ЛАБОРАТОРНА РОБОТА №14
Підключення семисегментного індикатора

Мета роботи: вивчення структури Arduino Uno організацію портів та форматів його команд, а також удосконалення початкових навичок програмування. Реалізація пристрою відображення інформації на семисегментному індикаторі

Короткі теоретичні відомості

Семисегментний індикатор - це набір світлодіодів, зібраних в єдиному корпусі. Світлодіоди утворюють сегменти-палички, шляхом підсвічування яких можна формувати цифри. Індикатор можна безпосередньо підключати до Arduino (рис. 14.1, рис 14.2), але при цьому буде задіяно 7 виводів контролера на один дисплей, і доведеться замислитись над додаванням в програму коду, що реалізує відображення цифр на індикаторі (рис. 14.3). Цей спосіб поганий також і тим, що обмежена кількість підключених індикаторів - більше двох розрядів відобразити вже не вийде, на контролері не вистачить виводів. Для вирішення цієї проблеми пропонується використовувати спеціальну мікросхему, що називається 7-ми сегментним драйвером (74HC595).

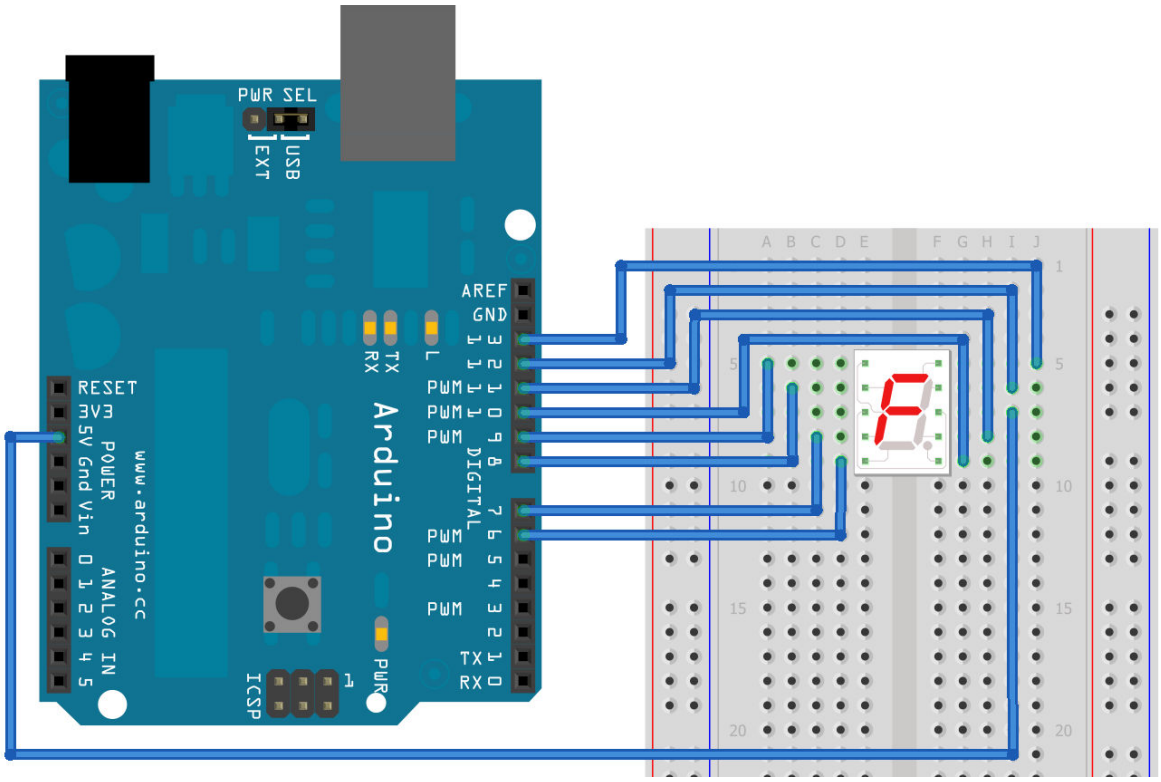

Рисунок 14.1 – Підключення та зовнішній вигляд макета

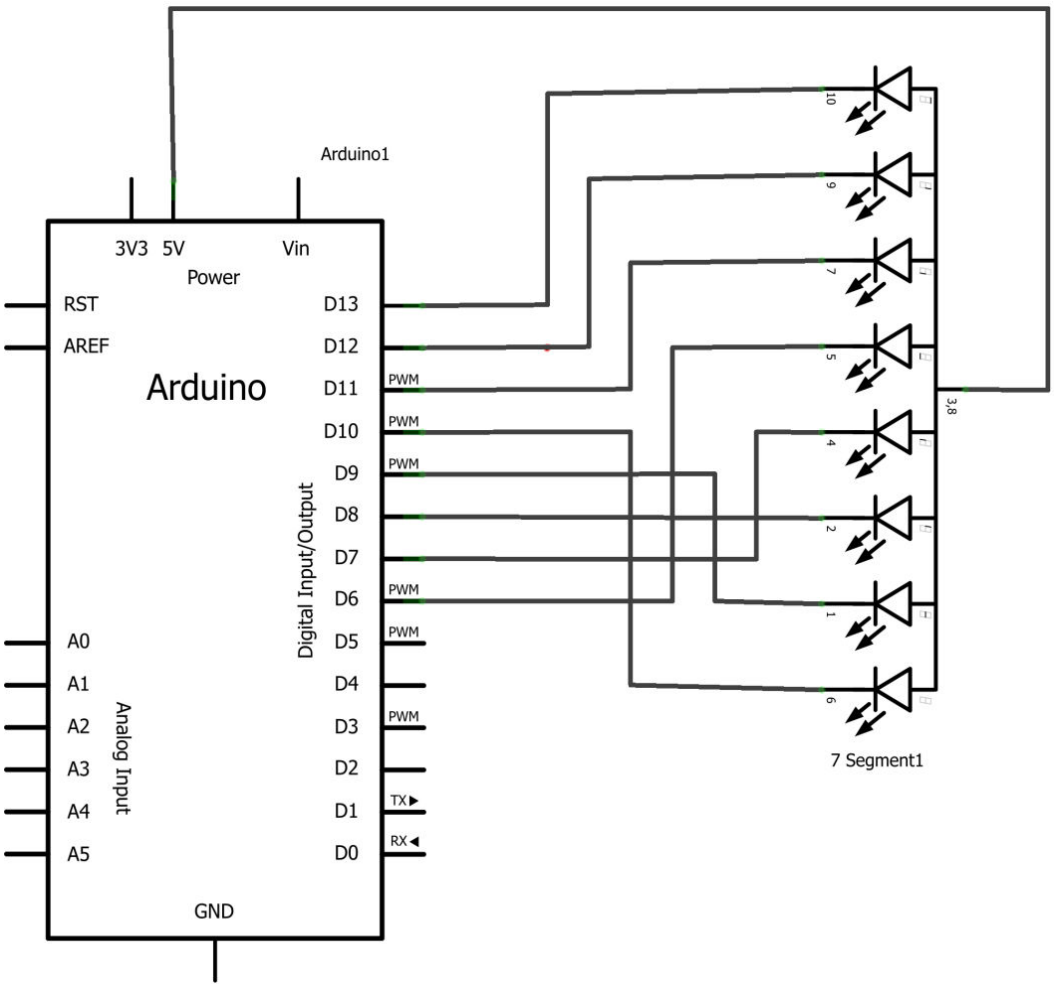

Рисунок 14.2 – Електрична принципова схема макета

7-сегментний дисплей

Зазвичай одиночний семисегментний дисплей має 10 контактів, з яких 2 - загальні, а інші 8 - окремі сегменти, включаючи десяткову крапку в правому нижньому кутку. Семисегментні індикатори можуть бути із загальним анодом або із загальним катодом. Необхідно чітко розуміти, якого саме типу семисегментний індикатор знаходиться у вашому користуванні.

На рис. 14.4 com - загальні контакти. Якщо подати до com потенціал +5 В, а до виводу A – 0 В, то загориться, відповідно, сегмент A. Якщо у дисплея загальний катод, то потенціали потрібно підключати навпаки.

