Arduino

Осциллограф

Проекты

Robert J Davis II

Arduino Oscilloscope Projects

Robert J Davis II

Arduino Oscilloscope Projects

По сути, это продолжение моей предыдущей книги «Arduino LCD Projects», где я представил ряд LCD (liquid crystal display, жидкокристаллический дислей) и быстрый аналог АЦП CA3306 для создания достаточно быстрого осциллографа со скоростью преобразования 5 MSPS.

Статья в журнале про осциллограф на базе Arduino и CA3306 вызвала столь большой интерес, что я решил написать книгу, показывающую некоторые наиболее популярные АЦП и некоторые наиболее популярные LCD, соединённые в типичном приложении к осциллографу и логическому анализатору. Некоторые главы этой книги повторяют ряд разработок, которые можно найти в книге «Arduino LCD Projects».

Годы назад, в юношеском возрасте, я потратил много времени на создание осциллографа. Однако я не получил работающего осциллографа, пока не создал тот, что был частью моего обучения (CIE). Тогда в моей книге «Digital and Computer Projects» я предложил концепцию осциллографа на основе параллельного порта компьютера, и даже разработал плату для этого. Однако параллельный порт ныне канул в лету, а на смену ему пришёл USB порт. А вместе с ним пришёл Arduino в качестве простого USB осциллографа.

Есть много быстрых аналого-цифровых преобразователей, которые я хочу описать в этой книге. Есть ряд быстрых АЦП, которые работают до 50 MSPS (миллионов выборок в секунду), и есть даже более быстрые АЦП. Но, поскольку модуль Arduino Uno не столь быстр, эти АЦП следует использовать вместе с FIFO (Fist In First Out, первым вошёл, первым вышел) – высокоскоростным запоминающим буфером.

Использование быстрого АЦП вместе с LCD при создании осциллографа – это решение, вызывающее много вопросов при разработке. Я надеюсь ответить на многие из них в этой книге.

И ещё, безопасность конструкции и работы с этими устройствами на совести исключительно читателей. Читатель должен принять все меры безопасности и нести ответственность за эти устройства. Нет подразумеваемых гарантий ни в части разработанной схемы, ни в части программ, представленных в этой книге.

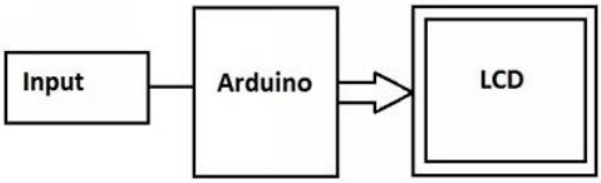
Наиболее важным я считаю получить от работы удовольствие! Попробуйте разные схемы и выберите то, что вам понравится. Внесите собственные улучшения и в схему, и в программу. Я уверен, вы можете придумать что-то, что будет лучше!

Оглавление

Глава 1. Обзор осциллографов с Arduino

В основном есть три разновидности осциллографов на основе Arduino Uno. В первом случае вы можете использовать встроенный АЦП, но изменив настройки так, чтобы сделать его быстрее. Во втором случае вы может добавить быстрый 6 или 8 разрядный внешний АЦП в зависимости от того, хотите ли вы использовать порт C или порт D. Третий тип конструкции использует FIFO (First in First Out, первым вошёл, первым вышел) буфер для выборок на скорости большей, чем мог бы обеспечить Arduino.

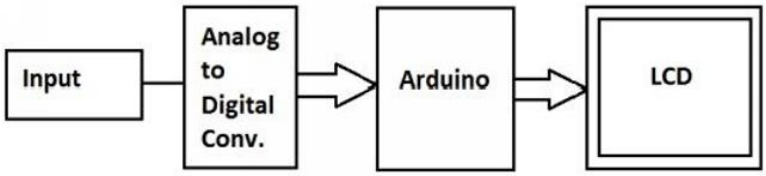
Вот блок-диаграмма первой разновидности осциллографов. Для этого типа осциллографа вы обычно привязываете обрабатываемый сигнал к A0. С программной модификацией тактирования оцифровки встроенным АЦП, вы можете получить 100000 выборок в секунду, то есть, 0.1 MSPS.



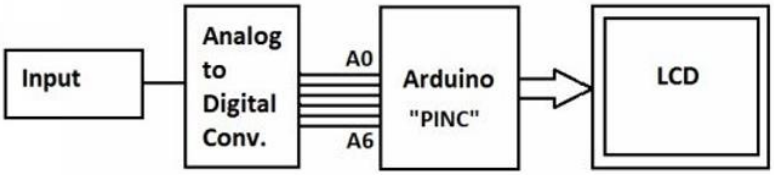
Вот код для изменения установок тактирования, чтобы ускорить АЦП почти в 16 раз. Изменение уменьшит точность конвертора. Код очистит три наиболее значимых управляющих бита. Чтобы это произошло, добавьте код сразу после «void loop() { ».

```
// Очищает бит 2 предделителя ADC, ускоряя оцифровку со 125KHz до 2 MHz

ADCSRA &= ~(1<<ADPS2);
```

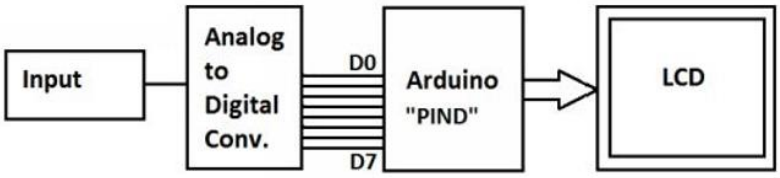
Для следующего типа осциллографа мы добавим АЦП и используем команду параллельного ввода для ускорения до 5 миллионов выборок в секунду. PINC (Parallel Input Port C) команда использует выводы от A0 до A5. Команда PIND (Parallel Input Port D) использует выводы от D0 до D7.



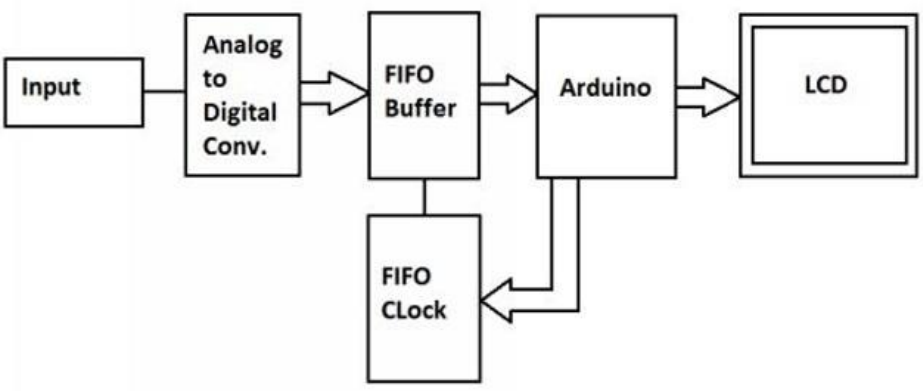
Есть два варианта подключения внешнего АЦП. Первый использует A0-A5, что ограничивает точность до шести битов. Но это идеальное подключение для CA3306, поскольку это шестиразрядный преобразователь. Вы можете использовать восьмибитовый АЦП, но подключать только шесть верхних выводов. Скажем, D7 становится D5, D6 становится D4 и т.д. Для быстрого доступа к содержимому шести аналоговых выводов используется команда «PINC».



