Dr. Bob Davidov

Построение интерфейса пользователя локальной системы управления на базе контроллера Arduino UNO.

Цель работы: освоение правил подключения и настройки средств отображения воздействий и состояния локальной системы.

Задача работы: построение системы отображения параметров локальной системы средствами Simulink сетевого компьютера и дисплея TFT LCD, подключенных к контроллеру Arduino UNO.

Приборы и принадлежности: Контроллер Arduino UNO, дисплей 2.4“ TFT LCD, персональный компьютер, USB кабель, пакет моделирования MatLAB Simulink, среда программирования контроллера Arduino, библиотеки TFT дисплея A137.

ВВЕДЕНИЕ

Популярный контроллер Arduino UNO, имеющий дискретные устройства ввода-вывода, аналого-цифровые преобразователи и каналы последовательной передачи данных, можно использовать в качестве интерфейса компьютера для подключения внешних устройств. Контроллер может решать задачи предварительной обработки сигналов, накопления данных и локального управления как автономно, так и совместно с компьютером. Использование недорогих и выразительных средств отображение данных и параметров системы повышает качество таких систем. В этой работе затронуты вопросы, связанные с подключением, настройкой и адаптацией к задачам пользователя цветного графического сенсорного дисплея с микро SD считывателем для Arduino UNO R3.

В разделе “Общие сведения” освещаются следующие вопросы.

- 2.4” TFT LCD дисплей для Arduino UNO
- Подключение дисплея к Arduino UNO
- Библиотека примитивов
- Изменение шрифтов
- Установка и перекодировка цветов дисплея
- Определение координат сенсорной точки.
- Особенности построения сенсорных клавиш

- Работа с микро SD картой
- Управление боковой подсветкой
- Высвобождение входа ADC4 для нужд пользователя
- Мерцание выводимых на дисплей значений

ОБЩИЕ СВЕДЕНИЯ

2.4” TFT LCD дисплей для Arduino UNO

Среди множества предлагаемых на рынке дисплеев для компьютера Arduino UNO выбран дисплей с неплохим отношением цены к качеству. Среди основных его достоинств – низкая цена. С доставкой цена составила US$5.3

2.4” цветной графический сенсорный дисплей с микро флеш считывателем A 137 компании имеет следующие характеристики.

- Полное название 2.4" TFT LCD Shield Touch Panel Module TF Reader Micro SD For Arduino UNO R3 (http://www.mcufriend.com)
- Диагональ LCD TFT дисплея — 2.4"
- Разрешение — 240x320 точек
- Количество цветов — 18-бит (262 000)
- Контроллер дисплея — SPFD5408 со встроенным видео RAM буфером
- Драйвер — Sitronix ST7783 262K Color Single-Chip TFT Controller/Driver
- Цифровой интерфейс — 8 бит данных и 4 бита управления
- Порты Arduino занятые дисплеем — DIO 2 .. 9, AI 0 .. 3*
- Свободные порты Arduino — DIO 0, DIO 1, AI 4*, AI 5 и DIO 12 (если не использовать считыватель микро SD карты)
- Напряжение — 5В, работает с логикой 3.3В или 5В
- Потребление — 3.3В / 300mA, LDO регулятор
- Боковая подсветка — 4 белых светодиода (LED), которые можно подключить через транзистор для управления подсветкой
- Сенсорный экран — 4-проводной резистивный.
- Считыватель — микро SD (проверена работа с 8 ГБ картой)

Контакты используемые для подключения TFT LCD A137 к Arduino UNO:

LCD_CS (Chip Select) - A3 (Analog 3)
LCD_CD (Command/Data) - A2 (Analog 2)
LCD_WR (Write) - A1 (Analog 1)
LCD_RD (Read) - A0 (Analog 0)

LCD_RESET	- A4* (Analog 4), можно пересоединить LCD_RESET к линии RESET Arduino UNO как описано ниже.
LCD_D0	- DIO 8
LCD_D1	- DIO 9
LCD_D2	- DIO 2
LCD_D3	- DIO 3
LCD_D4	- DIO 4
LCD_D5	- DIO 5
LCD_D6	- DIO 6
LCD_D7	- DIO 7
SD SS	- DIO 10
SD DI	- DIO 11
SD DO	- DIO 12
SD SCLK	- DIO 13

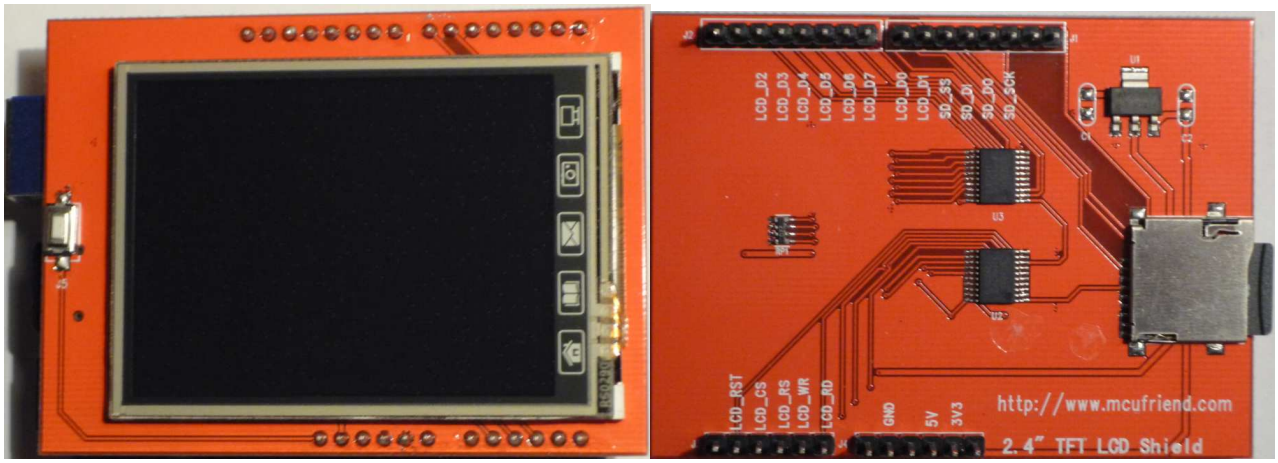


Рис. 1. Цветной графический сенсорный дисплей A137 (вид с двух сторон) с микро SD считывателем (вставлена 8 ГБ карта) для подключения “разъем в разъем” к контроллеру Arduino UNO. Размеры плат дисплея A137 и контроллера Arduino UNO совпадают.

Подключение дисплея к Arduino UNO

Перед подключением дисплея к контроллеру его библиотеки должны быть установлены на компьютере. Для ПК с ОС Windows это можно сделать следующим образом.

1. Загрузите (например, с http://arduino.cc/en/main/software) на компьютер с ОС Windows пакет программ Arduino не старше, 1.0.6 версии. Распакуйте пакет программ и поместите его в каталог, например, c:\arduino-1.0.6\
2. Подсоедините к портам обесточенного Arduino дисплейную плату совместив указанные на платах одноименные порты.

3. Подключите Arduino к USB порту компьютера. Когда операционная среда запросит местоположение драйвера укажите папку c:\arduino-1.0.6\drivers\
4. После установки драйвера, номер COM порта выделенный вашим компьютером для Arduino можно найти в списке менеджера устройств (Driver Manager).

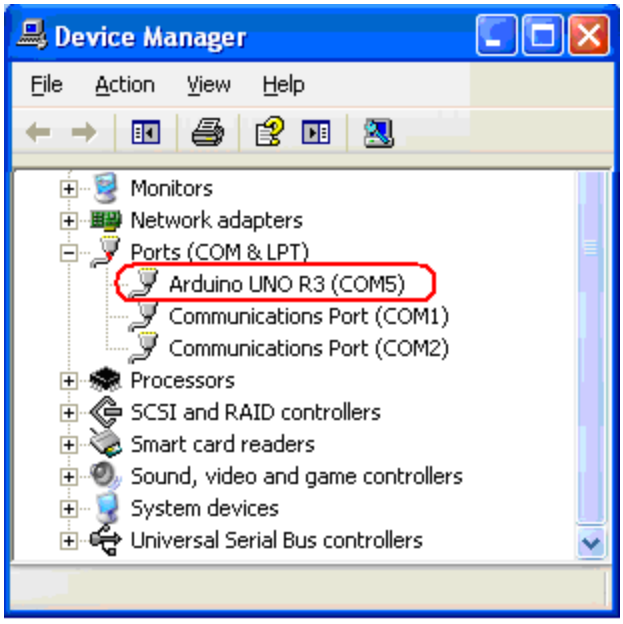


Рис. 2. Драйвер Arduino UNO R3 в списке менеджера устройств.

