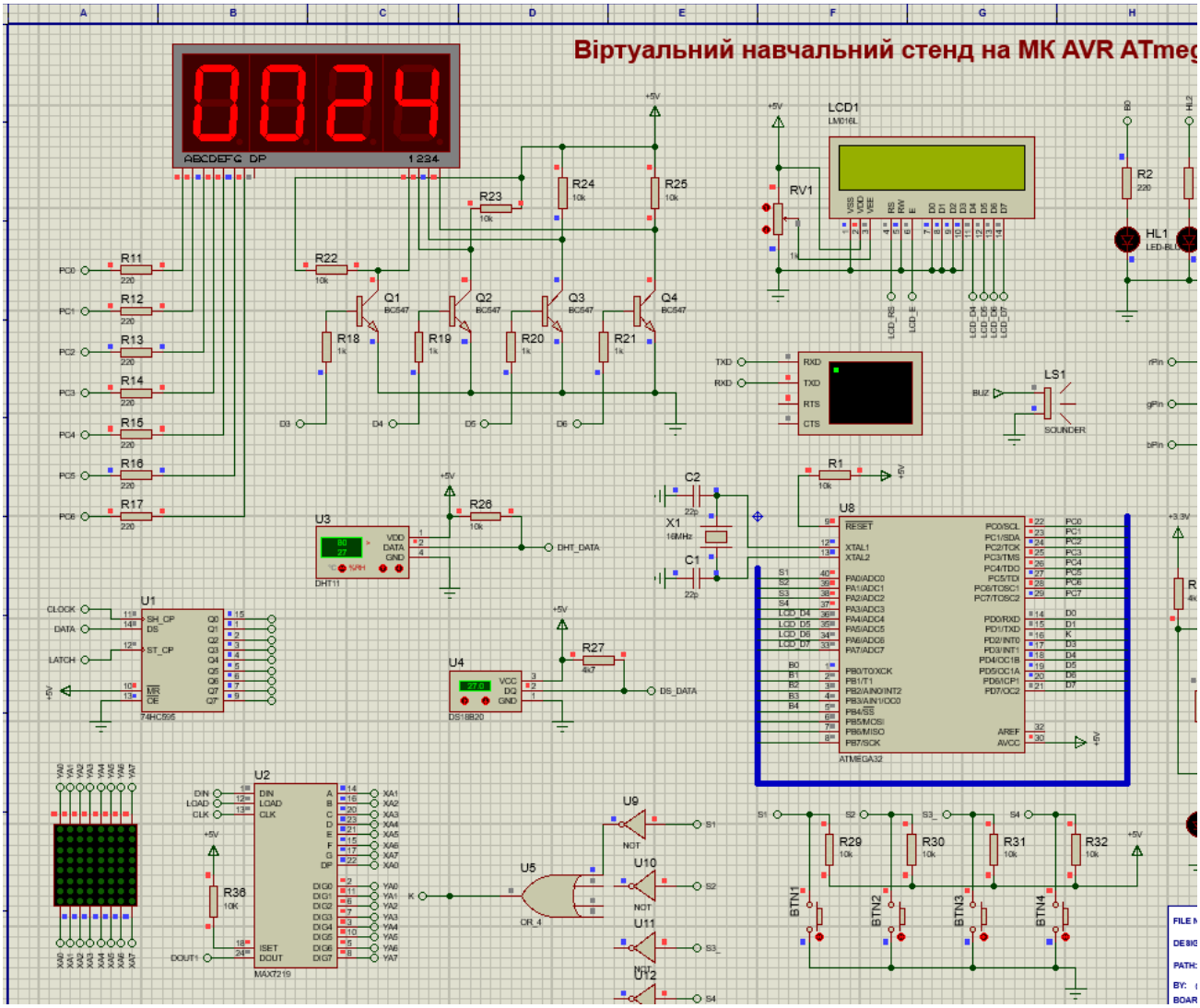

Рис. 8. Результат роботи

5. Реалізувати програму з застосуванням тактових кнопок S1-S4. Кнопки S3, S4 реалізують інкремент / декремент у старшій позиції індикатора в діапазоні від 0 до 23. Кнопки S1, S2 виконують аналогічну дію над числом в діапазоні від 0 до 59 у молодшій позиції індикатора.

