МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра **САП**

ЗВІТ

до виконання лабораторної роботи №1

На тему:” **Вивчення роботи портів вводу-виводу плати Arduino”**

з курсу “**Мікропроцесорні системи** ”

Варіант - 4

Виконав:

Студент гр.ПП-31

Гаврилюк Н. О.

Прийняв:

Доцент кафедри САП

Головатий А.І.

ЛЬВІВ - 2025

Мета: навчитися програмувати Arduino та дослідити роботу портів вводу-виводу мікроконтролера

Теоретичні відомості

Мікроконтролер. це логічне пристрій, який створено для управління іншими пристроями за допомогою логічних (цифрових) сигналів. Це означає, що максимум можна зняти з порта 40 мА, а рекомендується не більше 20 мА. Що станеться, якщо зняти з порта більше, ніж він може віддати? Він зламається. Що буде, якщо зняти з декількох портів більше, ніж може віддати мікроконтролер в цілому? Згорить мікроконтролер. Тому нічого потужніше світлодіода та маленькою пищалки до мікроконтролера підключати не можна. Ніяких моторчиків, лампочок, нагрівачів, потужних радіо-модулів та іншого живити від цифрових портів не можна. Цифрові порти служать для подачі команд іншим пристроям, наприклад реле / транзисторів для комутації навантажень.

Цифровий порт може перебувати в двох станах, вхід та вихід. Режим роботи вибирається за допомогою функції pinMode (pin, mode), де pin це номер порта, а mode це режим роботи (INPUT. вхід, OUTPUT. вихід, INPUT_PULLUP.вхід підтягнутий до живлення (PushUp)).

За замовчуванням всі порти налаштовані як входи (INPUT). Для зміни режимів роботи портів D2-D13, A0 -A5 можна використати фрагмент програми:

```
for ( byte i = 2; i < = 19; i ++ ) {
        pinMode (i, OUTPUT ) ; // робимо виходами
}
```

Для формування цифрового сигналу використовується функція digitalWrite (pin, value), де pin це цифровий порт Arduino, підписаний на платі як D; value. рівень сигналу: HIGH високий, LOW низький. Також можна використовувати цифри 0 та 1.

Порт, налаштований як OUTPUT, за замовчуванням має сигнал LOW. Цифровий порт може «вимірювати» напругу, але повідомити він може тільки про її відсутність (сигнал низького рівня, LOW) або наявність (сигнал високого рівня, HIGH). Відсутність напруги вважається проміжок від 0 до 2,1V. Відповідно від 2.1V до VCC (до 5V) мікроконтролер вважає за наявність сигналу високого рівня. Таким чином мікроконтролер може працювати з логічними пристроями, які шлють йому високий сигнал з напругою 3.3V, він такий сигнал прийме як HIGH. Не можна подавати на цифровий порт напругу вище напруги живлення мікроконтролера. Для читання рівня сигналу на порту використовується функція digitalRead (pin), де pin це номер порта на платі Arduino. Вони підписані як D , а

також порти A0-A5. Функція повертає 0, якщо сигнал низького рівня, і 1 якщо сигнал високого рівня.

Індивідуальне завдання

4	засвічує 7-й світлодіод на 18 секунд. Виконує паралельне включення та вимкнення 1-го, 3-го та 6-го світлодіодів з тривалістю світіння 200 мс та часом перебування в погашеному стані 0,3 с, протягом ~15 секунд.

Розробити програми біжучий вогник і мигання світлодіодами із заданим часовим інтервалом використати побітові операції |, &, <<, >>, ^.

Розробити програми для МК AVR і платформи Arduino, які виводять в латинській транслітерації на LCD ім'я, прізвище і шифр групи. Номер варіанту у двійковій системі засвітити на світлодіодах.

Хід роботи

Завдання 1

```
#define LED1 0
#define LED3 2
#define LED6 5
#define LED7 6

void setup() {
  pinMode(LED1, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED6, OUTPUT);
  pinMode(LED7, OUTPUT);
  digitalWrite(LED1,HIGH);
  digitalWrite(LED3,HIGH);
  digitalWrite(LED6,HIGH);
  digitalWrite(LED7,HIGH);

}
```

Рис.1 Створення "імен" для номерів виводів та функція setup

#define LED1-LED7 - Це **препроцесорні директиви**: LED1 відповідає виводу **0** (PD0)

LED2 → **1** (PD1), Тобто замість того, щоб кожен раз писати pinMode(0, OUTPUT), можемо писати pinMode(LED1, OUTPUT) — зручніше й зрозуміліше.

Я Відразу налаштовую потрібні піни на вивід високого сигналу, щоб умова виконувалась.

Роблю глобальні зміни та пишу функцію loop, яка буде реалізувати алгортм:

```
unsigned long currentTime2;
bool flag=false;
void loop() {
  unsigned long currentTime= millis();
  if(currentTime>=17000)
  {
    digitalWrite(LED7,LOW);
  }

  if(currentTime<=15000)
  {
    if(!flag && currentTime-currentTime2>=200)
    {
      flag=true;
      currentTime2+=200;
      digitalWrite(LED1,LOW);
      digitalWrite(LED3,LOW);
      digitalWrite(LED6,LOW);
    }
    if(flag && currentTime-currentTime2>=300)
    {
      flag=false;
      currentTime2+=300;
      digitalWrite(LED1,HIGH);
      digitalWrite(LED3,HIGH);
      digitalWrite(LED6,HIGH);
    }
  }
```

```
  else
  {
    digitalWrite(LED1,LOW);
    digitalWrite(LED3,LOW);
    digitalWrite(LED6,LOW);
  }
}
```

Рис.2 Функція loop із глобальною змінною часу

void setup() — виконується один раз при старті, налаштовує LCD і світлодіоди, а також вмикає **LED1, LED3, LED6, LED7** перед початком роботи.

void loop() — виконується безкінечно, даючи команди на виконання основного алгоритму.
digitalWrite(x,HIGH/LOW)-налаштовує рівень напруги на піні x.
HIGH → подати **1 (напругу ~5 В або 3.3 В)** → світлодіод загориться.