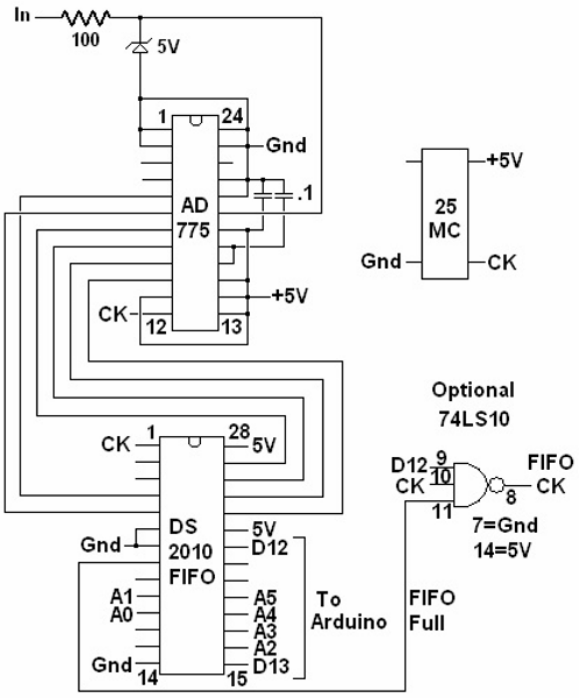
Во втором варианте подключения внешнего АЦП используются выводы D0-D7. Они быстро считывают данные при использовании команды «PIND». Однако D0 и D1 используются также для подключения USB кабеля связи с Arduino. Они либо должны отключаться при обновлении программы в Arduino, либо должны быть как-то ещё дезавуированы.



Для третьего типа осциллографа добавляется буфер памяти, называемый «FIFO». Что означает – первый вошёл, первый вышел. FIFO может оперировать со скоростью до 100 MSPS или больше. Этот вариант также требует делителя тактовой частоты и выбора цепи для выборочного торможения, иначе всегда будет осуществляться выборка на максимальной скорости от 20 до 100MSPS. Чтобы быть честным, скажу, что установка FIFO может оказаться чуть более сложной на плате. Я добавлял FIFO, используя витые провода и печатную плату. Установка FIFO была включена в мою раннюю книгу «Digital and Computer Projects».



Вот схема типового подключения FIFO в шестибитовом исполнении.

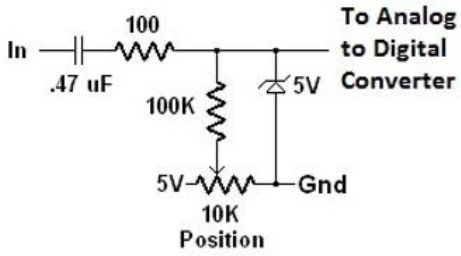


Эта книга ориентирована в первую очередь на первых двух типах разработки осциллографа, поскольку они легче в исполнении. Разработка будет использовать встроенный АЦП и внешний шести или восьмибитовый АЦП. Каждый из «блоков» в первых четырёх схемах выше будет описан детально в следующих главах этой книги. Отдельные части или блоки все, в некоторой степени, взаимозаменяемы.

Глава 2. Разработка схемы аналогового входа

Входная цепь служит трём целям. Во-первых, предполагается высокое входное сопротивление для тестируемого устройства, что предохраняет его от перегрузки. Во-вторых, предполагается делитель на входе. Обычно это делитель 1/10, чтобы позволить отображение сигналов в 10 раз больших, чем позволено для АЦП. С пробником 10X можно наблюдать сигналы более 100 вольт. Схема также направлена на поддержание «рабочей точки». Большинство АЦП могут работать только с положительным напряжением, так что, аналоговый сигнал нуждается в смещении с тем, чтобы сигнал оказался в середине входного диапазона напряжений конвертора.

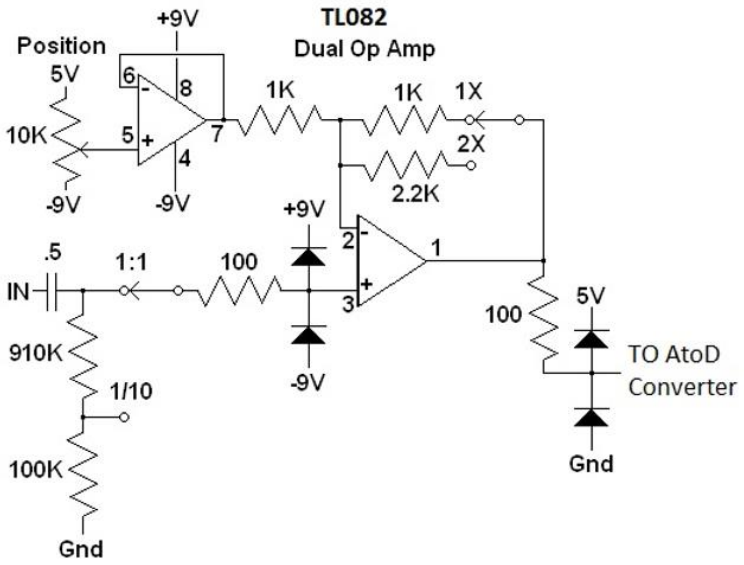
Вот очень простая схема входа. Она поддерживает только смещение.



Более удачный аналоговый вход может предоставить делитель 1/10 для 10-вольтового сигнала, буфер с большим сопротивлением и смещением, и всё только с одной микросхемой. Вы можете использовать FET (field-effect transistor, полевой транзистор) вход двойного усилителя, как, например, LF353 или TL082. Недостаток этого решения в том, что вам понадобится источник отрицательного напряжения 9-12 вольт. Хотя две батарейки по 9 вольт вполне подойдут. Вы также можете использовать 9-вольтовый AC адаптер и 555 инвертор питания для создания отрицательного напряжения.

Такая реализация входной цепи даёт меньшую нагрузку (1 МОм) для тестируемого устройства. Также есть выбор входного делителя, равно как и усиления. Кроме того, есть диод для защиты от напряжения входа и АЦП.

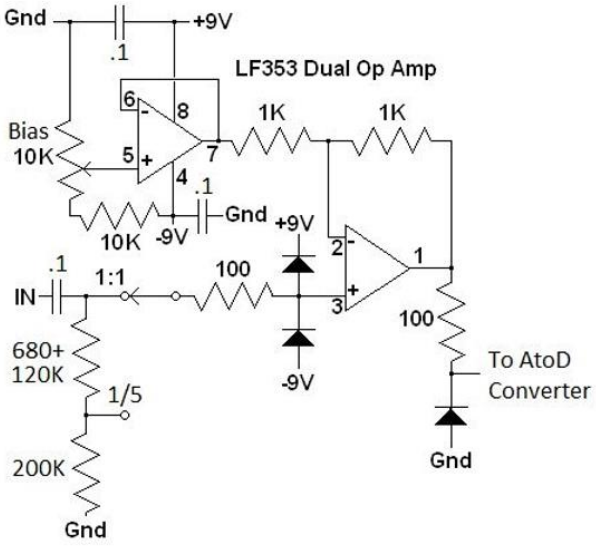
Вот схема для реализации аналогового входа.



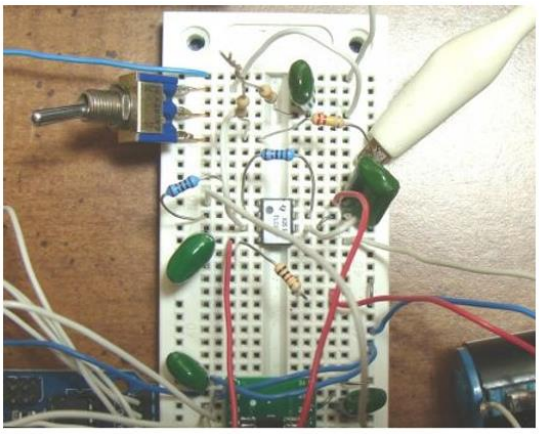
Далее мы улучшим входную цепь, которую только что показали. В предыдущем решении при изменении усиления ОУ требуется изменение рабочей точки, она должна быть подправлена. Лучше бы не иметь подстройки для решения этой проблемы. Усиление может быть перестроено с помощью программы. Подобное изменение даже упростит входную цепь. Вы можете использовать TL082, LF353 или множество других сдвоенных ОУ с FET входом.