5. Запустите программу Arduino c:\arduino-1.0.6\arduino.exe
6. Загрузите программу управления миганием светодиода платы Arduino: Меню > File > Examples > 01.Basics> Blink

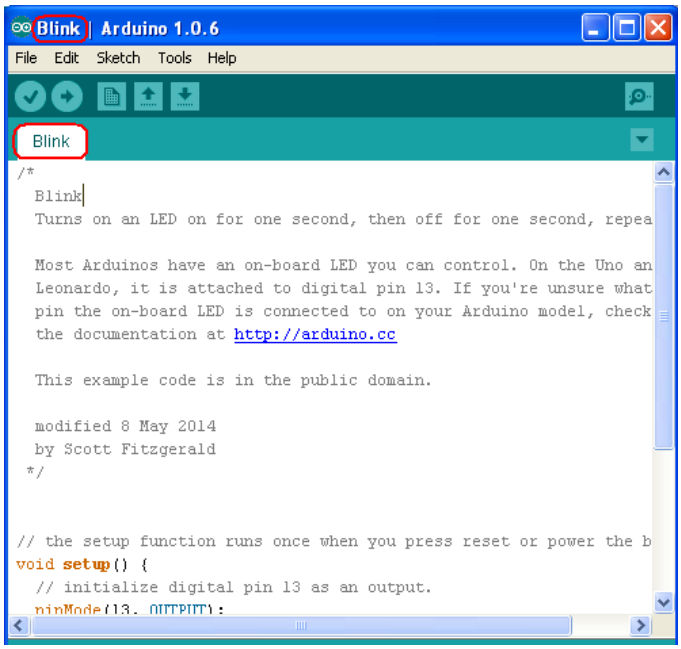


Рис. 3. Среда программирования Arduino.

7. Нажав на стрелку меню  запустите программу Blink. Если не появилось сообщений об ошибке, убедитесь, что загрузка программы прошла успешно и контроллер выполняет программу по переключению светодиода платы Arduino подключенного к порту DIO 13.

8. Подключите библиотеки для работы с устройствами дисплея платы A137. Загрузите с https://github.com/Smoke-And-Wires/TFT-Shield-Example-Code/archive/master.zip модифицированную TFT SHIELD библиотеку. Распакуйте zip и поместите его содержимое в раздел c:\arduino-1.0.6\libraries\. Убедитесь, что название распакованной папки и имена SWTFT.cpp и SWTFT.h файлов совпадают. При необходимости, исправьте имя папки.

9. Загрузите библиотеку графики с https://github.com/adafruit/Adafruit-GFX-Library/archive/master.zip. По аналогии с предыдущим пунктом, поместите папку в раздел библиотек Arduino и проверьте имя папки, сравнивая его с Adafruit_GFX.cpp и Adafruit_GFX.h файлами.

10. Аналогичным образом, подключите библиотеку сенсорного дисплея сайта https://github.com/adafruit/Touch-Screen-Library/archive/master.zip. TouchScreen - имя cpp и h файла.

11. Скопируйте эти три папки: Adafruit_GFX, SWTFT and TouchScreen

    из c:\arduino_1.0.6\libraries\ в C:\ProgramFiles\Arduino\Libraries
    в c:\ProgramFiles\Arduino\Libraries

12. Проверьте работу демонстрационных программ графического дисплея. Для этого в программе Arduino загрузите один из демонстрационных pde файлов: Меню > File > Open > c:\arduino-1.0.6\libraries\SWTFT\examples\. Данные программы помимо дисплея могут отображаться в окне COM соединения с компьютером который открывается кнопкой меню программы Arduino:  . Через это окно можно передавать данные программе которая считывает их из COM порта Arduino.

Примечание:

- graphicstest.pde – демонстрация графики дисплея с выводом времени построения графики в окно COM соединения.
- rotationtest.pde – демонстрация реакции дисплея на нажатие двух клавиш (произвольной и Enter) на клавиатуре компьютера.
- tftbmp.pde - отображает на дисплее bmp изображение с корневого каталога флеш-карты (miniwoof.bmp). Размер 24-bit bmp файла не должен превышать 260 КБ.
- tftpaint.ino - позволяет наносить цветные линии (палитра из 6 цветов) на экран сенсорного дисплея

Библиотека примитивов

Для вывода примитива на дисплей используются следующие команды.

```
// подключение библиотек
#include <Adafruit_GFX.h>  // ядро графической библиотеки
#include "SWTFT.h" // библиотека аппаратных средств

SWTFT tft;

// управление TFT дисплеем
tft.reset(); // сброс дисплея
uint16_t identifier = tft.readID();
tft.begin(identifier); // включение дисплея

// разворот направления печати дисплея
tft.setRotation(rotation); // установка угла начальных координат дисплея, rotation = 0 .. 3
tft.setRotation(tft.getRotation()+1); // разворот экрана на 90 градусов
```

Примечание: Драйвер полученного дисплея A137 имеет следующий номер

LCD driver chip: **7783**

Этот номер может использоваться для поиска и настройки других библиотек дисплея.

Таблица 1. Примитивы библиотеки Adafruit_GFX [1] tft объекта (SWIFT tft;).

Команда построения примитива	**Назначение**
ТЕКСТ	
tft.drawChar(int16_t x, int16_t y, unsigned char c, uint16_t color, uint16_t bg, uint8_t size)	Рисование символа шрифта
tft.void setCursor(uint16_t x0, uint16_t y0);	Установка курсора в зоне 240 x 320 точек
tft.void setTextColor(uint16_t color);	Установка цвета шрифта
tft.void setTextColor(uint16_t color, uint16_t backgroundcolor);	Установка цвета шрифта и фона
tft.void setTextSize(uint8_t size);	Установка размера шрифта
tft.void setTextWrap(boolean w);	Установка режима переноса символа длинной строки на новую или на начало текущей строки
tft.print(12.345); tft.print(“Hello”); tft.println(“World”); tft.print(0xF81F, HEX).	Вывод чисел и строк на дисплей
ТОЧКА (пиксел)	
tft.drawPixel(uint16_t x, uint16_t y, uint16_t color););	Вывод точки (пикселя)
ЛИНИЯ	
tft.drawLine(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t color);	Построение линии между точками (x0,y0) и (x1,y1)
tft.void drawFastVLine(uint16_t x0, uint16_t y0,	Построение вертикальной линии

uint16_t length, uint16_t color)	длиной length
tft.void drawFastHLine(uin86_t x0, uin86_t y0, uint8_t length, uint16_t color);	Построение горизонтальной линии длиной length
КРУГ	
tft.drawCircle(uint16_t x0, uint16_t y0, uint16_t r, uint16_t color);	Рисование круга радиусом r с центром в x0,y0
tft.fillCircle (uint16_t x0, uint16_t y0, uint16_t r, uint16_t color)	Заливка круга радиусом r с центром в x0,y0
ТРЕУГОЛЬНИК	
tft.drawTriangle(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, color);	Рисование треугольника с вершинами (x0,y0), (x1,y1) и (x2,y2)
tft.fillTriangle(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2, color);	заливка треугольника с вершинами (x0,y0), (x1,y1) и (x2,y2)
ПРЯМОУГОЛЬНИК	
tft.drawRect(drawRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);	Рисование прямоугольника шириной w и высотой h
tft.fillRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t color);	заливка прямоугольника шириной w и высотой h
ПРЯМОУГОЛЬНИК СО СКРУГЛЕННЫМИ УГЛАМИ	
tft.drawRoundRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t radius, uint16 color);	Рисование прямоугольника со скругленными углами
tft.fillRoundRect(uint16_t x0, uint16_t y0, uint16_t w, uint16_t h, uint16_t radius, uint16 color);	заливка прямоугольника
МОНОХРОМНОЕ РАСТРОВОЕ ИЗОБРАЖЕНИЕ	
drawBitmap(int16_t x, int16_t y, uint8_t *bitmap, int16_t w, int16_t h, uint16_t color);	Рисование небольшой монохромного (одноцветного) изображения. Bitmap располагается в памяти программ.
ВСЯ ЗОНА ДИСПЛЕЯ	
tft.fillScreen(uint16_t color);	заливка дисплея
tft.fillScreen(BLACK);	сброс (очистка) дисплея

Изменение шрифтов

Описание шрифтов выводимых на экран символов находится в glcdfont.c файлах библиотек Arduino. В файле указывается размер шрифтов, код первого символа, количество символов и приводится таблица шрифтов.

Таблица шрифтов фиксированной ширины, например, размером 5 x 7 пикселей с первого символа N 0x20 (32) по 0x7F (127) символ выглядит следующим образом

```
static const unsigned char font[] PROGMEM = {
        0x00, 0x00, 0x00, 0x00, 0x00,// (space)
        0x00, 0x00, 0x5F, 0x00, 0x00,// !
...
        0x3C, 0x4A, 0x49, 0x49, 0x30,// 6
```

```
        0x01, 0x71, 0x09, 0x05, 0x03,// 7
        0x36, 0x49, 0x49, 0x49, 0x36,// 8
...
        0x7E, 0x11, 0x11, 0x11, 0x7E,// A
        0x7F, 0x49, 0x49, 0x49, 0x36,// B
        0x3E, 0x41, 0x41, 0x41, 0x22,// C
...
        0x08, 0x08, 0x2A, 0x1C, 0x08,// ->
        0x08, 0x1C, 0x2A, 0x08, 0x08 // <-
};
```

где в каждой строке находится битовая маска символа, например, как показано на Рис. 4.