Код Arduino Ide:

```
const int numeral[10] = {
  B11111100,//0
  B01100000,//1
  B11011010,//2
  B11110010,//3
  B01100110,//4
  B10110110,//5
  B10111110,//6
  B11100000,//7
  B11111110,//8
  B11110110,//9
};
unsigned long currentTime;
// H,G,F,E,D,C,B,A
const int segmentPins[] = {13,8, 7, 6, 5, 4, 3,13};
const int nbrDigits = 4;

//SW1-Sw4
const byte swPins[]={A0,A1,A2,A3};

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

//pres sw1-sw4
volatile bool buttonsPres[]={false,false,false,false};

volatile unsigned long lastInteraptTime[]={0,0,0,0};
volatile int count[]={0,0,0,0};
```

```
 int a=0;

void setup() {
  for(int i=0;i<7;i++)
  {
    pinMode(segmentPins[i],OUTPUT);
  }
  for(int i=0;i<4;i++)
  {
    pinMode(swPins[i],INPUT_PULLUP);
  }
  pinMode(2, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(2), buttonISR, CHANGE);

  for(int i=0;i<4;i++)
  {
    pinMode(digitPins[i],OUTPUT);
  }
  Serial.begin(9600);
}

void buttonISR() // для A0–A3 на Arduino Uno
{



      if(count[0]*10+count[1]>59)
  {
    count[0]=0;
```

```
  count[1]=0;
 }
 if(count[2]*10+count[3]>23)
 {
  count[2]=0;
  count[3]=0;
 }
      if(digitalRead(swPins[0]) == LOW && buttonsPres[0] == false &&
currentTime - lastInteraptTime[0] >= 50)
      {
         count[1]++;
         if(count[1]==10)
         {
          count[0]++;
          count[1] = 0;
         }
         buttonsPres[0] = true;
         lastInteraptTime[0] = currentTime;
      }

      if(digitalRead(swPins[0]) == HIGH && buttonsPres[0] == true && currentTime
- lastInteraptTime[0] >= 50)
      {
         buttonsPres[0] = false;
         lastInteraptTime[0] = currentTime;
      }
```

```
        if(digitalRead(swPins[1]) == LOW && buttonsPres[1] == false &&
currentTime - lastInteraptTime[1] >= 50)
    {


      if(count[1]==0)
    {
     if(count[0]>0)
     {


      count[0]--;
      count[1]=9;
     }
    }
    else
    {


     count[1]--;
    }
      buttonsPres[1] = true;
      lastInteraptTime[1] = currentTime;
    }


    if(digitalRead(swPins[1]) == HIGH && buttonsPres[1] == true && currentTime
- lastInteraptTime[1] >= 50)
    {
      buttonsPres[1] = false;
      lastInteraptTime[1] = currentTime;
    }
```

```
      if(digitalRead(swPins[2]) == LOW && buttonsPres[2] == false &&
currentTime - lastInteraptTime[2] >= 50)
      {
      count[3]++;
      if(count[3]==10)
      {
        count[2]++;
        count[3] = 0;
      }
         buttonsPres[2] = true;
         lastInteraptTime[2] = currentTime;
      }


      if(digitalRead(swPins[2]) == HIGH && buttonsPres[2] == true && currentTime
- lastInteraptTime[2] >= 50)
      {
         buttonsPres[2] = false;
         lastInteraptTime[2] = currentTime;
      }


            if(digitalRead(swPins[3]) == LOW && buttonsPres[3] == false &&
currentTime - lastInteraptTime[3] >= 50)
      {
      if(count[3]==0)
      {
        if(count[2]>0)
        {


         count[2]--;
```

```
          count[3]=9;
        }
      }
      else
      {

        count[3]--;
      }
          buttonsPres[3] = true;
          lastInteraptTime[3] = currentTime;
      }

      if(digitalRead(swPins[3]) == HIGH && buttonsPres[3] == true && currentTime
- lastInteraptTime[3] >= 50)
      {
          buttonsPres[3] = false;
          lastInteraptTime[3] = currentTime;
      }
}

void loop()
{
  currentTime = millis();
  /*Serial.print("Значення count: ");
  for(int i=0;i<4;i++)
  {
     Serial.print(count[i]);
  }*/
  showNumber();
```

```
}


void showNumber()
{

  for(int i=0;i<4;i++)
  {
    showDigit(count[i],i);
  }
}

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);
}
```