Управление «рабочей точкой» подразумевает задание рабочей точки для АЦП. Вам нужно будет установить её около 1.3 вольт для 2.6-вольтового максимального входа. 1.3 вольт станет положением «нуля» в середине LCD. Эта рабочая точка является «0» сигнала, когда АЦП использует 2.6 вольт в качестве максимального входного напряжения, и не разрешает использования какого-либо отрицательного входного напряжения. Напряжение смещения будет выше, выше 2.5 вольт, при использовании CA3306 и CA3318, поскольку они имеют большее опорное напряжение.

Далее улучшенная схема.

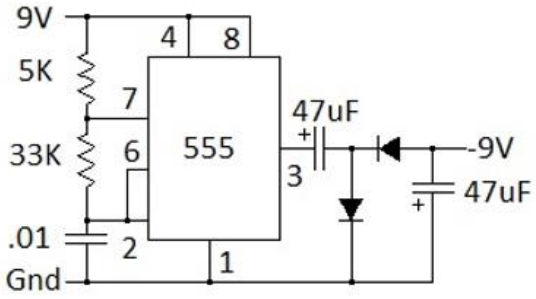


Ниже картинка, показывающая, как выглядит входная цепь, когда она собрана на плате. Не забудьте использовать конденсаторы фильтра 0.1 мкФ, которые подключаются между выводами 4 и 8 ОУ и землёй. Эти конденсаторы уменьшат шум ОУ.

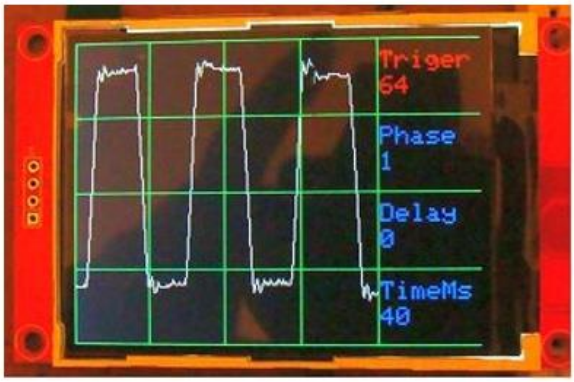


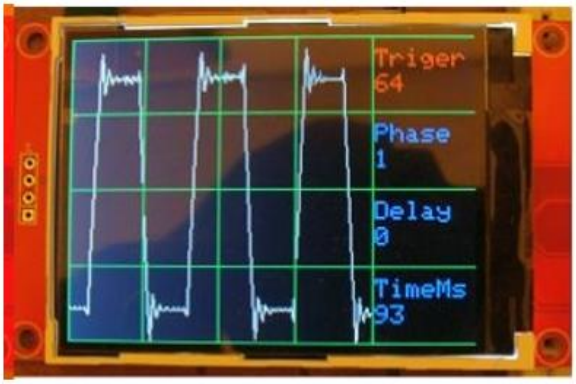
Питающее напряжение для ОУ должно быть двухполярным. +9 вольт получается от AC адаптера, питающего Arduino Uno. Его можно найти в гнезде «Vin» Arduino. Для получения -9 вольт вы

можете либо использовать 9-вольтовую батарейку, либо, что даже лучше, использовать осциллятор на таймере 555 с двумя диодами для формирования простого инвертора питания. Вот схема для 555 инвертора. Диоды могут быть 1N4001 или аналогичные. Под нагрузкой выходное напряжение может стать около -6 вольт, но всё прекрасно работает.



Для улучшения фронтов импульсов ОУ TL082 можно заменить на AD744. Потребуется изменить монтаж для правильной работы. Ниже две картинки, сравнивающие два ОУ при прямоугольных импульсах 65 кГц. Работа TL082 показана на первой картинке, а AD744 на второй. Заметьте, как улучшилось отображение углов импульсов на второй картинке.

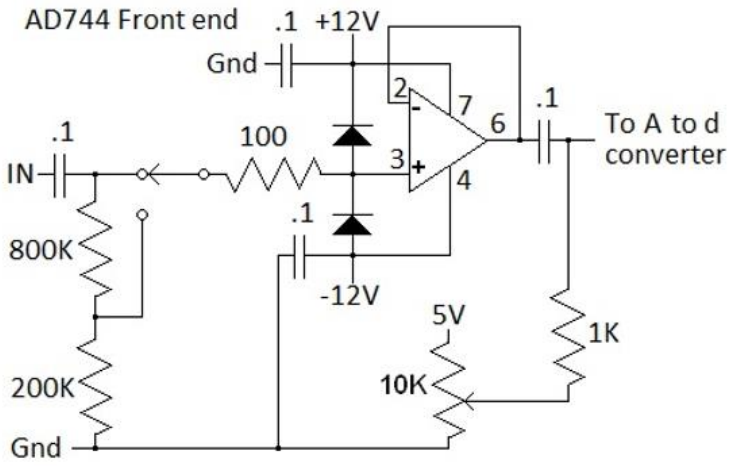




Хотя микросхема может работать с меньшим напряжением, AD744 работает лучше с положительным и отрицательным напряжением 9-12 вольт. Вы можете питать Arduino напряжением от 9 до 12 вольт через AC адаптер. В этом случае используйте гнездо «Vin» для питания AD744 и 555 инвертора для получения отрицательного питающего напряжения.

Но есть проблема с AD744 – микросхема не позволила мне добавить напряжение рабочей точки так, как я сделал с TL082. Вместо этого мне понадобилась пара AC и использование переменного резистора для добавления смещения непосредственно на входе АЦП.

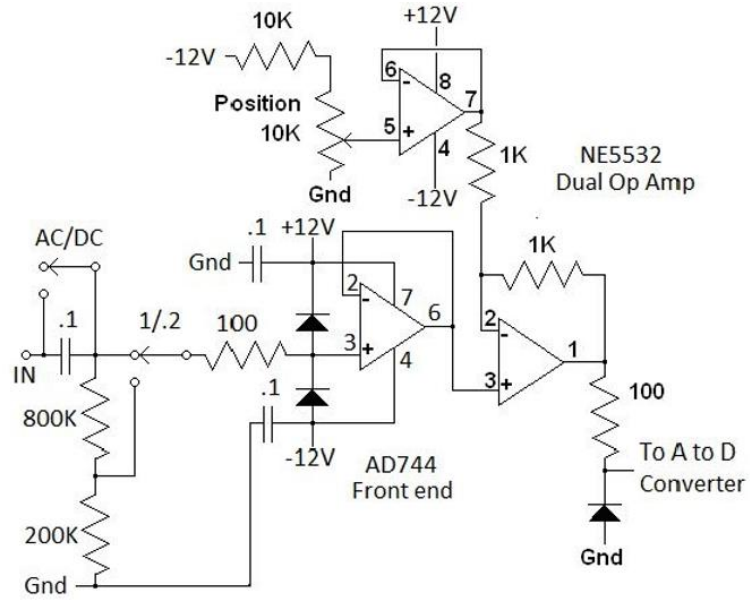
Ниже схема входной цепи с AD744.



Не забудьте 0.1 мкФ конденсаторы от плюса и минуса 12 вольт на землю. Также нужен конденсатор 5 пФ от вывода 5 к выводу 8 микросхемы AD744 или вы увидите «кучу» шума! Диоды защиты могут быть 1N914. Я использовал резистор 800 кОм из резисторов 470 кОм и 330 кОм, включённых последовательно. Некоторые осциллографы используют для входа комбинацию резисторов 900 кОм и 100 кОм для реализации 1X и .1X делителя.

Параллельно резистору 800 кОм должен быть добавлен подстроечный конденсатор 1-10 пФ. Также должен быть подстроечный конденсатор 10-100 пФ параллельно резистору 200 кОм. Эти конденсаторы могут понадобиться для улучшения фронтов прямоугольных импульсов.

Для улучшения совместимости AD744 с АЦП вам потребуется вернуться к цепям TL082. Но есть более удобный вариант – использовать NE5532 или AD827. Вместо максимального выхода до частоты 4 МГц, NE5532 позволяет получить 10 МГц, а AD827 50 МГц! Они также имеют более низкие шумовые свойства. Так что, ниже схемы двух цепей объединены вместе в одну схему.

