МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра **САП**

ЗВІТ

до виконання лабораторної роботи №5

На тему: ”Семисегментний індикатор з регістром зсуву 74HC595”

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 регистру зсуву 74HC595 разом з семисегментим індикатором та дослідити можливості програмної реалізації SPI інтерфейсу; закріпити навички програмування режимів 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

74HC595. це регістр зсуву послідовно-паралельно типу (SIPO), який використовується для збільшення кількості виходів з мікроконтролера. Схема модуля регістра зсуву 74HC595 наведена на рис.3.3.

Мікросхема 74HC595 перетворює послідовний сигнал, що входить, на 1 лінію (SER) у вихідний паралельний сигнал на 8 виводах (Qx). Послідовна передача синхронна: для тактових сигналів використовується вивід (SCK). Також окремим виводом (RCK) управляється регістр даних, що дозволяє змінювати сигнал на 8 виходах одночасно, коли усі дані передані.

Таким чином, 3 портами мікроконтролера можна керувати 8 цифровими виходами. З регістрів 74HC595 можна робити каскади, підключаючи один до одного (через пін QH*). Це дозволяє отримати 16, 24, 32 цифрові виходи.

Для зручної роботи з регістром 74HC595 в Arduino існує вбудована функція **shiftOut()**. Вона здійснює побітовий зсув та вивід байту даних, починаючи з найстаршого (лівого) або молодшого (правого) значущого біта. Функція по черзі відправляє кожен біт на вказаний вивід даних, після чого формує імпульс (високий рівень, потім низький) на тактовому виводі, повідомляючи зовнішньому пристрою про надходження нового біта.

Функція є програмною реалізацією SPI.

shiftOut(dataPin, bitOrder, value). Параметри:

- dataPin: вхід данних у послідовному коді (int);
- clockPin: тактовий вивід (int);
- bitOrder: характеризує порядок, в якому будуть зсуватися та виводитися біти; може приймати значення MSBFIRST (старший біт перший) або LSBFIRST (молодший біт перший);
- value: байт даних (byte).

Завдання

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

Код для Arduino IDE:

```
// Q7-A, Q6-B,...Q0-H
// SC - SH_CP(D7), SER - D6, ST_CP - D8, OE - GND, Button - A0

#define button A0
#define clockPin 7   // SH_CP
#define dataPin 6    // SER
#define latchPin 8   // ST_CP

byte current_digit;
int count = 9987;

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

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);

   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(count / 1000);
            digitalWrite(5, HIGH);
            break;
        case 2:
            disp((count / 100) % 10);
            digitalWrite(4, HIGH);
            break;
        case 3:
            disp((count / 10) % 10);
            digitalWrite(3, HIGH);
            break;
        case 4:
            disp(count % 10);
            digitalWrite(2, HIGH);
            break;
    }
```

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

void loop() {

    count=analogRead(A0);
    delay(200);

}

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);
}
```

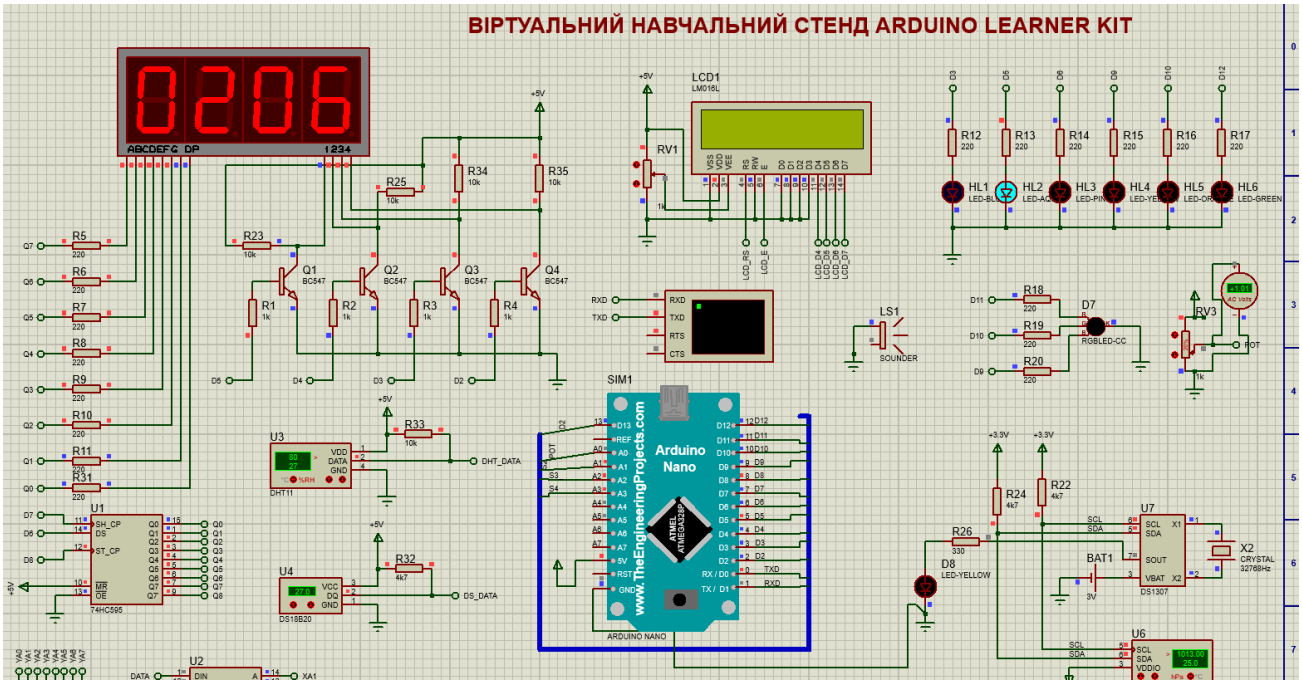

Рис. 1. Результат виконання

Код для МК AVR:

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

unsigned char current_digit = 1;
int count = 0;
```

```
unsigned char dataPin  = 6;
unsigned char clockPin = 7;
unsigned char latchPin = 0;
unsigned char digitPins[4] = {5,4,3,2};


uint8_t digits[10] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6};


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 disp_digit(unsigned char number) {

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


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

