Практична робота 10. Робота з LCD дисплеєм

Теоретичні відомості

Рідкокристалічний дисплей (LCD) 1602

LCD-дисплей 1602 (рис. 10.1) використовується для роботи з Arduino. Має два рядки по 16 символів в кожному. Працює зі стандартною бібліотекою LiquidCrystal, що йде в комплекті з Arduino IDE.

Рисунок 10.1 – Рідкокристалічний дисплей 1602

Технічні характеристики:

- розміри: 80 x 36 мм
- розмір символу: 4.35 x 2.95мм
- розміри точки: 0.5 x 0.5мм
- інтерфейс: HD44780
- видима область: 64.5 x 13.8мм
- живлення: 5В

Дані дисплеї мають два варіанти виконання: жовта підсвітка з чорними символами та синя підсвітка з білими символами (зустрічається частіше). Існують також дисплеї для виведення 4 рядків по 20 символів.

Більшість дисплеїв не мають підтримки кирилиці, мають її лише дисплеї з маркуванням СТК. Однак цю проблему можна частково вирішити, використовуючи власні символи.

Дисплей має 16 контактів для підключення (рис. 10.2):

1 (VSS) – живлення контролера (-)

2 (VDD) – живлення контролера (+)

3 (VO) – вихід керування контрастом

4 (RS) – вибір регістру

5 (R/W) – читання/запис (режим запису при з'єднанні з землею)

6 (E) – Enable

7-10 (DB0-DB3) – молодші біти 8-бітного інтерфейсу

11-14 (DB4-DB7) – старші біти інтерфейсу

15 (A) – анод (+) живлення підсвітки

16 (K) – катод (-) живлення підсвітки

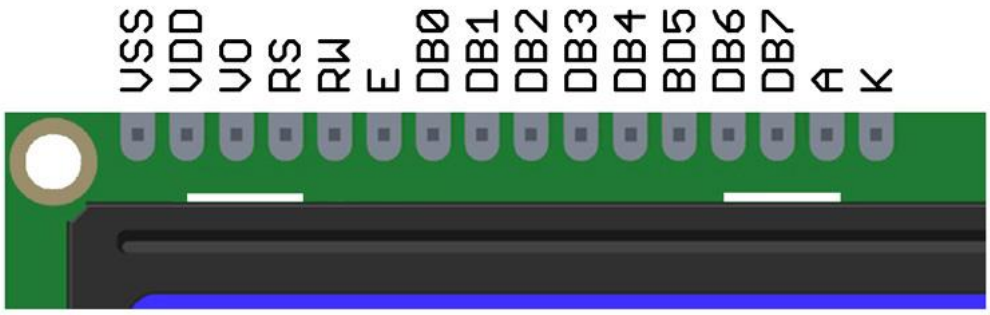

Рисунок 10.2 – Контакти підключення дисплею 1602

Для налаштування контрастності слід використовувати потенціометр на 10 кОм. Крайні контакти потенціометра з’єднуються з виходами +5V та GND, центральний контакт з'єднується з виходом VO.

Після подачі живлення на схему необхідно налаштувати правильний контраст, якщо він буде налаштований не вірно, то на екрані нічого не буде відображатися. Налаштування контрасту здійснюється потенціометром.

Сам же дисплей може працювати в двох режимах:

- 8-бітний режим – для цього використовуються і молодші і старші біти (BB0–DB7)
- 4-бітний режим – для цього використовуються і тільки молодші біти (BB4–DB7)

Використання 8-бітного режиму на даному дисплеї не доцільно. Для його роботи потрібно на 4 контакти більше, а виграшу у швидкості практично немає.

Для виведення тексту необхідно підключити контакти RS, E, DB4, DB5, DB6, DB7 до виходів контролера. Їх можна підключати до будь-яких виходів Arduino (рис. 10.3), головне в коді задати правильну послідовність.