LOW → подати **0 (0 В, «земля»)** → світлодіод згасне

delay(ms) - Затримка виконання програми на вказану кількість мілісекунд.

Поки триває delay — програма «стоїть» і нічого іншого не виконує.

OUTPUT → налаштування піна як вихід для керування світлодіодом.

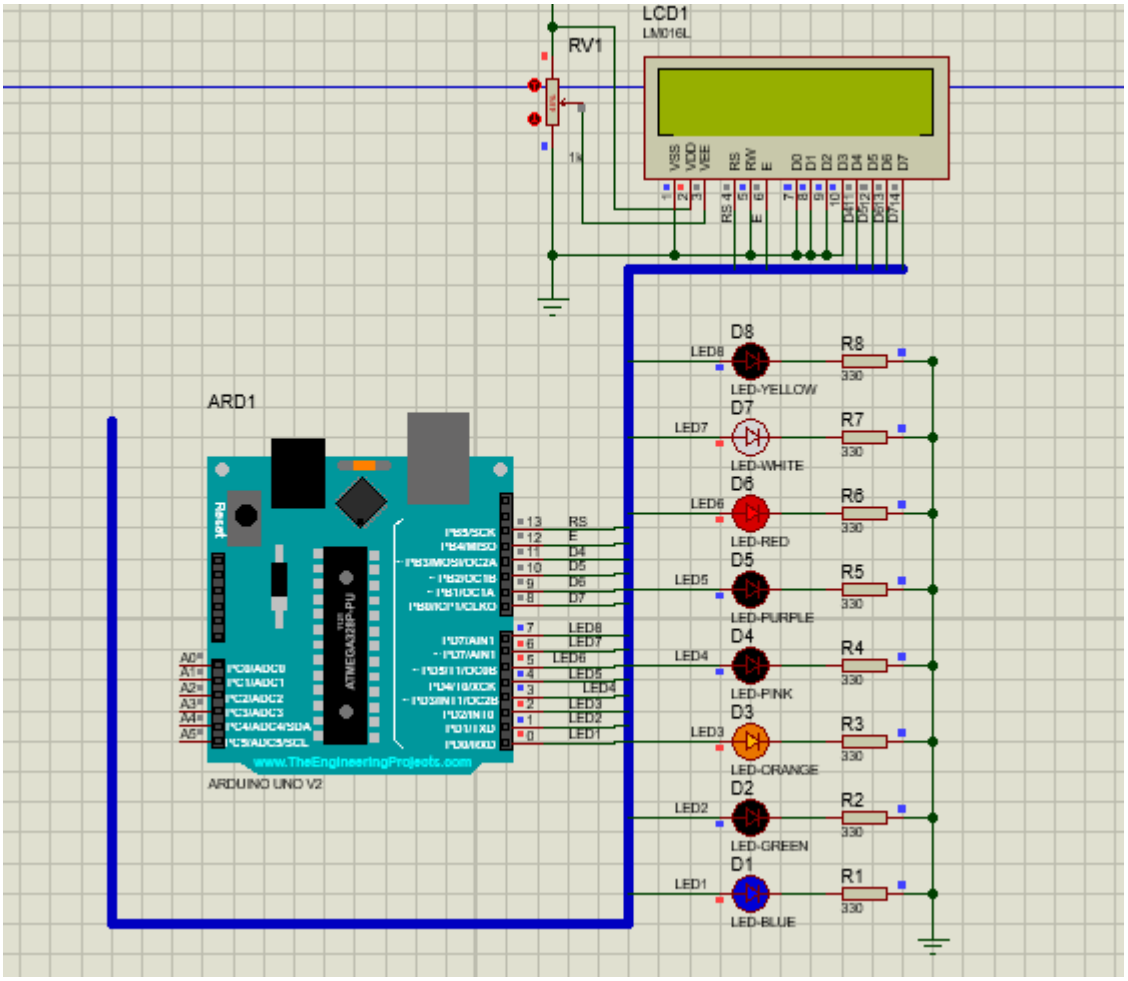

Рис. 3. Робота Світлодіодів

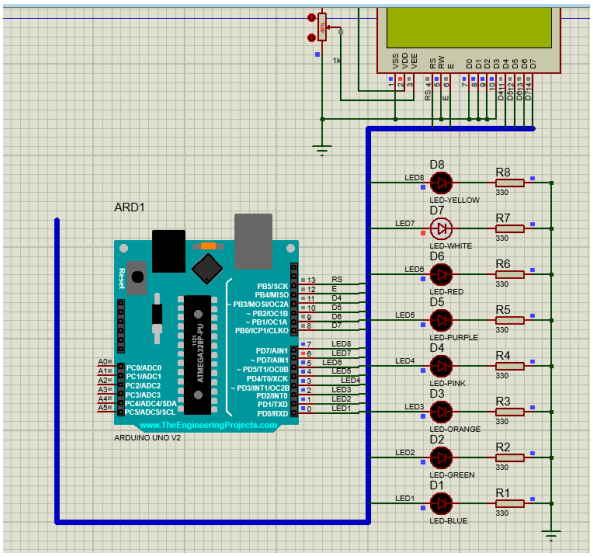

Рис.4. Світлодіоди LED1, LED3, LED6 за алгоритмом мигають 15 секунд, після чого вимикаються.