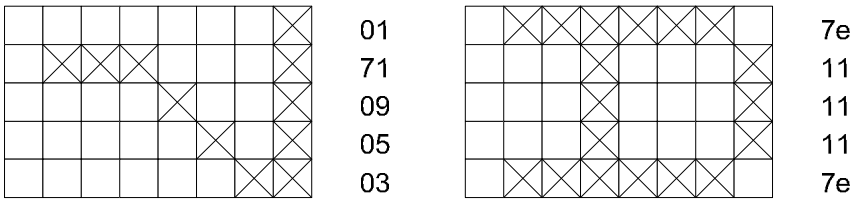


Рис. 4. Битовая маска символов 7 и А. В этой кодировка межсимвольный интервал добавляется автоматически.

Любой символ таблицы шрифтов можно изменить добавив или удалив биты отображаемых пикселов (см. Рис. 4). Это можно сделать при помощи текстового редактора или специальных программ, например, GLCDFontCreator.exe.

Установка и перекодировка цветов дисплея

Коды цветов графического дисплея A137 показаны ниже.

```
#define BLACK     0x0000
#define BLUE      0x001F
#define RED       0xF800
#define  GREEN    0x07E0
#define  CYAN     0x07FF
#define MAGENTA   0xF81F
#define YELLOW    0xFFE0
#define WHITE     0xFFFF
```

Видно, что любой цвет обозначается двумя байтами и кодировка всех цветов лежит в диапазоне 0x0000 до 0xFFFF, при чем, красному цвету выделено 5 бит (имеет 32 значения от 0 до 31), зеленому – 6 бит (64 значения) и синему – 5 бит.

Для определения кодировки дополнительных цветов можно использовать коды цветов, например, системы Microsoft Office и формулу перевода палитры 24-бит Microsoft в палитру 16-бит поддерживаемые библиотекой SWIFT дисплея A137.

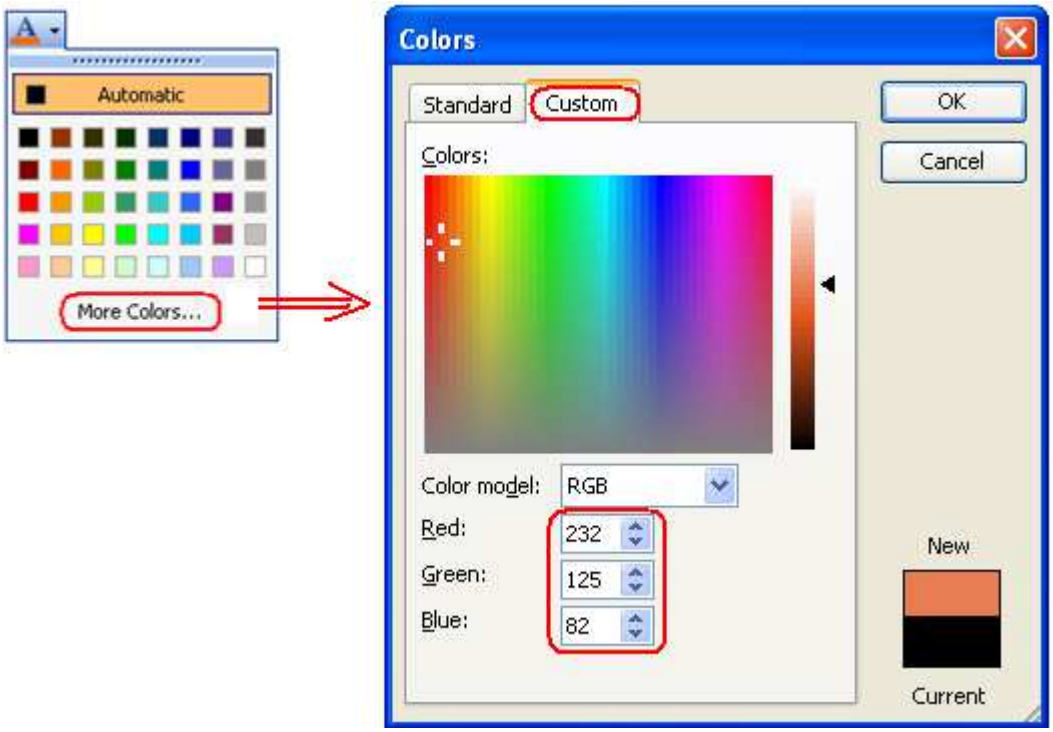


Рис. 5. Считывание RGB палитры 24 бит Windows в которой каждый цвет задается байтом (8 бит) от 0 до 255.

Преобразование кода палитры 24 бит в код палитры 16 бит можно выполнить по формуле

$$Код = floor(\frac{Red}{256} * 32) * 64 * 32 + floor(\frac{Green}{256} * 64) * 32 + floor(\frac{Blue}{256} * 32),$$

где цвета Red (красный) Green (зеленый) Blue (голубой) имеют значения от 0 до 255. Перевод десятичного кода в шестнадцатеричное или двоичное в МатЛАБ выполняется функциями dec2hex и dec2bin соответственно.

Определение координат сенсорной точки

Для обнаружения нажатия необходимо знать сопротивление между крайними координатами по X. Сопротивление можно измерить с помощью любого мультиметра. Величина сопротивления, например, 300 ом и размеры сенсорной зоны устанавливаются отношением

```
#define YP A1  // must be an analog pin
#define XM A2  // must be an analog pin
#define YM 7   // can be a digital pin
#define XP 6   // can be a digital pin
TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);
```

Для работы с сенсорным экраном необходимо подключить библиотеку

```
#include <TouchScreen.h> // библиотека сенсорного экрана
```

Обычно измеренные координаты сенсорной зоны дисплея масштабируются в координатах зоны изображения.

На Рис. 6 показан результат перевода координат нажатия не откалиброванной сенсорной зоны 1024 x 1024 значений с

```
#define TS_MINX 150
#define TS_MINY 120
#define TS_MAXX 920
#define TS_MAXY 940
```

в координаты зоны изображения 240 x 320 точек с использованием следующих команд.

```
tft.fillRect(165, 160, 130, 52, ORANGE);  // построение прямоугольника
TSPoint p = ts.getPoint();  // считывание координат нажатия в зоне 1024 x 1024
p.x = tft.width()-(map(p.x, TS_MINX, TS_MAXX, tft.width(), 0)); // преобразование
p.y = tft.height()-(map(p.y, TS_MINY, TS_MAXY, tft.height(), 0)); // преобразование

// нажатие в зоне прямоугольника ?
if ((p.x > 15) and (p.x < 90) and (p.y > 120) and (p.y < 215)){…}
```

Несовпадение преобразованных координат нажатия на углы прямоугольника по оси X с координатами построения изображения прямоугольника получилось как 15 к 160 и 90 к 212. Несовпадение можно уменьшить проведя калибровку резистивной матрицы сенсорного экрана.

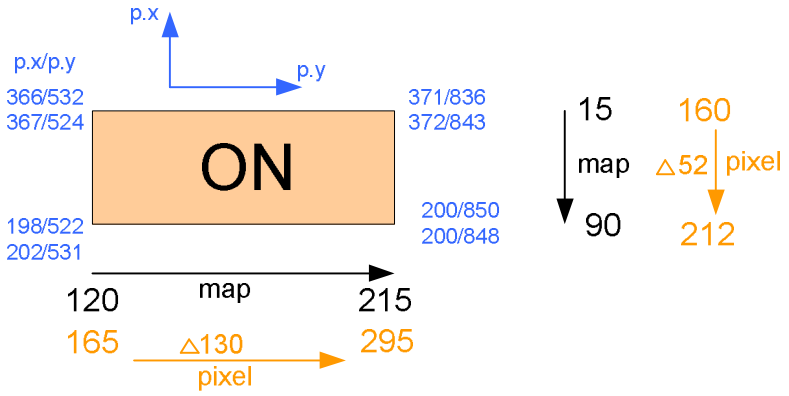


Рис. 6. Результат повторного считывания координат углов прямоугольника (числа, обозначенные голубым цветом) с использованием процедуры [p.x, p.y] = ts.getPoint() после нажатия на угловые точки. Прямоугольник в зоне 240x320 точек построен с использованием библиотечной функции tft.fillRect(165, 160, 130, 52, ORANGE). Желтые числа – усредненный результат перевода координат нажатий на углы в зоне 1024 x 1024 значений в координаты зоны изображения 240 x 320 точек полученный с использованием функции map. Режим направления печати задан командой tft.setRotation(1).

