

Лабораторная работа №5

Дисплеи, подключение дисплеев к Arduino

Микроконтроллеры позволяют сделать любые системы автоматизации и мониторинга. Но для взаимодействия техники и человека нужны как устройства ввода – различные кнопки, рычаги, потенциометры, так и устройства вывода – световые индикаторы (лампочки), различные звуковые сигнализаторы (пищалки) и наконец дисплеи. В этой статье мы рассмотрим символьные дисплеи для Arduino, как их подключить и заставить работать.

Виды дисплеев

Дисплеи можно разделить на:

- Сегментные (такие, как на цифровых часах);
- Алфавитно-цифровые (знакосинтезирующие);
- Графические.

Сегментные используются для индикации простых величин, например: температура, время, количество оборотов. Такие используются в калькуляторах и на бюджетной бытовой технике и по сей день. Информация выводится путем засвечивания определенных символов.

Они могут быть как жидкокристаллическими, так и светодиодными. Алфавитно-цифровые дисплеи можно встретить на старой бытовой технике, игрушках, промышленной технике и прочем. Их еще называют знакосинтезирующими, текстовыми, символьными. Состоят из набора крупных пикселей. Могут быть выполнены по LCD, TFT и OLED-технологии.

К графическим дисплеям можно отнести даже монитор или экран смартфона, особых пояснений я думаю не требуется.

Знакосинтезирующие дисплеи

Дисплеи этого вида могут одновременно отображать определенное количество символов, ограниченное геометрическими размерами. Маркируются они по такому образцу:

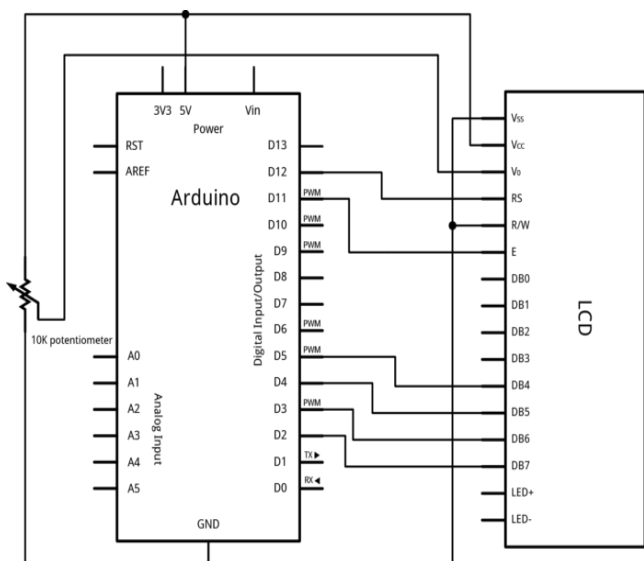
1602;

2002.

Где первые две цифры – количество символов в строке, а вторая пара – количество строк. Таким образом дисплей с названием 1602 может отображать одновременно 2 строки по 16 символов.

По типу ввода данных различают дисплеи:

- С параллельным вводом данных;

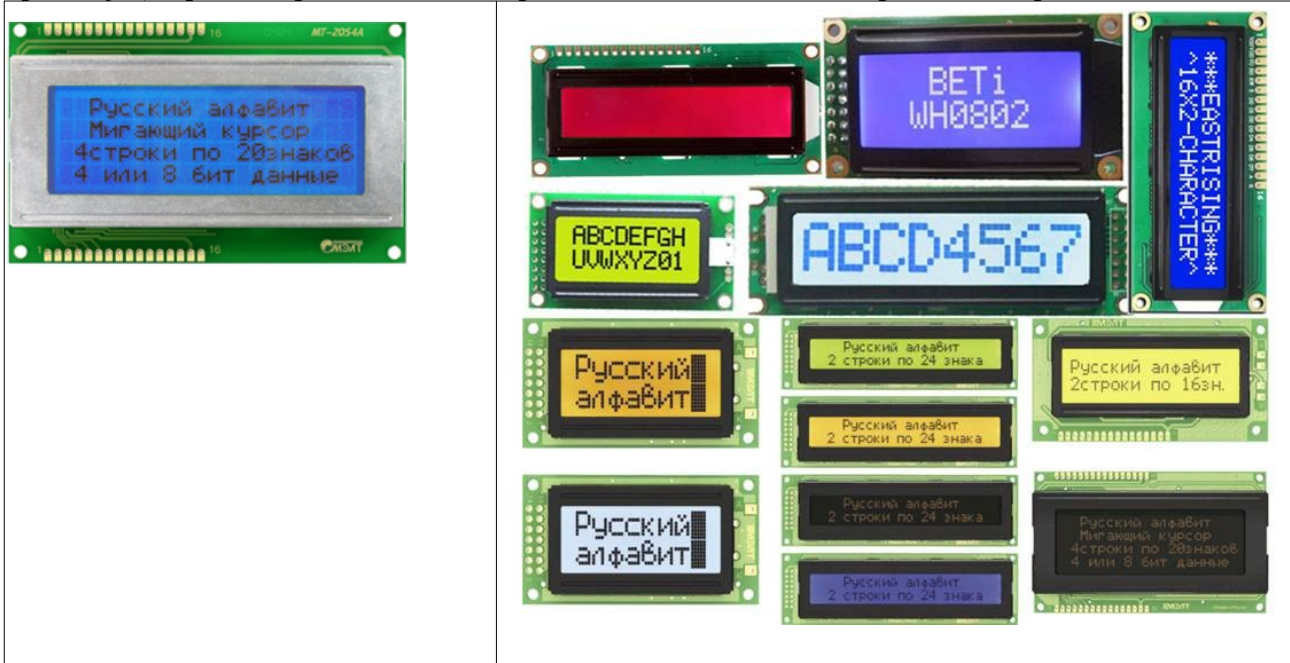


- С вводом данных по протоколу I2C, SPI, UART и т.д. Параллельный ввод данных предполагает передачу 8 или 4-битных слов по 10 или 6 выводам соответственно (рис. ниже – схема подключения для управления 4 битами). Кроме данных на дисплей подаётся питание. Учитывайте это при проектировании, в противном случае вам может не хватить пинов платы Ардуино.

Передача данных на дисплей с помощью I2C займет 4 пина вашей Arduino, 2 из которых питание, а 2 – данные. Но подробнее рассмотрим этот вопрос немного ниже.