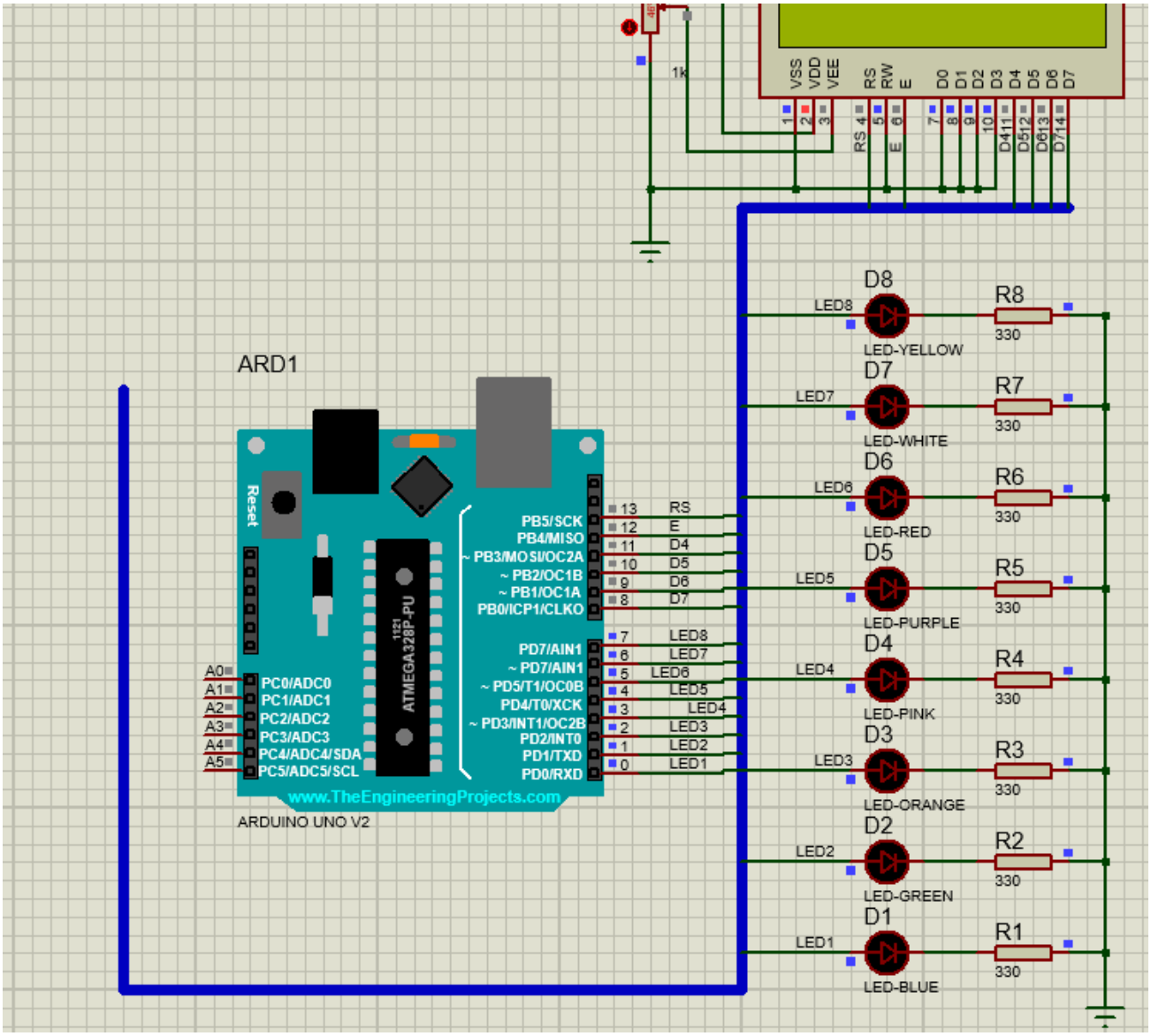

Рис.5. Світлодіод LED7 вимикаються на 17 секунді.

Також я написав код на МК AVR в Microship Studio від якого світлодіоди горять аналогічно.

Код:

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define LED1 PD0
#define LED3 PD2
#define LED6 PD5
#define LED7 PD6

unsigned long currentTime=0;
```

```
ISR(TIMER0_COMPA_vect) {
	currentTime++;
}

void initTimer0() {
	TCCR0A = (1 << WGM01);
	OCR0A = 249;
	TCCR0B = (1 << CS01) | (1 << CS00);
	TIMSK0 = (1 << OCIE0A);
	sei();
}

int main(void) {
	DDRD |= (1<<LED1)|(1<<LED3)|(1<<LED6)|(1<<LED7);
	PORTD |= (1<<LED1)|(1<<LED3)|(1<<LED6)|(1<<LED7);
   initTimer0();
	unsigned long currentTime2=currentTime;
	bool flag=false;
	while(1)
	{
		if(currentTime>=17000)
		{
			PORTD &= ~(1<<LED7);
		}

		if(currentTime<=15000)
		{
			if(!flag && currentTime-currentTime2>=200)
			{
				flag=true;
				currentTime2+=200;
```

```
                PORTD &= ~(1<<LED1);
                PORTD &= ~(1<<LED3);
                PORTD &= ~(1<<LED6);
            }
            if(flag && currentTime-currentTime2>=300)
            {
                flag=false;
                currentTime2+=300;
                PORTD |= (1<<LED1)|(1<<LED3)|(1<<LED6)|(1<<LED7);
            }
        }
        else
        {
            PORTD &= ~(1<<LED1);
            PORTD &= ~(1<<LED3);
            PORTD &= ~(1<<LED6);
        }

    }
}
```

Розробити програми біжучий вогник і мигання світлодіодами із заданим часовим інтервалом використати побітові операції |, &, <<, >>, ^.

