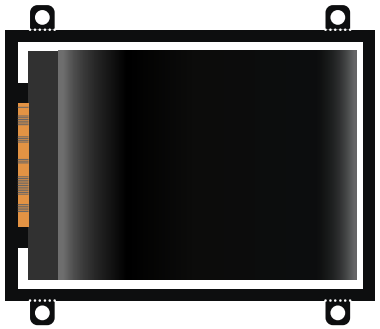
# Цветной графический дисплей 2.8 TFT 320x240



## Общие сведения:

***Цветной графический дисплей 2.8*** *и* ***цветной графический дисплей 2.8 с TouchScreen*** — дисплеи, разработанные нашей компанией специально для работы с Arduino UNO / MEGA.

Благодаря поддержке огромного количества цветов (более 260 тысяч), а так же библиотекам UTFT и TouchScreen, вы сможете выводить на дисплеи разноцветные фигуры, анимацию, а так же текст (цвет и форму которого вы тоже сможете выбрать).

Дисплеи оснащены преобразователем уровней, благодаря чему вы можете подключать их как к 3В, так и к 5В логике!

Данные дисплеи поддерживают возможность регулировки яркости подсветки, а конструкция их такова, что они удобно устанавливаются в ***ПВХ-конструктор*** даже без дополнительной фиксации. Впрочем, если вам понадобится дополнительная жёсткость, то для этого на дисплеях имеется 4 крепёжных петли.

## Спецификация:

### Дисплей без сенсорной панели

- **Разрешение**: 320x240 точек;
- **Цветопередача**: 262 тысячи цветов;
- **Диагональ**: 2.8 дюйма;
- **Интерфейс подключения**: SPI;
- **Крепёжные отверстия:** M3;

- **Размер**: 75мм x 65мм;

### Дисплей с сенсорной панелью

- **Разрешение**: 320x240 точек;
- **Цветопередача**: 262 тысячи цветов;
- **Диагональ**: 2.8 дюйма;
- **Тип сенсора:**резистивный;
- **Интерфейс подключения**: SPI;
- **Крепёжные отверстия:** ⌀ 3мм;
- **Размер**: 75мм x 65мм;

## Подключение:

Для удобства подключения к Arduino воспользуйтесь Trema Shield, Trema Power Shield, Motor Shield или Trema Set Shield.

| **Выводы дисплея без сенсорной панели** | **Выводы Arduino** |
|---|---|
| GND (обязательно) | GND |
| VCC (обязательно) | 5V |
| D/C (обязательно) | Любой вывод |
| RST (обязательно) | Любой вывод |
| CS (обязательно) | Любой вывод |
| SCK (обязательно) | Любой вывод |
| MOSI (обязательно) | Любой вывод |
| LED (регулировка яркости, опционально) | Любой вывод |
| MISO (опционально) | Любой вывод |

| **Выводы дисплея с сенсорной панелью** | **Выводы Arduino** |
|---|---|
| GND (обязательно) | GND |
| VCC (обязательно) | 5V |
| D/C (обязательно) | Любой вывод |
| RST (обязательно) | Любой вывод |
| CS (обязательно) | Любой вывод |
| SCK (обязательно) | Любой вывод |
| MOSI (обязательно) | Любой вывод |
| LED (регулировка яркости, опционально) | Любой вывод |
| MISO (опционально) | Любой вывод |
| YU (Y+) (обязательно) | Любой аналоговый вывод |
| XR (X-) (обязательно) | Любой аналоговый вывод |
| YD (Y-) (обязательно) | Любой вывод |
| XL (X+) (обязательно) | Любой вывод |

Далее, в примерах подключения и в примерах скетчей будет выполнено подключение согласно следующей таблиц:

| **Выводы дисплея** | **Выводы Arduino** |
|---|---|
| GND | GND |
| VCC | 5V |
| D/C | 4 |

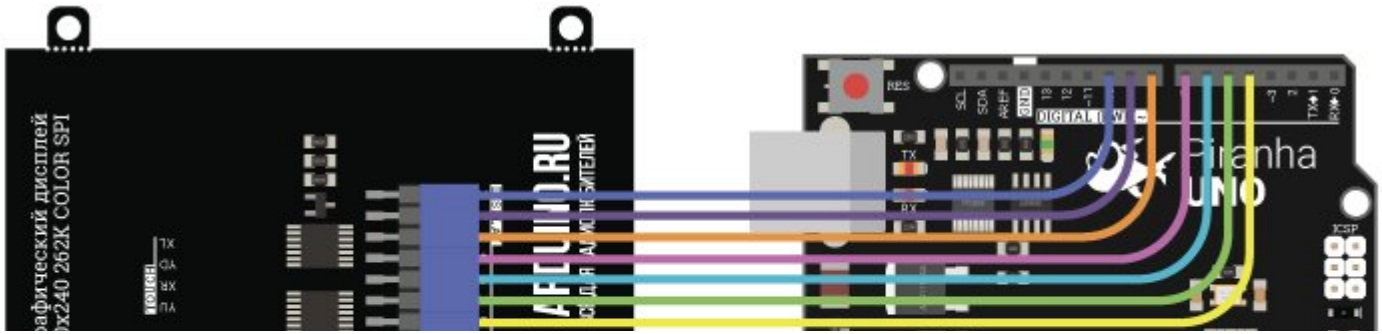
| | |
|---|---|
| RST | 5 |
| CS | 6 |
| SCK | 7 |
| MOSI | 8 |
| LED | 9 |
| MISO | 10 (кроме примера с SD картой) |
| YU (Y+) | A0 |
| XR (X-) | A1 |
| YD (Y-) | A2 |
| XL (X+) | A3 |