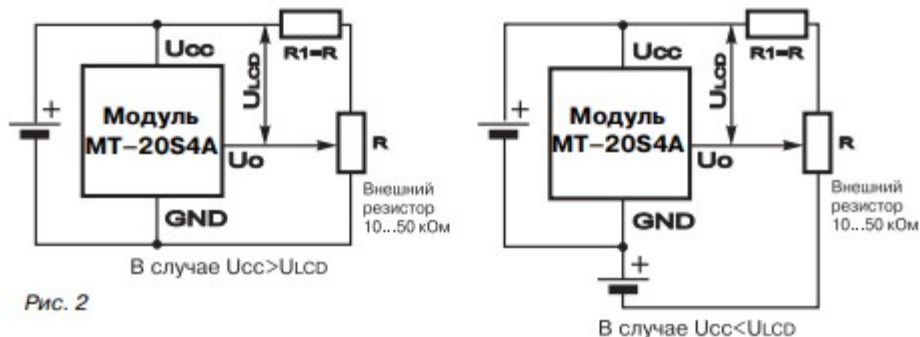
Среди отечественных производителей можно выделить фирму МЭЛТ. Среди продукции, которой есть целый ряд различных

дисплеев. Например, ниже изображен дисплей с маркировкой 20S4, по аналогии с предыдущей рассмотренной, это говорит нам о том, что он отображает 4 строки по 20 знаков.



Он построен на контроллере КБ1013ВГ6, от ОАО «АНГСТРЕМ», который аналогичен HD44780 фирмы HITACHI и KS0066 фирмы SAMSUNG. На которых построены подавляющее большинство китайских дисплеев. Кстати он, как и дисплеи на перечисленных чипах поддерживает стандартную библиотеку параллельного управления Arduino IDE.

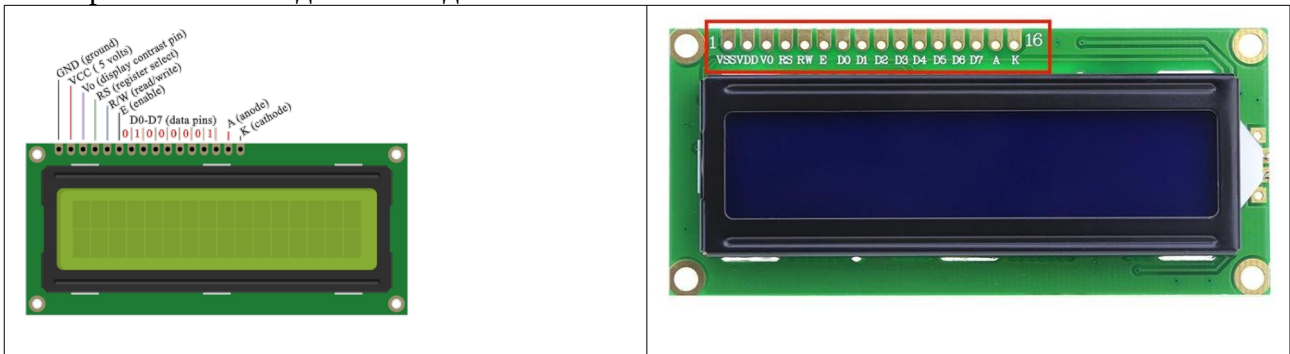
Знакосинтезирующие дисплеи бывают с подсветкой и без неё, также могут отличаться цветом изображаемых символов. Яркость подсветки и контрастность изображения обычно регулируется. Ниже приведет пример схемы из даташита, на упомянутый выше МЭЛТ.



Переменный резистор R и служит для регулировки яркости.

Подключение

Подключение будем рассматривать на дисплее типа 1602. В первую очередь обратите внимание на подписи выводов. Встречается два варианта, нумерации. На двух рисунках ниже всё нормально – от 1 до 16 вывода.



Отметим, что под VSS понимается земля. В остальном назначения выводов идентичны.

Разберем подробнее.

1 – (Vss) земля или «—» питания.

2 – (Vcc) «+» питания. Чаще всего это 5 вольт.

3 – регулировка контрастности символов. Осуществляется через потенциометр, установленный между «+» питания и этим контактом. Чем выше напряжение – тем меньше яркость и энергопотребление.

4 – (RS) Адресный сигнал. По наличию сигнала от ардуино на этом входе контроллер дисплея понимает, на линии данных сигнал команды (перемещение курсора, например) или кода символа для отображения.

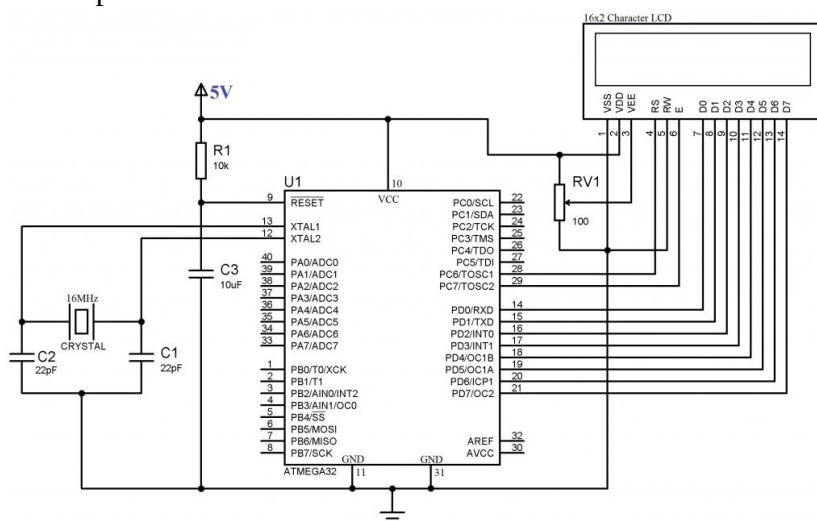
5 – (E) разрешения доступа к данным. Когда здесь логическая «1» - дисплей выполняет команду или выводит символ.

6-14 – через эти пины обеспечивается параллельный ввод данных.

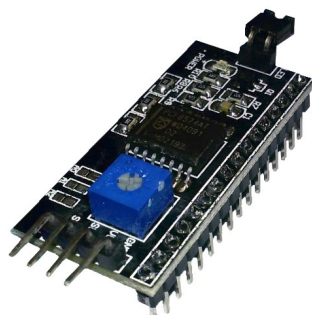
15 – (BLA) анод подсветки. Чтобы она зажглась на всю яркость – сюда подают +5В.

16 – (BLC) катод подсветки. Подключают к земле.

Один из примеров подключения к Ардуино в 4 битовом режиме мы рассмотрели выше. Теперь взгляните на схему подключения в 8 битовом режиме управления. Кстати вы могли заметить переменный резистор. Он и нужен для регулировки яркости подсветки, как было сказано ранее.



Таким образом у вас оказываются занятыми половина входов платы Arduino UNO. Конечно если вы будете использовать MEGA – это будет не столь существенной проблемой, но всё же это не рационально, особенно если вы собираетесь подключать группу датчиков и клавиш управления.