```
#define DELAY_RUN   200   // мс для біжучого вогника
#define DELAY_BLINK 500   // мс для мигання

void setup() {
  DDRD = 0xFF;   // Всі PD0–PD7 (D0–D7) як виходи
}

void loop() {
  // === 1. Біжучий вогник вперед (0 → 7) ===
  uint8_t led = 0x01;   // Починаємо з LED1 (PD0)
  for (int i = 0; i < 8; i++) {
    PORTD = led;         // Засвітити тільки цей світлодіод
    delay(DELAY_RUN);
    led = led << 1;      // Зсув вліво (<<)
  }

  // === 2. Біжучий вогник назад (7 → 0) ===
  led = 0x80;            // Починаємо з LED8 (PD7)
  for (int i = 0; i < 8; i++) {
    PORTD = led;
    delay(DELAY_RUN);
    led = led >> 1;      // Зсув вправо (>>)
  }
```

Рис.6. Константи затримок функція setup та частина функції loop

DELAY_RUN = швидкість бігання світлодіодів (0.2 с).
DELAY_BLINK = швидкість миготіння (0.5 с).
DDRD = 0xFF → у двійковій формі це 11111111.
Це означає: всі 8 бітів (D0…D7) налаштовані як **виходи**.

Loop - Програма виконується безкінечно у 4 етапи.

1. Біжучий вогник вперед (зліва направо)

led = 0x01 → 00000001 → горить лише LED1.
Кожна ітерація робить << 1 (зсув вліво):

00000001 → 00000010 → 00000100 → … → 10000000.
Світлодіод "біжить" від **LED1 до LED8**.

2. Біжучий вогник назад (справа наліво)

led = 0x80 → 10000000 → горить лише LED8.
Кожна ітерація робить >> 1 (зсув вправо):

10000000 → 01000000 → 00100000 → … → 00000001.
Світлодіод "біжить" від **LED8 до LED1**.

```
  // === 3. Мигання всіма LED ===
  for (int i = 0; i < 5; i++) {
    PORTD = PORTD | 0xFF;   // Всі біти = 1 → усі LED ON
    delay(DELAY_BLINK);

    PORTD = PORTD & 0x00;   // Всі біти = 0 → усі LED OFF
    delay(DELAY_BLINK);
  }