Другой вариант использования тестовых угловых координат для определения - произошло ли нажатие в прямоугольной зоне, позволяет отказаться от функции перевода (масштабирования):

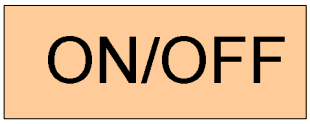
```
TSPoint p = ts.getPoint();// считывание координат нажатия в зоне 1024 x 1024
// нажатие в зоне прямоугольника ?
if ((p.x > 200) and (p.x < 365) and (p.y > 525) and (p.y < 840)){…}
```

Особенности построения сенсорных клавиш

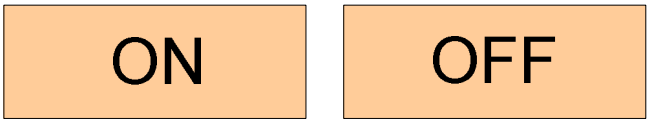
При использовании сенсорного дисплея для ввода реакции пользователя необходимо на экране показать зоны ввода (сенсорные клавиши). От варианта построения клавиш и считывая координат нажатия на экран зависит надежность срабатывания ввода. Рассмотрим два варианта организации ввода через дисплей A137.

1. Одна клавиша используется для ввода двух состояний, например, ВКЛ./ВЫКЛ. (ON/OFF)
2. Используются две клавиши: одна для ON, а другая для OFF.

В первом варианте за время нажатия происходило многократное переключение клавиши с ON на OFF и обратно, что затрудняло установку заданного положения клавиши. При введении 100 мс задержки между переключениями потребовалось значительно увеличить силу нажатия и, зачастую, требовалось многократное нажатие для изменения состояния клавиши.



Использование двух клавиш увеличило зону экрана занятую вводом, однако в этом варианте не зависимо от задержки происходило уверенное переключение даже при минимальной силе нажатия.



Работа с микро SD картой

Микро SD карата может использоваться для отображения заранее подготовленных графических изображений и хранения произвольных данных.

Таблица 2. Примеры характеристик bmp файлов для дисплея TFT LCD A137.

Bmp file	Width, pixels	Height, pixels	Horizontal resolution, dpi	Vertical resolution, dpi	Bit depth	Size, kB
woof*	240	320	71	71	24	230.5
miniwoof	120	160	71	71	24	57.7
test	240	240	71	71	24	230.5

* параметры полноэкранного изображения

Конвертирование 24-bit bitmaps в 16-bit формат (5 бит красный , 6 бит зеленый, 5 бит голубой) позволяет обеспечивать прямую загрузку в LCD полноэкранных изображений, что существенно увеличивает скорость загрузки. Код конвертора можно найти по ссылке [2].

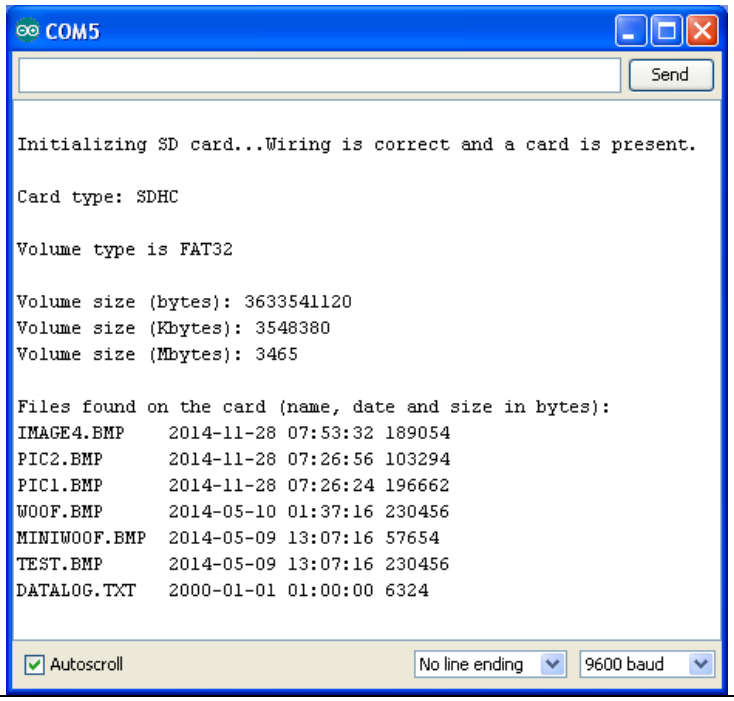
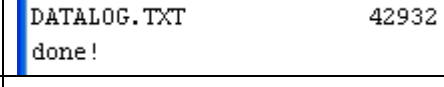
В программе работы с микро SD картой необходимо подключить библиотеку.

```
#include <SD.h> // библиотека SD карты
```

SD библиотека пакета arduino-1.0.6 содержит примеры работы с SD картой (см. Таблица 3).

Внимание! Для работы примеров с SD картой дисплея A137 необходимо исправить const int chipSelect = 4 на const int chipSelect = **10**; и if (!SD.begin(4)) на if (!SD.begin(**10**))

Таблица 3. Примеры SD библиотеки arduino-1.0.6\libraries\SD\examples\

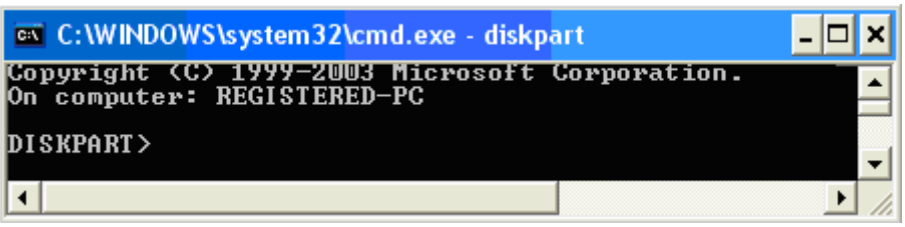
Название примера	**Назначение**
CardInfo.ino	Считывание и передача в последовательный канал параметров и содержимого SD карты, например, 
Datalogger.ini	Выполнение циклической записи показаний трех АЦП контроллера в файл SD карты "datalog.txt" с параллельной передачей показаний АЦП в последовательный канал *. 
DumpFile.ino	Считывание данных файла "datalog.txt" SD карты и передача считанных данных в последовательный канал.

Files.ino	Создание и удаление файла SD карты.
listfiles.ino	Передача списка файлов SD карты в последовательный канал.
ReadWrite.ino	Запись строки "testing 1, 2, 3." в файл "test.txt" SD карты, чтение записанных данных и передача данных в последовательный канал. Initializing SD card...initialization done. Writing to test.txt...done. test.txt: testing 1, 2, 3. testing 1, 2, 3. Autoscroll No line ending 9600 baud TEST.TXT 2000-01-01 01:00:00 36

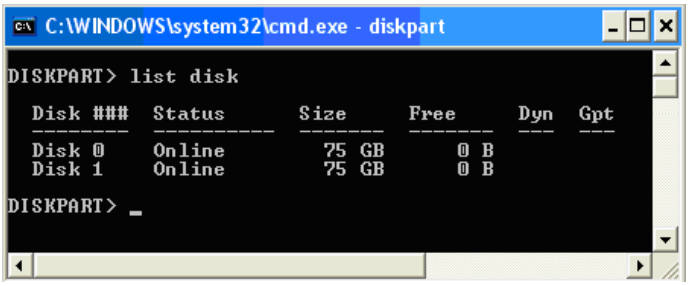
* Файл размером 76 КБ был записан примерно за 70 секунд со скоростью ~1100 байт/с. Параллельно с записью выполнялась передача данных в канал последовательной передачи на скорости 9600 бит/с. Исключение циклической передачи данных в последовательный канал параллельно с записью данных в файл увеличило скорость записи до 1375 байт/с (110 КБ за 80 с).

В некоторых случаях программа не может записать данные на SD карту, поскольку карта защищена от записи. Снять защиту от записи SD карты можно в следующей последовательности.

1. На компьютере с устройством к которому подключена карта откройте интерпретатор командной строки cmd (start > Run > Open > cmd)
2. Введите команду diskpart



3. Распечатайте список дисководов командой list disk



4. Введите номер считывателя SD карты командой list disk select disk 1. Необходимо в команде на месте 1 указать номер вашего устройства, распечатанный в списке (см. п.3).
5. Снимите защиту от записи: attributes disk clear readonly
6. Завершите работу в cmd. Введите exit , затем еще раз exit

Управление боковой подсветкой

После доработки разводки электрической платы дисплея яркость подсветки (0 .. 255 уровней) можно регулировать ШИМ сигналом порта DIO Arduino UNO. Управляющий ШИМ должен подключаться к светодиодам подсветки через транзисторный усилитель тока, поскольку выходной ток порта DIO не должен превышать 40 мА, тогда как, для регулировки подсветки может потребовать ток 100 мА.

В Интернет приводятся варианты и программного релейного включения/выключения TFT подсветки дисплеев после доработки библиотек. Варианта для дисплея A137 найти не удалось.