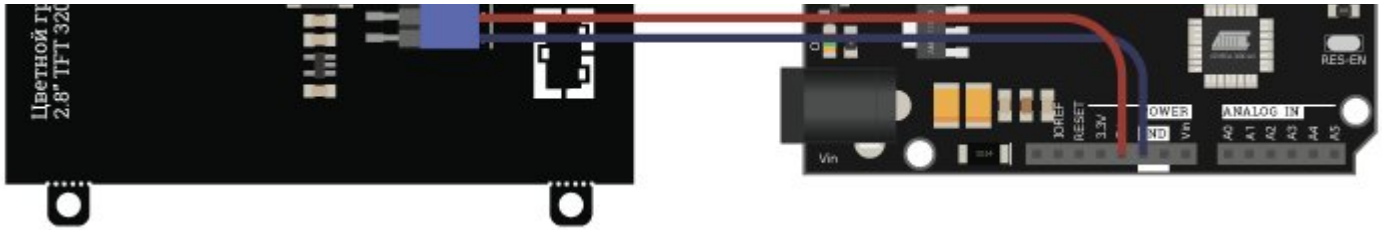
Модуль удобно подключать 3 способами, в зависимости от ситуации:

## Способ - 1 : Используя проводной шлейф и Piranha UNO

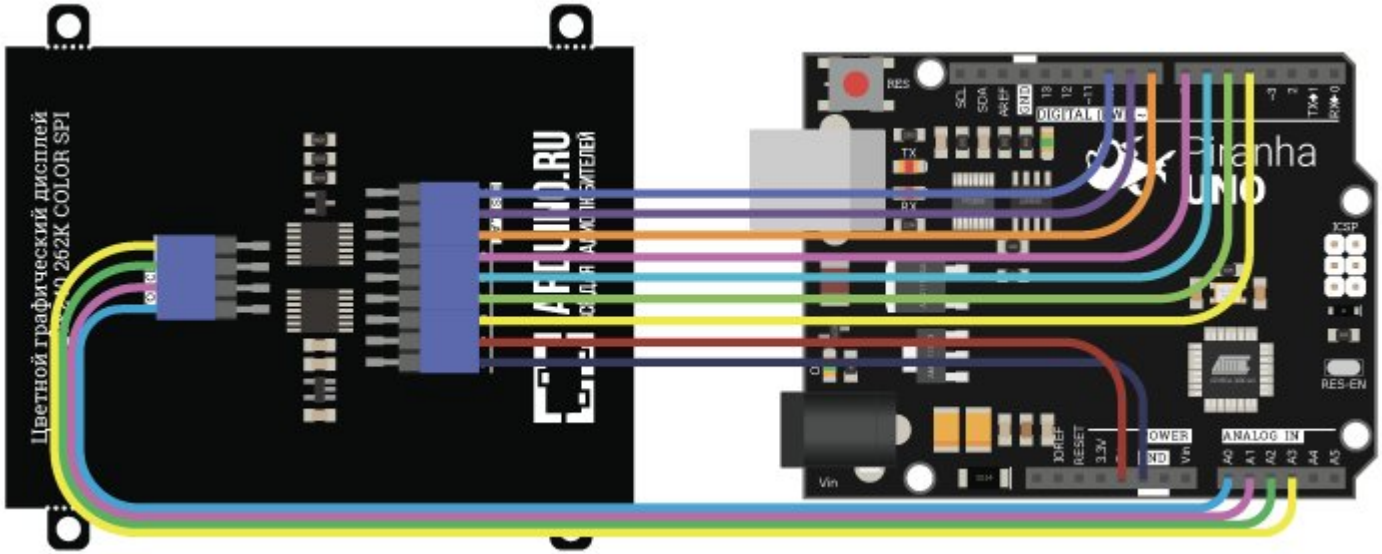
Используя провода «Папа — Мама», подключаем напрямую к контроллеру Piranha UNO