  // === 4. Інверсія XOR ===
  uint8_t pattern = 0xAA;   // 10101010
  for (int i = 0; i < 8; i++) {
    PORTD = pattern;
    delay(300);
    pattern = pattern ^ 0xFF;  // інвертувати (XOR з 11111111)
  }
}
```

Рис.7 Миготіння всіма LED та інверсія 3.

Мигання всіма LED

5 разів:

| 0xFF (OR з 11111111) → усі біти стають 1 → **всі LED увімкнені**.

& 0x00 (AND з 00000000) → усі біти стають 0 → **всі LED вимкнені**.

Світлодіоди блимають синхронно. 00 11

4. Інверсія XOR

pattern = 0xAA → 10101010 (світяться LED2,4,6,8). Далі
pattern ^ 0xFF → інверсія:

10101010 → 01010101 (світяться LED1,3,5,7).
Кожна ітерація перемикає «шахматний» малюнок.

<< i >> — для **руху "вогника"** по діодах.

| — для **встановлення всіх бітів у 1** (усі LED вкл.).

& — для **обнулення всіх бітів** (усі LED викл.).

^ — для **швидкої інверсії стану** (наприклад, "шахматка" 10101010 ↔ 01010101).

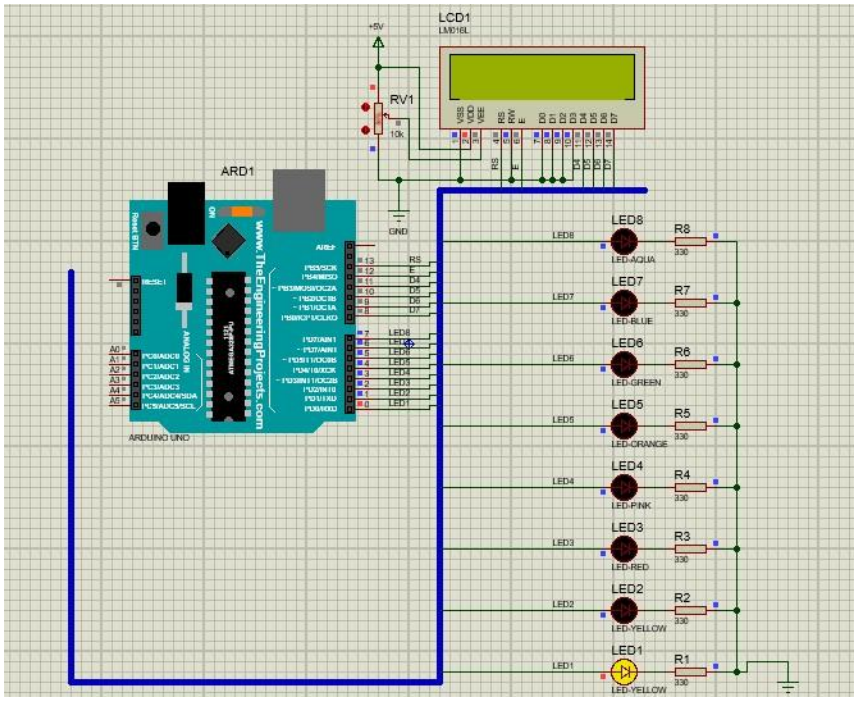

Рис.8 Світиться тільки 1

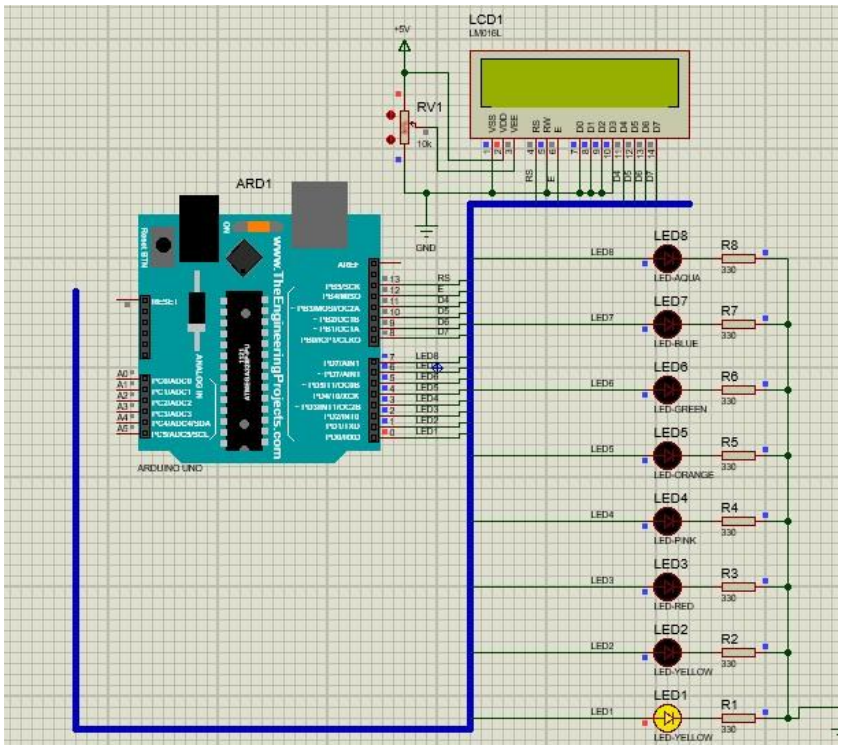

Рис.9 Світиться тільки 2

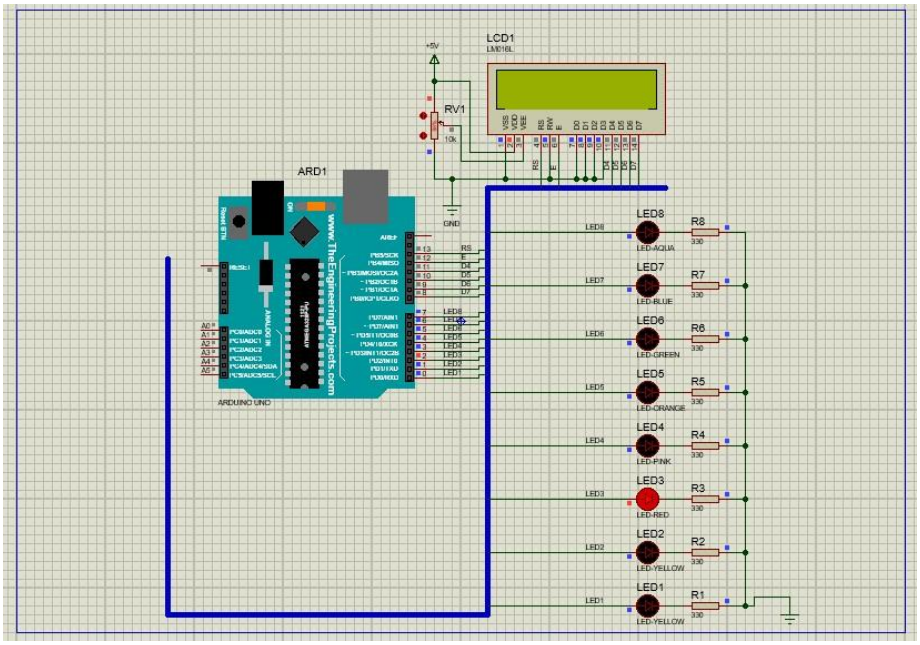

Рис.10 Світиться тільки 3…

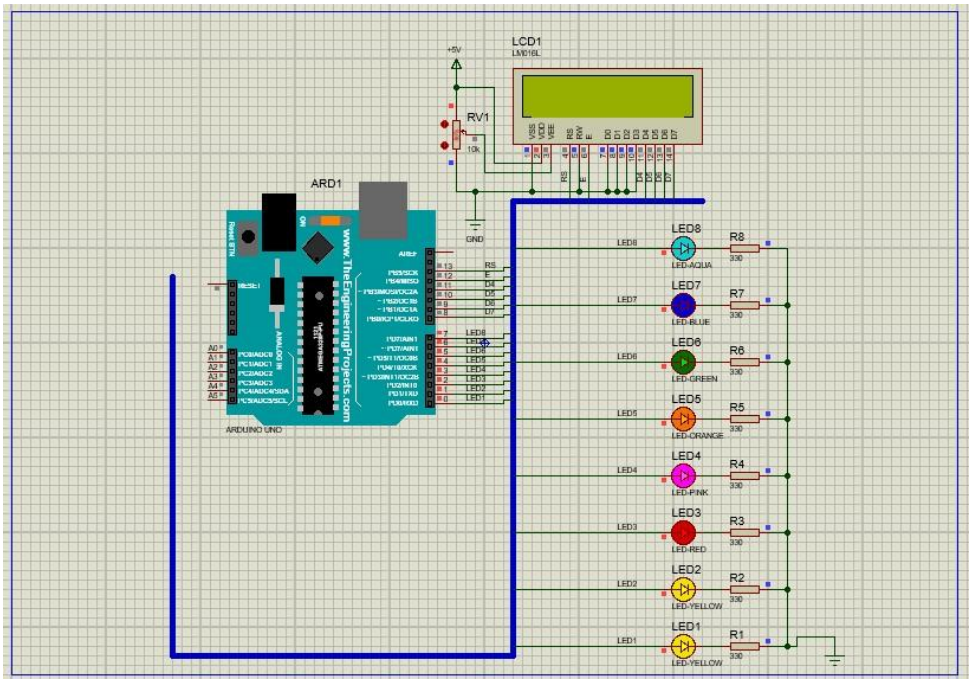

Рис.11 Всі одночасно світяться

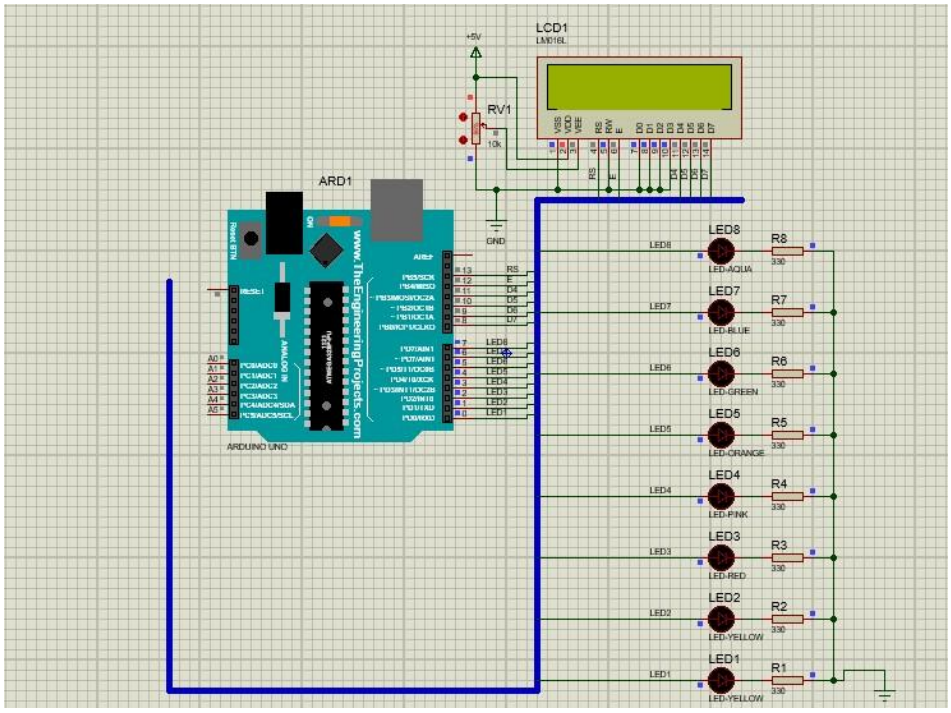

Рис.12 Всі загасли і так 5 раз…

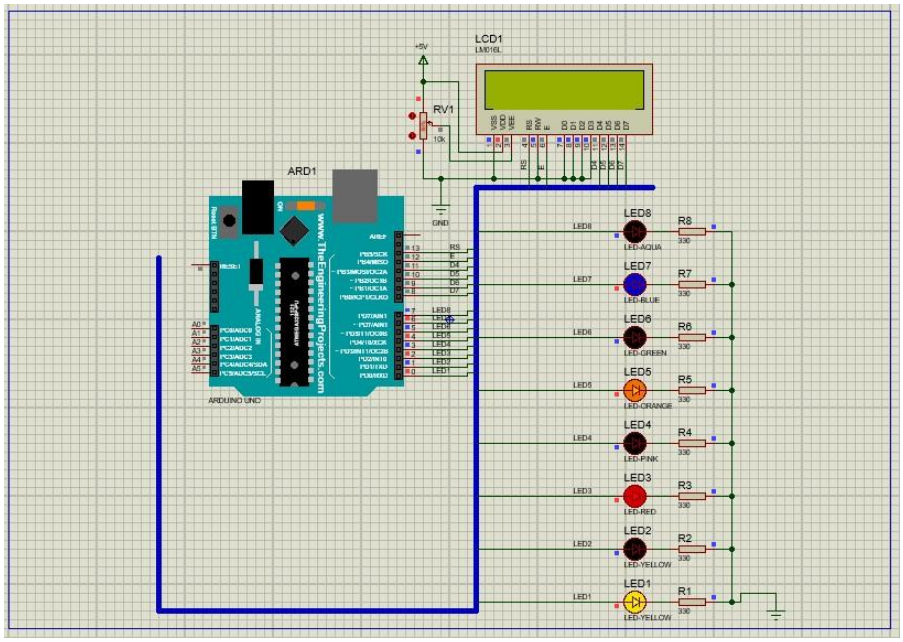

Рис.13 Інверсія(горять 1 3 5 7)

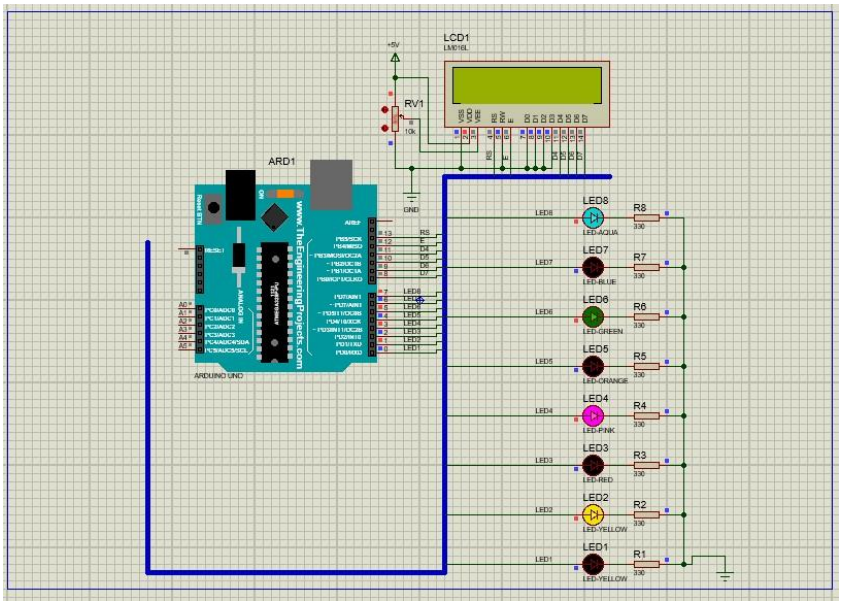

Рис.14 Інверсія(горять 2 4 6 8)

Завдання 2

Розробити програми для МК AVR і платформи Arduino, які виводять в латинській транслітерації на LCD ім’я, прізвище і шифр групи. Номер варіанту у двійковій системі засвітити на світлодіодах.

Код у Arduino Ide:

```
#include <LiquidCrystal.h>

#define LED1 0

#define LED2 1

#define LED3 2

#define LED4 3

#define LED5 4

#define LED6 5

#define LED7 6

#define LED8 7


LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

void setup() {

  pinMode(LED1, OUTPUT);
```

```
  pinMode(LED2, OUTPUT);
  pinMode(LED3, OUTPUT);
  pinMode(LED4, OUTPUT);
  pinMode(LED5, OUTPUT);
  pinMode(LED6, OUTPUT);
  pinMode(LED7, OUTPUT);
  pinMode(LED8, OUTPUT);
     //LCD
  lcd.begin(16,2);
  lcd.print("Nazar Hvryliuk");
  lcd.setCursor(0,1);
  lcd.print("PP-31");
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, HIGH);
  digitalWrite(LED3, HIGH);
  digitalWrite(LED4, HIGH);
  digitalWrite(LED5, HIGH);
}

void loop() {
}
```

Я не програмував фукцію loop через те що алгоритм на виконання має датись лише 1 раз, тому немає потреби занисити його у loop.

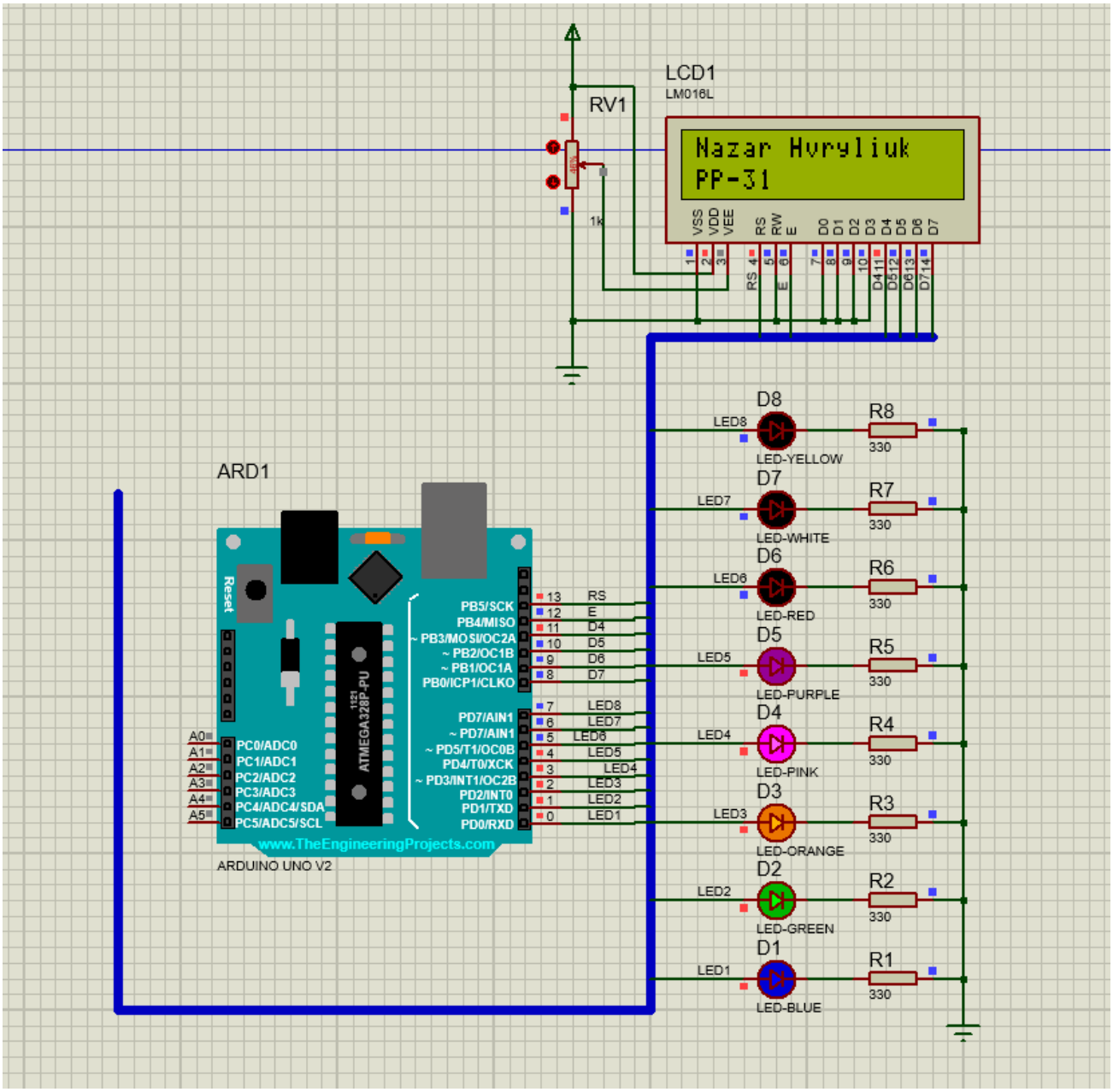

Рис. 15. Результат роботи.

Код написаний у МК AVR:

```
#include <avr/io.h> /* Defines pins, ports, etc */

#define F_CPU 16000000

#include <util/delay.h> /* Functions to waste time */

#include <stdio.h> //input/output library

#include "lcd_lib.h" // LCD library
```

```
#define rgbLED_DDR DDRB
#define rgbLED_PORT PORTB

#define LEDS_DDR DDRD
#define LEDS_PORT PORTD

int main(void) {
	int i = 3;
	DDRD |= (1<<PD0)|(1<<PD1)|(1<<PD2)|(1<<PD3)|(1<<PD4);
	PORTD |= (1<<PD0);
	PORTD |= (1<<PD1);
	PORTD |= (1<<PD2);
	PORTD |= (1<<PD3);
	PORTD |= (1<<PD4);

	/* Ініціалізація LCD */
	LCDinit();
	LCDcursorOFF();
	LCDstring("Nazar Havryliuk");
	LCDGotoXY(0,1);
	LCDstring("PP-31");
	_delay_ms(2000);
	//LCDclr();