```
PORTB |= (1<<latchPin);



PORTD |= (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<<digitPins[0])|(1<<digitPins[1])|(1<<digitPins[2])|(1
<<digitPins[3]);
DDRB |= (1<<latchPin);
```

```
TCCR1A = 0;
TCCR1B = (1<<CS10);
TCNT1 = 0;
TIMSK |= (1<<TOIE1);
sei();

while(1) {

        ADMUX = (1<<REFS0);
        ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
        ADCSRA |= (1<<ADSC);
        while(ADCSRA & (1<<ADSC));
        count = ADC;
        if(count > 9999) count = 9999;
        _delay_ms(50);
}
}
```

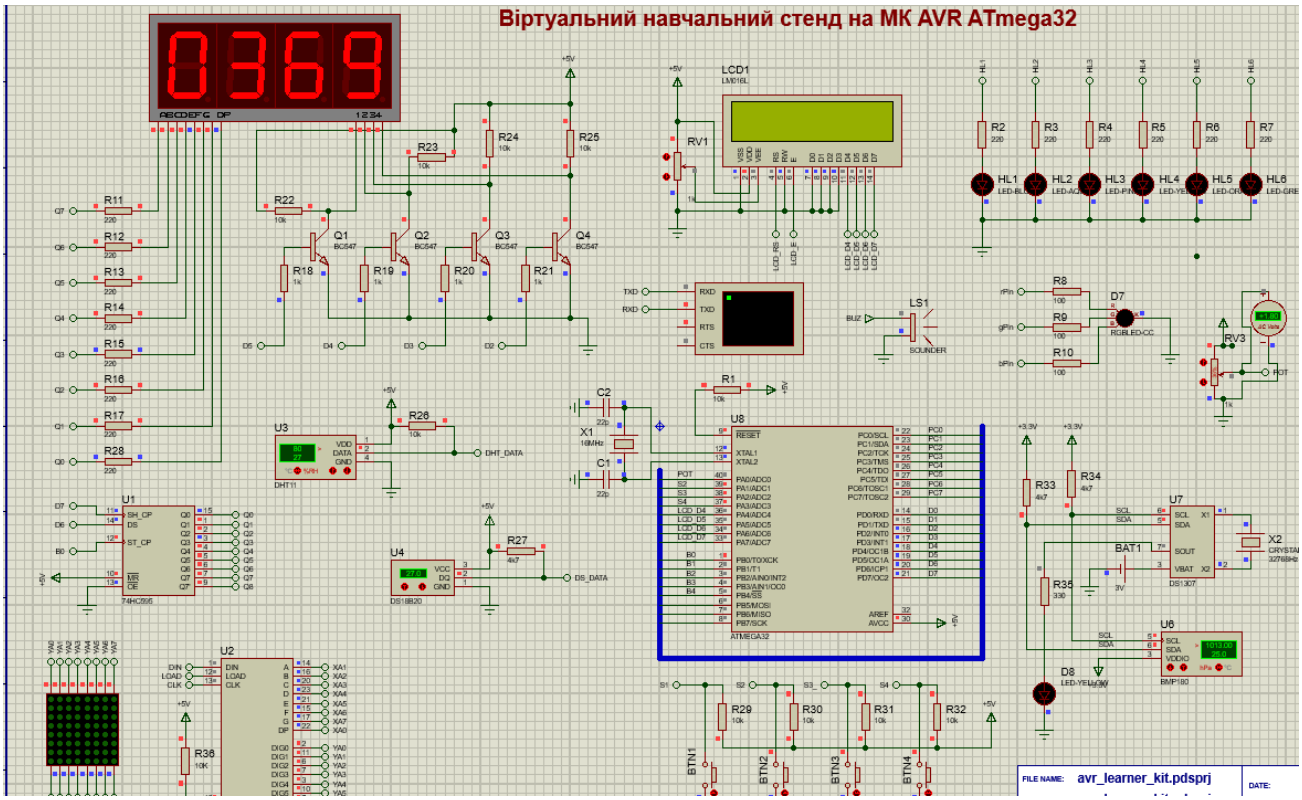

Рис. 2. Результат виконання

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

Код для Arduino IDE:

```
// Q7-A, Q6-B,...Q0-H
// SC - SH_CP(D7), SER - D6, ST_CP - D8, OE - GND, Button - A0

#define button A0
#define clockPin 7   // SH_CP
#define dataPin 6    // SER
#define latchPin 8   // ST_CP

byte current_digit;
int count = 9987;

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

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);

  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(count / 1000);
            digitalWrite(5, HIGH);
            break;
        case 2:
            disp((count / 100) % 10);
            digitalWrite(4, HIGH);
            break;
        case 3:
            disp((count / 10) % 10);
            digitalWrite(3, HIGH);
            break;
        case 4:
            disp(count % 10);
            digitalWrite(2, HIGH);
            break;
```