Якщо ми представимо цифри на 7-сегментному дисплеї у вигляді байтів, то, подаючи їх по SPI шині в регістр побітно, в момент замикання будемо отримувати потрібну цифру. Таким чином, ми отримуємо можливість керувати семисегментним дисплеєм по чотирьох провідниках, причому кожен наступний семисегментний індикатор не буде потребувати великої кількості додаткових провідників.

```
int a=13;
int b=12;
int c=11;
int d=10;
int e=9;
int f=8;
int g=7;
int p=6;
 void setup ()
 {
   pinMode (a, OUTPUT);
   pinMode (b, OUTPUT);
   pinMode (c, OUTPUT);
   pinMode (d, OUTPUT);
   pinMode (e, OUTPUT);
   pinMode (f, OUTPUT);
   pinMode (g, OUTPUT);
   pinMode (p, OUTPUT);
 }
 void loop ()
 {
   digitalWrite (a, LOW);
   digitalWrite (b, LOW);
   digitalWrite (c, LOW);
   digitalWrite (d, LOW);
   digitalWrite (e, LOW);
   digitalWrite (f, LOW);
   digitalWrite (g, LOW);
   digitalWrite (p, LOW);
 }
```

Рисунок 14.3 – Лістинг програми

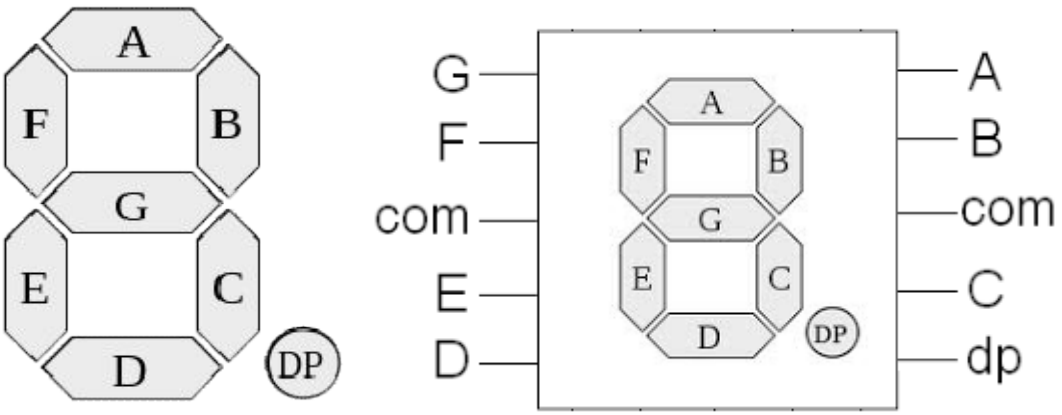

Рисунок 14.4 – Нумерація сегментів та схема розташування виводів семисегментного індикатора

Шина SPI

SPI (Serial Peripheral Interface) - це послідовний синхронний стандарт передачі даних, який призначений для спілкування контролера з периферією. Більшість сучасних мікроконтролерів підтримують SPI, оскільки вона дуже проста і зручна у використанні. У Arduino є бібліотека для роботи з SPI, яка складається всього з шести простих функцій.

Один з пристроїв на шині є провідним (найчастіше, сам контролер), а інші - відомими. Дані передаються по шині від провідного до одного обраному веденого або від веденого до ведучого синхронно по тактується сигналу ведучого.

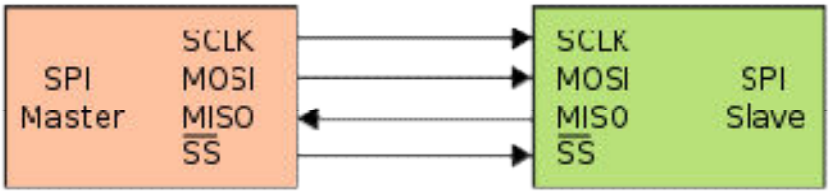

Рисунок 14.5 – Структурна схема шини SPI

Для передачі даних в інтерфейсі SPI використовуються чотири сигнали:

MOSI або SI — вихід ведучого, вхід веденого (англ. Master Out Slave In). Служить для передачі даних від ведучого пристрою до веденого.

MISO або SO — вхід ведучого, вихід веденого (англ. Master In Slave Out). Служить для передачі даних від веденого пристрою до ведучого.

SCLK або SCK — послідовний тактовий сигнал (англ. Serial Clock). Служить для передачі тактового сигналу для ведених пристроїв.

CS або SS — сигнал вибору мікросхеми (англ. Chip Select, Slave Select). Служить для вибору необхідного веденого пристрою.

Передача даних від провідного проводиться так: ведучий виставляє низький рівень на дроті CS веденого пристрою, з яким він збирається спілкуватися, після чого по MOSI передаються біти. По закінченні передачі ведучий "відпускає" провід CS - повертає його на високий рівень.