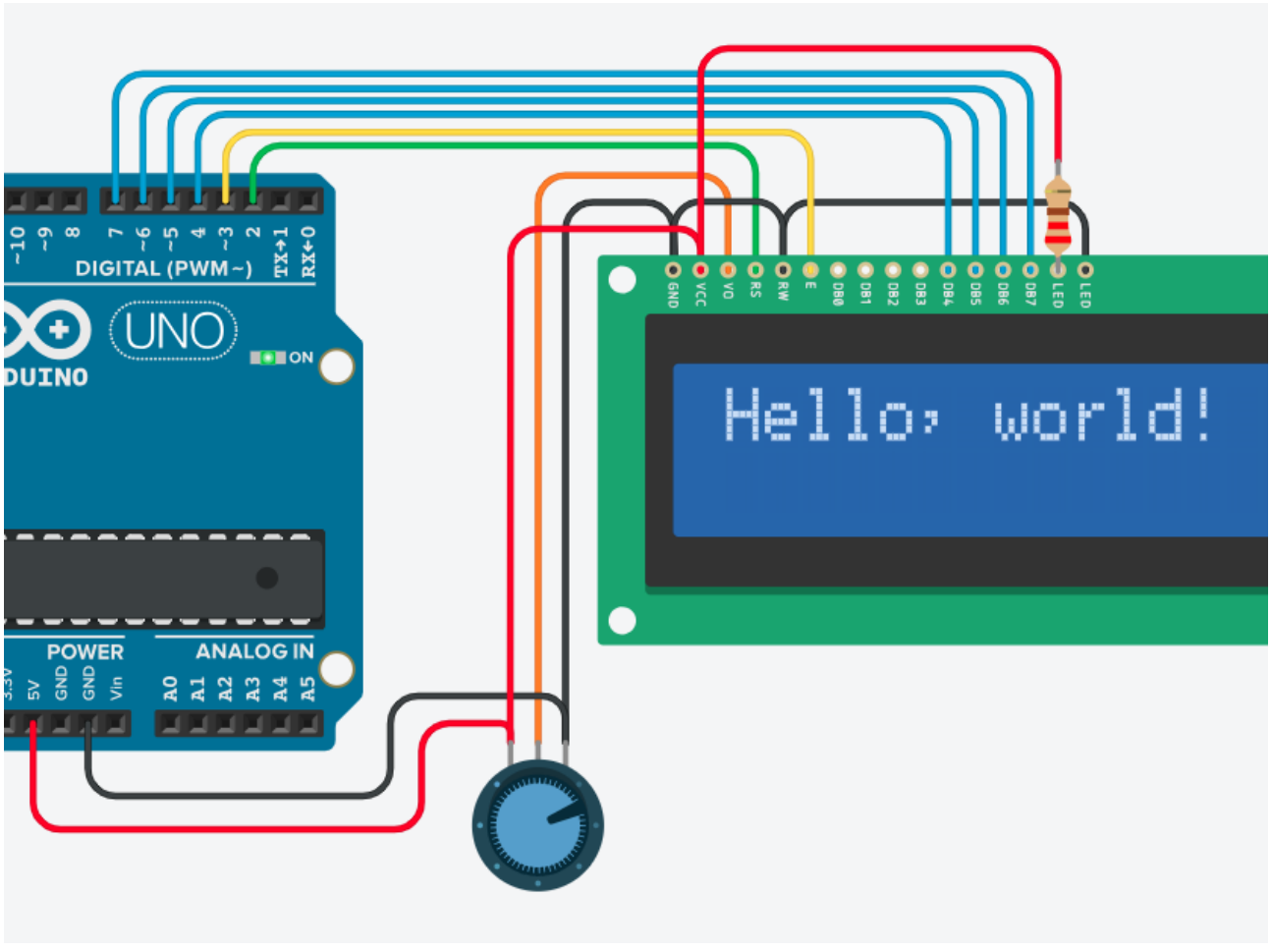

Рисунок 10.3 – Схема підключення LCD-дисплея до Arduino

Код програми, яка виводить на екран дисплею надпис «Hello, world!» представлено в лістингу 10.1:

Лістинг 10.1 – Виведення на екран LCD-дисплею текстових даних

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);  // вказуємо виходи RS, E, DB4, DB5, DB6, DB7

void setup(){
  lcd.begin(16, 2);                   // задаємо розмірність дисплею
  lcd.setCursor(0, 0);                // встановлюємо курсор на початок першого рядка
  lcd.print("Hello, world!");         // виводим текст
}

void loop(){}
```

кінець лістингу 10.1

Літери англійського алфавіту зашиті в пам'ять контролера дисплею. Однак існує можливість необхідний символ зробити вручну (всього до 8-ми символів).

Комірка для відображення символу має розширення 5x8 точок. Задача по створенню символу зводиться до того, щоб написати бітову маску і розставити в ній одинички в місцях де повинні світитись точки. У нижче наведеному прикладі (рис. 10.4) прикладі показано яким чином формується смайлик.

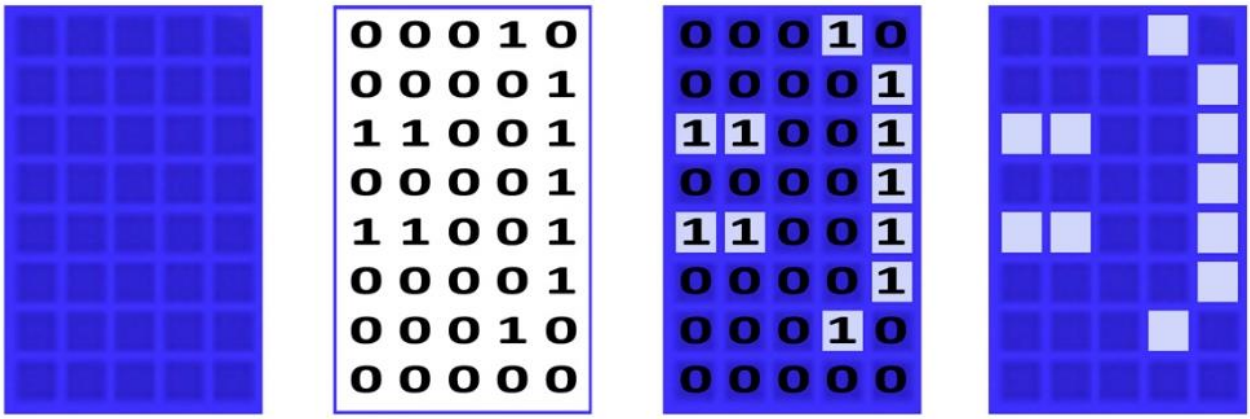

Рисунок 10.4 – Створення власних символів для виведення на дисплей

Залишається перенести бітову маску в код програми та передати її в пам’ять контролера дисплею (лістинг 10.2).

Лістинг 10.2 – Виведення на дисплей власних символів

```
#include <LiquidCrystal.h>

byte smile[8] =                    // бітова маска смайлика
{
  B00010,
  B00001,
  B11001,
  B00001,
  B11001,
  B00001,
  B00010,
};
LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup(){
  lcd.begin(16, 2);
  lcd.createChar(1, smile);        // створення символу під номером 1
  lcd.setCursor(0, 0);
  lcd.print("\1");                 // виводиться символ під номером 1
}