Вот пример [3] программы трехуровневого (100%, 50%, 0%, 100%, ...) ежесекундного изменения подсветки 3.3 В при питании от источника 5В.

```
#define PIN_BACKLIGHT 8
void SetBacklightPercent(int percent)
{
	if (percent > 100)
		percent = 100;
	else if (percent < 0)
		percent = 0;
	//(3.3v/5v) * 255 = 168.3
	analogWrite(PIN_BACKLIGHT, (int) round(168.3 * (percent/100.0)));
}

void setup()
{	pinMode(PIN_BACKLIGHT, OUTPUT);
}

void loop()
{
	SetBacklightPercent(100);
	delay(1000);

	SetBacklightPercent(50);
	delay(1000);
```

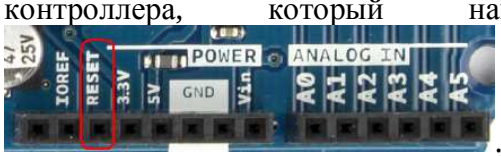
```
        SetBacklightPercent(0);
        delay(1000);
}
```

Высвобождение входа ADC4 для нужд пользователя

В оригинальной версии порт LCD_RESET дисплея A 137 подключается к порту A4 контроллера Arduino UNO. Через порт A4 контроллер производит “сброс” дисплея – установку его в начальное состояние. Доработка соединения дисплей – контроллер позволяет использовать порт A4 для ввода аналоговых сигналов в контроллер. Доработка включает следующие шаги.

В файле SWIFT.cpp библиотеки SWIFT необходимо исключить команды установки работы аналогового порта A4 в режим дискретного выхода: pinMode(LCD_RESET, OUTPUT); и команды записи в дискретный порт A4: digitalWrite(LCD_RESET, HIGH); и digitalWrite(LCD_RESET, LOW);

Описанное выше исключение команд-установок порта A4 блокирует переход порта в режим цифрового выхода которое выполнялось в момент формировании tft объекта командой: SWTFT tft. Теперь порт A4 постоянно работает в режиме АЦП – приёма аналоговых сигналов, который устанавливается “по умолчанию” при включении или перезагрузке Arduino. Однако, если на вход АЦП подать сигнал низкого уровня, то произойдет сброс дисплея. Чтобы избежать неконтролируемого сброса, необходимо физически отключить линию LCD_RST дисплея 

от порта A4 контроллера Arduino UNO (показан на рисунке ниже) и подключить его к контакту RESET контроллера, который находится рядом с контактом +3.3V: 

.

Мерцание выводимых на дисплей значений

При частом выводе переменной на дисплей возникает эффект мерцания, что делает дискомфортным чтение экрана. Для устранения эффекта мерцания необходимо удалять и выводить только обновляемые символы (цифры числа). Если цифра числа, которое необходимо вывести на экран, не отличается от цифры той же позиции выведенного числа, то вывод такой цифры необходимо заблокировать.

Ниже дан пример подавления мерцания при выводе на дисплей значений 10-разрядного АЦП A5 каждые 100 мс.

```
#include <Adafruit_GFX.h>    // Графическая библиотека
#include <SWTFT.h>           // Библиотека аппаратных средств дисплея
```

```
// обозначение цветов
#define BLACK   0x0000
#define GREEN   0x07E0

SWTFT tft;

const int adc_5 = A5;  // номер используемого АЦП
int adc_5_sample, adc_5_sample_prev = 9999;
unsigned long set_time = 0;
int countdigits[] = {0, 0, 0, 0}; // массив цифр последнего числа
int prevdigits[] = {0, 0, 0, 0}; // массив цифр предыдущего числа
int digitpos[] = {175, 205, 235, 265}; // X координаты цифр
int x = 0;

void setup() {
  uint16_t identifier = tft.readID();
  tft.begin(identifier);

  tft.setRotation(1);  // направление вывода строк - по длине экрана
  tft.fillScreen(BLACK); // сброс экрана – заливка черным
  tft.setTextSize(5);
}

void loop() {
  unsigned long time = millis();  // чтение таймера
  // запуск цикла каждые 100 мс
  if (time > set_time) {
    set_time = set_time + 100;

  // чтение АЦП:
  adc_5_sample = analogRead(adc_5);

// обновление показаний если новое и предыдущие значения не совпадают
    if (adc_5_sample != adc_5_sample_prev) {
```

```
// разложить считанное число на цифры
if (adc_5_sample > 999) countdigits[0] = (adc_5_sample / 1000) % 10;
if (adc_5_sample > 99) countdigits[1] = (adc_5_sample / 100) % 10;
if (adc_5_sample > 9) countdigits[2] = (adc_5_sample / 10) % 10;
countdigits[3] = adc_5_sample % 10;

// разложить предыдущее число на цифры
if (adc_5_sample_prev > 999) prevdigits[0] = (adc_5_sample_prev / 1000) % 10;
if (adc_5_sample_prev > 99) prevdigits[1] = (adc_5_sample_prev / 100) % 10;
if (adc_5_sample_prev > 9) prevdigits[2] = (adc_5_sample_prev / 10) % 10;
prevdigits[3] = adc_5_sample_prev % 10;

// поцифровое сравнение чисел
for(x=0; x < 4; x++){

  // обновить цифру если она отличается
  if(countdigits[x] != prevdigits[x]){
    // стереть старую цифру
    tft.setCursor(digitpos[x], 114);
    tft.setTextColor(BLACK);
    tft.print(prevdigits[x]);

    // напечатать новую цифру если это не ноль в старшем разряде
    tft.setTextColor(GREEN);
    if((x == 0) and (adc_5_sample > 999)) {
      tft.setCursor(digitpos[x], 114);
      tft.print(countdigits[x]);
    }
    if((x == 1) and (adc_5_sample > 99)) {
      tft.setCursor(digitpos[x], 114);
      tft.print(countdigits[x]);
    }
    if((x == 2) and (adc_5_sample > 9)) {
      tft.setCursor(digitpos[x], 114);
      tft.print(countdigits[x]);
    }
```

```
        if(x == 3) {
          tft.setCursor(digitpos[x], 114);
          tft.print(countdigits[x]);
        }
      }
    }
  }
  // сохранить считанное значение
  adc_5_sample_prev = adc_5_sample;
 }
}
```

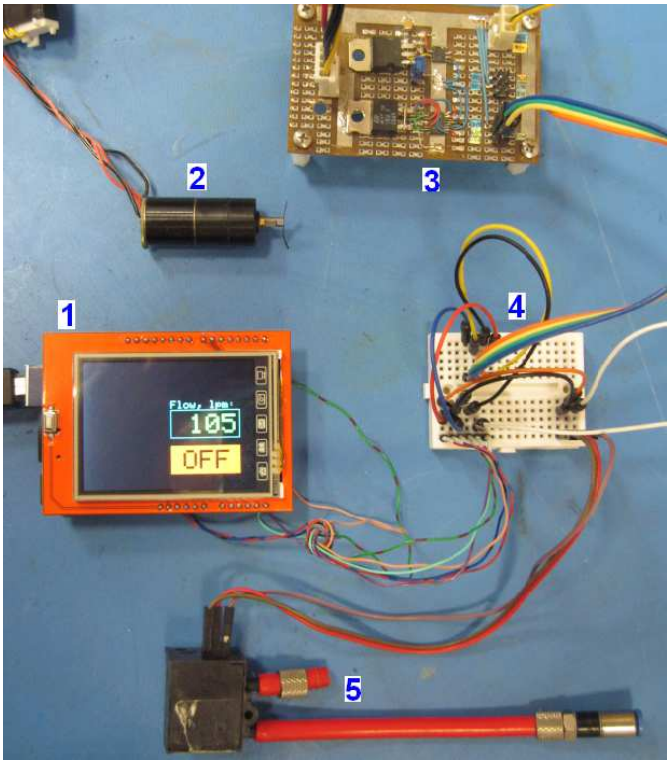


Рис. 7. Пример компонентов макета локальной системы управления газовым потоком: 1 – контроллер Arduino UNO, накрытый цветным графическим сенсорным дисплеем; 2 - шаговый двигатель AM0820; 3 - драйвер BA6845FS (усилители тока шагового двигателя по мостовой схеме [4]); 4 – коммутационная плата; 5 – датчик потока газа D6F-P0010A1.

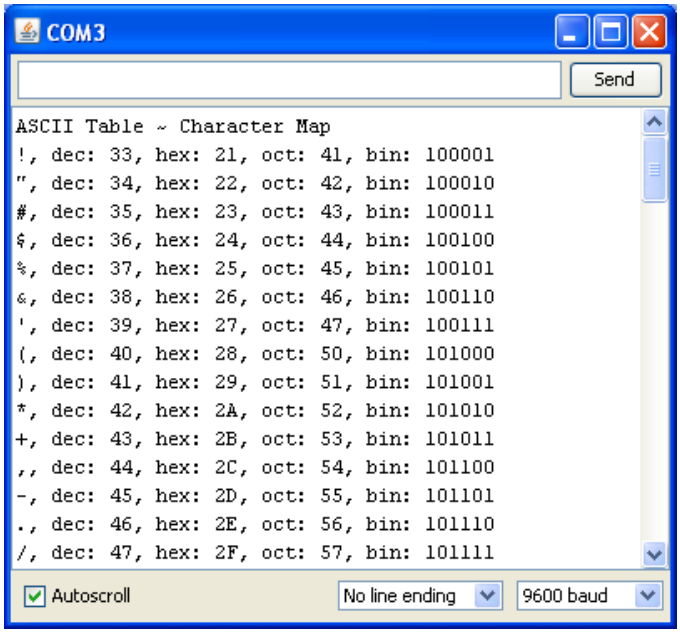
ПРИМЕРЫ ПОЛУЧЕНИЯ ПРОВЕРЕННЫХ РЕЗУЛЬТАТОВ И ВАРИАНТЫ ДЛЯ САМОКОНТРОЛЯ

Задание 1. Отображение параметров контроллера и ввод данных через монитор последовательного канала.