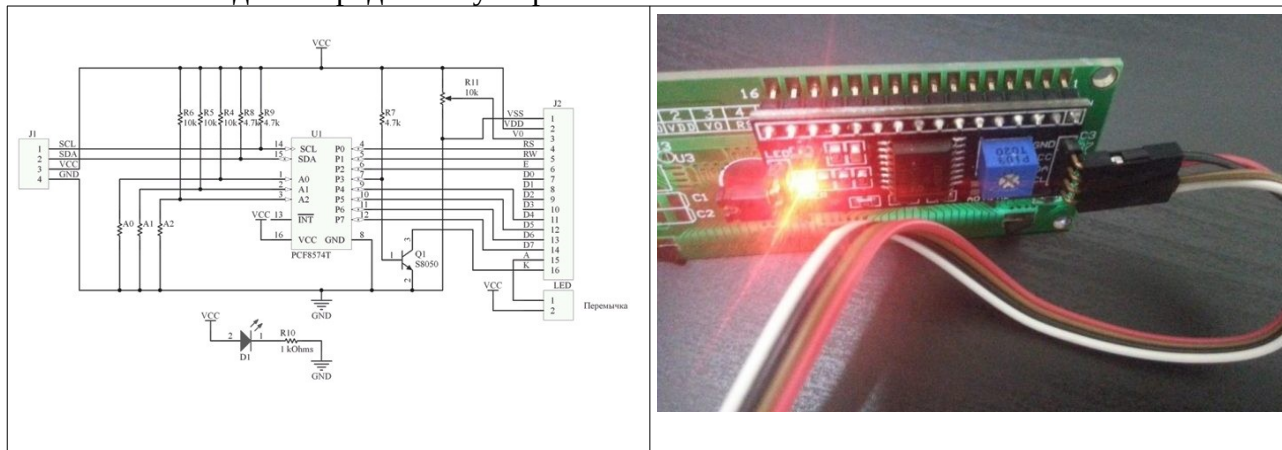
Чтобы высвободить входы используйте конвертер I2C для LCD экрана (именно так он называется, и вы сможете найти его в магазинах под таким названием). Внимание:

Если будете покупать этот модуль отдельно от дисплея не забудьте о расположении и нумерации выводов, которую мы рассмотрели ранее.

Гребёнка, изображенная снизу просто припаивается к дисплею, а четыре контакта на торце платы – подключаются к пинам Arduino, также есть третья группа из двух контактов (на фото сзади) – это включение подсветки, модели поставляются с установленной перемычкой.

Схема такого модуля выглядит следующим образом:

А вот так он выглядит припаянным непосредственно к контактам дисплея. Большинство моделей продаются уже распаянными.



Однако для его использования вам нужно будет найти в сети библиотеку LiquidCrystal_I2C её нет в стандартном наборе актуального на момент написания статьи Arduino IDE.

Для работы по I2C нужно сформировать 2 информационных сигнала – SDA и SCL, обратите внимание в нижний правый угол рисунка. Эти выводы в ардуино совмещены с A4 и A5 аналоговыми входами.

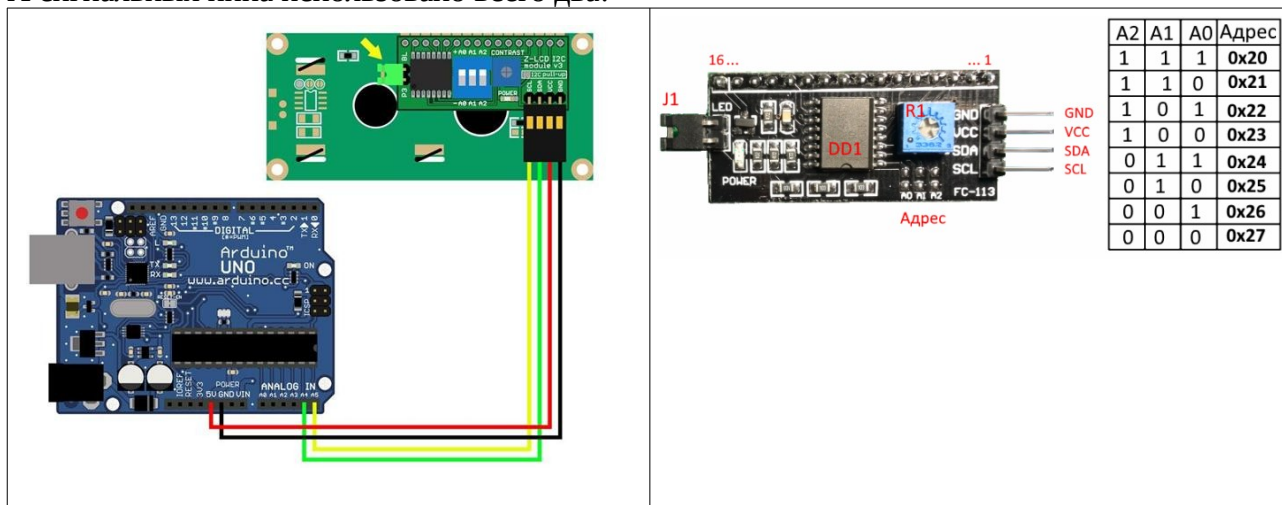
Важно:

Переназначить их вы на другие выводы не можете.

Тогда монтажная схема подключения будет иметь вид:

Согласитесь, проводов намного меньше! От ардуино к дисплею идут всего 4 провода.

А сигнальная пина использовано всего два!



Но просто подключить у вас ничего не получится вы должны знать адрес устройства, для этого есть еще одна группа контактов, где адрес задаётся с помощью перемычек. Это указывается в инициализирующей команде соответствующей библиотеки, об этом далее.

Программа

Естественно нам нужен какой-то скетч, который может показывать изображение на символьном дисплее. Если вы хотите «напрямую» работать с дисплеем – придется изучить даташиты и таблицы символов на каждое конкретное изделие. Но Ардуино была создана для простого и быстрого прототипирования электронных устройств. Поэтому мы пойдем другим путём и воспользуемся благами цивилизации. Мы уже упомянули, что в стандартном наборе библиотек в Arduino IDE есть готовое решение для работы с LCD-дисплеями.