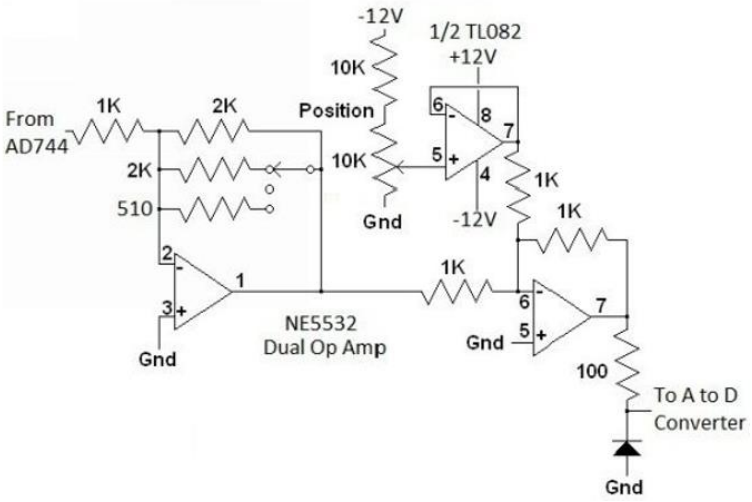


Есть ещё один предмет обсуждения. Вы можете добавить усилитель для «выбора усиления» между AD744 и NE5532. Этот дополнительный усилитель даст вам выбор усиления 0.5, 1 или 2.

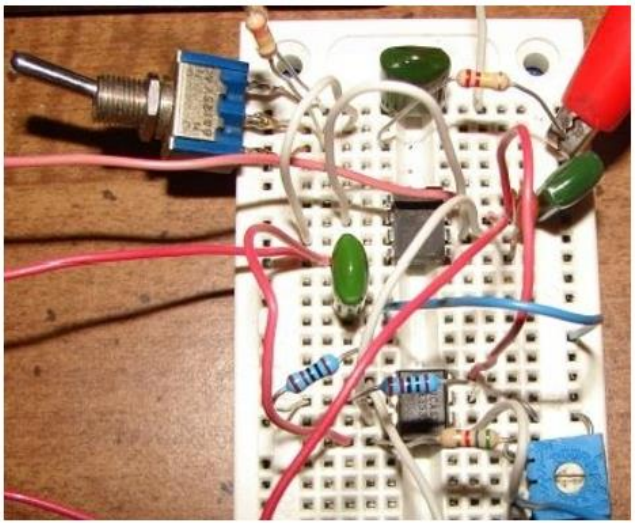
Некоторые осциллографы используют этот каскад для задания выбора усиления в 1, 2 или 5 раз. Этот дополнительный усилитель может быть полезен для наблюдения за маленькими сигналами.

Однако добавление ещё одного усилителя оставит нас без одного ОУ. Нам потребуется добавить половинку TL082 для буфера смещения. Операционному усилителю, используемого для буфера смещения, не нужна хорошая полоса пропускания, поскольку он используется только для создания смещения на постоянном напряжении.

Другая проблема с усилителем «усиления» в том, что он инвертирующий усилитель. Так что, суммирующий усилитель должен быть тоже инвертирующим усилителем. Таким образом, два инвертора будут отменять друг друга. Ниже схема, показывающая добавление «усилителя усиления» и инвертирующего смещения суммирующего усилителя.



Вот картинка AD744 с NE5532 на передней части платы. Это моя любимая конфигурация АЦП.



Глава 3. Быстрые АЦП

Ниже показан список некоторых быстрых АЦП, которые я тестировал. Есть другие, которые тестировались, но часть из них скрылась в дыму. AD775 и TLC5510 взаимозаменяемы. Они мои любимцы, поскольку стали предметом многих злоупотреблений, но до сих пор работают.

CA3306 (Работает лучше с портом C «PINC», поскольку это 6-битовый АЦП)
Bits: 6
Clock: 15MHz проверено до 16MHz

CA3318 (8-битовая версия CA3306)
Bits: 8
Clock: 15MHz проверено до 16MHz

AD775 (Снято с производства и трудно найти)
Bits: 8
Clock: 20MHz проверено до 25MHz

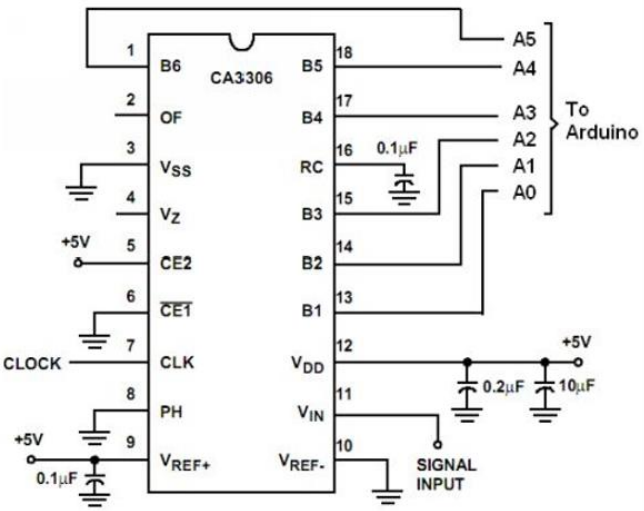
TDA8703 (Похоже сохраняет работу без тактирования)
Bits: 8
Clock: 40MHz проверено до 50MHz

TLC5510 (Некоторые выводы как у CDX1175/ADC1175/ADC1173)
Bits: 8
Clock: 20MHz

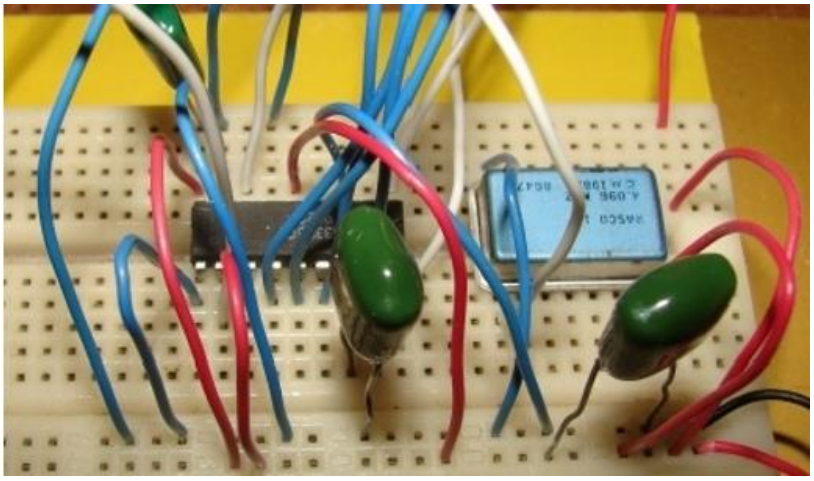
TLC5540 (Почти как LTC5510, но быстрее)
Bits: 8
Clock: 40MHz

ADC1175-50 (Некоторые выводы как у CDX1175/ADC1175/ADC1173)
Bits:8
Clock: 50 MHz

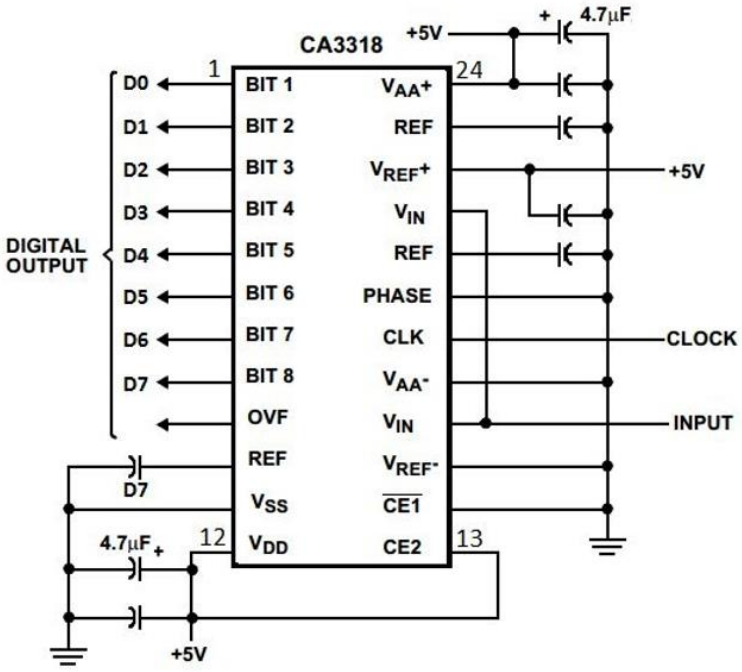
Вот схема с CA3306 АЦП. Схема адаптирована на основании PDF файла спецификации для CA3306. Входные тактовые импульсы можно получить от любого 4-15 МГц осциллятора.