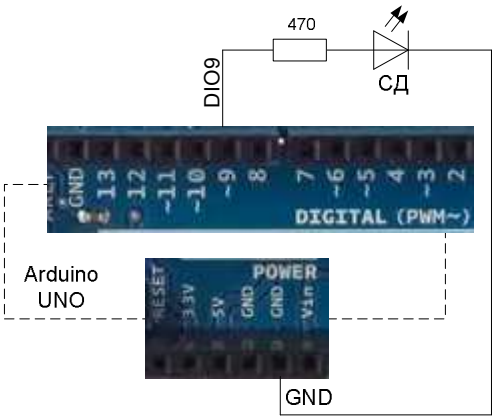
1. Запустите программу arduino.exe
2. Загрузите (Меню > File > Examples > 04.Communication > ASCIITable) пример передачи данных Arduino в канал COM - последовательной передачи данных.

   c:\arduino-1.0.6\examples\04.Communication\ASCIITable\ASCIITable.ino

3. Запустите пример  и откройте монитор: Serial Monitor 



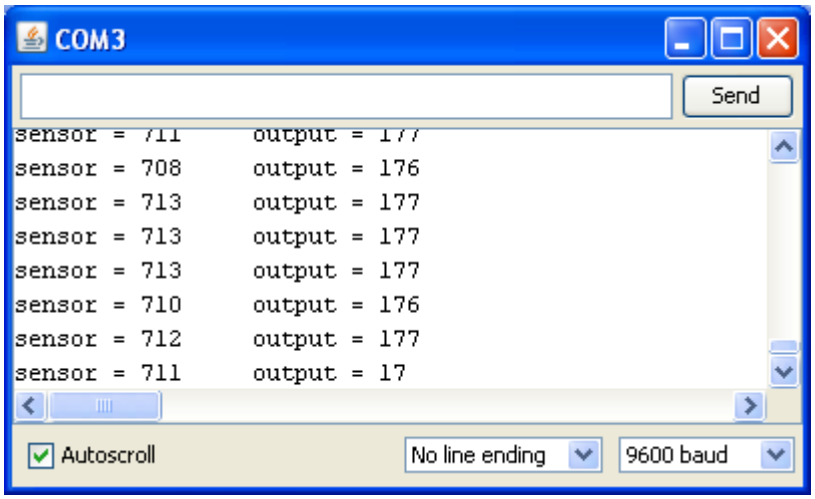
4. Рассмотрите организацию отображения данных контроллера, используя коды программы и выводимые на монитор данные.
5. К порту DIO 9 выключенного Arduino UNO подключите светодиод.



6. Загрузите (Меню > File > Examples > 03.Analog > AnalogInOutSerial) пример считывания показаний 10-бит АЦП (A0) Arduino, масштабирования данных до 8 бит функцией map программы и передачи полученных байт в канал COM.

   c:\arduino-1.0.6\examples\03.Analog\AnalogInOutSerial\AnalogInOutSerial.ino

7. Запустите пример и откройте Serial Monitor 



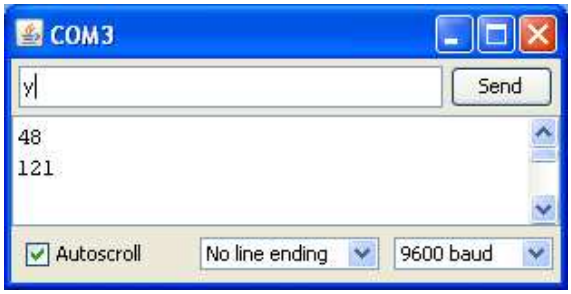
8. Рассмотрите организацию отображения данных АЦП, используя коды программы, выводимые на монитор данные и свечение светодиода. Установку светодиода в крайние состояния Вкл./Выкл. можно можно получить подключением заданного напряжения ко входу АЦП A0 разъема ANALOG IN контроллера с контактов 5V (5В) / GND (0В) разъема POWER.

9. Загрузите (Меню > File > Examples > 04.Communication > Dimmer) пример передачи данных контроллеру Arduino через Serial Monitor и COM канал и отображение передаваемого кода свечением светодиода подключенного к DIO 9.

   c:\arduino-1.0.6\examples\04.Communication\ Dimmer\Dimmer.ino

10. Используя первые два примера доработайте пример Dimmer.ino так, чтобы монитор отображал данные принятые контроллером.

Serial.println(brightness); // добавлена передача монитору



Задание 2. Отображение параметров контроллера в среде Simulink компьютера в реальном времени по тактам контроллера Arduino без использования специальных средств MatLAB для обеспечения работы систем в реальном времени rtwintgt (Real-Time Windows Target).

Примечание. Преимущество рассматриваемого соединения в котором реальное время задается контроллером Arduino в сравнении с вариантом использования rtwintgt состоит в том, что

- для запуска модели в режиме реального времени не надо тратить время на компиляцию с rtwintgt.
- можно использовать большее количество блоков библиотеки Simulink
- случается, что модель Simulink не может успешно пройти компиляцию с rtwintgt.

1. Запустите МатЛАБ.
2. В среде Simulink постройте следующую модель

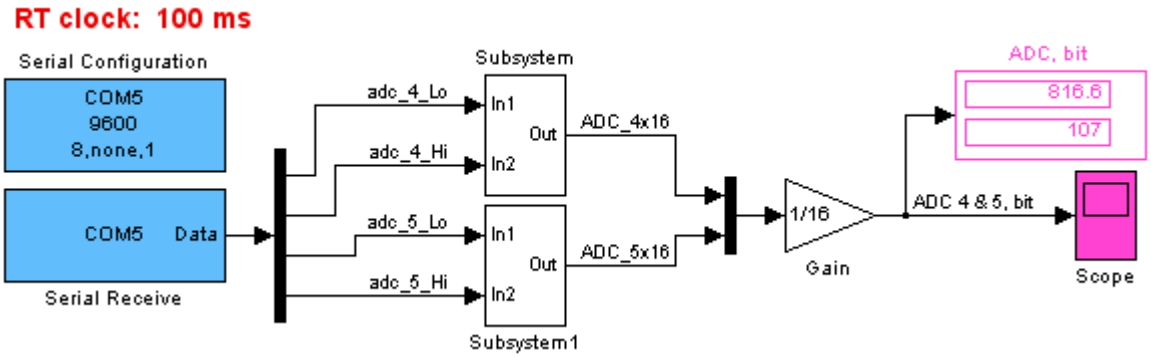


Рис. 8. Модель считывания и отображения данных последовательного канала. В модели вычисляется дробная часть усредненных показаний АЦП контроллера с разрешением 1/16 бит.

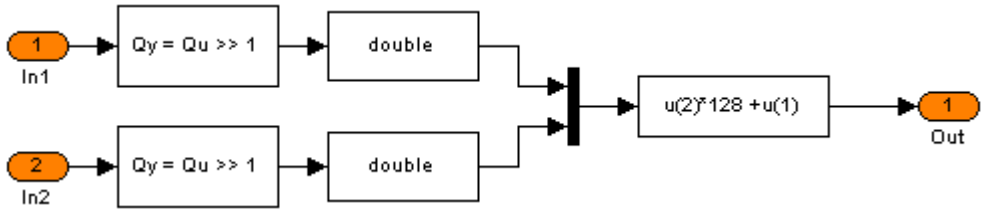


Рис. 9. Структура блоков Subsystem(s) модели. Восстановление двухбайтового числа.