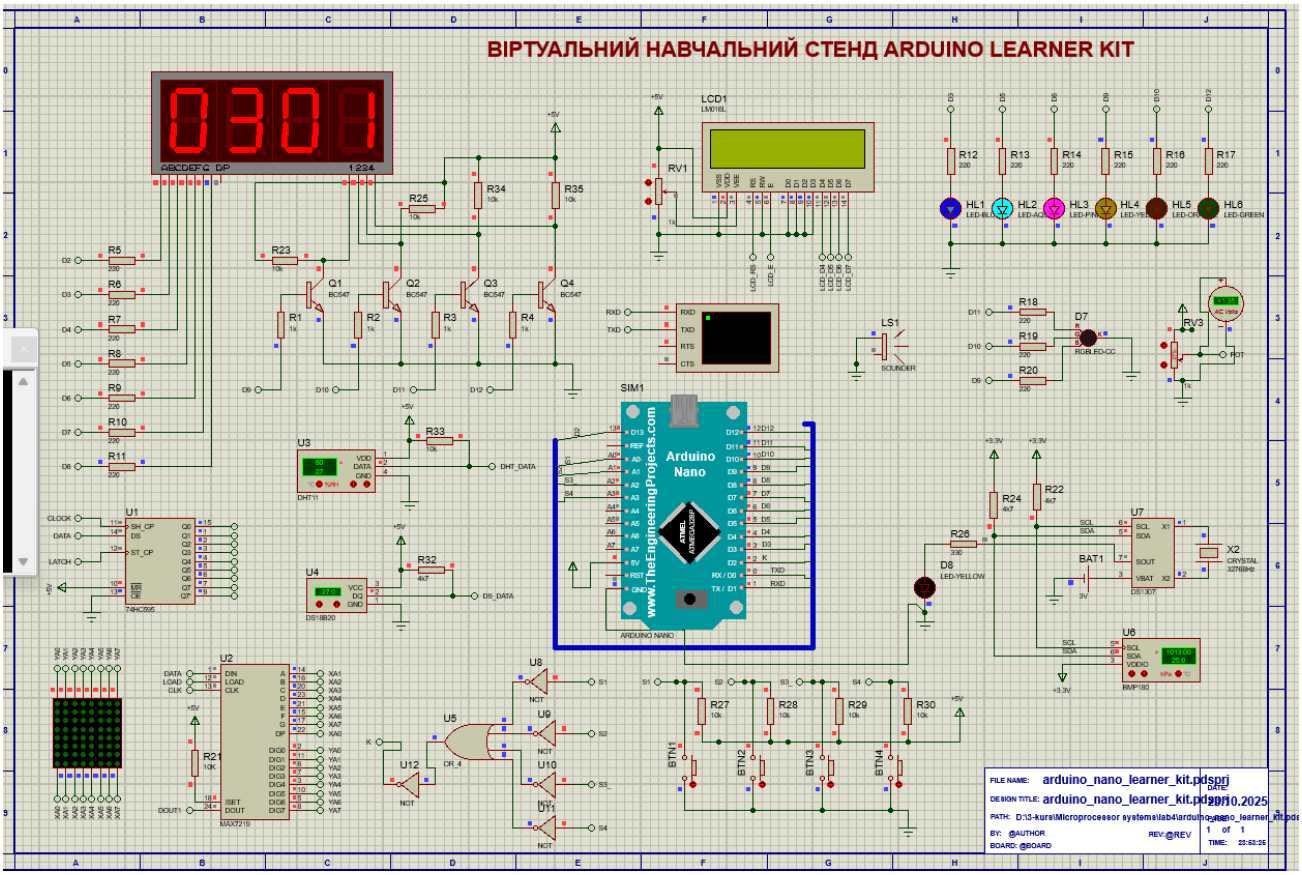

Рис. 9. Результат роботи

Код МК AVR:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdbool.h>

const uint8_t numeral[10] = {
0b11111100,  //0
0b01100000, //1
0b11011010, //2
0b11110010, //3
0b01100110, //4
0b10110110, //5
```