Рассмотрим простейший «Хэлоу ворлд». Пример полностью совпадает с тем, что есть в стандартном наборе IDE, я лишь перевёл текст комментариев на русский язык. Обратите внимание – это пример работы в 4-битном режиме.

```
// include the library code:
#include <LiquidCrystal.h>

// инициализируем библиотеку с номерами пинов к которым подключен дисплей указанными в скобках
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // Указываем количество символов и строк дисплея, у нас 16 символов и 2 строки
  lcd.begin(16, 2);
  // Печатаем слово на LCD
  lcd.print("hello, world!");
}

void loop() {
  // Переносим курсор на 1 символ 2 второй строки. Нумерация от нуля
  // а это значит что 1=0, а 2=1
  lcd.setCursor(0, 1);
  // печатаем сколько секунд прошло от запуска программы
  lcd.print(millis() / 1000);
}

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4); // Для экрана 20x4 (четырёхстрочный)
//LiquidCrystal_I2C lcd(0x27, 16, 2); // Для экрана 16x2 (двухстрочный)

void setup()
{
  lcd.begin();
  lcd.setCursor(0, 0); // 1 строка
  lcd.print("LCD I2C Test - 20x4");
  lcd.setCursor(0, 1); // 2 строка
  lcd.print("01234567899876543210");
  lcd.setCursor(0, 2); // 3 строка
  lcd.print("01234567899876543210");
  lcd.setCursor(0, 3); // 4 строка
  lcd.print("  gekelectronics.org");
}

void loop()
{
}
```

Работа с I2C практически аналогична:

Обратите внимание, что в этом примере кода первой командой указан несколько размер дисплея, количество строк и символов, но и его I2C адрес. А именно – 0x27, что соответствует отсутствующим перемычкам. Вообще это нужно для того, чтобы подключить на два сигнальных провода несколько дисплеев (8 штук).

Методы библиотеки LiquidCrystal и LiquidCrystal_I2C

Функции, реализованные только в библиотеке LiquidCrystal_I2C:

init(); – Инициализация дисплея. Должна быть первой командой библиотеки LiquidCrystal_I2C после создания объекта. На самом деле данная функция есть и в библиотеке LiquidCrystal, но в той библиотеке она вызывается автоматически (по умолчанию) при создании объекта.

backlight(); – Включение подсветки дисплея.

noBacklight(); – Выключение подсветки дисплея.

setBacklight(flag); – Управление подсветкой (true - включить / false - выключить), используется вместо функций noBacklight и backlight.

Подключение:

<pre>// Для шины I2C: #include <Wire.h> #include <LiquidCrystal_I2C.h> LiquidCrystal_I2C lcd(address , col , row); void setup(){ lcd.init(); }</pre>	<p>Параметр:</p> <ul style="list-style-type: none"> • address: Адрес дисплея на шине I2C - 0x27 или 0x3F • col: количество столбцов реализованное у дисплея • row: количество строк реализованное у дисплея
<pre>// Для параллельной шины из 4 проводов: #include <LiquidCrystal.h> LiquidCrystal lcd(RS , E , D4 , D5 , D6 , D7); void setup(){ lcd.begin(col , row); }</pre>	<p>Параметр:</p> <ul style="list-style-type: none"> • RS: № вывода Arduino к которому подключён вывод RS • E: № вывода Arduino к которому подключён вывод E • D0...D3: № выводов Arduino к которым подключены выводы D0-D3
<pre>// Для параллельной шины из 8 проводов: #include <LiquidCrystal.h> LiquidCrystal lcd(RS , E , D0 , D1 , D2 , D3 , D4 , D5 , D6 , D7); void setup(){ lcd.begin(col , row); }</pre>	<p>Параметр:</p> <ul style="list-style-type: none"> • D4...D7: № выводов Arduino к которым подключены выводы D4-D7 • col: количество столбцов реализованное у дисплея • row: количество строк реализованное у дисплея
<p>begin(col , row , [size]); Инициализация дисплея с указанием размеров экрана и символов.</p>	<p>Параметр:</p> <ul style="list-style-type: none"> • col: количество столбцов реализованное у дисплея • row: количество строк реализованное у дисплея • size: размер символов, указывается константой: LCD_5x8DOTS (по умолчанию), или LCD_5x10DOTS

Пример:

```
/* Для шины I2C: */
#include <Wire.h> // Подключаем библиотеку для работы с шиной I2C
#include <LiquidCrystal_I2C.h> // Подключаем библиотеку для работы с LCD дисплеем по шине I2C
LiquidCrystal_I2C lcd(0x3F,20,4); // указываем параметры дисплея
void setup(){
  lcd.init(); // Инициализируем работу с LCD дисплеем
  lcd.backlight(); // Включаем подсветку LCD дисплея
  ... // Выводим информацию, которая должна отображаться при старте
}
void loop(){
  ... // Выводим информацию которая должна меняться по алгоритму Вашего кода
}
```

Функции управления дисплеем:

display();	Примечание: Функция выполняется быстро и без
-------------------	--

Включает дисплей после того как он был выключен функцией noDisplay.	изменений в ОЗУ дисплея.
noDisplay(); Выключает дисплей. Данные на дисплее не будут отображаться до вызова функции display, но и не сотрутся из памяти ОЗУ, а после вызова функции display, опять будут отображаться.	Примечание: Функция выполняется быстро и без изменений в ОЗУ дисплея.
scrollDisplayLeft(); Сдвигает координаты дисплея на один столбец влево. Постоянный вызов данной функции создаст эффект бегущей строки. Координаты сдвигаются как для имеющейся на дисплее информации, так и для той, которая будет выведена после.	Примечание: Функция выполняется без изменений ОЗУ дисплея. Если вызвать функцию 40 раз подряд, то координата вернётся в изначальную точку
scrollDisplayRight(); Сдвигает координаты дисплея на один столбец вправо. Постоянный вызов данной функции создаст эффект бегущей строки. Координаты сдвигаются как для имеющейся на дисплее информации, так и для той, которая будет выведена после.	Примечание: Функция выполняется без изменений ОЗУ дисплея. Если вызвать функцию 40 раз подряд, то координата вернётся в изначальную точку
clear(); Очистка дисплея с установкой курсора в положение 0,0. Информация имеющаяся на дисплее безвозвратно сотрётся.	Примечание: Занимает много времени.
backlight(); Включение подсветки дисплея.	Примечание: Функция реализована только в библиотеке LiquidCrystal_I2C.
noBacklight(); Выключение подсветки дисплея.	Примечание: Функция реализована только в библиотеке LiquidCrystal_I2C.
setBacklight(flag); Управление подсветкой (вместо функций noBacklight и backlight).	Параметр: • flag: значение true - включает, а false - выключает подсветку. Примечание: Функция реализована только в библиотеке LiquidCrystal_I2C.