```
    }

    current_digit = (current_digit % 4) + 1;
}

void loop() {

    count=random(0, 9999);
    delay(200);

}

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);
}
```

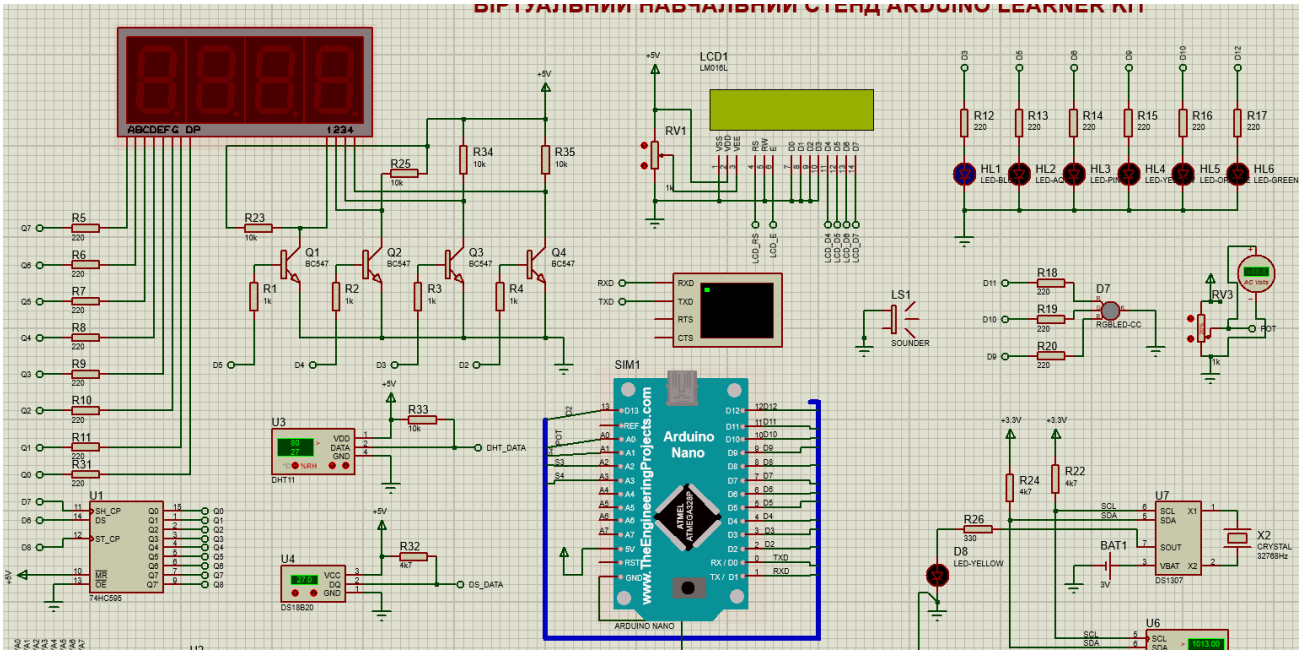

Рис. 3. Результат виконання

Код для МК AVR:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdbool.h>
unsigned char current_digit = 1;
```

```
int count = 0;
volatile uint32_t millis_count = 0;
unsigned long currentTime;
unsigned long lastInteraptTime;
unsigned char dataPin  = 6;
unsigned char clockPin = 7;
unsigned char latchPin = 0;
unsigned char digitPins[4] = {5,4,3,2};
bool buttonsPres=false;


uint8_t digits[10] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6};


ISR(TIMER0_COMP_vect)
{
millis_count++;
}

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

uint32_t millis(void)
```

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


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 disp_digit(unsigned char number) {

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



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

```
PORTD |= (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<<digitPins[0])|(1<<digitPins[1])|(1<<digitPins[2])|(1
<<digitPins[3]);
DDRB |= (1<<latchPin);

DDRA &= ~(1<<0);
PORTA |= (1<<0);
```

```
TCCR1A = 0;
TCCR1B = (1<<CS10);
TCNT1 = 0;
TIMSK |= (1<<TOIE1);
sei();

timer0_init();



while(1) {


        currentTime = millis();
        if(!(PINA & (1 << 0)) && (currentTime - lastInteraptTime>= 50) && buttonsPres== false)
        {
                count =(TCNT0 + TCNT1) % 10000;
                buttonsPres = true;
                lastInteraptTime= currentTime;
        }


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