***Дисплей без сенсорной панели***





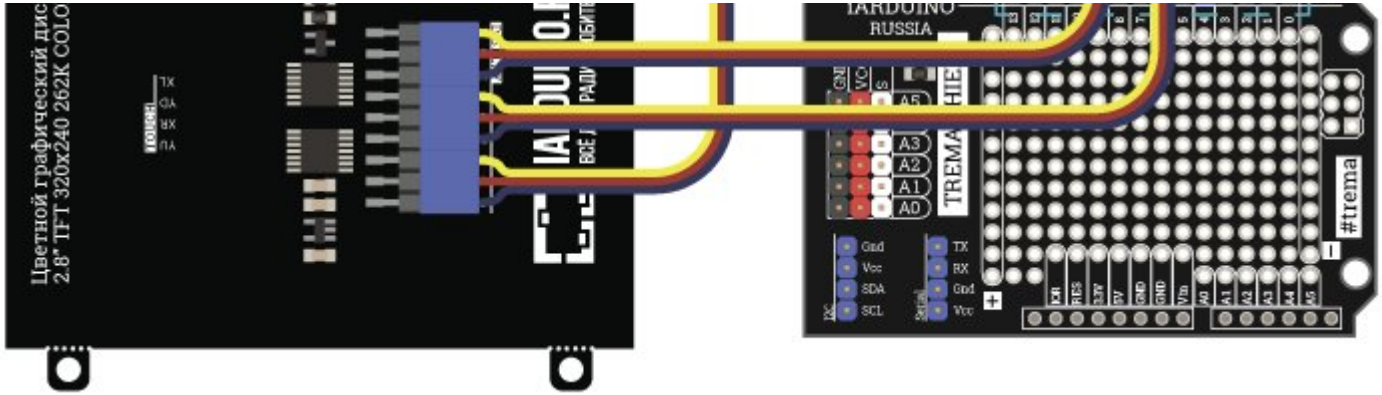
***Дисплей с сенсорной панелью***



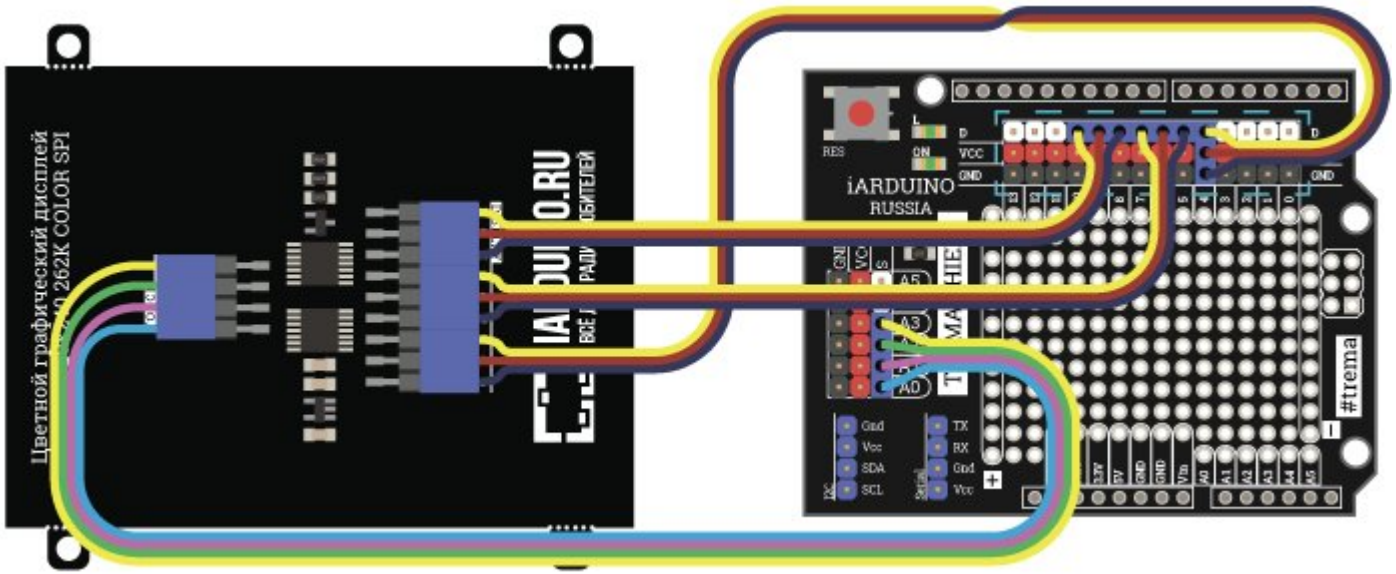

## Способ - 2 : Используя проводной шлейф и Shield

Используя 3-х проводные шлейфы "Мама — Мама", к Trema Shield, Trema-Power Shield, Motor Shield, Trema Shield NANO и тд.