Пример:

```

/* Выводим надпись для наблюдения за функциями управления дисплеем: */
  lcd.cursor(0,0);    // Устанавливаем курсор в крайний верхний угол дисплея (0 столбец, 0 строка)
  lcd.print("iarduino.ru"); // Выводим текст "iarduino.ru" (первая буква "i" будет находиться в позиции "0,0", а последняя "u" в позиции "10,0", невидимый курсор в позиции "11,0")
  lcd.noDisplay();    // Выключаем дисплей (надпись исчезнет с дисплея)
  lcd.display();      // Включаем дисплей (надпись появится на дисплее в том же месте)
  lcd.scrollDisplayLeft(); //Сдвигаем координаты столбцов влево (на дисплее будет отображаться "arduino.ru" без первой буквы "i", которая выйдет за пределы дисплея, но останется в его ОЗУ)
  lcd.scrollDisplayRight(); // Сдвигаем координаты столбцов вправо (на дисплее будет отображаться "iarduino.ru" на том же месте, где и была выведена изначально)
  lcd.clear();        // Чистим дисплей (надпись безвозвратно исчезнет с дисплея)
  lcd.noBacklight();  // Отключаем подсветку дисплея
  lcd.backlight();    // Включаем подсветку дисплея
  lcd.setBacklight(0); // Отключаем подсветку дисплея
  lcd.setBacklight(1); // Включаем подсветку дисплея

```

Функции управления курсором:

setCursor(col , row); Установка курсора в указанную позицию.	Параметр: • col: номер столбца (начиная с 0). • row: номер строки (начиная с 0)
home(); Установка курсора в позицию 0,0. Работает как функция setCursor(0,0);	Примечание: Занимает много времени.
blink(); Включение мигающего курсора.	Примечание: Курсор занимает всё поле символа и мигает с частотой около 1 Гц, в той позиции где он был установлен ранее.
noBlink(); Выключение мигающего курсора.	Примечание: Курсор становится невидим, но его позиция сохраняется.
cursor(); Включение подчеркивания курсора.	Примечание: Курсор принимает вид символа подчеркивания и находится в той позиции, где он был установлен ранее.
noCursor(); Выключение подчеркивания курсора.	Примечание: Курсор становится невидим, но его позиция сохраняется.

```

lcd.setCursor( 0, 1);           // Устанавливаем курсор на первый символ второй строки (нумерация строк
и столбцов начинается с 0)
lcd.home();                   // Устанавливаем курсор на первый символ первой строки (как при вызове
lcd.setCursor(0,0);
lcd.blink();                   // Делаем курсор видимым (на месте курсора будет мигать прямоугольник)
lcd.noBlink();                // Делаем курсор невидимым (убираем мигающий прямоугольник)
lcd.cursor();                 // Делаем курсор видимым (на месте курсора появится знак подчеркивания)
lcd.noCursor();              // Делаем курсор невидимым (убираем знак подчеркивания)
// Если курсор попадает на место где есть символ, то этот символ не исчезает

```

Функции указывающие направление и выравнивание:

leftToRight(); Указывает, что после каждого нового символа, положение курсора должно сдвигаться на один столбец вправо.	Примечание: Если вывести текст "abc" на дисплее отобразится "abc" и текст будет находиться правее от изначального положения курсора. (Как обычно)
rightToLeft(); Указывает, что после каждого нового символа, положение курсора должно сдвигаться на один столбец влево.	Примечание: Если вывести текст "abc" на дисплее отобразится "cba" и текст будет находиться левее от изначального положения курсора. (Письменность справа налево)
noAutoscroll(); Указывает, что в дальнейшем, текст нужно выравнивать по левому краю от изначальной позиции курсора.	Примечание: если установить курсор в позицию 10,0 и вывести текст, то в данной позиции будет находиться первый символ выведенного текста. (Как обычно)
autoscroll(); Указывает, что в дальнейшем, текст нужно выравнивать по правому краю от изначальной позиции курсора.	Примечание: если установить курсор в позицию 10,0 и вывести текст, то в данной позиции будет находиться курсор. (Координаты дисплея будут сдвинуты влево, как будто Вы вызвали функцию scrollDisplayLeft столько раз, сколько букв в выведенном тексте)

```

lcd.leftToRight(); // Указываем курсору сдвигаться вправо
lcd.clear(); lcd.setCursor(5,0); lcd.print("ABC"); /
lcd.rightToLeft(); // Указываем курсору сдвигаться влево
lcd.clear(); lcd.setCursor(5,0); lcd.print("ABC");
lcd.noAutoscroll(); // Устанавливаем выравнивание по левому краю (Как обычно)
lcd.clear(); lcd.setCursor(5,0); lcd.print("ABC"); // На дисплее увидим: " ABC " (Как обычно)
lcd.autoscroll(); // Устанавливаем выравнивание по правому краю
lcd.clear(); lcd.setCursor(5,0); lcd.print("ABC");

```

Функции ввода текста и символов:

createChar(num,array); Запись пользовательского символа в CGRAM дисплея под указанным номером. Если Вы хотите вывести текст (функцией print()) в котором должен находиться установленный Вами символ, укажите слэш и номер под которым был записан этот символ: print("C\1MBO\2") .	Параметр: • num: номер под которым будет записан символ. • array: массив представляющий записываемый символ. Примечание: Массив состоит из нескольких байт, количество которых равно количеству строк в символе. Каждый установленный бит байта соответствует установленному (отображаемому) пикселю символа.
print(text); Вывод текста, символов или цифр на экран дисплея.	Параметр: • text: символ, число или строка для вывода на дисплей. Примечание: Синтаксис схож с одноимённой функцией класса Serial .

```
#include <Wire.h>           // Подключаем библиотеку для работы с шиной I2C
#include <LiquidCrystal_I2C.h> // Подключаем библиотеку для работы с LCD дисплеем по шине I2C
LiquidCrystal_I2C lcd(0x27,16,2); // Объявляем объект библиотеки, указывая параметры дисплея (адрес I2C = 0x27, количество столбцов = 16, количество строк = 2)

//
uint8_t symbol_d[8] = {0b00000, // 1 строка символа "д"
                      0b00000, // 2 строка символа "д"
                      0b00110, // 3 строка символа "д"
                      0b01010, // 4 строка символа "д"
                      0b01010, // 5 строка символа "д"
                      0b01010, // 6 строка символа "д"
                      0b11111, // 7 строка символа "д"
                      0b10001}; // 8 строка символа "д"  Весь массив можно записать одной строкой: uint8_t
symbol_d[8]={0,0,6,10,10,10,31,17};
//
uint8_t symbol_i[8] = {0b00000, // 1 строка символа "и"
                      0b00000, // 2 строка символа "и"
                      0b10001, // 3 строка символа "и"
                      0b10011, // 4 строка символа "и"
                      0b10101, // 5 строка символа "и"
                      0b11001, // 6 строка символа "и"
                      0b10001, // 7 строка символа "и"
                      0b00000}; // 8 строка символа "и"  Весь массив можно записать одной строкой: uint8_t
symbol_i[8]={0,0,17,19,21,25,17,0};
void setup(){
  //
  lcd.init();           // Иницилируем работу с LCD дисплеем
  lcd.backlight();      // Включаем подсветку LCD дисплея
  lcd.createChar(1,symbol_d); // Загружаем в память дисплея первый символ
  lcd.createChar(2,symbol_i); // Загружаем в память дисплея второй символ
  lcd.clear();          // Чистим экран
  lcd.setCursor(0,0);   // Устанавливаем курсор в крайний верхний угол
  lcd.print("Pa\1\2o"); // Выводим текст "Радио" при этом символы 'Р', 'а', 'о' пишем латиницей,
}                        // а символы 'д', 'и' выводим из памяти дисплея, указывая их номера
//
void loop(){
  //
  lcd.setCursor(0,1); lcd.print("        "); // стираем всю нижнюю строку
  lcd.setCursor(0,1); lcd.print("i"); lcd.print("arduino"); lcd.print(".ru"); // выводим текст "i" "arduino"
  ".ru" в нижней строке
  delay(2000); // ждём 2 секунды
  lcd.setCursor(0,1); lcd.print("        "); // стираем всю нижнюю строку
  lcd.setCursor(0,1); lcd.print(12.345); // выводим число 12.34 (выводится 2 знака
  после запятой)
  delay(2000); // ждём 2 секунды
  lcd.setCursor(0,1); lcd.print("        "); // стираем всю нижнюю строку
  lcd.setCursor(0,1); lcd.print(12, HEX); // выводим число 12 в виде
  шестнадцатиричного числа
  delay(2000); // ждём 2 секунды
```

```

lcd.setCursor(0,1); lcd.print("      ");           // стираем всю нижнюю строку
lcd.setCursor(0,1); lcd.print(1);       // выводим число 1
delay(2000);                             // ждём 2 секунды
}