```
        if(count > 9999) count = 9999;
        _delay_ms(50);
}
}
```

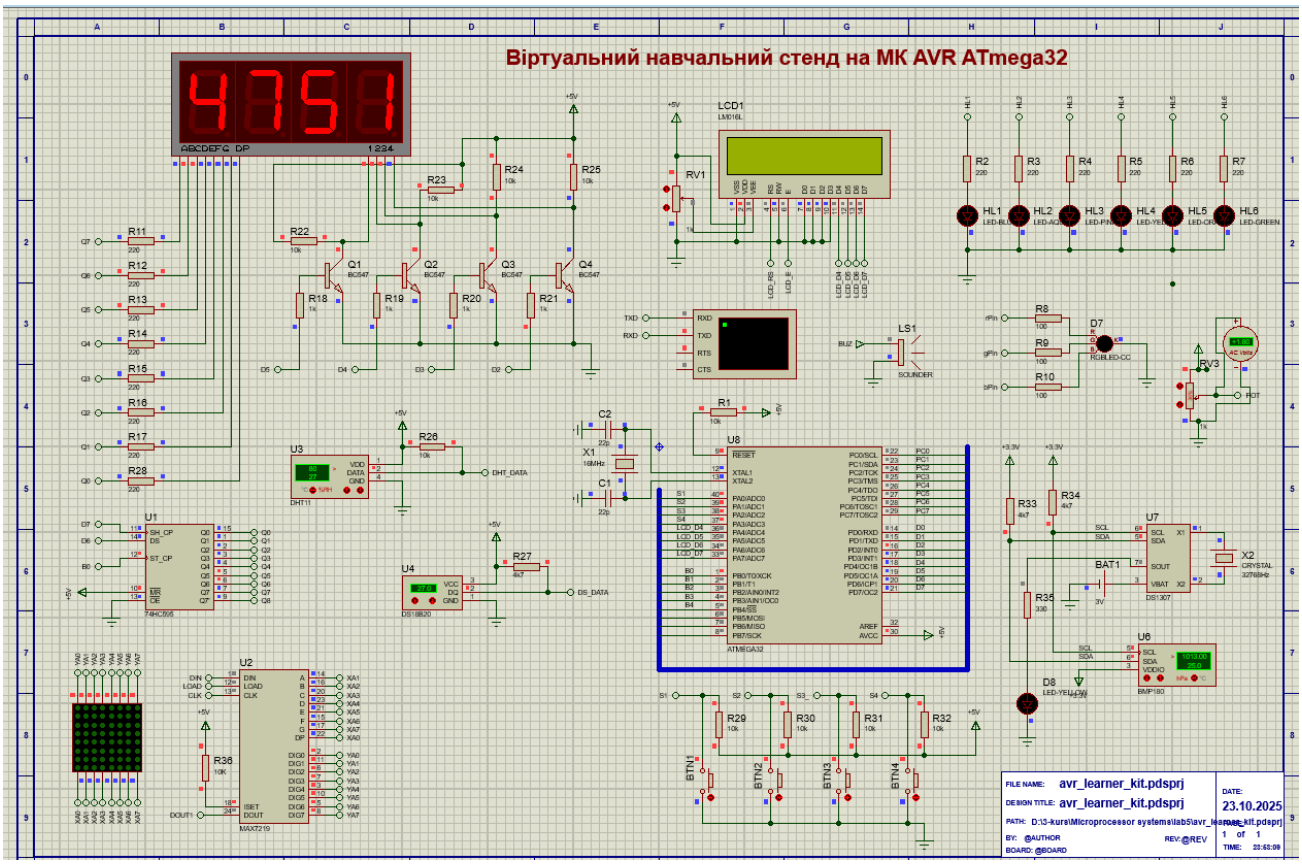

Рис. 4. Результат виконання

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

Код для Arduino IDE:

```
// Q7-A, Q6-B,...Q0-H

// SC - SH_CP(D7), SER - D6, ST_CP - D8, OE - GND, Button - A0


#define clockPin 7   // SH_CP

#define dataPin 6    // SER

#define latchPin 8   // ST_CP
```

```
unsigned long currentTime;
byte current_digit=1;
int count = 9987;
const int segmentPins[] = {13,8, 7, 6, 5, 4, 3,13};
const int nbrDigits = 4;
volatile bool buttonsPres[]={false,false,false,false};

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

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

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

void setup() {

   pinMode(5, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(2, OUTPUT);
   pinMode(clockPin, OUTPUT);
   pinMode(dataPin, OUTPUT);
   pinMode(latchPin, OUTPUT);

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

```
    pinMode(swPins[i],INPUT_PULLUP);
  }

  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(count / 1000);
      digitalWrite(5, HIGH);
      break;
    case 2:
      disp((count / 100) % 10);
      digitalWrite(4, HIGH);
      break;
    case 3:
      disp((count / 10) % 10);
      digitalWrite(3, HIGH);
```

```
            break;
        case 4:
            disp(count % 10);
            digitalWrite(2, HIGH);
            break;
    }

    current_digit = (current_digit % 4) + 1;
}

void loop() {
    currentTime = millis();
    Serial.println(count);

    if(count>9999)
    {
        count=0;
    }
    for(int i = 0; i < 4; i++)
    {
        if(digitalRead(swPins[i]) == LOW && buttonsPres[i] == false && currentTime
- lastInteraptTime[i] >= 50)
        {

            int digitValue = 1;
            for(int j=0; j < 3-i; j++) digitValue *= 10;
            count += digitValue;
            buttonsPres[i] = true;
            lastInteraptTime[i] = currentTime;
```

```
        }

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


}


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 = B11100110; 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);
}
```