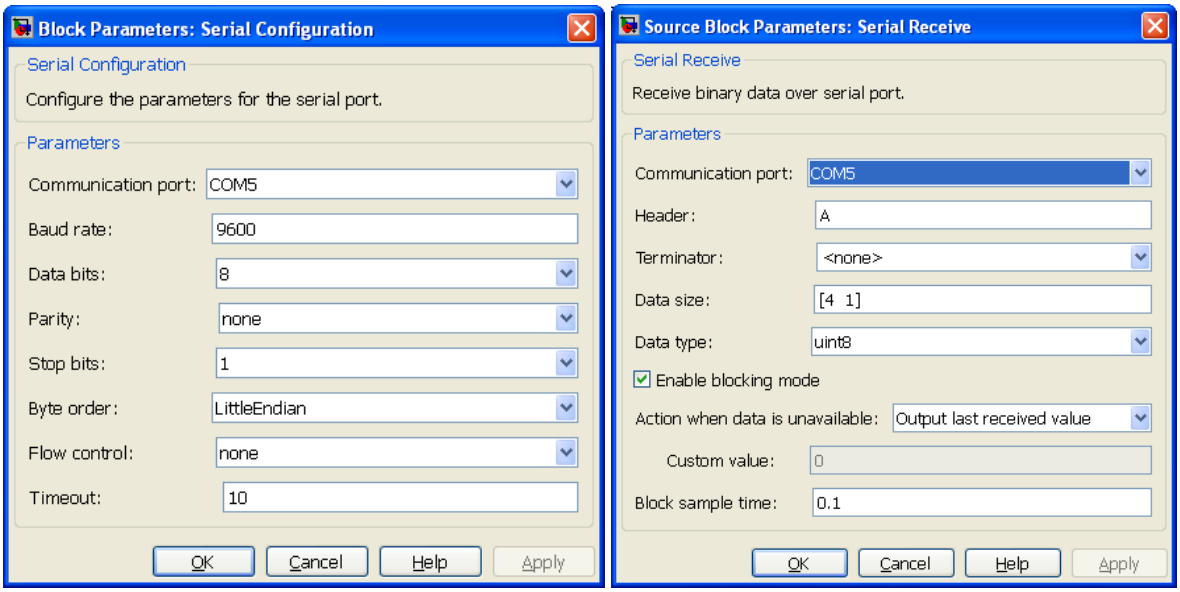


Рис. 10. Параметры настройки последовательного канала COM модели для приема данных контроллера Arduino на частоте 9600 бод пакетом из пяти байт: 4 байта данных + заголовок - символ A.

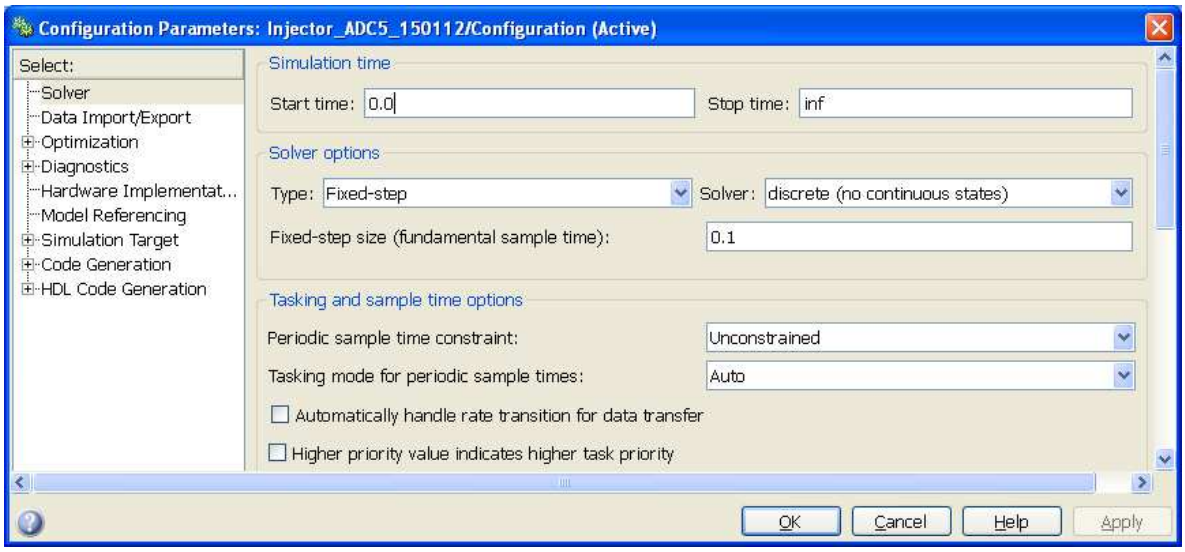


Рис. 11. Параметры моделирования с шагом 0.1 сек. Шаг моделирования должен совпадать с периодом передачи данных в последовательный канал контроллером Arduino.

3. Создайте программу считывания показаний двух АЦП контроллера Arduino, суммирования показаний каждого АЦП и передачу усредненных показаний в последовательной канал. Для повышения точности усредненные показания перед передачей должны быть увеличены в 16 раз. Байты данных должны передаваться пакетом с уникальным заголовком - символом “A”.

```
const int adc_4 = A4, adc_5 = A5;
int adc_4_sample, adc_5_sample;
unsigned long adc_4_sum = 0, adc_5_sum = 0;
unsigned long set_time = 0;
int loop_num = 0;

void setup() {
  Serial.begin(9600);
}
void loop() {
  // read ADC with maximum rate:
  adc_4_sample = analogRead(adc_4);
  adc_4_sum = adc_4_sum + adc_4_sample;
  loop_num = loop_num + 1; // ~ 120 loops per 100 ms

  adc_5_sample = analogRead(adc_5);
  adc_5_sum = adc_5_sum + adc_5_sample;

  unsigned long time = millis();  // read time in ms
  // run calculation each 100 ms
  if (time > set_time) {
    set_time = set_time + 100;  // dt = 100 ms

    adc_4_sample = (adc_4_sum * 16) / loop_num;
    adc_5_sample = (adc_5_sum * 16) / loop_num;

    // bytes for Simulink
    byte adc_4_Hi = ((adc_4_sample >> 6) & 0xFE); // high byte
    byte adc_4_Lo = ((adc_4_sample << 1) & 0xFE); // low byte
    byte adc_5_Hi = ((adc_5_sample >> 6) & 0xFE);
    byte adc_5_Lo = ((adc_5_sample << 1) & 0xFE);

    //Send data into COM port
    Serial.print("A"); // it is header
```

```
    Serial.write(adc_4_Lo);
    Serial.write(adc_4_Hi); // output byte: uint8
    Serial.write(adc_5_Lo);
    Serial.write(adc_5_Hi); // output byte: uint8
  // reset averages
    adc_4_sum = 0;
    adc_5_sum = 0;
    loop_num = 0;
   }
  }
```

4. Запустите программу Arduino, затем модель Simulink.
5. Откройте графопостроитель (блок Scope) модели. Наблюдайте сигналы АЦП контроллера Arduino в среде Simulink в реальном времени (период 100 мс).

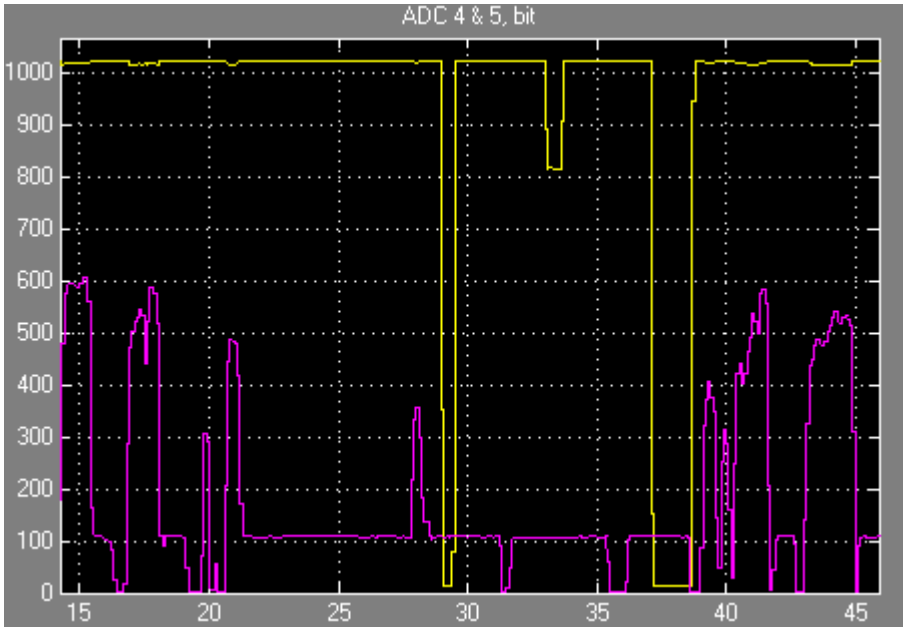


Рис. 12. Пример отображения сигналов 10 разрядных АЦП контроллера Arduino с разрешением 1/16 разряда.

Примечание. Через блок Serial Send (рис. 13) можно передавать данные Simulink контроллеру Arduino. Для приема данных, в программу контроллера следует включить следующие команды [5].

```
 if (Serial.available() > 0) {
   ctrl_num = Serial.read(); // 0 .. 255 from simulink
…
}
```

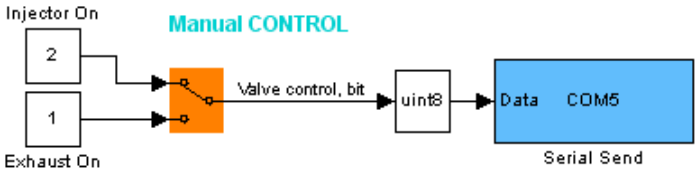


Рис. 13. Пример канал передачи данных Simulink контроллеру Arduino через COM порт.

Задание 3. Построение интерфейса пользователя на базе цветного графического сенсорного дисплея A137.