```
0b10111110, //6
0b11100000, //7
0b11111110, //8
0b11110110  //9
};

unsigned long currentTime;

const int segmentPins[] = {7, 6, 5, 4, 3, 2, 1, 0};
const int nbrDigits = 4;


const uint8_t  swPins[]={0,1,2,3};


const int digitPins[] = {3, 4, 5, 6};


volatile bool buttonsPres[]={false,false,false,false};
volatile uint32_t millis_count = 0;

volatile unsigned long lastInteraptTime[]={0,0,0,0};
volatile int count[]={4,5,9,8};

void timer0_init(void)
{
TCCR0 = (1 << WGM01) | (1 << CS01) | (1 << CS00);
OCR0 = 249;
```

```
TIMSK |= (1 << OCIE0);
sei();
}

ISR(TIMER0_COMP_vect)
{
millis_count++;
}

uint32_t millis(void)
{
uint32_t ms;
cli();
ms = millis_count;
sei();
return ms;
}

int main()
{
timer0_init();
for(int i=0; i<8; i++)
{
        DDRC |= (1 << segmentPins[i]);
}

for(int i=0; i<4; i++)
{
```

```
        DDRD |= (1 << digitPins[i]);
}
DDRB |= (1 << 0);
for(int i=0; i<4; i++)
{
        DDRA &= ~(1 << swPins[i]);
        PORTA |= (1 << swPins[i]);
}


PORTB &= ~(1 << 0);
while(1)
{
        currentTime = millis();
        but();
        showNumber();
}
}
void but() {

if(count[0]*10+count[1]>59)
{
        count[0]=0;
        count[1]=0;
}
if(count[2]*10+count[3]>23)
{
        count[2]=0;
```

```
        count[3]=0;
}

        if(!(PINA & (1 << swPins[0])) && (currentTime - lastInteraptTime[0] >= 50)
&& buttonsPres[0] == false)
        {
                count[1]++;
                if(count[1]==10)
                {
                        count[0]++;
                        count[1] = 0;
                }


                buttonsPres[0] = true;
                lastInteraptTime[0] = currentTime;
        }

        if((PINA & (1 << swPins[0])) && buttonsPres[0] == true && (currentTime -
lastInteraptTime[0] >= 50))
        {
                buttonsPres[0] = false;
                lastInteraptTime[0] = currentTime;
        }


                if(!(PINA & (1 << swPins[1])) && (currentTime - lastInteraptTime[1]
>= 50) && buttonsPres[1] == false)
                {
```

```
                if(count[1]==0)
                {
                        if(count[0]>0)
                        {

                                count[0]--;
                                count[1]=9;
                        }
                }
                else
                {

                        count[1]--;
                }
                buttonsPres[1] = true;
                lastInteraptTime[1] = currentTime;
        }

        if((PINA & (1 << swPins[1])) && buttonsPres[1] == true &&
(currentTime - lastInteraptTime[1] >= 50))
        {
                buttonsPres[1] = false;
                lastInteraptTime[1] = currentTime;
        }

        if(!(PINA & (1 << swPins[2])) && (currentTime - lastInteraptTime[2]
>= 50) && buttonsPres[2] == false)
        {
```

```
                count[3]++;
                if(count[3]==10)
                {
                        count[2]++;
                        count[3] = 0;
                }


                buttonsPres[2] = true;
                lastInteraptTime[2] = currentTime;
        }

        if((PINA & (1 << swPins[2])) && buttonsPres[2] == true &&
(currentTime - lastInteraptTime[2] >= 50))
        {
                buttonsPres[2] = false;
                lastInteraptTime[2] = currentTime;
        }

        if(!(PINA & (1 << swPins[3])) && (currentTime - lastInteraptTime[3]
>= 50) && buttonsPres[3] == false)
        {
                if(count[3]==0)
                {
                        if(count[2]>0)
                        {

                                count[2]--;
                                count[3]=9;
```

```
                }
            }
            else
            {

                count[3]--;
            }

            buttonsPres[3] = true;
            lastInteraptTime[3] = currentTime;
        }

        if((PINA & (1 << swPins[3])) && buttonsPres[3] == true &&
(currentTime - lastInteraptTime[3] >= 50))
        {
            buttonsPres[3] = false;
            lastInteraptTime[3] = currentTime;
        }
}



void showNumber()
{

for(int i=0;i<4;i++)
{
    showDigit(count[i],i);
}
}
```

```
void showDigit(uint8_t number, uint8_t digit)
{
PORTD |= (1 << digitPins[digit]);

for(uint8_t seg = 0; seg < 8; seg++)
{
        if (numeral[number] & (1 << seg))
                PORTC |= (1 << (7 - seg));
        else
                PORTC &= ~(1 << (7 - seg));
}