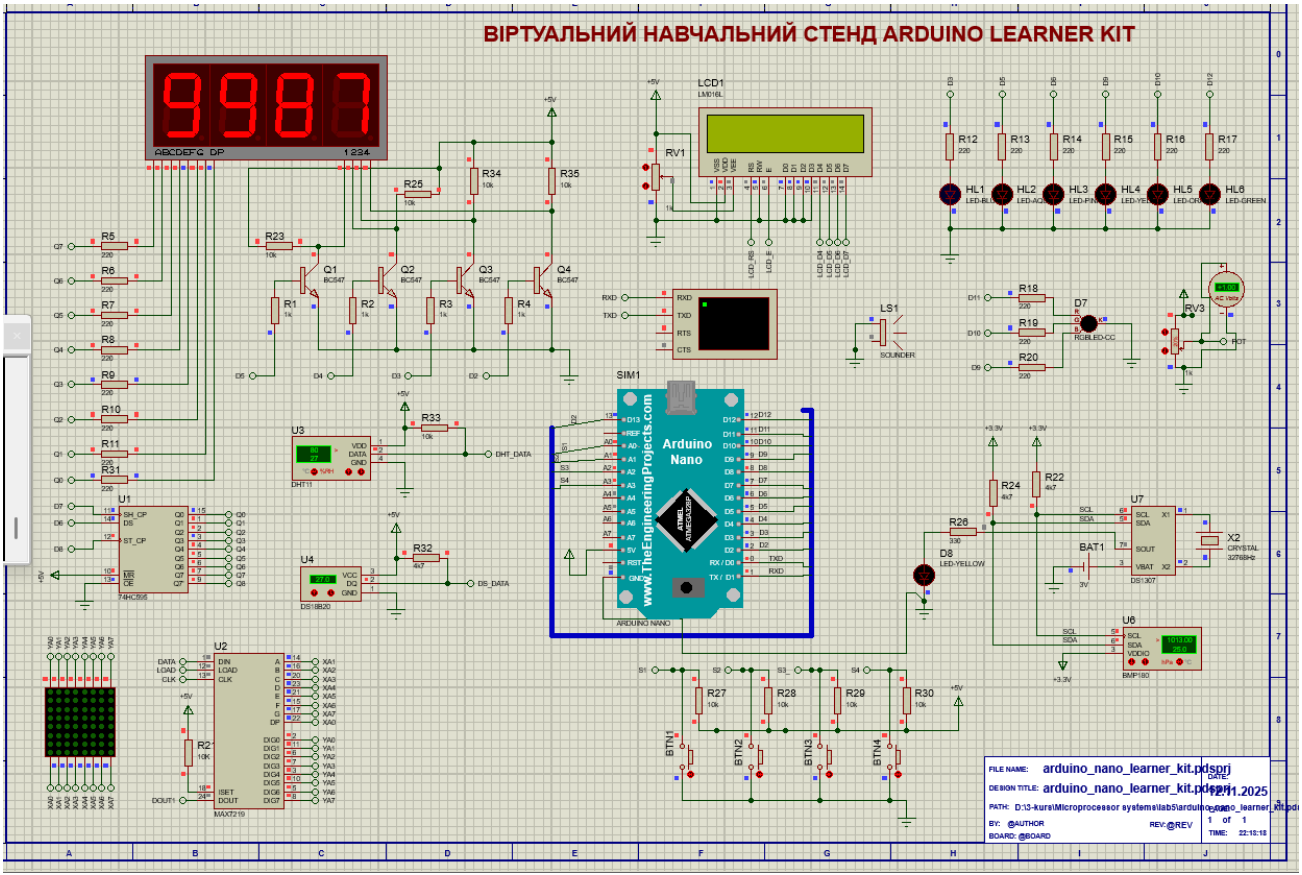

Рис. 5. Результат виконання

Код для МК AVR:

```
#define F_CPU 16000000UL
#include <avr/io.h>
```

```
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdbool.h>


const uint8_t  swPins[]={0,1,2,3};
unsigned char current_digit = 1;
int count = 1111;
volatile uint32_t millis_count = 0;
unsigned long currentTime;
volatile unsigned long lastInteraptTime[]={0,0,0,0};
unsigned char dataPin  = 6;
unsigned char clockPin = 7;
unsigned char latchPin = 0;
unsigned char digitPins[4] = {5,4,3,2};
volatile bool buttonsPres[]={false,false,false,false};

uint8_t digits[10] = {0xFC,0x60,0xDA,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xF6};


ISR(TIMER0_COMP_vect)
{
millis_count++;
}

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

```
OCR0 = 249;
TIMSK |= (1 << OCIE0);
sei();
}

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


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 disp_digit(unsigned char number) {

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

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


PORTD |= (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<<digitPins[0])|(1<<digitPins[1])|(1<<digitPins[2])|(1
<<digitPins[3]);
DDRB |= (1<<latchPin);



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

}



TCCR1A = 0;
TCCR1B = (1<<CS10);
TCNT1 = 0;
TIMSK |= (1<<TOIE1);
sei();

timer0_init();



while(1) {

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

```
                    if(!(PINA & (1 << swPins[i])) &&
(currentTime - lastInteraptTime[i] >= 50) && buttonsPres[i] == false)
                    {
                        int digitValue = 1;
                        for(int j=0; j < 3-i; j++) digitValue *= 10;
                        count += digitValue;

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

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

                if(count > 9999) count = 9999;
                _delay_ms(50);
}
}
```