	// ------ Event loop ------ //
	while (1) {
	}
	return (0); /* This line is never reached */
}
```

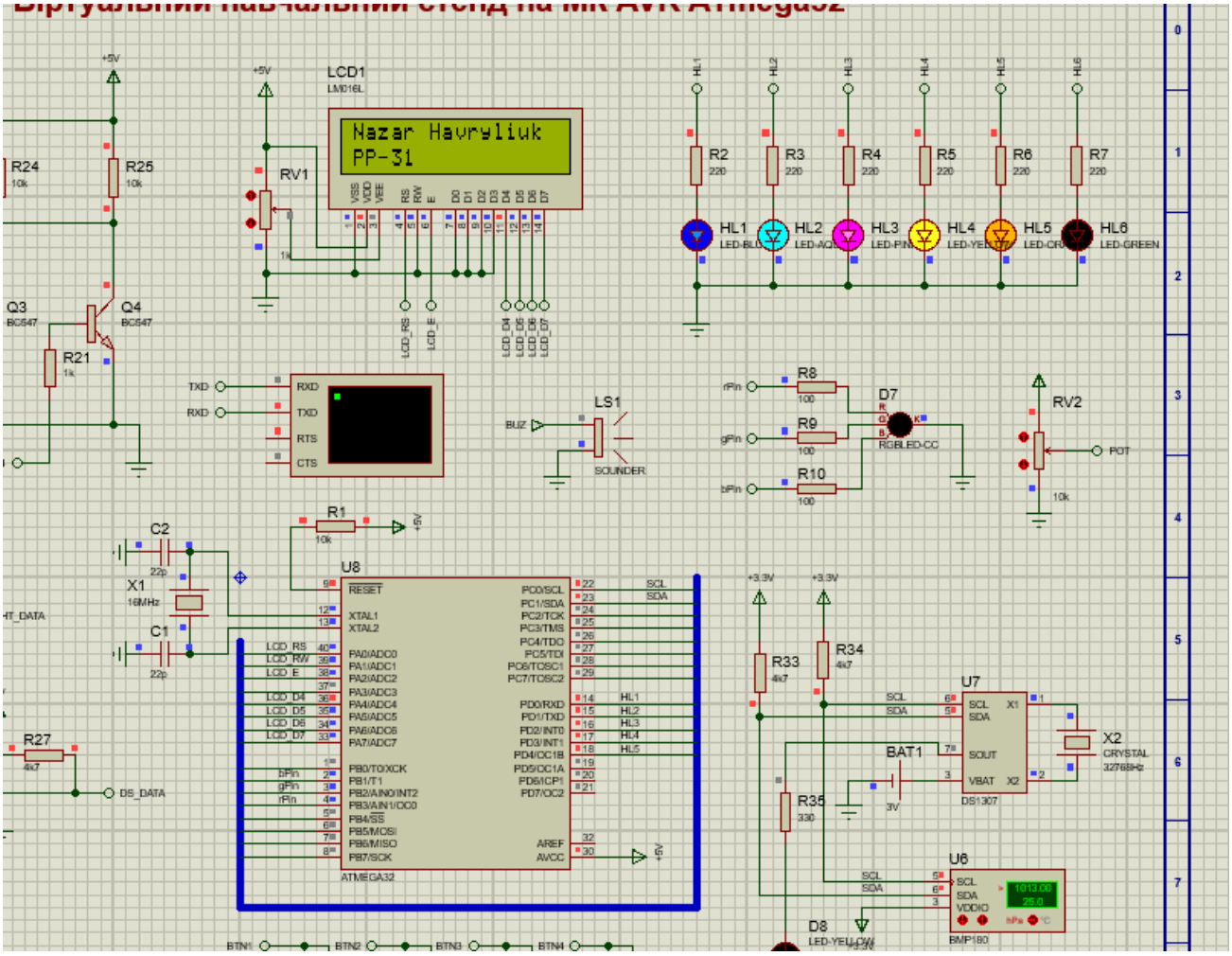

Рис. 16. Результат роботи

Короткі відповіді на контрольні запитання

1.Зазначте основні сфери застосування Arduino?

Arduino використовується для:

навчання програмуванню мікроконтролерів; створення прототипів електронних пристроїв; автоматизації (роботи, розумний дім, системи керування); IoT (інтернет речей); керування сенсорами, двигунами, дисплеями.

2. Які є основні функції цифрового введення та виведення? pinMode(pin, mode) – налаштовує режим роботи піна (вхід/вихід).

digitalWrite(pin, value) – встановлює рівень сигналу (HIGH/LOW) на вихідному піні. **digitalRead(pin)** – зчитує рівень сигналу (HIGH/LOW) з вхідного піна.

3. Які значення має параметр value функції digitalWrite()?

Може бути:

HIGH – подати логічну «1» (5 В або 3.3 В залежно від плати).

LOW – подати логічний «0» (0 В).

4. Чи повертає значення функція pinMode(pin, mode)?

Ні, ця функція нічого не повертає (void). Вона лише встановлює режим роботи піна.

5. Опишіть алгоритм ініціалізації порту вводу-виводу Arduino?

Вибрати номер піна.

Викликати pinMode(pin, INPUT/OUTPUT/INPUT_PULLUP) для налаштування.

При потребі – використовувати digitalWrite або digitalRead.

6. Чому в лабораторній роботі світлодіоди підключені без використання транзисторів?

Тому що Arduino може безпосередньо подавати невеликий струм (до ~20 мА) на LED. Для простих експериментів цього достатньо. Транзистори потрібні лише при керуванні потужними навантаженнями.

7. Що виконує інструкція ++pin?

int – ціле число зі знаком (може бути від’ємним).

unsigned int – ціле число без знака (тільки додатні значення). Діапазон на Arduino Uno (16-бітний int):

int = від -32768 до 32767 unsigned int =

від 0 до 65535

9. Що повертає функція millis()?

Функція повертає кількість мілісекунд, що минула з моменту запуску програми на Arduino (тип unsigned long). Використовується для вимірювання часу.

Висновок: У ході виконання лабораторної роботи було розглянуто роботу мікроконтролера AVR на платформі Arduino та основні принципи керування світлодіодами й LCD-дисплеєм. Було налаштовано порти вводу-виводу, реалізовано алгоритми біжучого вогника, миготіння та інверсії світлодіодів. Також відпрацьовано відображення інформації на LCD та статичний вивід даних. Окремо були проаналізовані ключові функції Arduino (pinMode, digitalWrite, delay, millis) та базові поняття щодо змінних. У результаті закріплено практичні навички програмування й роботи з апаратною частиною Arduino.