Практическое занятие №4
Тема: «Подключение и составление алгоритма и программы для ЖК-дисплеев»

Цель работы: приобрести практические навыки по подключению и программированию ЖК-дисплея LCD1602A на платформе Arduino.

Последовательность выполнения работы:

– Изучить теоретические сведения, приведенные в практическом занятии.

– Сделать монтажную и принципиальную схему в программе Fritzing. (напоминание: принципиальная схема формируется автоматически после создания монтажной).

– Собрать схемы желательно на макетной плате, иначе при отсутствии набора Arduino в web-приложениях (https://wokwi.com/projects/new/arduino-uno или https://www.tinkercad.com/) для приведенных примеров.

– Запрограммировать микроконтроллер согласно тексту, указанному в примере.

– Выполнить задание для самостоятельной работы.

Содержание отчета:

– Название практического занятия, его цель.

– Фото или скриншоты собранной схемы.

– Написанный программный код вставить текстом, Courier New, 12 кегль, одинарный отступ без абзацев.

– Вывод о проделанной работе.

– Файл Fritzing с принципиальной и монтажной схемой.

Теоретические сведения

LCD1602A – это монохромный жидкокристаллический дисплей (ЖК-дисплей) с поддержкой алфавитно-цифровых символов.

Основные характеристики:

1. Дисплей: 16 символов × 2 строки (отсюда название: 1602).
2. Подсветка: синяя или зеленая с темными символами, встречаются варианты с другими цветами.
3. Управление: Контроллер Hitachi HD44780 или совместимый.
4. Напряжение питания: 5 В.
5. Интерфейс: Параллельный (4- или 8-битный), также поддерживает I^2C через переходник.

Структура дисплея:

1. DDRAM (Display Data RAM) – память, хранящая символы, отображаемые на экране.
2. CGROM (Character Generator ROM) – содержит предопределенные символы (латинские буквы, цифры, знаки препинания).
3. CGRAM (Character Generator RAM) – позволяет создавать собственные символы (5×8 или 5×10 пикселей).

Каждая позиция на дисплее имеет адрес:

1-я строка: 0x00 … 0x0F

2-я строка: 0x40 … 0x4F

Для вывода текста данные записываются в DDRAM через команды или напрямую.

Дисплей поддерживает два режима передачи данных:

– 8-битный – быстрый, но требует больше линий.

– 4-битный – экономит GPIO микроконтроллера, но работает медленнее.

Основные управляющие сигналы:

– RS (Register Select) – выбор между командным (0) и данными (1).

– RW (Read/Write) – чтение (1) или запись (0).

– E (Enable) – стробирующий импульс для подтверждения данных.

– D0-D7 – шина данных (в 4-битном режиме используются только D4-D7).

Таблица 1 - Основные команды

Команда (Hex)	**Команда (Binary)**	**Описание**
0x01	00000001	Очистка дисплея – удаляет все символы, курсор в (0, 0).
0x02	00000010	Возврат курсора в начало – DDRAM сбрасывается в 0x00.
0x04	00000100	Сдвиг курсора влево (без перемещения текста).
0x06	00000110	Автоинкремент курсора (текст вводится слева направо).
0x0E	00001110	Включение дисплея + курсор (без мигания).
0x0F	00001111	Включение дисплея + курсор + мигание.
0x10	00010000	Сдвиг курсора влево (без изменения DDRAM).

Команда (Hex)	Команда (Binary)	Описание
0x14	00010100	Сдвиг курсора вправо (без изменения DDRAM).
0x18	00011000	Сдвиг всего дисплея влево.
0x1C	00011100	Сдвиг всего дисплея вправо.
0x80	10000000	Установка курсора в начало 1-й строки (0x00).
0xC0	11000000	Установка курсора в начало 2-й строки (0x40).

Примеры использования команд

Инициализация LCD1602 в 4-битном режиме:

```
#include <LiquidCrystal.h>

// Подключение: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 6, 7, 8);

void setup() {
  lcd.begin(16, 2);  // 16 столбцов, 2 строки
  lcd.print("Hello, World!");  // Вывод текста
  lcd.setCursor(0, 1);  // Переход на 2-ю строку (0-й символ)
  lcd.print("LCD1602A Test");
}
```