_delay_ms(5);
PORTD &= ~(1 << digitPins[digit]);
}
```

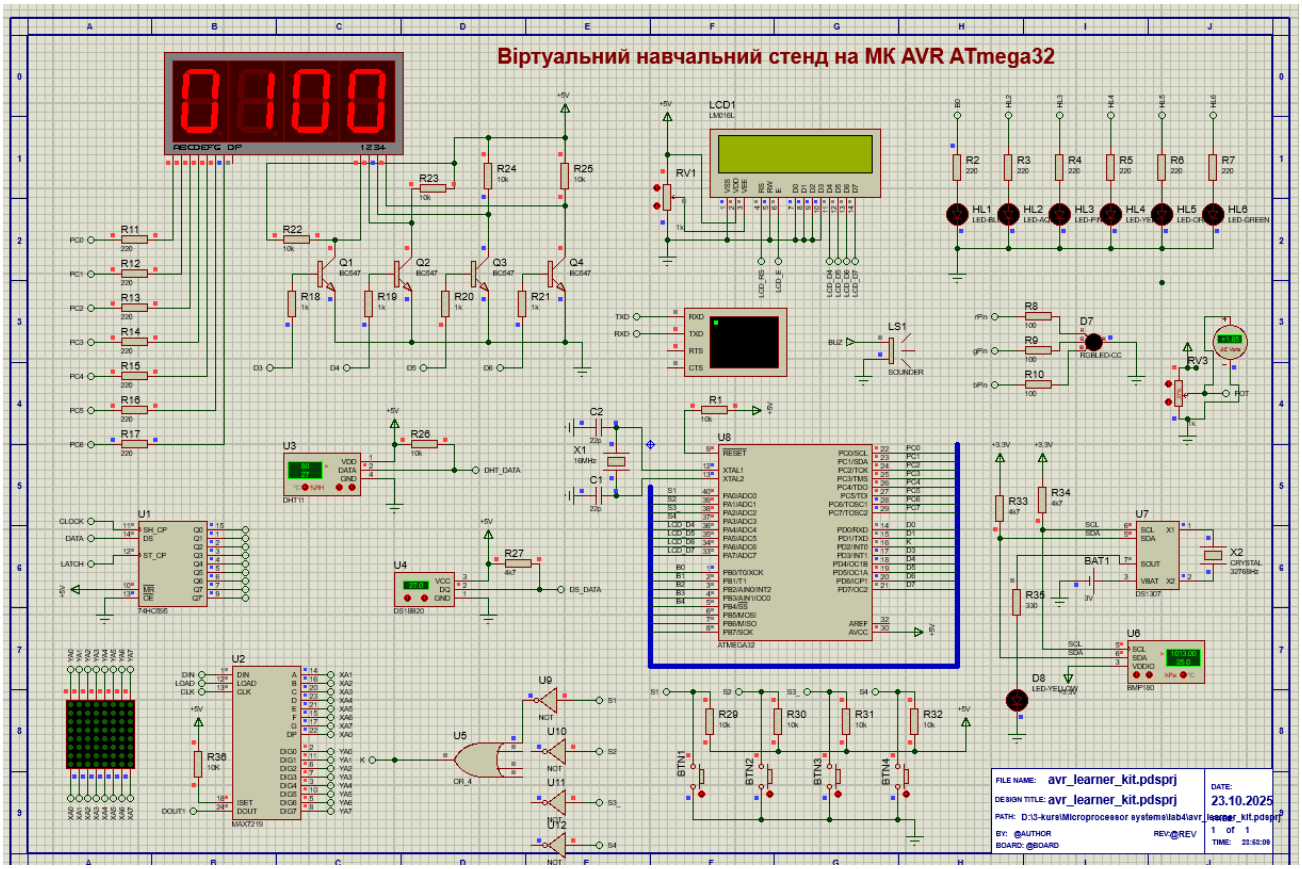

Рис. 10. Результат роботи

Висновок: У ході виконання лабораторної роботи я ознайомився з принципом роботи семисегментного індикатора та дослідив можливості програмування його роботи у динамічному режимі; навчився програмувати режими роботи семисегментного індикатора з використанням переривань таймера/лічильника, що входить до складу мікроконтролера Arduino; закріпив навички роботи з цифровими портами, тактовими кнопками, масивами.