```

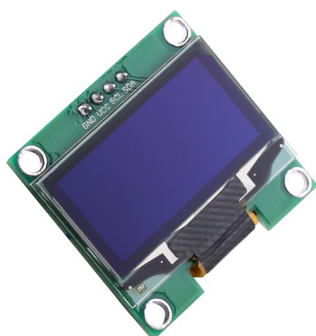
Графические дисплеи.

В OLED (Organic Light-Emitting Diode) дисплеях используется технология в которой светодиоды сами излучают свет без дополнительного подсвечивания как например в LCD дисплеях. Дисплей OLED состоит из тонкой многослойной органической пленки, помещенной между анодом и катодом. OLED обладает высоким потенциалом применения практически для всех типов дисплеев и рассматривается в качестве конечной технологии для следующего поколения плоских дисплеев. Качество отображения информации, дешевизна и идеальные углы обзора OLED дисплея делают его лидером среди дисплеев.

OLED-экраны на базе контроллера SSD1306 популярны благодаря простому подключению, относительно низкой цене и высокому разрешению – для экрана диагональю 0.96 дюйма разрешение составляет аж 128×64.

Важным плюсом OLED-экранов является работа без подсветки – каждый пиксель – сам себе подсветка. За счёт такой системы, экран потребляет крайне мало тока (фактически, его можно запитать от пина Arduino). Есть и один минус – при постоянном использовании отдельные пиксели начинают выгорать и терять яркость, но до наступления этого состояния вы успеете отладить и вывести всё, что только можно.

Особенности OLED L2C дисплея



Не требуется подсветка дисплея
Высокое разрешение: 128 * 64 пикселей.

Угол обзора: больше 160 градусов.

Полностью совместим с Arduino, контроллерами 51 серии, MSP430 серии, STM32 / 2, KCO IC и т.д.

Ультра-низкое энергопотребление: при полном свечении экрана 0.08W

Рабочее напряжение: 3V ~ 5 В постоянного тока.

Рабочая температура: -30 C ~ 70 C.

I2C / IIC интерфейс, нужно только 2 провода.

Драйвер IC: SSD1306.

Размер платы: 2.7см x 2.8см.

Размер дисплея: 2.7см x 1,95см (0,96" дюйма)

Подключение OLED L2C дисплея

Дисплей подключается по высокоскоростному интерфейсу I2C (относительно высокоскоростному – до 400Кбод) и использует всего 2 сигнальных провода. Это ещё один неоспоримый плюс! Несмотря на то, что интерфейс последовательный, да ещё и данные в обе стороны идут по одной линии, на рядовой Arduino можно достичь порядка 15-20fps, чего более чем достаточно для проектов.

Стоит заметить, что дисплей монохромный – цветные картинки на него не выведешь, а для текста или графика хватит и двух цветов.

VCC — +5v (+5 вольт ,но работает и от 3х вольт - проверено на есп 8266)

GND — GND (земля)

SDA — pin SDA (pin A4 для Arduino nano V3)

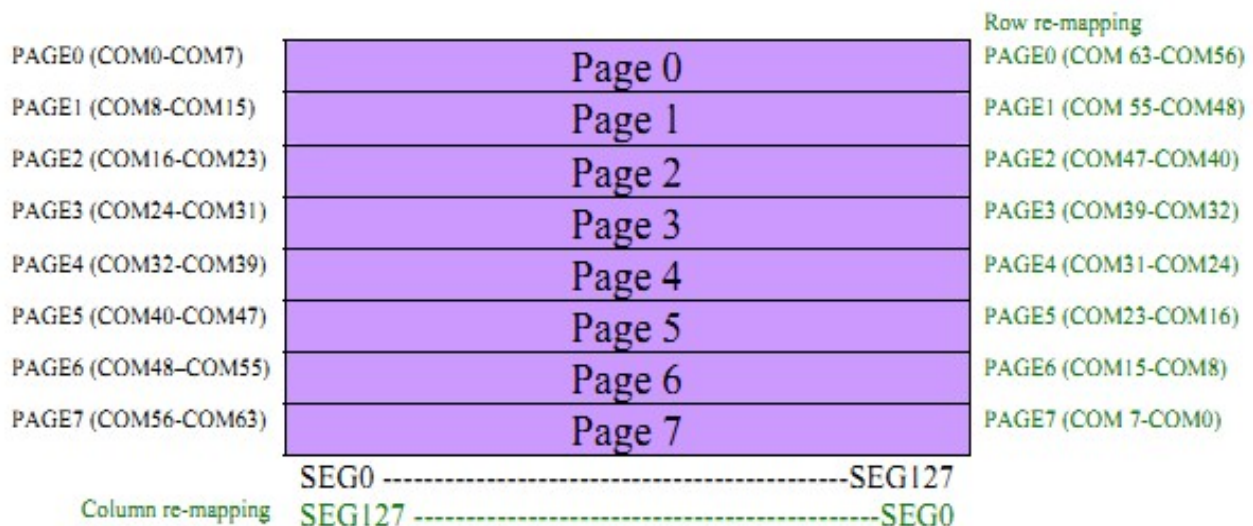
SCL — pin SCL (pin A5 для Arduino nano V3)

Как я уже упоминал, дисплей имеет разрешение 128*64 точки, но ввиду особенностей чипа SSD1306 невозможно получить доступ к каждой точке отдельно. Что бы передать

дисплею значение одной точки достаточно одного бита, но данные передаются байтами. Как вы помните, в одном байте - 8 бит, и бита достаточно для изменения значения сразу 8 точек одновременно (в данном случае эти 8 точек расположены в виде вертикального столбца). Поэтому всё поле дисплея по горизонтали разбито на страницы, в каждой странице 8 строчек. Столбцы же остаются столбцами. Итого 8 страниц и 128 столбцов, а значит для изменения значений всех 8192 пикселей на дисплее необходимо передать 1024 байта.