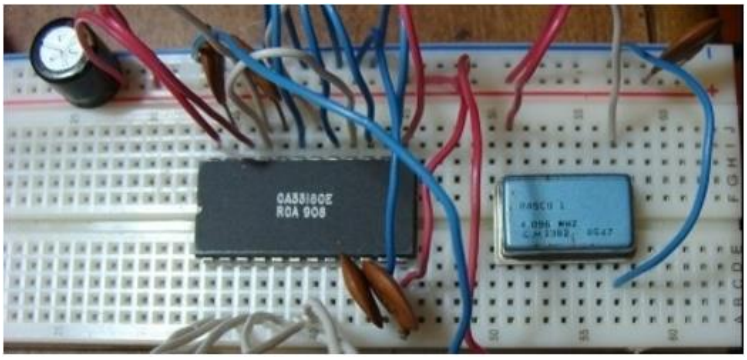
Далее картинка CA3306 АЦП с осциллятором и конденсаторами фильтра.



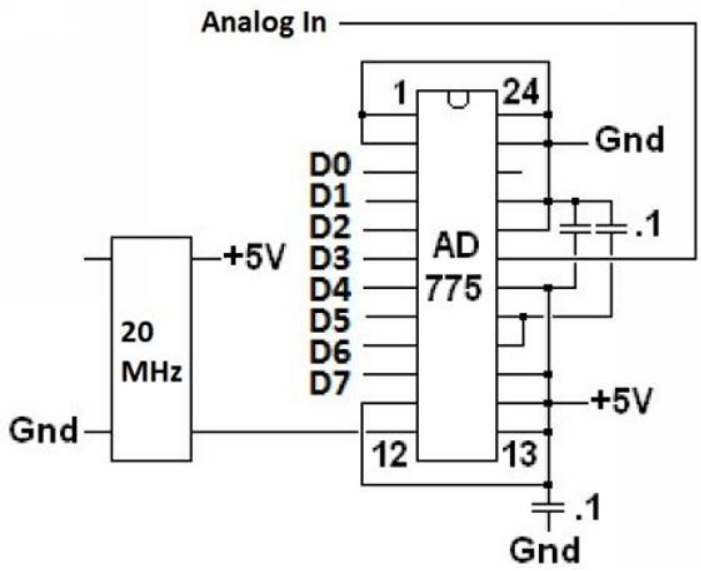
Следующий АЦП – CA3318. Это, собственно, CA3306 с 8-битовой точностью. Я проверял конвертор на 16 МГц, но он перестаёт работать на 20 МГц. Вот схема. Немаркированный конденсатор имеет величину 0.1 мкФ.



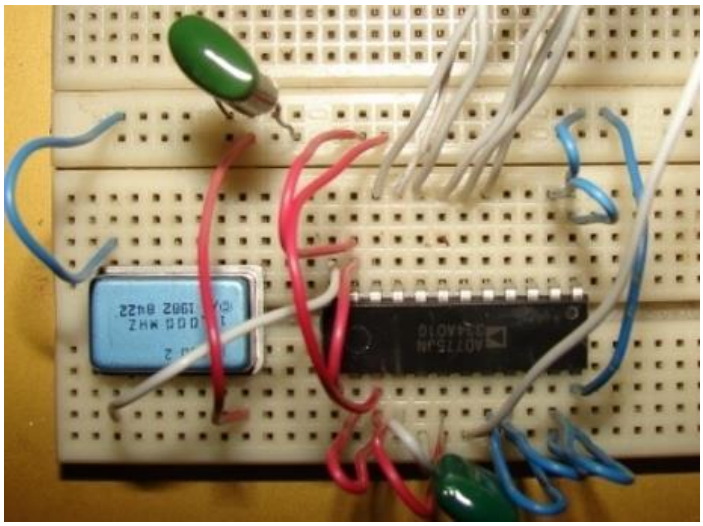
А это картинка с собранным и работающим CA3318.



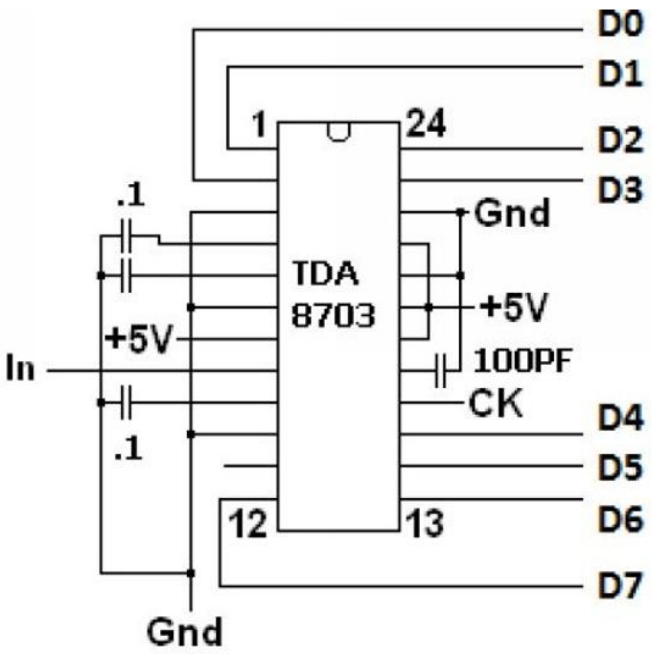
AD775 был первым быстрым АЦП, с которым я экспериментировал. Я собрал на параллельном порте видео адаптер, используя его в 90е. Заметьте, что выводы почти идентичны выводам TLC5510.



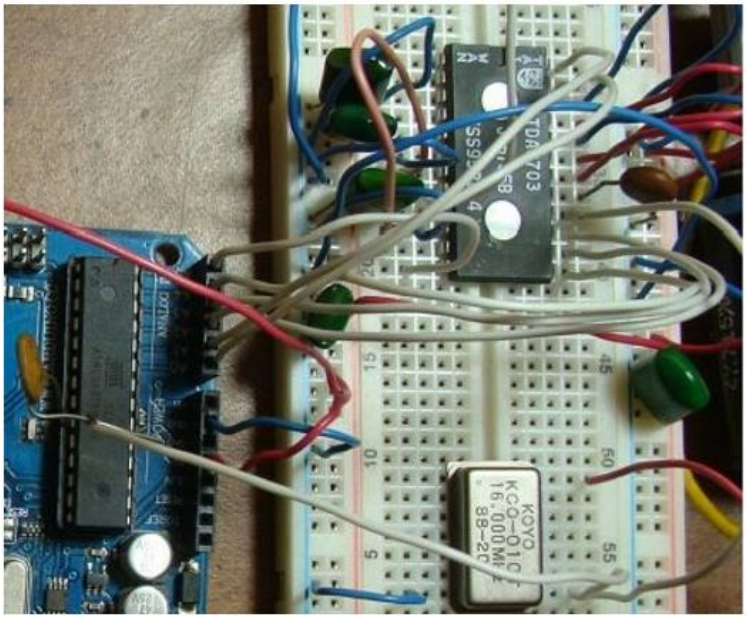
Вот картинка с собранным и работающим AD775.