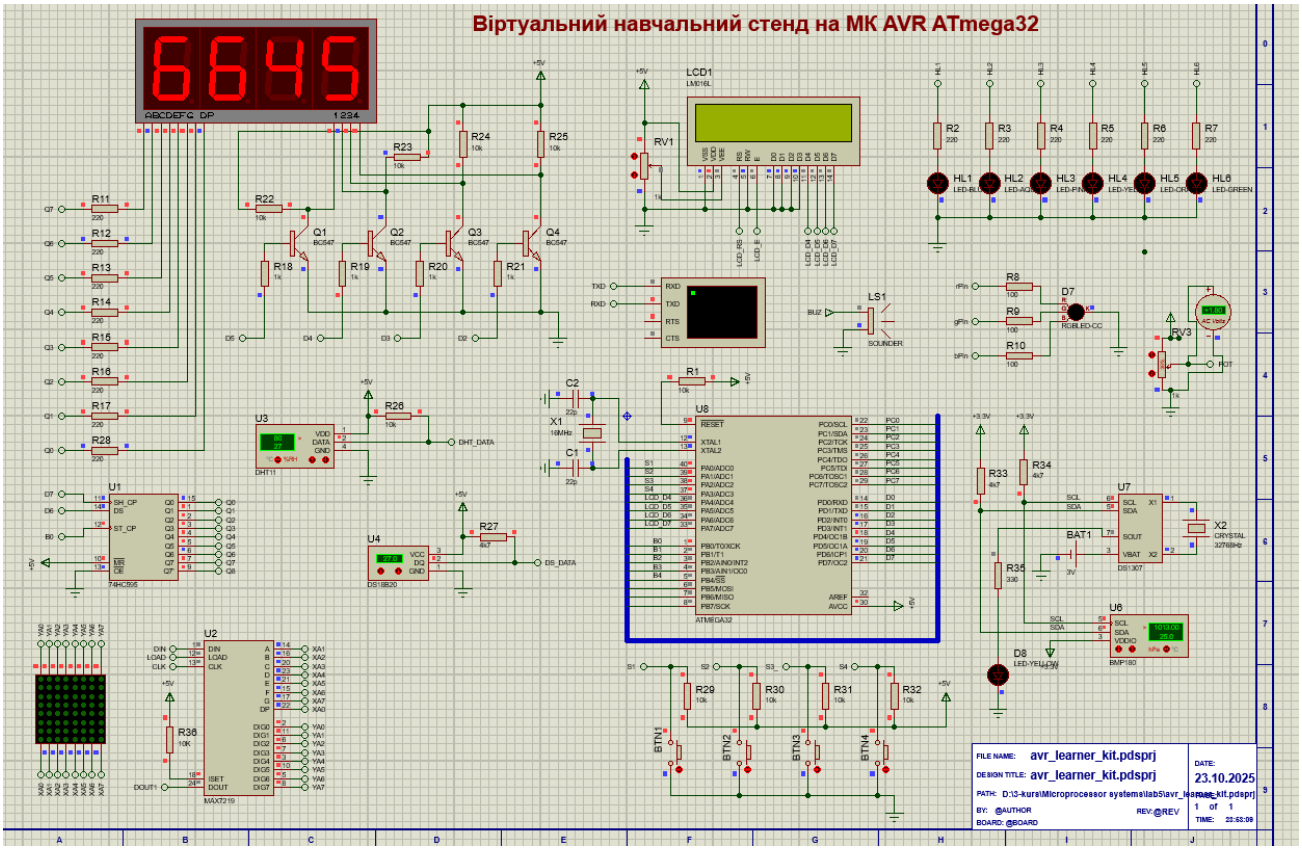

Рис. 6. Результат виконання

4. Реалізувати програму з застосуванням регістру зсуву 74HC595, яка відображає на семисегментному індикаторі напис «LOAD». За натисненням кнопки S1 відбражається напис «PLAY», а S2. «STOP».

Код для Arduino IDE:

```
// Q7-A, Q6-B,...Q0-H

// SC - SH_CP(D7), SER - D6, ST_CP - D8, OE - GND, Button - A0


#define clockPin 7   // SH_CP

#define dataPin 6    // SER

#define latchPin 8   // ST_CP


unsigned long currentTime;

byte current_digit=1;

int count = 9987;

const int segmentPins[] = {13,8, 7, 6, 5, 4, 3,13};

const int nbrDigits = 4;
```

```
volatile bool buttonsPres[]={false,false,false,false};

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

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

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

void setup() {

   pinMode(5, OUTPUT);
   pinMode(4, OUTPUT);
   pinMode(3, OUTPUT);
   pinMode(2, OUTPUT);
   pinMode(clockPin, OUTPUT);
   pinMode(dataPin, OUTPUT);
   pinMode(latchPin, OUTPUT);

   for(int i=0;i<4;i++)
  {
   pinMode(swPins[i],INPUT_PULLUP);
  }

   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();
    if(count==2)
    {
        switch (current_digit) {
        case 1:
            disp('S');
            digitalWrite(5, HIGH);
            break;
        case 2:
            disp('T');
            digitalWrite(4, HIGH);
            break;
        case 3:
            disp('O');
            digitalWrite(3, HIGH);
            break;
        case 4:
            disp('P');
            digitalWrite(2, HIGH);
            break;
```

```
}
}
else if(count==1)
{
   switch (current_digit) {
   case 1:
      disp('P');
      digitalWrite(5, HIGH);
      break;
   case 2:
      disp('L');
      digitalWrite(4, HIGH);
      break;
   case 3:
      disp('A');
      digitalWrite(3, HIGH);
      break;
   case 4:
      disp('Y');
      digitalWrite(2, HIGH);
      break;
}
}

else
{
   switch (current_digit) {
   case 1:
```

```
            disp('L');
            digitalWrite(5, HIGH);
            break;
        case 2:
            disp('O');
            digitalWrite(4, HIGH);
            break;
        case 3:
            disp('A');
            digitalWrite(3, HIGH);
            break;
        case 4:
            disp('D');
            digitalWrite(2, HIGH);
            break;
    }
    }

    current_digit = (current_digit % 4) + 1;
}

void loop() {
    currentTime = millis();
    Serial.println(count);

    if(count>9999)
    {
        count=0;
```