***Дисплей без сенсорной панели***



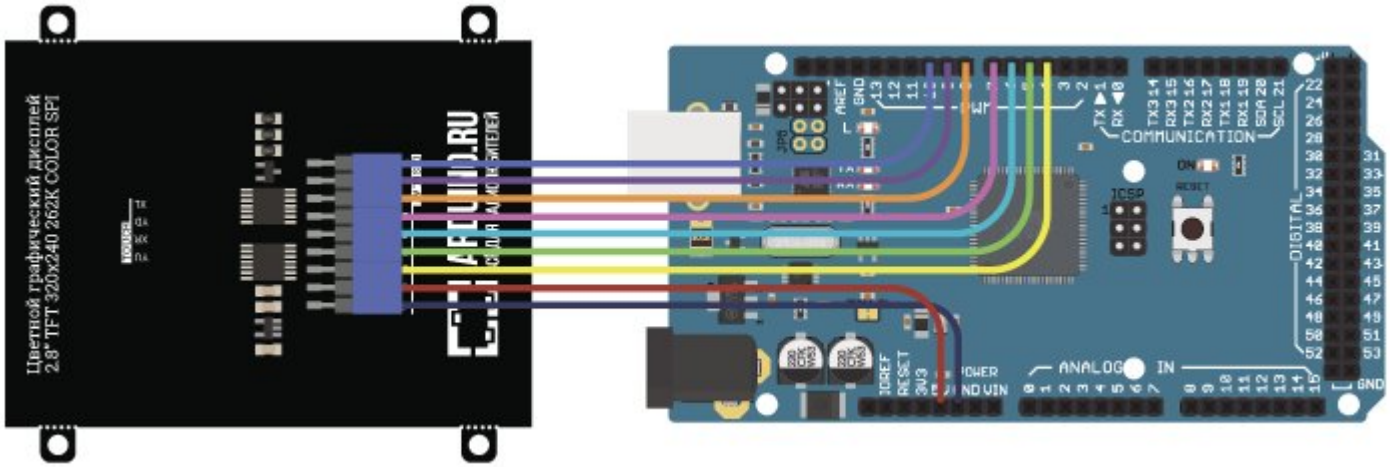

***Дисплей с сенсорной панелью***



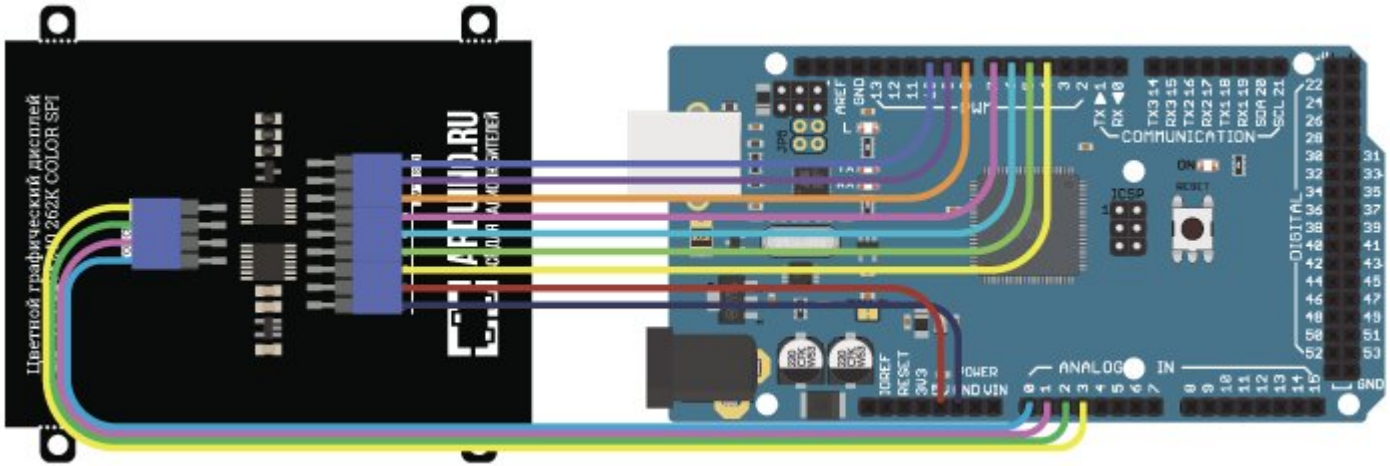

## Способ - 3 : Используя проводной шлейф и Arduino Mega

Используя провода «Папа — Мама», подключаем напрямую к контроллеру Arduino MEGA

***Дисплей без сенсорной панели***




***Дисплей с сенсорной панелью***




## Питание:

Данный дисплей питается от постоянного источника напряжения 5В.

## Библиотеки для работы с дисплеями:

Для работы с дисплеями существует несколько библиотек, каждая из которых может вам пригодится в разных проектах. Опишем кратко каждую из них:

- Библиотека UTFT для работы с цветными графическими дисплеями (описание функций и примеры смотрите ***тут***)
- Библиотека TouchScreen для работы с сенсорными экранами (описание функций и примеры смотрите ***тут***)

## Форм-фактор:

Данный дисплей выпущен в нашем новом формате, который был специально спроектирован с учётом размеров и расположения отверстий в ***ПВХ-конструкторе***, благодаря чему дисплей легко и удобно в нём закрепить.

Помимо этого, мы разработали специально для данных дисплеев ***крепёжную пластину***, которая так же подходит к нашим ПВХ-корпусам ***Set Box*** и ***Set Box XL***.

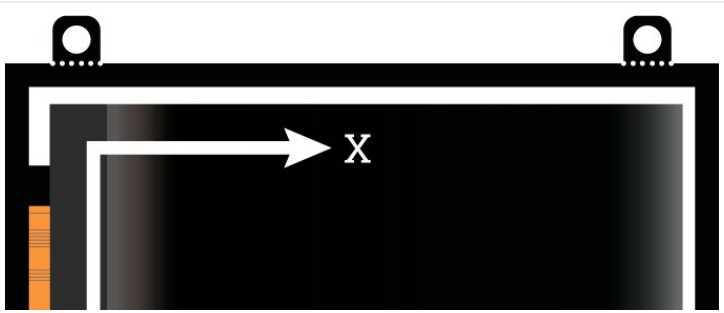
## Подробнее о дисплее:

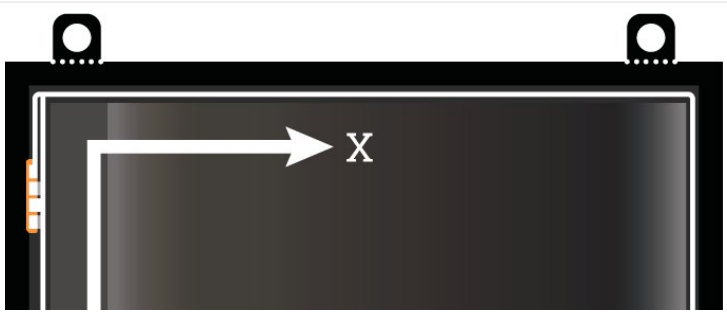
На нашей Wiki имеются статьи, в которых мы очень подробно разобрали все особенности работы с дисплеями того или иного типа. Ниже вы найдёте ссылки на эти статьи:

- Информацию о том, как работать с **графическим дисплеем** вы найдёте ***тут***.
- Информацию о том, как работать с **графическим дисплеем с touchscreen** вы найдёте ***тут***.

### Координатная сетка

Начало координатной сетки лежит в верхнем левом углу как показано на рисунках ниже.