Микросхема TDA8703 физически больше. Доступна и маленькая версия, но она требует адаптера. Микросхема TDA8703 была весьма популярным быстрым конвертором на протяжении ряда лет. Она использовалась в других цифровых осциллографах. Вот схема с TDA8703. «CK» - это вход тактовых импульсов.



Далее картинка собранного TDA8703, подключённого к аналоговому входу Arduino Uno. На левой стороне картинки вы можете видеть конденсатор 10 пФ, подключённый к выводу тактирования 9 для синхронизации двух тактовых генераторов.

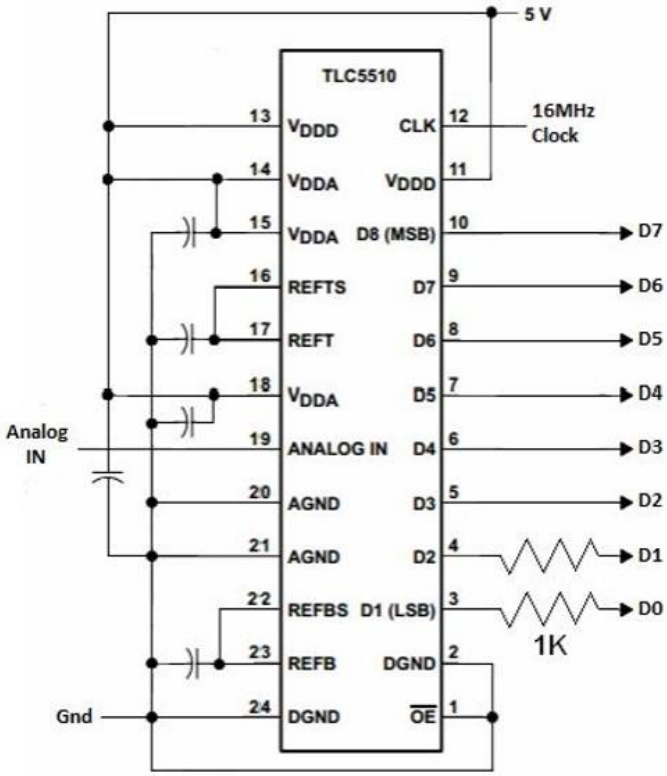


TLC5510 – обычно линия небольших микросхем (SOIC). Вам понадобится использовать адаптер для установки её на плату. Пайка микросхемы на адаптер может стать весьма хитрой операцией. Одна из хитростей в использовании паяльного флюса. Заметьте, что схема показывает микросхему, нарисованную нижней стороной. Я подключил её верхней стороной, чтобы соответствовать схеме, а затем перевернул плату, чтобы подключить микросхему. Конденсаторы 0.1 мкФ на 16 вольт.

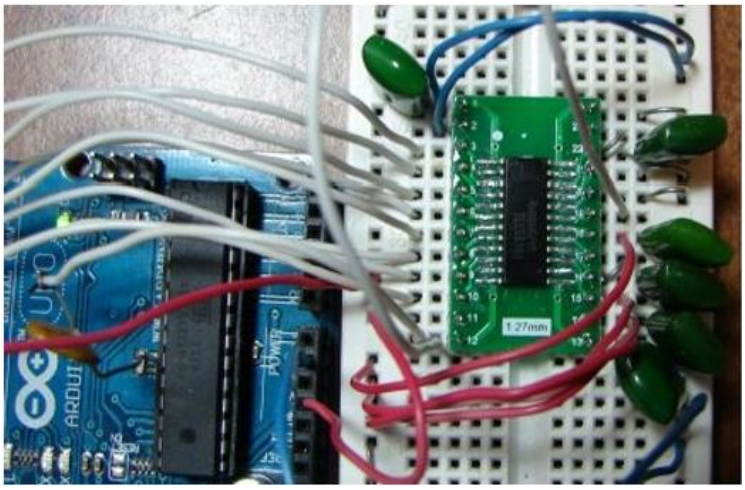
Также взгляните на два дополнительных по 1 кОм (или 2.2 кОм) резистора, которые помогают предотвратить смешение D0 и D1 при загрузке программы в Arduino Uno. Они не нужны, если вы

отключите D0 и D1, когда загружаете программу. Они не решают проблему в полной мере; временами вам, возможно, придётся отключать их, чтобы загрузить новую программу.

Далее схема с TLC5510 на базе указаний по применению.



Затем картинка с собранной и работающей схемой на TLC5510. На картинке вы может также видеть конденсатор 10 пФ, подключённый к выводу 9 Arduino Uno для синхронизации генераторов, когда не используется FIFO.



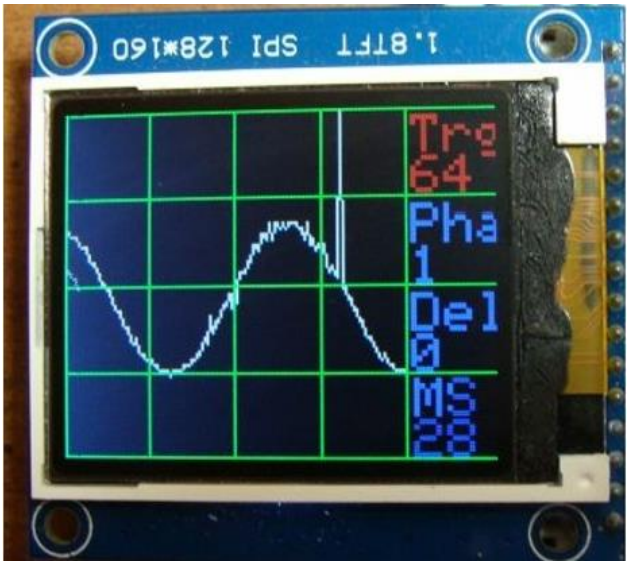
Одна из проблем с любым свободно работающим АЦП в «Clock Glitches, ошибки тактирования». Проблема ошибок происходит от не полного совпадения тактирования АЦП с тактированием Arduino. И процессор, и АЦП работают в их собственном режиме «свободной работы».

Однако процессор всегда будет читать содержимое входного порта на одной и той же фазе его тактового генератора. АЦП будет всегда обновлять свой результат на одной и той же фазе, но его

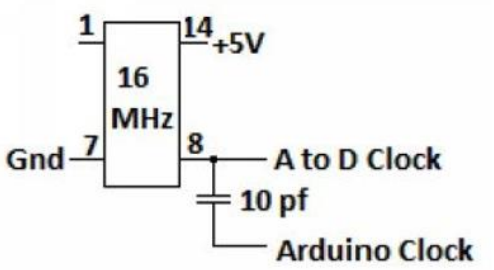
собственного тактового генератора. И, если эти две фазы не совмещены правильно, получится ошибка. Если выход АЦП был в процессе изменения с 10000000 на 01111111, результатом станет появление всех нулей или всех единиц на время обновления. Решение только в привязке фаз двух тактовых генераторов так, чтобы быть уверенным, что фаза аналогового обновления не совпадает с фазой параллельного ввода Arduino.

Длительное время я пытался подключиться к тактовому генератору на выводе 9 Arduino, чтобы использовать его для запуска АЦП. Затем я пытался «перетянуть» тактовый генератор Arduino внешним генератором, но это не сработало. Тогда я нашёл простое решение проблемы. Если вы подключите внешний генератор 16 МГц через конденсатор 10-20 пФ к выводу 9 Arduino, то это работает! Проблема в том, что вход тактирования вывода 9 используется для получения очень маленького сигнала на нём, так что внешний генератор должен быть «таким же маленьким». Решение работает только с Arduino Uno, поскольку модуль имеет специфическую аппаратную модификацию.

Ниже картинка, показывающая, как выглядит ошибка тактирования на экране LCD.



Улучшение схемы синхронизации просто подразумевает осциллятор 16 МГц с 10 пФ конденсатором, который идёт к микросхеме генератора Arduino на выводе 9. Вы можете также использовать генератор 32 МГц, поделив предварительно частоту на два. Решение работает с любым АЦП, описанным в этой книге.