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



            if(i==0)
            {
                count=1;
            }
            if(i==1)
            {
                count=2;
            }
            buttonsPres[i] = true;
            lastInteraptTime[i] = currentTime;
        }


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



}
```

```
void disp(char number) {
   byte pattern;
   switch (number) {
      case 'L': pattern = B00011100; break;//L
      case 'O': pattern = B11111100; break;//O
      case 'A': pattern = B11101110; break;//A
      case 'D': pattern = B11111100; break;//D
      case 'P': pattern = B11001110; break;//P
      case 'Y': pattern = B01001110; break;//Y
      case 'S': pattern = B10110110; break;//S
      case 'T': pattern = B11100000; break;//T
      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);
}
```

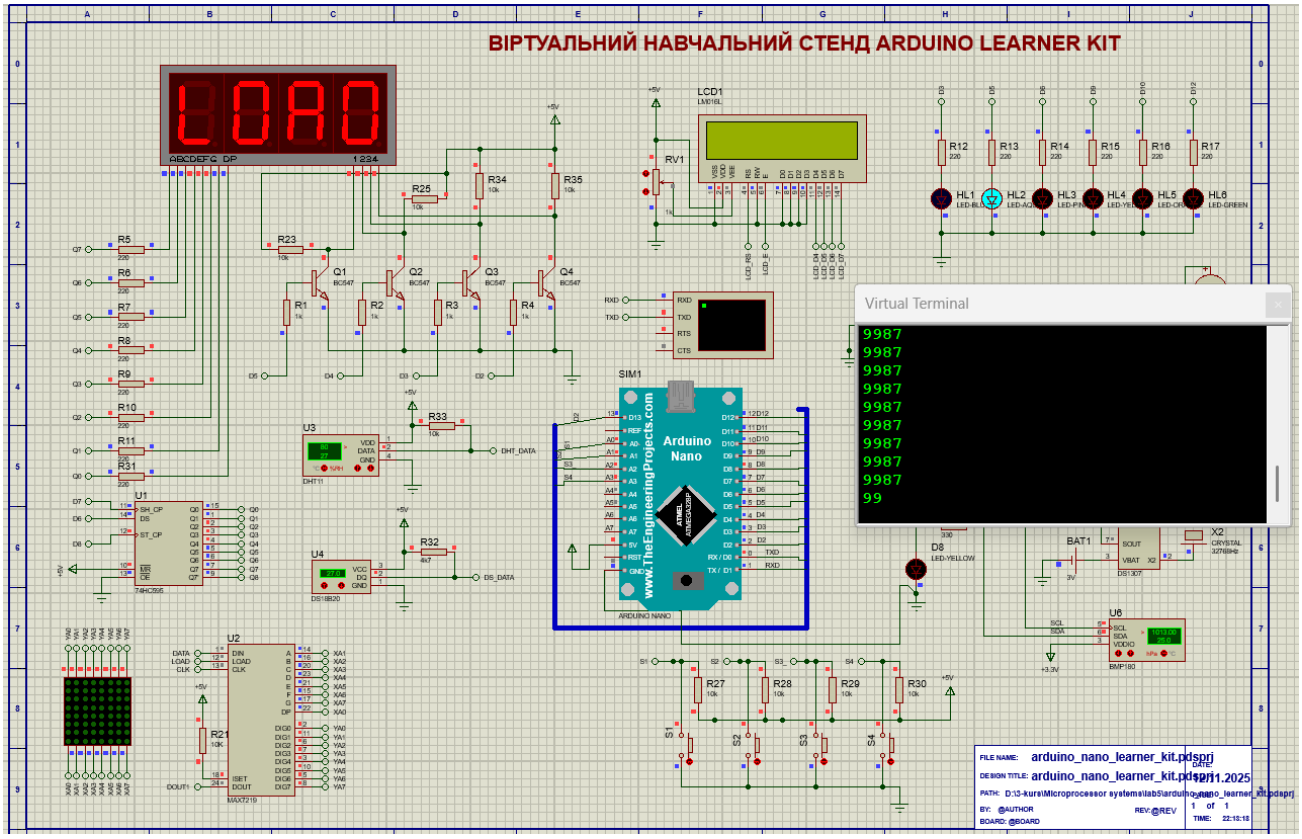

Рис. 7. Результат виконання

Код для МК AVR:

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


const uint8_t  swPins[]={0,1,2,3};
unsigned char current_digit = 1;
int count = 1111;
volatile uint32_t millis_count = 0;
unsigned long currentTime;
volatile unsigned long lastInteraptTime[]={0,0,0,0};
unsigned char dataPin  = 6;
```

```
unsigned char clockPin = 7;
unsigned char latchPin = 0;
unsigned char digitPins[4] = {5,4,3,2};
volatile bool buttonsPres[]={false,false,false,false};

int digits[10] = {0x1C,0xFC,0xEE,0xFC,0xCE,0x4E,0xB6,0xE0};



ISR(TIMER0_COMP_vect)
{
millis_count++;
}

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

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

```
}


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 disp_digit(unsigned char number) {

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


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


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


}
```

```
ISR(TIMER1_OVF_vect) {
unsigned char d;
if(count==1)
{
                        switch(current_digit) {
                                case 1: d =4 ; break;
                                case 2: d =0; break;
                                case 3: d =2; break;
                                case 4: d =5; break;
                        }
}
else if(count==2)
{
                        switch(current_digit) {
                                case 1: d = 6; break;
                                case 2: d = 7; break;
                                case 3: d = 1; break;
                                case 4: d = 4; break;
                        }
}
else
{
                        switch(current_digit) {
                                case 1: d =0 ; break;
                                case 2: d =1; break;
                                case 3: d =2; break;
                                case 4: d =3; break;
```