## Примеры:

### Вывод текста на дисплей

```
// подключаем библиотеку UTFT
#include <UTFT.h>

// Определяем выводы используемые для
// управления дисплеем 2.8" TFT 320x240 UNO:
#define dispMISO    8
#define dispSCK     7
#define dispCS      6
#define dispRST     5
#define dispDC      4

// подключаем шрифты
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];

// объявляем объект myGLCD класса библиотеки UTFT указывая тип дисплея
UTFT myGLCD(TFT01_24SP, dispMISO,
            dispSCK, dispCS,
            dispRST, dispDC);
```

```
void setup(){
    // инициируем дисплей
    myGLCD.InitLCD();
}

void loop(){

    // стираем всю информацию с дисплея
    myGLCD.clrScr();

    // устанавливаем маленький шрифт
    myGLCD.setFont(SmallFont);
    // устанавливаем белый цвет текста
    myGLCD.setColor(VGA_WHITE);
    // выводим текст на дисплей (выравнивание по ширине -
    // центр дисплея, координата по высоте 50 точек)
    myGLCD.print("iarduino.ru | SmallFont", CENTER, 50);

    // устанавливаем большой шрифт
    myGLCD.setFont(BigFont);
    // устанавливаем синий цвет текста
    myGLCD.setColor(VGA_BLUE);
    // выводим текст на дисплей (выравнивание по ширине -
    // центр дисплея, координата по высоте 100 точек)
    myGLCD.print("BigFont", CENTER, 100);
    // выводим текст на дисплей (выравнивание по ширине -
    // центр дисплея, координата по высоте 115 точек)
    myGLCD.print("iarduino.ru", CENTER, 115);

    // устанавливаем шрифт имитирующий семисегментный индикатор
    myGLCD.setFont(SevenSegNumFont);
    // устанавливаем пурпурный цвет текста
```

```
    myGLCD.setColor(VGA_FUCHSIA);
    // выводим текст на дисплей (выравнивание по ширине -
    // центр дисплея, координата по высоте 150 точек)
    myGLCD.print("1234567890", CENTER, 150);

    // ждём 20 секунд
    delay(20000);
}
```

## Вывод фигур на дисплей

```
// подключаем библиотеку UTFT
#include <UTFT.h>

// Определяем выводы используемые для
// управления дисплеем 2.8" TFT 320x240 UNO:
#define dispMISO    8
#define dispSCK     7
#define dispCS      6
#define dispRST     5
#define dispDC      4

// объявляем объект myGLCD класса библиотеки UTFT указывая тип дисплея
UTFT myGLCD(TFT01_24SP, dispMISO,
            dispSCK, dispCS,
            dispRST, dispDC);

void setup(){
    // инициируем дисплей
    myGLCD.InitLCD();
    myGLCD.clrScr();
}
```

```
void loop(){
   int n = random(7);
   int j = random(239);
   for (int i = random(239); i > 0; i--) {
       // Устанавливаем цвет RGB
       myGLCD.setColor(i, -i, j);
       switch (n) {
           case 0:
               // Рисуем прямоугольник
               myGLCD.drawRect(i*j, i, i*j+i, i+j);
               break;

           case 1:
               // Заполняем прямоугольник
               myGLCD.fillRect(i*j, i, i*j+5, i+5);

           case 2:
               // Рисуем круг
               myGLCD.drawCircle(i*j, i, i/5);
               break;

           case 3:
               // Заполняем круг
               myGLCD.fillCircle(i*j, i, 5);
               break;

           case 4:
               // Рисуем прямоугольник
               myGLCD.drawRoundRect(i*j, i, i*j+10, i+10);
               break;

           case 5:
```

```
                // Рисуем прямоугольник
                myGLCD.drawLine(i*j, i, i*j+i, i+j);
                break;

            default:
                break;
        }
    }
}
```

## Вывод изображения часов на дисплей

```
//  Подключаем библиотеки:
#include "UTFT.h"                                   // Подключаем библиотеку для работы с дисплеем
//  Определяем выводы используемые для управления дисплеем 2.8" TFT 320x240 UNO:
const uint8_t RS   = 8;                             //
const uint8_t WR   = 7;                             //
const uint8_t CS   = 6;                             //
const uint8_t RST  = 5;                             //
const uint8_t SER  = 4;                             //

UTFT    myGLCD(TFT01_24SP, RS, WR, CS, RST, SER);   // Создаём объект для работы с дисплеем

// НАСТРОЙКИ ДИСПЛЕЯ
extern uint8_t  SmallFont[];                        // подключаем маленький шрифт
extern uint8_t  BigFont[];                          // подключаем большой шрифт
extern uint8_t  SevenSegNumFont[];                  // подключаем шрифт имитирующий семисегментный индикатор

uint32_t TIMER_DISPLAY;                             // таймер для вывода времени игры трека на дисплей
uint8_t  HOURS   = 0;                               // переменная для секунд
uint8_t  MINUTES = 0;                               // переменная для минут
uint8_t  SECONDS = 0;                               // переменная для секунд
```

```
void setup(void) {                                          //
  Serial.begin(9600);                                       // Инициируем передачу данных в монитор последовательного порта на скор
  // НАСТРОЙКА LCD-ДИСПЛЕЯ
  myGLCD.InitLCD();                                         // инициируем дисплей
  myGLCD.setBackColor (VGA_WHITE);                          // устанавливаем цвет заливки фона шрифта
  myGLCD.setColor     (VGA_RED);                            // устанавливаем красный цвет текста
  myGLCD.clrScr();                                          // стираем всю информацию с дисплея
  myGLCD.fillScr      (VGA_WHITE);                          // указываем цвет заливки экрана
  // ВЫВОД ДВОЕТОЧИЯ
  myGLCD.setColor  (VGA_BLACK);                             // устанавливаем чёрный цвет текста

  myGLCD.fillCircle(120, 157, 4);                           // рисуем точку в указанных координатах с радиусом 4
  myGLCD.fillCircle(120, 172, 4);                           // рисуем точку в указанных координатах с радиусом 4

  myGLCD.fillCircle(215, 157, 4);                           // рисуем точку в указанных координатах с радиусом 4
  myGLCD.fillCircle(215, 172, 4);                           // рисуем точку в указанных координатах с радиусом 4
  myGLCD.setFont  (BigFont);                                // устанавливаем большой шрифт
  myGLCD.print("iArduino Clock", CENTER, 20);               // выводим текст в указанных координатах
  myGLCD.print("Day:  Monday", CENTER, 70);                 // выводим текст в указанных координатах
  myGLCD.print("Date: 22.04.19", CENTER, 100);              // выводим текст в указанных координатах
  myGLCD.print("H", 63, 200);                               // выводим текст в указанных координатах
  myGLCD.print("M", 158, 200);                              // выводим текст в указанных координатах
  myGLCD.print("S", 253, 200);                              // выводим текст в указанных координатах
}

void loop() {
  if (millis() - TIMER_DISPLAY > 1000) {                    // выполняем 1 раз в 1000мс
    TIMER_DISPLAY = millis();                               // обновляем счётчик
    SECONDS++;                                              // увеличиваем значение секунд на 1 каждую 1000мс
    if (SECONDS > 59) {                                     // если значение секунд превысило значение 59, то
      SECONDS = 0;                                          // сбрасываем значение секунд в 0 и
      MINUTES++;                                            // увеличиваем значение минут на 1
```

```
    }
    if (MINUTES > 60) {                                 // если значение минут превысило 59, то
      MINUTES = 0;                                      // сбрасываем значение минут в 0 и
      HOURS++;                                          // увеличиваем значение часов на 1
    }
    myGLCD.setFont  (SevenSegNumFont);                  // устанавливаем шрифт с 7-сегментного дисплея
    myGLCD.printNumI(HOURS,    40,  140,  2,  '0');     // выводим часы
    myGLCD.printNumI(MINUTES, 135,  140,  2,  '0');     // выводим минуты
    myGLCD.printNumI(SECONDS, 230,  140,  2,  '0');     // выводим секунды
  }
}
```