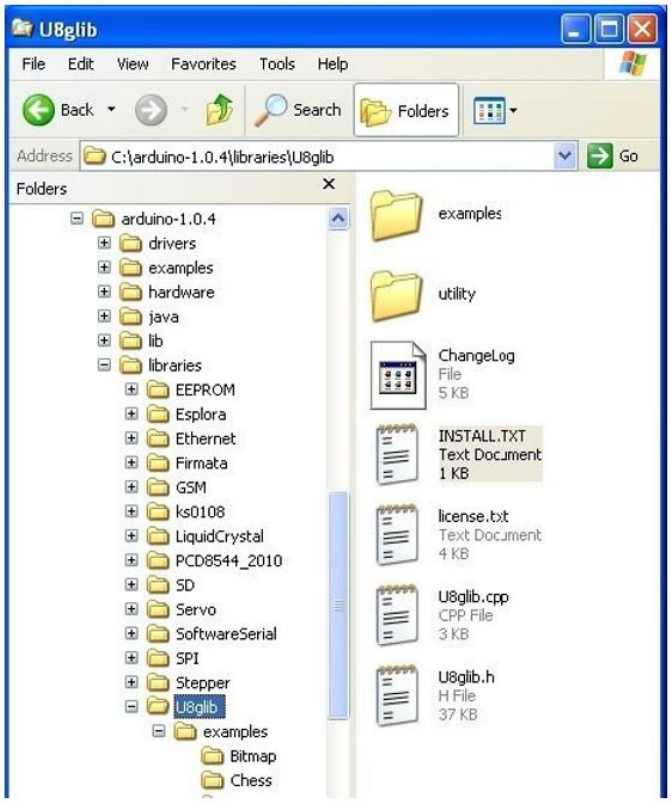
Глава 4. Монохромный LCD QC12864B

Утверждение, что графические дисплеи широко варьируются в части выводов, неполно. Когда оно относится только к графическим LCD 128x64, следует учесть, что дисплеи варьируются по выводам, физическому размеру, модели контроллера, который используется, и даже в по разным режимам операций.

Когда я начинал, у меня не было графического дисплея 128x64 в наличии, так что я должен был купить его для этих проектов. Первый LCD «прибывал» несколько недель, но, когда я открыл его, там вместо LCD был ленточный кабель с 40 выводами в коробке! Я быстро поехал к моему любимому поставщику и заказал другой 128x64 дисплей, даже не удосужившись прочитать мелкий шрифт. Когда заказ пришёл, я подключил дисплей и загрузил программу, но всё, что я получил, это мусор на экране. Пришлось вернуться и прочитать мелкий шрифт. LCD назывался QC12864B и использовал ST7920 контроллер!

Решение в использовании универсальной графической библиотеки, называемой «U8Glib.zip». Что означает – универсальная 8-битовая графическая библиотека. Вновь, вам нужно распаковать её в вашу директорию «Arduino\libraries», и затем убедиться в том, что она видна в интерфейсе Arduino.

Вот картинка, что показывает библиотеку U8Glib, которая установлена правильно.



Библиотека U8G имеет очень длинный список команд, которые она поддерживает. Вот краткий список доступных команд:

```
Begin(void);
disableCursor(void);
drawBitmap, drawBitmapP (x, y, count, height, *bitmap);
drawBox (x, y, width, height);
```

```
drawCircle (x ,y, radius);
drawDisc (x, y, radius);
drawFrame (x, y, width, height);
drawHLine, drawVLine (x, y, width);
drawLine (x1, y1, x2, y2);
drawPixel (x, y);
drawRBox, drawRFrame (x, y, width, height, radius);
drawStr, drawStr90, drawStr180, drawStr270 (x, y, *string);
drawStrP, drawStr90P, drawStr180P, drawStr270P (x, y, *string);
drawXBM, drawXBMP (x, y, width, height, *bitmap);
enableCursor(void);
firstPage(void);
getColorIndex(void);
getFontAscent(void);
getFontDescent(void);
getFontLineSpacing(void);
getHeight(void);
getMode(void);
getWidth(void);
getStrWidth (*string);
InitSPI (*dev, sck, mosi, cs, a0, reset);
InitHWSPI (*dev, cs, a0, reset);
Init8Bit (*dev, d1, d2, d3, d4, d5, d6, d7, en, cs1, cs2, di, rw, reset);
nextPage(void);
print(...);
setColorIndex (Color_index);
setContrast (contrast);
setCursorColor (foreground, background);
setCursorFont (*font);
setCursorPos (x,y);
setCursorStyle (encoding);
setDefaultBackgroundColor, setDefaultForegroundColor (void);
setDefaultMidColor (void);
setFont (*font);
setFontLineSpacingFactor (factor);
setFontPosBaseline, setFontPosBottom (void);
setFontPosCenter, setFontPosTop (void);
setFontRefHeightAll, setFontRefHeightExtendedText (void);
setFontRefHeightText (void);
setPrintPos (x, y);
setRot90, setRot180, setRot270 ();
 setScale2x2 ();
sleepOn, sleepOff (void);
undoRotation();
undoScale();
```

Есть также полный список драйверов контроллеров и графических форматов, которые поддерживает U8Glib. Вы должны удалить «//» перед заданием контроллера и графики, которые вы используете. Затем вам предстоит соединить LCD согласно с выводами, которые заданы в следующей строке. Это только частичный список множества поддерживаемых контроллеров. Мы будем использовать набор микросхем ST7920, поэтому он не имеет «//» пред своими строками.

```
U8GLIB_ST7920_128X64_4X u8g (8, 9, 10, 11, 4, 5, 6, 7, 18, 17, 16);
//8Bit Com: D0..D7: 8, 9, 10, 11, 4, 5, 6, 7 en=18, di=17, rw=16
//U8GLIB_PCD8544 u8g (13, 11, 10, 9, 8);
```

```
//SPI Com: SCK = 13, MOSI = 11, CS = 10, A0 = 9, Reset =8
//U8GLIB_KS0108_128 u8g (8, 9, 10, 11, 4, 5, 6, 7, 18, 14, 15, 17, 16);
//8Bit Com: D0-D7: 8, 9, 10, 11, 4, 5, 6,7 en=18, cs1=14, cs2=15, di=17, rw=16
//U8GLIB_LC7981_160X80 u8g (8, 9, 10, 11, 4, 5, 6, 7, 18, 14, 15, 17, 16);
//8Bit Com: D0..D7: 8, 9, 10, 11, 4, 5, 6, 7 en=18, cs=14, di=15, rw=17, reset=16
//U8GLIB_SSD1306_128X64 u8g (13, 11, 10, 9);
//SW SPI Com: SCK = 13, MOSI = 11, CS = 10, A0 = 9
//U8GLIB_SSD1309_128X64 u8g (13, 11, 10, 9);
//SPI Com: SCK = 13, MOSI = 11, CS = 10, A0 = 9
//U8GLIB_NHD_C12864 u8g (13, 11, 10, 9, 8);
//SPI Com: SCK = 13, MOSI = 11, CS = 10, A0 = 9, RST = 8
//U8GLIB_NHD_C12832 u8g (13, 11, 10, 9, 8);
//SPI Com: SCK = 13, MOSI = 11, CS = 10, A0 = 9, RST = 8
```

Ниже картинка списка выводов графического дисплея 128x64 QC12864B.