8 страниц x 128 сегментов x 8 бит данных = 8192 бит = 1024 байт = 1 Кб памяти

Figure 8-13 : GDDRAM pages structure of SSD1306



Каждый бит представляет собой определенный OLED пиксель на экране, который может быть включен или выключен программно. Контроллер SSD1306 OLED дисплея имеет гибкие, но сложные драйверы. Для использования контроллера SSD1306 необходимы огромные знания по адресации памяти. К счастью, была написана библиотека Adafruit SSD1306, которая позволяет довольно простыми и понятными командами управлять OLED дисплеем.

Хотя SSD1306 имеет встроенный GDDRAM для экрана, мы не можем прочесть его содержимое (согласно Adafruit). Следовательно, невозможно управлять экранным буфером для выполнения математических операций.

В качестве альтернативы библиотека выделяет 1 КБ (128 × 64) / 8 бит) памяти ATmega328P в качестве буфера. Таким образом, появляется возможность манипулировать экранным буфером и затем выполнять массовую передачу из памяти ATmega328P во внутреннюю память контроллера SSD1306.

Скетч начинается с подключения четырех библиотек, а именно. SPI.h, Wire.h, Adafruit_GFX.h и Adafruit_SSD1306.h. Хотя библиотека SPI.h не требуется для I2C OLED-дисплеев, нам нужно добавить ее для компиляции нашей программы:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SPI.h>
```

Далее нам нужно создать объект Adafruit_SSD1306.h. Конструктор Adafruit_SSD1306 принимает номер контакта Arduino, к которому подключен вывод сброса дисплея. Поскольку используемый нами OLED-дисплей не имеет вывода RESET, мы отправим в конструктор -1, чтобы ни один из выводов Arduino не использовался в качестве сброса для дисплея.

```
Adafruit_SSD1306 display(-1);
```

В функции `setup()` нам нужно инициализировать объект OLED с помощью функции `begin()`. Функция принимает два параметра. Первый параметр `SSD1306_SWITCHCAPVCC` включает схему charge pump, а второй параметр устанавливает адрес I2C OLED дисплея. I2C адрес такого OLED модуля обычно равен `0x3C`.

Далее мы очищаем буфер перед печатью нашего первого сообщения:

```
// инициализируем I2C addr 0x3C
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
// Очистить буфер.
display.clearDisplay();
```

Отображение простого текста (Hello World)



```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,28);
display.println("Hello world!");
display.display();
delay(2000);
```

Для отображения текста на экране нам нужно установить размер шрифта. Это может быть сделано путем вызова `setFontSize()` и передачи размера шрифта (начиная с 1) в качестве параметра.

Далее нам нужно установить цвет шрифта, вызвав функцию `setTextColor()`. Передайте параметр `WHITE` для темного фона и `BLACK` для яркого фона. Теперь перед печатью сообщения нам нужно установить позицию курсора, вызвав функцию `setCursor(X, Y)`.

Пиксели на экране адресуются по горизонтальным (X) и вертикальным (Y) координатам. Система координат размещает начало координат (0,0) в верхнем левом углу, причем положительный X увеличивается вправо, а положительный Y увеличивается вниз.

Мы можем использовать функцию `print()` или `println()` для печати сообщения на экране так же, как мы печатаем данные на последовательном мониторе. Помните, `println()` переместит курсор на новую строку.

Чтобы библиотека могла выполнять чрезвычайно быстрые математические операции с буфером экрана (более 100 кадров в секунду), вызовы функций печати не сразу передают содержимое экранного буфера в SSD1306 контроллер.

Для этого требуется команда `display()`, чтобы дать указание библиотеке выполнить массовую передачу из экранного буфера ATmega328P во внутреннюю память контроллера SSD1306. Как только память будет перенесена, на OLED-дисплее появятся пиксели, соответствующие экранному буферу.

Инверсия сообщения



```
display.clearDisplay();
display.setTextColor(BLACK, WHITE);
display.setCursor(0,28);
display.println("Hello world!");
display.display();
delay(2000);
```

Для выполнения инверсии мы снова вызываем функцию `setTextColor(FontColor,BackgroundColor)`. Если вы обратили внимание, то вы заметите, что до этого мы передали только один параметр этой функции, но теперь мы передаем два параметра.

Изменение размера шрифта



```
display.clearDisplay();
display.setTextColor(WHITE);
display.setCursor(0,24);
display.setTextSize(2);
display.println("Hello!");
display.display();
delay(2000);
```

Ранее мы вызывали функцию `setTextSize()` для установки размера шрифта и передавали 1 в качестве параметра. Вы можете использовать эту функцию для масштабирования шрифта, передавая любое неотрицательное целое число.

Символы отображаются в соотношении 7:10. Это означает, что при передаче размера шрифта 1 текст будет отображаться с разрешением 7×10 пикселей на символ, при передаче 2 будет отображаться текст с разрешением 14×20 пикселей на символ и т. д.

Отображение чисел



```
display.clearDisplay();
display.setTextSize(1);
display.setCursor(0,28);
display.println(123456789);
display.display();
delay(2000);
```

Числа могут быть отображены на OLED дисплее путем вызова функций `print()` или `println()`.

Указание базиса чисел