```
                                }
}

disp_digit(d);
current_digit = (current_digit % 4) + 1;
}

int main(void) {



DDRD |= (1<<dataPin)|(1<<clockPin)|(1<<digitPins[0])|(1<<digitPins[1])|(1<<digitPins[2])|(1<<digitPins[3]);
DDRB |= (1<<latchPin);



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

}



TCCR1A = 0;
TCCR1B = (1<<CS10);
TCNT1 = 0;
TIMSK |= (1<<TOIE1);
sei();
```

```
timer0_init();



while(1) {

                        currentTime = millis();
                            for(int i = 0; i < 2; i++)
                            {
                                if(!(PINA & (1 << swPins[i])) &&
(currentTime - lastInteraptTime[i] >= 50) && buttonsPres[i] == false)
                                {
                                    count=i+1;
                                    buttonsPres[i] = true;
                                    lastInteraptTime[i] = currentTime;
                                }


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

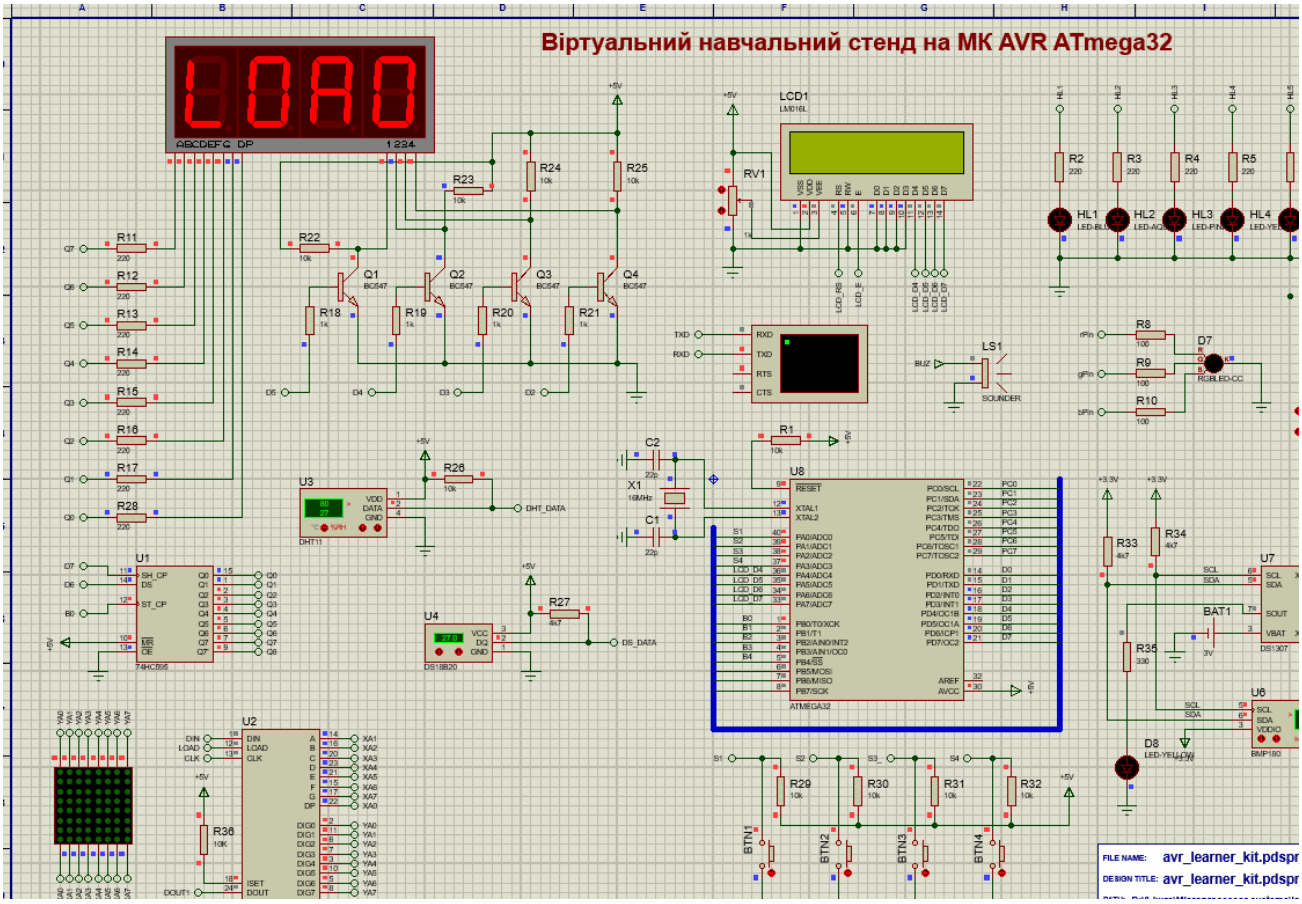

Рис. 8. Результат виконання

Висновок: У ході виконання лабораторної роботи я ознайомився з принципом роботи регистру зсуву 74HC595 разом з семисегментим індикатором та дослідив можливості програмної реалізації SPI інтерфейсу; закріпив навички програмування режимів роботи семисегментного індикатора з використанням переривань таймера/лічильника, що входить до складу мікроконтролера Arduino; закріпив навички роботи з цифровими портами, тактовими кнопками, масивами.