void loop() {}
```

кінець лістингу 10.2

Бібліотека LiquidCrystal

Дана бібліотека дозволяє Arduino керувати різними рідкокристалічними дисплеями (LCD), побудованими на базі контролера Hitachi HD44780 (або сумісного). У бібліотеці реалізований як 4-х, так і 8-бітний режими роботи (тобто є можливість використовувати 4 або 8 каналів для передачі даних).

Основні функції:

LiquidCrystal() – створює змінну типу LiquidCrystal. Синтаксис функції:

LiquidCrystal(rs, enable, d4, d5, d6, d7)

LiquidCrystal(rs, rw, enable, d4, d5, d6, d7)

LiquidCrystal(rs, enable, d0, d1, d2, d3, d4, d5, d6, d7)

LiquidCrystal(rs, rw, enable, d0, d1, d2, d3, d4, d5, d6, d7)

rs – номер виходу Arduino, з'єднаного з виходом RS LCD-дисплею;

rw – номер виходу Arduino, з'єднаного з виходом RW LCD- дисплею;

enable – номер виходу Arduino, з'єднаного з виходом E LCD- дисплею;

d0, d1, d2, d3, d4, d5, d6, d7 – номера виходів Arduino, які підключені до відповідних цифровим виходів LCD-дисплею.

У випадку 4-бітного режиму роботи слід пропустити параметри d0-d3, залишивши відповідні виходи не підключеними. Вихід RW можна також не підключати до Arduino, а з'єднати його безпосередньо з землею. У цьому випадку параметр rw в функції можна не вказувати.

begin() – ініціалізує інтерфейс для взаємодії з LCD-екраном і задає розміри (ширину і висоту) області виведення екрану. При роботі з LCD-дисплеєм, функція begin() повинна викликатися першої. Синтаксис функції:

lcd.begin (cols, rows)

lcd – змінна типу LiquidCrystal;

cols – кількість стовпців екрану;

rows – кількість рядків екрану.

clear() – очищає LCD-дисплей та переміщує курсор в лівий верхній кут. Синтаксис функції: *lcd.clear()*.

setCursor() – дозволяє задати позицію курсора на LCD-дисплеї; тобто встановлює позицію, в якій буде виводитися наступний текст. Синтаксис функції:

lcd.setCursor (col, row)

col – координата X позиції курсора (0 означає перший стовпець);

row – координата Y позиції курсора (0 означає перший рядок).

print() – виводить текст на LCD-дисплей. Синтаксис функції:

lcd.print (data)

data – дані, які необхідно вивести (типу char, byte, int, long або string).

noDisplay() – вимикає LCD-дисплей. Текст, що відображається на екрані, зберігається в пам'яті. Синтаксис функції: *lcd.noDisplay()*.

display() – функція вмикає LCD-дисплей після того, як він був вимкнений функцією noDisplay(). Після виконання цієї функції на екрані відобразиться текст (і курсор), які були на ньому до вимнення. Синтаксис функції: *lcd.display()*.

cursor() – відображає на LCD-дисплеї курсор: символ підкреслення в тій позиції, куди буде виведено наступний символ. Синтаксис функції: *lcd.cursor()*.

noCursor() – функція відключає відображення курсора на LCD-дисплеї. Синтаксис функції: *lcd.noCursor()*.

createChar() – функція створює користувальницький символ. Дисплей підтримує до 8 користувальницьких символів (пронумерованих від 0 до 7) розміром 5x8 пікселів. Зовнішній вигляд кожного символу задається масивом з восьми байт, кожен з яких характеризує відповідний рядок. П'ять молодших біт кожного байту визначають стан пікселів у відповідному рядку. Синтаксис функції:

lcd.createChar (num, data)

num – номер призначеного для користувача символу, який необхідно створити (від 0 до 7);

data – дані про пікселі призначеного для користувача символу.

Хід роботи

1. Побудуйте схему підключення до Arduino потенціометра та рідкокристалічного дисплею 1602.
2. Запрограмуйте Arduino таким чином, щоб отримувати значення отримані від потенціометра та виводити результат на рідкокристалічний дисплей.

3. Оформити звіт по роботі. Звіт повинен містити тему та мету роботи, короткі відомості про функції, що розглядаються в роботі, рисунок побудованої схеми підключення та код програми Arduino.