Щоб підключити декілька пристроїв до шини SPI, для кожного пристрою зводиться свій провід CS, і ведучий, почергово "смикаючи" ними, спілкується з тим чи іншим пристроєм. Детальніше підключення декількох пристроїв розглянемо далі.

Регістр зсуву 74HC595

Для реалізації системи формування символів будемо використовувати в якості веденого пристрою - регістр зсуву 74HC595, до якого підключимо 7-сегментний дисплей. Повна його назва - "8-bit serial-in, serial or parallel-out shift register with output latches" - "вісьмибітний регістр з послідовним введенням і послідовним або паралельним виведенням із вихідними засувками".

Принцип дії такого регістра полягає в тому, що він послідовно отримує біти по лінії DS, а при виставленні низького рівня на виводі STCP «фіксує» отримані біти так, що на його паралельних виводах формується весь отриманий байт. Вихід Q7S "повторює" біти на вході DS, що робить можливим роботу регістра як би в режимі байпас (*англ.* bypass — обхід, тобто функція, що дозволяє виконати комутацію вхідного сигналу безпосередньо на вихід, минаючи всі функціональні блоки), що можна використовувати у разі послідовного з'єднання декількох регістрів (дана можливість буде використовуватись пізніше).

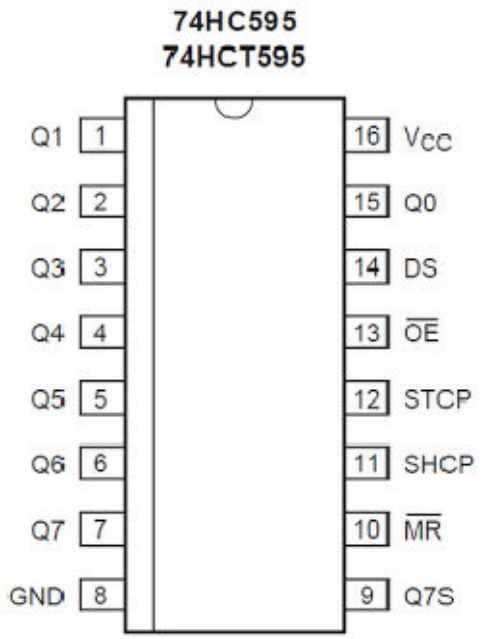

Рисунок 14.6 – Нумерація та позначення виводів регістру зсуву

Де:

– VCC - живлення;

– GND - земля;

– DS - послідовний ввід. Повинен підключатися до проводу MOSI шини SPI;

– Q0: Q7 - паралельні виходи;

– SHCP - тактовий вхід. Повинен підключатися до проводу SCK шини SPI;

– STCP – закриваючий вивід. Повинен підключатися до проводу CS шини SPI;

– OE - дозвіл роботи виходів. При низькому рівні виходи працюють;

– Q7S - послідовний вивід;

– MR - загальне скидання. Якщо притягнути його до землі - входи обнуляться.

Робота з байтами

В документації до семисегментного індикатора є таблиця, який логічний рівень потрібно подавати на той чи інший вхід, щоб загорілася та чи інша цифра. Швидше за все, ця таблиця виглядає так (замість 0 і 1 можуть бути L і H, відповідно):

Таблиця 14.1 - Представлення символів шістнадцяткової системи числення на семисегментному індикаторі

Сегмент / Символ	A	B	C	D	E	F	G	dp	hex
0	0	0	0	0	0	0	1	1	C0
1	1	0	0	1	1	1	1	1	F9
2	0	0	1	0	0	1	0	1	A4
3	0	0	0	0	1	1	0	1	B0
4	1	0	0	1	1	0	0	1	99
5	0	1	0	0	1	0	0	1	92
6	0	1	0	0	0	0	0	1	82
7	0	0	0	1	1	1	1	1	F8
8	0	0	0	0	0	0	0	1	80
9	0	0	0	0	1	0	0	1	90
a	0	0	0	1	0	0	0	1	88
b	1	1	0	0	0	0	0	1	83
c	0	1	1	0	0	0	1	1	C6
d	1	0	0	0	0	1	0	1	A1
e	0	1	1	0	0	0	0	1	86
f	0	1	1	1	0	0	0	1	8E

Якщо у дисплея загальний катод, то лог. 1 потрібно замінити на лог. 0, а лог. 0 - на лог. 1. Самий правий стовпець під назвою hex - це і є представлення цифри в байтовому вигляді. Його в документації немає потрібної інформації, потрібно розраховувати самостійно. Зробити це дуже просто: пам'ятаючи, що байт передається зі старшого біта (у нашому випадку - самого правого), розбити його на два числа по чотири біти і перевести їх у шістнадцяткове представлення.