Результат:
Hello, World!
LCD1602A Test

Очистка дисплея (0x01):

```
lcd.clear(); // Аналог команды 0x01
```

Эквивалент в ручном режиме:

```
void sendCommand(uint8_t cmd) {
  digitalWrite(RS, LOW);  // Режим команды
  digitalWrite(RW, LOW);  // Запись
  // Отправка команды (4-битный режим)
  digitalWrite(D7, (cmd >> 7) & 1);
  digitalWrite(D6, (cmd >> 6) & 1);
  digitalWrite(D5, (cmd >> 5) & 1);
  digitalWrite(D4, (cmd >> 4) & 1);
  pulseEnable();
  //Вторая тетрада
  digitalWrite(D7, (cmd >> 3) & 1);
  digitalWrite(D6, (cmd >> 2) & 1);
  digitalWrite(D5, (cmd >> 1) & 1);
  digitalWrite(D4, (cmd >> 0) & 1);
```

```
  pulseEnable();
}

sendCommand(0x01);  // Очистка экрана
delay(2);  // Ждем завершения
```

Управление курсором:

Включение/выключение курсора

```
lcd.cursor();      // Показать курсор (подчеркивание)
lcd.noCursor();    // Скрыть курсор
lcd.blink();       // Включить мигание курсора
lcd.noBlink();     // Выключить мигание
```

Аналог через команды:

```
sendCommand(0x0E);  // Курсор без мигания
sendCommand(0x0F);  // Курсор с миганием
sendCommand(0x0C);  // Курсор выключен
```

Перемещение курсора:

```
lcd.setCursor(3, 1);  // 4-й символ, 2-я строка
```

Аналог через команды:

```
sendCommand(0xC0 + 3);  // 2-я строка (0x40) + смещение 3
```

Создание своих символов (CGRAM):

Можно загрузить 8 пользовательских символов (5×8 пикселей).

```
byte heart[8] = {  // Создаем символ "сердечко"
  0b00000,
  0b01010,
  0b11111,
  0b11111,
  0b01110,
  0b00100,
  0b00000,
  0b00000
};

void setup() {
  lcd.createChar(0, heart);  // Загружаем в CGRAM (номер 0)
  lcd.begin(16, 2);
  lcd.write(byte(0));  // Выводим символ
}
```

Подключение к микроконтроллеру

Пример схемы для 4-битного режима (Arduino):

LCD1602A	Arduino
VSS	GND
VDD	5V
VO	Подстроечный резистор (контраст)
RS	Pin 12
RW	GND (только запись)
E	Pin 11
D4-D7	Pins 2-5
A (подсветка+)	+5V через резистор
K (подсветка-)	GND

Особенности

– контрастность регулируется потенциометром на выводе VO.

– инициализация требует соблюдения временных задержек после подачи питания.

– поддержка кириллицы – отсутствует в стандартной прошивке, но можно загрузить свои символы в CGRAM.

LCD1602A широко используется в проектах Arduino, Raspberry Pi и других микроконтроллеров благодаря простоте управления и низкой стоимости.

ЗАДАНИЕ

Собрать схему:

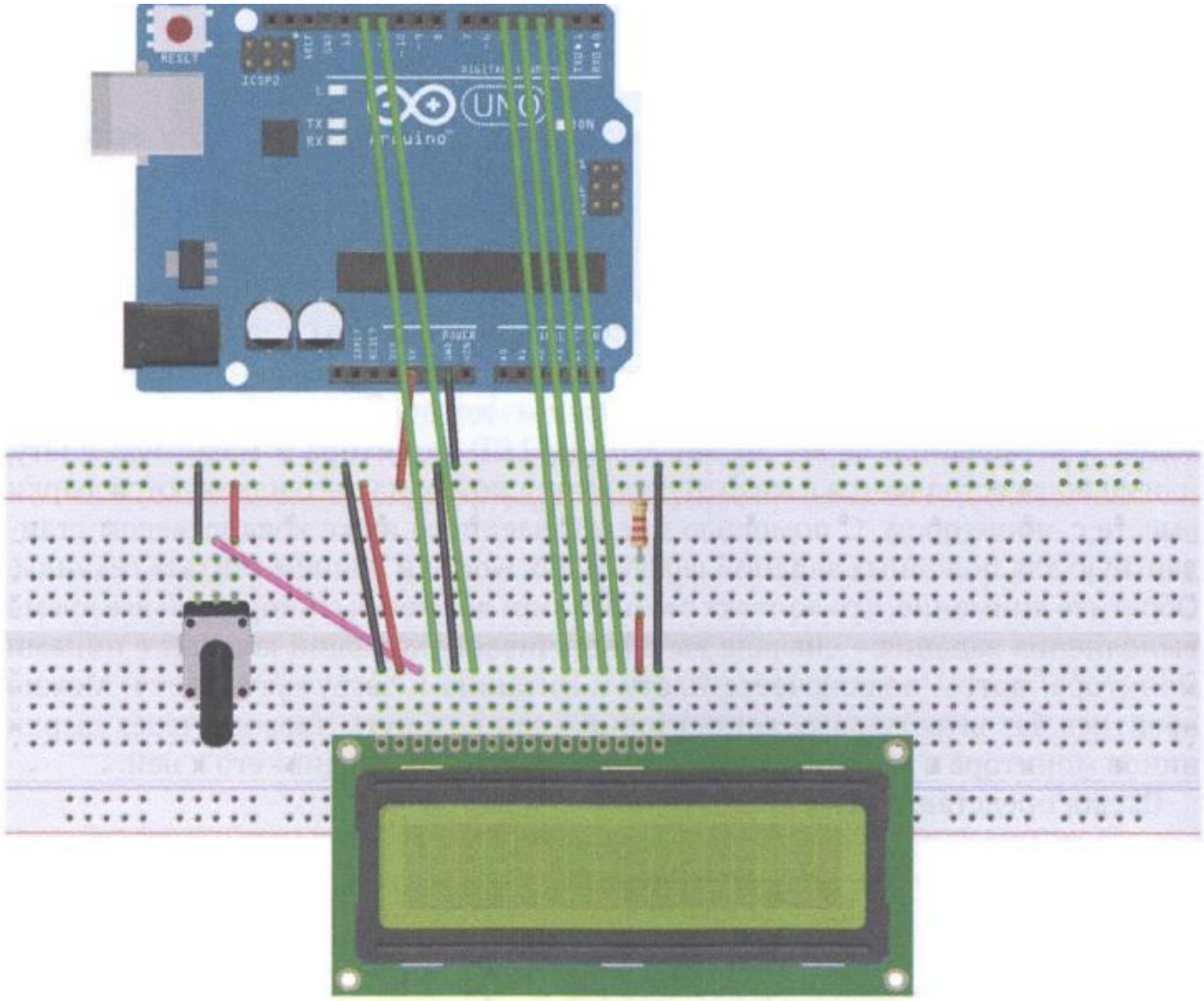

Рисунок 1 – Монтажная схема к заданию

Программа №1: Статический вывод текста

```
#include <LiquidCrystal.h>
// Инициализация LCD (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // Устанавливаем размер дисплея (16 столбцов, 2 строки)
  lcd.begin(16, 2);

  // Выводим текст на первую строку
  lcd.print("Hello, Arduino!");

  // Переходим на вторую строку
  lcd.setCursor(0, 1);

  // Выводим текст на вторую строку
  lcd.print("LCD1602A Test");
}

void loop() {
  // Ничего не делаем в цикле
}
```

Программа №2: Динамический вывод (бегущая строка + счётчик)

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  lcd.begin(16, 2);
  lcd.print("Scroll & Counter");
}

void loop() {
  // Бегущая строка на второй строке
  for (int i = 0; i < 16; i++) {
    lcd.setCursor(i, 1);
    lcd.print("Hello!");
    delay(300);
    lcd.setCursor(i, 1);
    lcd.print("      "); // Стираем предыдущий текст
  }

  // Счётчик от 0 до 99
  for (int count = 0; count < 100; count++) {
    lcd.setCursor(0, 1);
    lcd.print("Count: ");
    lcd.print(count);
    lcd.print("   "); // Очистка лишних символов
    delay(500);
  }
}
```

САМОСТОЯТЕЛЬНАЯ РАБОТА

Ответьте на контрольные вопросы:

1. Какое максимальное количество символов может отображать LCD1602A? Укажите его формат.
2. Какой контроллер чаще всего используется в LCD1602A?
3. Какие два режима передачи данных поддерживает LCD1602A? Какой из них экономит GPIO микроконтроллера?
4. Для чего нужны сигналы RS, RW и E в управлении дисплеем?
5. Какая команда используется для очистки дисплея? Какой у неё шестнадцатеричный код?
6. Как установить курсор на 5-й символ 2-й строки с помощью команд?
7. Какой потенциометр и куда подключается для регулировки контрастности LCD1602A?
8. Какие команды нужно отправить, чтобы включить дисплей, курсор и мигание?
9. Сколько пользовательских символов можно записать в CGRAM и какого они размера?
10. Какой командой можно сдвинуть весь дисплей вправо без перемещения курсора?