## Создание программы для рисования, которая использует TouchScreen

```
//  Подключаем библиотеки:
#include "UTFT.h"                                       // Подключаем библиотеку для работы с дисплеем
#include "TouchScreen.h"                                // Подключаем библиотеку для работы с TouchScreen
//  Определяем выводы используемые для управления дисплеем 2.8" TFT 320x240 UNO:
const uint8_t RS   = 8;                                 //
const uint8_t WR   = 7;                                 //
const uint8_t CS   = 6;                                 //
const uint8_t RST  = 5;                                 //
const uint8_t SER  = 4;                                 //
//  Определяем выводы используемые для чтения данных с TouchScreen:
const uint8_t YP   = A0;                                // Вывод Y+ должен быть подключен к аналоговому входу
const uint8_t XM   = A1;                                // Вывод X- должен быть подключен к аналоговому входу
const uint8_t YM   = A2;                                // Вывод Y-
const uint8_t XP   = A3;                                // Вывод X+
//  Определяем экстремумы для значений считываемых с аналоговых входов при определении точек нажатия на TouchScreen:
const int tsMinX   = 100;                               // соответствующий точке начала координат по оси X
const int tsMinY   = 120;                               // соответствующий точке начала координат по оси Y
const int tsMaxX   = 950;                               // соответствующий максимальной точке координат по оси X
```

```
const int tsMaxY   = 915;                                  // соответствующий максимальной точке координат по оси Y
const int mipPress = 10;                                   // соответствующий минимальной степени нажатия на TouchScreen
const int maxPress = 1000;                                 // соответствующий максимальной степени нажатия на TouchScreen
const uint8_t pen_radius = 3;                              // указываем размер точки, которая будет появляться при рисовании
//  Создаём объекты библиотек:
UTFT    myGLCD(TFT01_24SP, RS, WR, CS, RST, SER);          // Создаём объект для работы с дисплеем
TouchScreen ts     = TouchScreen(XP, YP, XM, YM);          // Создаём объект для работы с TouchScreen
void setup(void) {                                         //
  Serial.begin(9600);                                      // Инициируем передачу данных в монитор последовательного порта на скор
  myGLCD.InitLCD();                                        // Инициируем работу с TFT дисплеем
  myGLCD.clrScr();                                         // Чистим экран дисплея
  myGLCD.setColor(VGA_WHITE);                              // Указываем белый цвет
  myGLCD.drawLine(0,   0, 40,  40);                        // Рисуем линию
  myGLCD.drawLine(40,  0, 0,   40);                        // Рисуем линию
  myGLCD.setColor(VGA_RED);                                // Указываем красный цвет
  myGLCD.fillRect(0,  40, 40,  80);                        // Рисуем квадрат
  myGLCD.setColor(VGA_GREEN);                              // Указываем зелёный цвет
  myGLCD.fillRect(0,  80, 40, 120);                        // Рисуем квадрат
  myGLCD.setColor(VGA_YELLOW);                             // Указываем жёлтый цвет
  myGLCD.fillRect(0, 120, 40, 160);                        // Рисуем квадрат
  myGLCD.setColor(VGA_PURPLE);                             // Указываем фиолетовый цвет
  myGLCD.fillRect(0, 160, 40, 200);                        // Рисуем квадрат
  myGLCD.setColor(VGA_BLUE);                               // Указываем синий цвет
  myGLCD.fillRect(0, 200, 40, 240);                        // Рисуем квадрат
}                                                          //
void loop() {                                              //
  //  Считываем показания с TouchScreen:
  TSPoint p = ts.getPoint();                               // Считываем координаты и интенсивность нажатия на TouchScreen в структ
  //  Возвращаем режим работы выводов:
  pinMode(XM, OUTPUT);                                     // Возвращаем режим работы вывода X- в значение «выход» для работы с ди
  pinMode(YP, OUTPUT);                                     // Возвращаем режим работы вывода Y+ в значение «выход» для работы с ди
  //  Если зафиксировано нажатие на TouchScreen, то ...
```

```
  if (p.z > mipPress && p.z < maxPress) {             // Если степень нажатия достаточна для фиксации координат TouchScreen
    p.x = map(p.x, tsMinX, tsMaxX, 0, 320);           // Преобразуем значение p.x от диапазона tsMinX...tsMaxX, к диапазону 0
    p.y = map(p.y, tsMinY, tsMaxY, 0, 240);           // Преобразуем значение p.y от диапазона tsMinY...tsMaxY, к диапазону 0
    if (p.x <= 40) {                                  // Если нажатие было в зоне выбора цветов, то
      if (p.y < 40 ) {                                // Если нажатие было в зоне кнопки "очищение экрана", то
        myGLCD.clrScr();                              // Чистим экран дисплея
        myGLCD.setColor(VGA_WHITE);                   // Указываем цвет
        myGLCD.drawLine(0,   0, 40,  40);             // Рисуем линию
        myGLCD.drawLine(40,   0, 0,  40);             // Рисуем линию
        myGLCD.setColor(VGA_RED);                     // Указываем цвет
        myGLCD.fillRect(0,  40, 40,  80);             // Рисуем квадрат
        myGLCD.setColor(VGA_GREEN);                   // Указываем цвет
        myGLCD.fillRect(0,  80, 40, 120);             // Рисуем квадрат
        myGLCD.setColor(VGA_YELLOW);                  // Указываем цвет
        myGLCD.fillRect(0, 120, 40, 160);             // Рисуем квадрат
        myGLCD.setColor(VGA_PURPLE);                  // Указываем цвет
        myGLCD.fillRect(0, 160, 40, 200);             // Рисуем квадрат
        myGLCD.setColor(VGA_BLUE);                    // Указываем цвет
        myGLCD.fillRect(0, 200, 40, 240);             // Рисуем квадрат
      } else if (p.y > 40 && p.y <= 80 ) {            // Если нажатие было в зоне красного квадрата, то
        myGLCD.setColor(VGA_RED);                     // Устанавливаем цвет
      } else if (p.y > 80 && p.y <= 120 ) {           // Если нажатие было в зоне зелёного квадрата, то
        myGLCD.setColor(VGA_GREEN);                   // Устанавливаем цвет
      } else if (p.y > 120 && p.y <= 160 ) {          // Если нажатие было в зоне жёлтого квадрата, то
        myGLCD.setColor(VGA_YELLOW);                  // Устанавливаем цвет
      } else if (p.y > 160 && p.y <= 200 ) {          // Если нажатие было в зоне фиолетового квадрата, то
        myGLCD.setColor(VGA_PURPLE);                  // Устанавливаем цвет
      } else if (p.y > 200 && p.y <= 240 ) {          // Если нажатие было в зоне синего квадрата, то
        myGLCD.setColor(VGA_BLUE);                    // Устанавливаем цвет
      }
    }
    if (p.x - pen_radius > 40) {                      // Если нажатие было в зоне рисования, то
      myGLCD.fillCircle(p.x, p.y, pen_radius);        // Прорисовываем окружность диаметром 3 пикселя с центром в точке коорд
```

```
    }
  }
}
```