Зауваження: взагалі кажучи, можна змусити контролер передавати біти, починаючи з молодшого, за допомогою функції SPI.setBitOrder (), проте в цій роботі дана функція не буде використовуватись.

Розберемо цифру 5. У таблиці - це 01001001. Розвертаємо і розбиваємо навпіл, отримуємо 1001 0010, бачимо, що це 9 і 2. Хто призабув переклад з двійкової системи в шістнадцяткову - дивимося таблицю:

Таблиця 14.2 - Двійково-шістнадцяткові числа

Двійкове чотири-розрядне число	Шістнадцяткове число	Двійкове чотири-розрядне число	Шістнадцяткове число
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

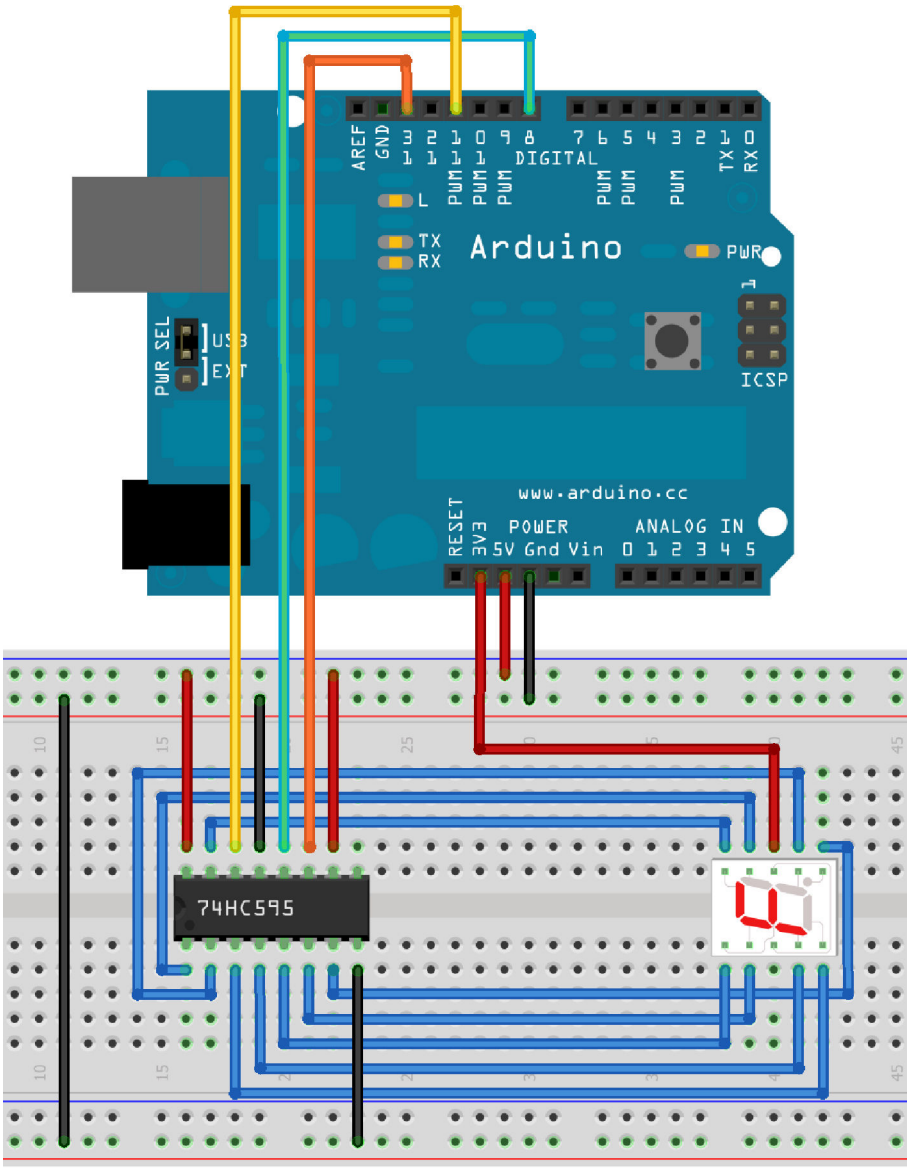

Рисунок 14.7 - Зовнішній вигляд макета

DS лічильника підключаємо до 11 виводу Arduino (жовтий провід); виходи ми завжди дозволяємо, тому підключаємо OE відразу до землі; STCP підключаємо до 8 виводу Arduino (зелений провід); SHCP (тактуючий вхід) підключаємо до 13 виводу Arduino (помаранчевий провід); MR підключаємо до живлення, так як не плануємо використовувати скидання; Q7S залишаємо непідключеним, в даний момент він нам не потрібен [3].

```
// Семисегментний індикатор на SPI

#include <SPI.h> // Підключаємо бібліотеку SPI

enum { reg = 8 }; // Провід CS підєднуємо до 8-го
виводу Arduino

void setup()
{
  SPI.begin();  // Ініціалізуємо SPI
  pinMode(reg, OUTPUT); // Налаштовуємо 8-й вивід
як вихід
}

void loop()
{
  // Зберігаємо в масиві всі цифри
  static uint8_t digit[16] = {0xC0,0xF9,0xA4,0xB0,
0x99,0x92,0x82,0xF8,0x80,0x90,0x88,0x83,0xC6,0xA1,
0x86,0x8E};
  // Виводимо цифри по одній
  for (int i=0;i<16;i++){
    digitalWrite(reg, LOW); // Встановлюємо лог. 0
на CS початок передавання
    SPI.transfer(digit[i]); // Передаємо байт
    digitalWrite(reg, HIGH); // Встановлюємо лог. 1
на CS - кінець передавання
    delay(1000); // Очікуємо секунду
  }
  // Очищуємо дисплей на секунду
  digitalWrite(reg, LOW);
  SPI.transfer(0xFF);
  digitalWrite(reg, HIGH);
  delay(1000);
}
```

Рисунок 14.8 – Лістинг програми

Необхідні комплектуючі деталі:

– Семисегментний індикатор;