1. Создайте программу отображения показаний двух АЦП (A4 и A5) на доработанном дисплее A137 (с сигналом LCD_RESET, подключенным к порту RESET контроллера). Необходимо блокировать мерцание выводимых показаний (см. соответствующий раздел выше). Интерфейс должен включать две сенсорные кнопки. Одна – для очистки выводимых на дисплей показаний АЦП и остановки вывода, другая - для продолжения вывода показаний.

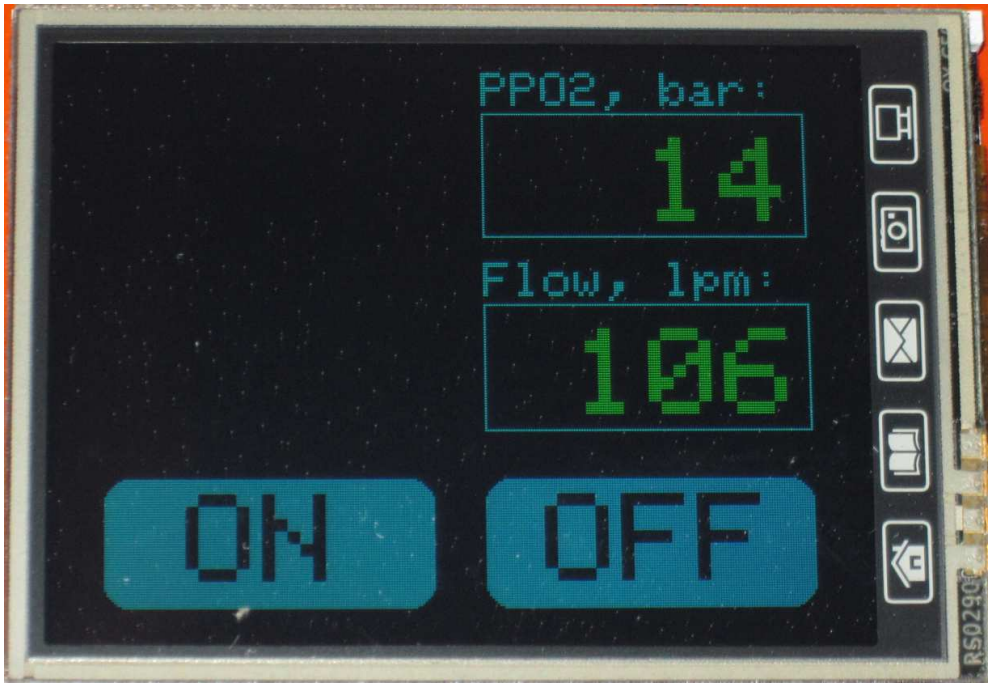


Рис. 14. Пример интерфейса пользователя контроллера Arduino UNO R3.

ПРИМЕР ПРОГРАММЫ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ:

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <SWTFT.h>           // Hardware-specific library
```

```
#include <TouchScreen.h>

#define YP A1  // must be an analog pin
#define XM A2  // must be an analog pin
#define YM 7   // can be a digital pin
#define XP 6   // can be a digital pin

#define TS_MINX 150
#define TS_MINY 120
#define TS_MAXX 920
#define TS_MAXY 940

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

// Assign human-readable names to some common 16-bit color values:
#define BLACK   0x0000
#define GREEN   0x07E0
#define CYAN    0x07FF
#define SKY     0x067F

SWTFT tft;

const int adc_4 = A4, adc_5 = A5;
word adc_4_sample, adc_5_sample; //as unsigned int 0 .. 65535
word adc_4_sample_prev, adc_5_sample_prev;

unsigned long set_time = 0;
word loop_num = 0;
boolean button_state = 1;
boolean first_read = 1;

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
  Serial.println(F("TFT User Interface Test"));

//  tft.reset();
  uint16_t identifier = tft.readID();
  tft.begin(identifier);
```

```
  tft.setRotation(1);  // screen rotation 0 .. 3
  tft.fillScreen(BLACK);

  //Draw static screen elements
  // Data output windows
  tft.drawRect(170, 25, 130, 52, CYAN);
  tft.drawRect(170, 104, 130, 52, CYAN);
  tft.setTextColor(CYAN);
  tft.setTextSize(2);
  tft.setCursor(170, 8);
  tft.print("PPO2, bar:");
  tft.setCursor(170, 87);
  tft.print("Flow, lpm:");

  // Buttons ON and OFF
  tft.setTextSize(5);
  tft.setTextColor(BLACK);
  tft.fillRoundRect(20, 175, 130, 52, 10, SKY);
  tft.setCursor(52, 181);
  tft.print("ON");
  tft.fillRoundRect(170, 175, 130, 52, 10, SKY);
  tft.setCursor(193, 181);
  tft.print("OFF");
}

#define MINPRESSURE 10
#define MAXPRESSURE 1000

void loop() {
  // read ADC:
  adc_4_sample = analogRead(adc_4);
  adc_5_sample = analogRead(adc_5);

  // read touch screen
  TSPoint p = ts.getPoint();
  pinMode(XM, OUTPUT);
  pinMode(YP, OUTPUT);
```

```
// pressure of 0 means no pressing!
if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
  // button On: run output
  if ((p.x > 160) and (p.x < 320) and (p.y > 170) and (p.y < 475)){
    button_state = 1;
    first_read = 1;
  }
  // button Off: clear output windows, stop output
  if ((p.x > 160) and (p.x < 320) and (p.y > 525) and (p.y < 840)){
    button_state = 0;
    tft.fillRect(170, 25, 130, 52, BLACK); // clear
    tft.drawRect(170, 25, 130, 52, CYAN);  // restore botton frame
    tft.fillRect(170, 104, 130, 52, BLACK);
    tft.drawRect(170, 104, 130, 52, CYAN);
  }
}
unsigned long time = millis();
if (time > set_time) {
  set_time = set_time + 100;

  if (button_state == 1) {
    // just after run,  previous sample should be
    // different from the latest sample
    if (first_read == 1) {
      adc_4_sample_prev = 9999 - adc_4_sample;
      adc_5_sample_prev = 9999 - adc_5_sample;
      first_read = 0;
    }
    if (adc_4_sample != adc_4_sample_prev) {
      print_tft(adc_4_sample, adc_4_sample_prev, 35);
    }
    adc_4_sample_prev = adc_4_sample;

    if (adc_5_sample != adc_5_sample_prev) {
      print_tft(adc_5_sample, adc_5_sample_prev, 114);
    }
    adc_5_sample_prev = adc_5_sample;
  }
}
```

```
}

void print_tft(word sample_latest, word sample_prev, word y) {
  byte countdigits[] = {0, 0, 0, 0};
  byte prevdigits[] = {0, 0, 0, 0};
  word digitpos[] = {175, 205, 235, 265};
  byte x;
  word p;

  for(x=0; x < 4; x++){
    p = round(pow(10.0,3-x)); // need round of .9999... float
    if (sample_latest > p-1) {countdigits[x] = (sample_latest / p) % 10;}
    if (sample_prev > p-1) {prevdigits[x] = (sample_prev / p) % 10;}
  }

  // Compare each digit to the value from the previous loop.
  // The digit will only be redrawn if it has changed.
  for(x=0; x < 4; x++){
    if(countdigits[x] != prevdigits[x]){
      // clear old digit
      tft.setCursor(digitpos[x], y);
      tft.setTextColor(BLACK);
      tft.print(prevdigits[x]);

      // print new digit in green
      p = round(pow(10,3-x));
      if(sample_latest > p-1) {
        tft.setCursor(digitpos[x], y);
        tft.setTextColor(GREEN);
        tft.print(countdigits[x]);
      }
      if(sample_latest == 0) {
        tft.setCursor(digitpos[3], y);
        tft.setTextColor(GREEN);
        tft.print(countdigits[x]);
      }
    }
  }
}
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какими средствами можно обеспечить ввод воздействий пользователя в локальную систему на базе контроллера Arduino?
2. Какими средствами можно обеспечить отображение и накопление параметров и переменных состояния локальной системы на базе контроллера Arduino?
3. Для решения каких задач целесообразно использовать недорогой цветной графический сенсорный дисплей с микро SD считывателем?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Adafruit GFX Graphics Library, created by Phillip Burgess
2. Justin. Bitmap Converter for Arduino LCD. http://misc.ws/2013/11/03/bitmap-converter-for-arduino-lcd/
3. How to control the display's backlight in the sketch? http://forum.arduino.cc/index.php?topic=125856.0.
4. Dr. Bob Davidov. Стенд для разработки алгоритмов высокоскоростного управления шаговым приводом. http://portalnp.ru/wp-content/uploads/2013/12/07.06_Stand-for-design-of-high-speed-stepper-motor-control-algorithms_Ed_1b1.pdf
5. Dr. Bob Davidov. Многоканальное устройство ввода и накопления аналоговых данных на базе MS Excel. http://portalnp.ru/2014/03/1762
6. Dr. Bob Davidov. Компьютерные технологии управления в технических системах http://portalnp.ru/author/bobdavidov.