## Регулировка яркости подсветки:

### Как подключить

Вывод **LED** подключается к любому выводу ARDUINO, который поддерживает ШИМ (выводы со знаком "**~**" тильды на шелкографии платы). Яркость регулируется изменением процента заполнения ШИМ. Например, вызов **analogWrite**(dispLED, 128) установит половину яркости.

### Пример работы подсветки дисплея (при использовании датчика освещённости)

```
// Подключаем библиотеку для работы с дисплеем
#include "UTFT.h"

// Определяем выводы используемые для
// управления дисплеем 2.8" TFT 320x240 UNO:
#define dispLED     9
#define dispMISO    8
#define dispSCK     7
#define dispCS      6
#define dispRST     5
#define dispDC      4
#define LIGHT_SENS  A0

// переменная для значений с датчика освещённости
uint16_t      LIGHT_SENS_VAL;
// переменная для значений яркости дисплея
uint16_t      LED_VAL;

// объявляем объект myGLCD класса библиотеки UTFT указывая тип дисплея
```

```
UTFT myGLCD(TFT01_24SP, dispMISO,
            dispSCK, dispCS,
            dispRST, dispDC);

void setup(void) {

    // конфигурируем вывод регулировки подсветки как ВЫХОД
    pinMode(dispLED, OUTPUT);

    // НАСТРОЙКА LCD-ДИСПЛЕЯ

    // инициируем дисплей
    myGLCD.InitLCD();
    // устанавливаем цвет заливки фона шрифта
    myGLCD.setBackColor (VGA_WHITE);
}

void loop() {

    // считываем значение с датчика освещённости
    LIGHT_SENS_VAL = analogRead(LIGHT_SENS);

    // переопределяем диапазон значений для ШИМ-сигнала
    LED_VAL = map(LIGHT_SENS_VAL, 0, 1023, 0, 255);

    // выводим значение ШИМ-сигнала
    analogWrite(dispLED, LED_VAL);

}
```

## Работа с SD-картами:

Используя в связке с дисплеем модуль SD-карт, вы можете выводить анимацию и изображения, которые физически не могут быть размещены в памяти микроконтроллера.