– Регістр зсуву 74HC595;
– Набір провідників.

Хід лабораторної роботи

1 Безпосереднє підключення макету до семисегментного індикатора:

– Зібрати макет представлений на рис. 14.1;
– Підключити макет до ПК;
– Запустити програму Arduino IDE;
– Набрати лістинг програми представлений на рис. 14.3;
– Перевірити формування файлу, який буде записаний на мікроконтролер;
– Записати файл на мікроконтролер;
– Перевірити функціональність пристрою.

2 Безпосереднє підключення макету до семисегментного індикатора у відповідності з індивідуальним завданням:

– Зібрати макет представлений на рис. 14.1 ;
– Набрати лістинг програми відносно свого варіанту;
– Перевірити функціональність пристрою.

3 Підключення семисегментного індикатора за допомогою регістру зсуву:

– Зібрати макет представлений на рис. 14.7;
– Підключити макет до ПК;
– Запустити програму Arduino IDE;
– Набрати лістинг програми представлений на рис. 14.8;
– Перевірити формування файлу, який буде записаний на мікроконтролер;
– Записати файл на мікроконтролер;
– Перевірити функціональність пристрою.

4 Підключення семисегментного індикатора за допомогою регістру зсуву у відповідності з індивідуальним завданням:

– Зібрати макет представлений на рис. 14.7 ;
– Набрати лістинг програми відносно свого варіанту;
– Перевірити функціональність пристрою.
–

Зміст звіту

Звіт повинен складатись з титульної сторінки, що повинна містити номер та тему лабораторної роботи, а також групу, прізвище та ім’я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макету та схема електрична принципова

пристрою.

В кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрій для відображення символів на семисегментному індикаторі, як на регістрі зсуву так і без його використання.

Номер варіанту задається викладачем, завдання необхідно взяти із табл. 14.1.

Таблиця 14.1 - Варіанти завдань

Номер варіанту	Вивести число за схемою на рис. 14.1	Вивести послідовно за схемою на рис. 14.5
01	2	1,2,3,4,a,b,c
02	3	5,6,7,8, d,e,f
03	4	9,0,2,5,a,d,c,f
04	5	4,6,8,0,b,d,f,c
05	6	1,3,5,7,9,e,f,a
06	7	0,3,7,9,a,c,d,f