```
display.clearDisplay();
display.setCursor(0,28);
display.print("0x"); display.print(0xFF, HEX);
display.print("(HEX) = ");
display.print(0xFF, DEC);
display.println("(DEC)");
display.display();
delay(2000);
```


Функции `print()` и `println()` имеет второй необязательный

	<p>параметр , который определяет базу (формат). Допустимые значения:</p> <p>BIN (двоичное или базовое 2), OCT (восьмеричное или базовое 8), DEC (десятичное или базовое 10), HEX (шестнадцатеричное или базовое 16).</p> <p>Для чисел с плавающей запятой этот параметр указывает количество десятичных знаков. Например:</p> <pre>print(78, BIN) — дает «1001110» print(78, OCT) — дает «116» print(78, DEC) — дает «78» print(78, HEX) — дает «4E» println(1.23456, 0) — дает «1» println(1.23456, 2) — дает «1.23» println(1.23456, 4) — дает «1.2346»</pre>
--	--

Отображение ASCII символов


	<pre>display.clearDisplay(); display.setCursor(0,24); display.setTextSize(2); display.write(3); display.display(); delay(2000);</pre> <p>Функции print() и println() отправляют данные на дисплей в виде удобочитаемого текста ASCII, а функция write() отправляет двоичные данные. Таким образом, вы можете использовать эту функцию для отображения символов ASCII. В нашем примере отправка числа 3 будет отображать символ сердца.</p>
--	--

Полноэкранная прокрутка

	<pre>display.clearDisplay(); display.setCursor(0,0); display.setTextSize(1); display.println("Full"); display.println("screen"); display.println("scrolling!"); display.display(); display.startscrollright(0x00, 0x07); delay(2000); display.stopscroll(); delay(1000); display.startscrollleft(0x00, 0x07); delay(2000); display.stopscroll(); delay(1000); display.startscrolldiagright(0x00, 0x07); delay(2000); display.startscrolldiagleft(0x00, 0x07); delay(2000); display.stopscroll();</pre> <p>Вы можете прокручивать дисплей по горизонтали, вызывая</p>
---	--

	<p>функции <code>startscrollright()</code> и <code>startscrollleft()</code>, и по диагонали, вызывая <code>startscrolldiagright()</code> и <code>startscrolldiagleft()</code>. Все эти функции принимают два параметра, а именно: начальная страница и конечная страница.</p> <p>Поскольку на дисплее отображается восемь страниц от 0 до 7, вы можете прокручивать весь экран, прокручивая все страницы, то есть передавая параметры <code>0x00</code> и <code>0x07</code>. Чтобы остановить отображение прокрутки вы можете использовать функцию <code>stopscroll()</code>.</p>
--	---

Прокрутка определенной части

	<pre>display.setCursor(0,0); display.setTextSize(1); display.println("Scroll"); display.println("some part"); display.println("of the screen."); display.display(); display.startscrollright(0x00, 0x00);</pre> <p>Иногда у нас нет необходимости в прокрутке всей странице. В этом случае мы можете сделать это, передав стартовую страницу и информацию об остановке страницы функциям прокрутки.</p> <p>Поскольку на дисплее отображается восемь страниц от 0 до 7, мы можете прокрутить некоторую часть экрана, передавая конкретные номера страниц в качестве параметров.</p> <p>В нашем примере мы передали оба параметра как <code>0x00</code>. Это позволит прокрутить только первую страницу (первые 8 строк) дисплея.</p>
---	--

Рисование прямоугольника

	<pre>display.clearDisplay(); display.setTextSize(1); display.setTextColor(WHITE); display.setCursor(0,0); display.println("Rectangle"); display.drawRect(0, 15, 60, 40, WHITE); display.display(); delay(2000);</pre> <p>Вы можете нарисовать на дисплее прямоугольник с помощью функции <code>drawRect()</code>. Функция принимает пять параметров, а именно: координаты X и Y, ширина, высота и цвет. На самом деле эта функция рисует не закрашенный прямоугольник с границей в 1 пиксель. Вы можете нарисовать закрашенный прямоугольник, используя функцию <code>fillRect()</code>.</p>
---	--

Рисование скругленный прямоугольник



```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Round Rectangle");
display.drawRoundRect(0, 15, 60, 40, 8, WHITE);
display.display();
delay(2000);
```

Вы можете нарисовать на дисплее скругленный прямоугольник с помощью функции `drawRoundRect()`. Эта функция принимает те же параметры, что и функция `drawRect()`, за исключением одного дополнительного параметра — радиуса скругления угла. На самом деле эта функция рисует не закрашенный скругленный прямоугольник с границей в 1 пиксель. Вы можете нарисовать закрашенный круглый прямоугольник, используя функцию `fillRoundRect()`.

Рисование круга



```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Circle");
display.drawCircle(20, 35, 20, WHITE);
display.display();
delay(2000);
```

Вы можете нарисовать круг на дисплее с помощью функции `drawCircle()`. Функция принимает четыре параметра, а именно: координата центра X и Y, радиус и цвет. Эта функция рисует не закрашенный круг с границей в 1 пиксель. Вы можете нарисовать закрашенный круг, используя функцию `fillCircle()`.

Рисование треугольника



```
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(WHITE);
display.setCursor(0,0);
display.println("Triangle");
display.drawTriangle(30, 15, 0, 60, 60, 60, WHITE);
display.display();
delay(2000);
```

Вы можете нарисовать треугольник на дисплее с помощью функции `drawTriangle()`. Функция принимает семь параметров, а именно: X и Y координаты (x_0 , y_0 , x_1 , y_1 , x_2 , y_2) вершин треугольника и цвета. (x_0 , y_0) представляет верхнюю вершину, (x_1 , y_1) представляет левую вершину и (x_2 , y_2) представляет правую вершину.

Эта функция рисует не закрашенный треугольник с границей в 1 пиксель. Вы можете нарисовать закрашенный треугольник,

	используя функцию fillTriangle().
--	-----------------------------------

Дополнительные функции:

Чтобы нарисовать пиксель на OLED-дисплее, можно использовать метод drawPixel(x, y, color), который принимает в качестве аргументов координаты x и y, где появляется пиксель, и цвет. Например

display.drawPixel(64, 32, WHITE);

Нарисовать линию

Используйте метод drawLine(x1, y1, x2, y2, color) для создания линии. Координаты (x1, y1) указывают начало линии, а координаты (x2, y2) указывают, где заканчивается линия.

Например

display.drawLine(0, 0, 127, 20, WHITE);

Инвертирование

Библиотека предоставляет дополнительный метод, который можно использовать с фигурами или текстом: метод invertDisplay (). Передайте true в качестве аргумента, чтобы инвертировать цвета экрана, или false, чтобы вернуться к исходным цветам.

Если вы вызываете следующую команду после определения треугольника

display.invertDisplay(true);

Вы получите черный треугольник, а фон будет подсвечен.

Задания к лабораторной работе №4

Для Знакосинтезирующего дисплея:

1. Выведите сообщение «Hello, world!» по середине экрана на верхней строке.
2. Вывод сообщения во вторую строку из консоли
3. Вывод температуры и влажности с датчика DHT11 на экран.
4. Создание термостата, температура срабатывания задается переменным резистором от 10 до 30 градусов Цельсия. Гистрезис задается также переменным резистором. Выбор что устанавливается делается кнопкой.

Для графического дисплея:

Задания:

1. Выведите сообщение «Hello, world!», начиная с позиции 28 по оси X и 16 по оси Y;
2. Увеличьте размер шрифта и выведите текстовое сообщение на OLED;
3. Выведите два тестовых сообщения на разных строках дисплея;
4. Сделайте прокрутку сообщения сначала справа налево, а потом слева направо;
Показать ответ »
- 5 Нарисуйте квадрат 20×20 , координаты левой крайней точкой (10, 10).
Прокручивайте его в бесконечном цикле слева направо.
6. Вывести график данных с датчика освещенности макета