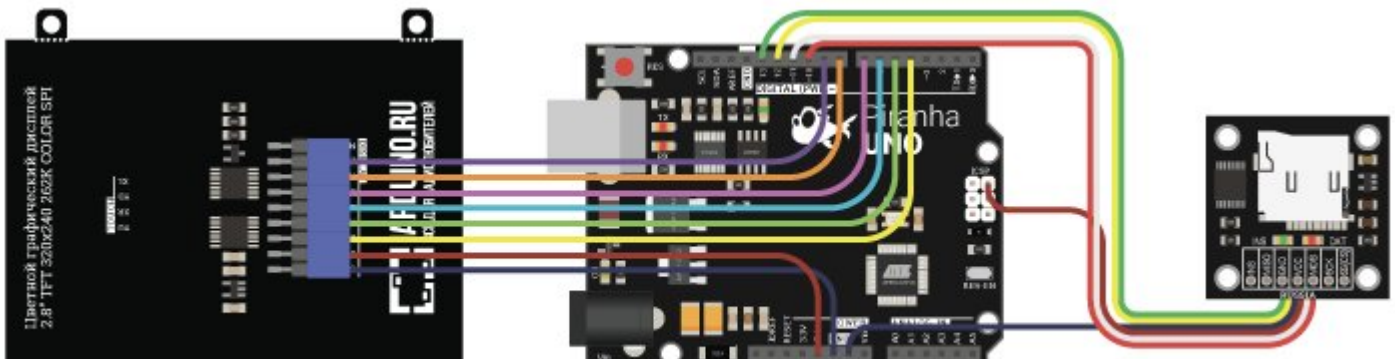
## Как подключить

Используя провода «Папа — Мама», подключаем  напрямую к контроллеру Piranha UNO

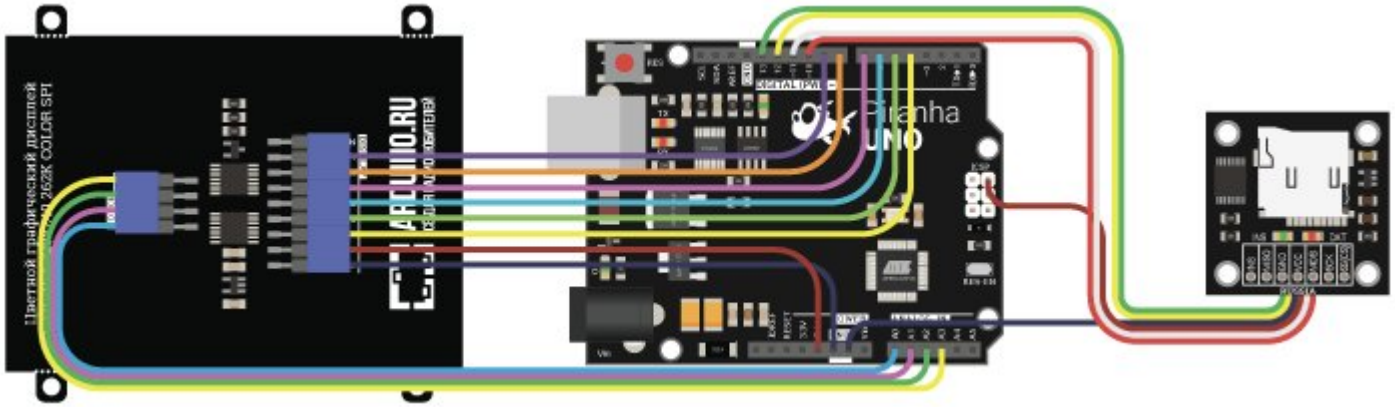
В данном примере Адаптер карт microSD подключается к Piranha UNO следующим образом:

| **Выводы Адаптера** | **Выводы Arduino** |
|---|---|
| INS | - |
| MISO | 12 |
| GND | GND |
| VCC | 5V |
| MOSI | 11 |
| SCK | 13 |
| SS(CS) | 10 |

***Дисплей без сенсорной панели***



*Дисплей с сенсорной панелью*




## Пример работы

## Вывод на дисплей и плавная смена 2 изображений.

```
#include   "SPI.h"                                    // подключаем библиотеку SPI           для общения с SD-картой по ш
#include   "SdFat.h"                                  // подключаем библиотеку SdFat         для работы с SD-картой
#include   "UTFT.h"                                   // подключаем библиотеку UTFT          для работы с дисплеем
#include   "UTFT_SdRaw.h"                             // подключаем библиотеку UTFT_SdRaw    для вывода изображений с SD-
#define    SD_CHIP_SELECT SS                          // определяем константу  SD_CHIP_SELECT которой присваиваем номер вы
                                                      //
UTFT       myGLCD(TFT01_24SP, 8, 7, 6, 5, 4);         // объявляем объект myGLCD  класса библиотеки UTFT        указывая
SdFat      mySD;                                      // объявляем объект mySD    класса библиотеки SdFat       для работ
UTFT_SdRaw myFiles(&myGLCD);                          // объявляем объект myFiles класса библиотеки UTFT_tinyFAT с передач
                                                      //
void setup() {                                        //
    myGLCD.InitLCD();                                 // инициируем дисплей
    myGLCD.clrScr();                                  // стираем всю информацию с дисплея
    while (!mySD.begin(SD_CHIP_SELECT)) {}            // инициируем работу с SD картой, ожидая завершения инициализации в
```

```
}                                                                //
                                                                 //
void loop() {                                                    //
    myFiles.load(0, 0, 320, 240, "image_320x240_a.raw");  // выводим на дисплей картинку начиная с координаты 0,0 размером 320
    delay(2000);                                                 // ждём 2 секунды
    myFiles.load(0, 0, 320, 240, "image_320x240_b.raw");  // выводим на дисплей картинку начиная с координаты 0,0 размером 320
    delay(2000);                                                 // ждём 2 секунды
}
```

## Применение:

- Вывод текста, изображений, анимации;
- Создание управления проектами с помощью touchscreen;