ITEM	SYMBOL	LEVEL	FUNCTIONS
1	VSS	0V	Power Ground
2	VDD	+5V	Power supply for logic
3	V0	—	Contrast adjust
4	RS(CS)	H/L	H:data L:command
5	RW/(SID)	H/L	H:read L:write
6	E/(SCLK)	H.H→L	Enable signal
7-14	DB0-DB7	H/L	Data Bus
15	PSB	H/L	H:Paraller mode L:serial mode
16	NC	—	No connection
17	/REST	L	Reset signal
18	VOUT	—	Output LCD voltage
19	LEDA	+5V	Power supply for LED backlight
20	LEDK	0V	

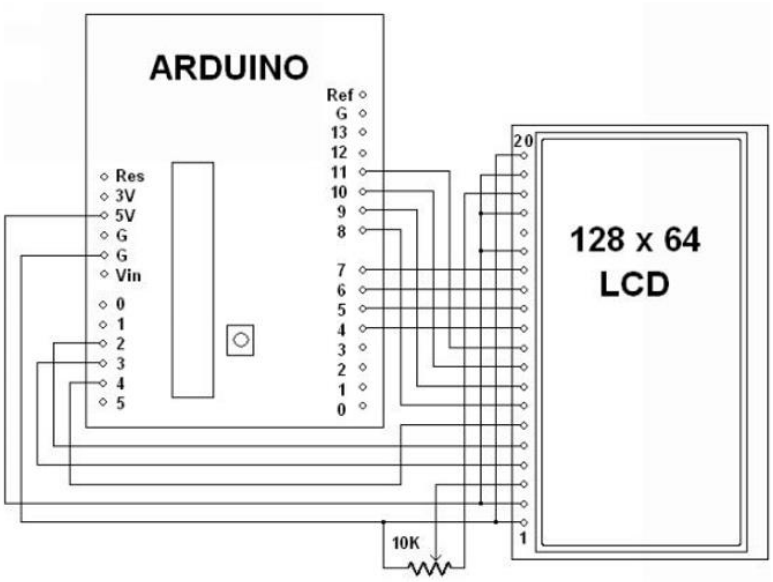
Если вы думаете, что таблица сбивает с толку, то вы не одиноки. В верхней части мы имеем соединение с Arduino Uno для обсуждения. Вот таблица выводов, показывающая соединения с Arduino, написанная на простом английском.

Pin	Symbol	Arduino	Function
1	Vss	GND	Ground
2	Vdd	5V	Five Volts
3	Vo		Contrast variable resistor wiper
4	RS/CS	17 - A3	Register Select
			Higha is data, low is a command
5	RW/SID	16 - A2	Read Write, or Serial Data
			High is read, Low is write
6	E/SCLK	18 - A4	Enable or Serial Clock
7	DB0	8	data Bus bit 0
8	DB1	9	data Bus bit 1
9	DB2	10	data Bus bit 2
10	DB3	11	data Bus bit 3
11	DB4	4	data Bus bit 4
12	DB5	5	data Bus bit 5
13	DB6	6	data Bus bit 6
14	DB7	7	data Bus bit 7
15	PSB	5V	Parallel/Serial Bus
			High is Parallel, Low is Serial

16	NC		Not Connected – a really easy one
17	RST	5V	Reset Signal – Low is Reset
18	Vout		Voltage out to the contrast variable resistor
19	LED+	5V	Anode to the LED backlight
20	LED	GND	Ground for the LED backlight

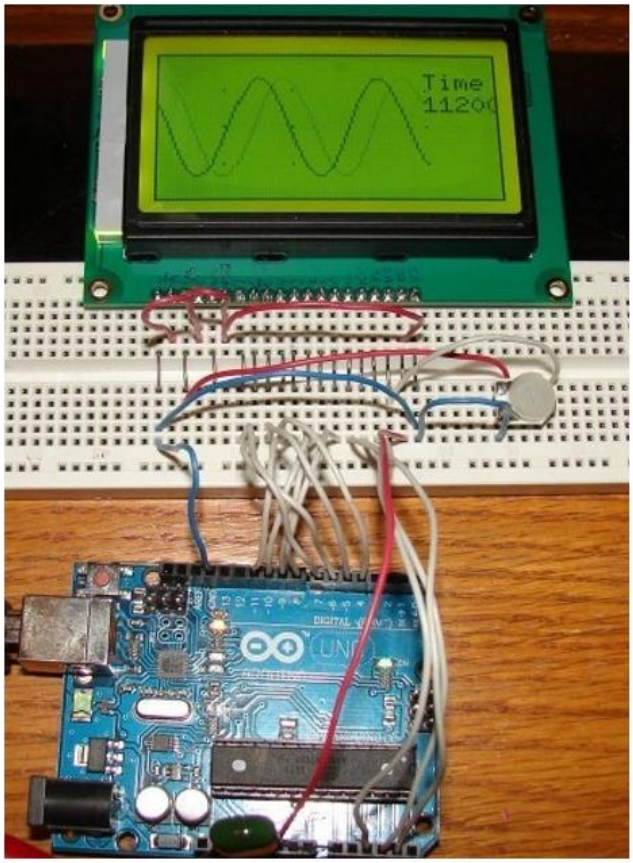
Когда поддерживающая программа установлена, вы можете соединить выводы дисплея и включить. Мой дисплей всё ещё показывает мусор, но это происходит из-за того, что некоторые провода данных были перепутаны. Я использовал все подключения одного цвета, белые. Кто-то рекомендует использовать провода для подключения разноцветные, чтобы они случайно не перепутывались.

Ниже схема для QC12864B с параллельным 8-битовым подключением.



Следующий проект – это простой осциллограф на базе Arduino. Он имеет программу, основанную на триггере цикла, так что изображение сигнала почти не прыгает.

Чтобы проверить этот проект я запускал программу, названную «Sweep Generator», на моём лэптопе. Файл программы называется «sweepgen.exe». Я проверял эту разработку осциллографа в звуковом диапазоне 100 Гц – 1000 Гц. Этот простой осциллограф вполне приемлем до 1 кГц, но отнюдь не выше, где дисплей показывает что-то больше напоминающее шум.



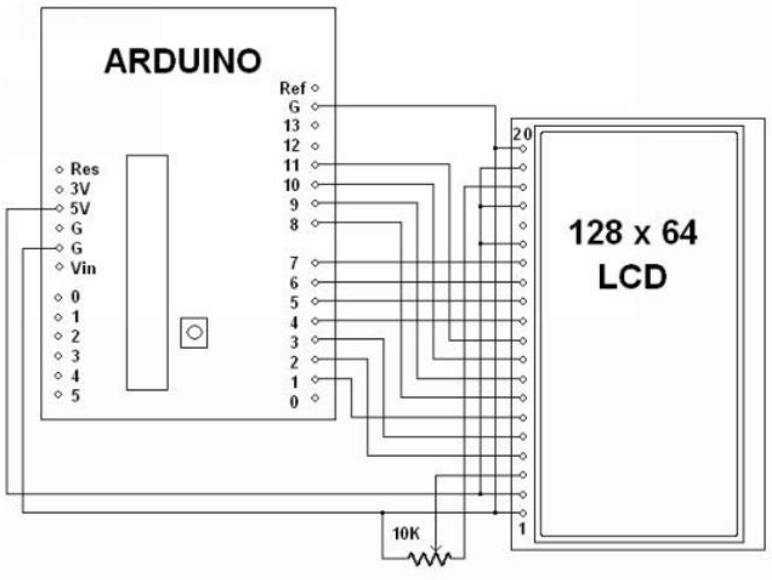
Вот скетч простого осциллографа:

```
/**********************************
128 by 64 LCD Oscilloscope - Default
By Bob Davis
Uses Universal 8bit Graphics Library, http://code.google.com/p/u8glib/
Copyright (c) 2012, olikraus@gmail.com All rights reserved.
**********************************************/
#include "U8glib.h"
// 8Bit Com: D0..D7: 8,9,10,11,4,5,6,7 en=18, di=17, rw=16
U8GLIB_ST7920_128X64_4X u8g (8, 9, 10, 11, 4, 5, 6, 7, 18, 17, 16);
int Sample[128];
int Input = 0;
int OldInput = 0;
int StartSample = 0;
int EndSample = 0;
int SampleTime = 0;

void u8g_prepare (void) {
 u8g.setFont (u8g_font_6x10);
 u8g.setFontRefHeightExtendedText ();
 u8g.setDefaultForegroundColor ();
 u8g.setFontPosTop();
}

void DrawMarkers (void) {
 u8g.drawFrame  (0, 0, 128, 64);
 u8g.drawPixel (25, 16);
 u8g.drawPixel (50, 16);
```

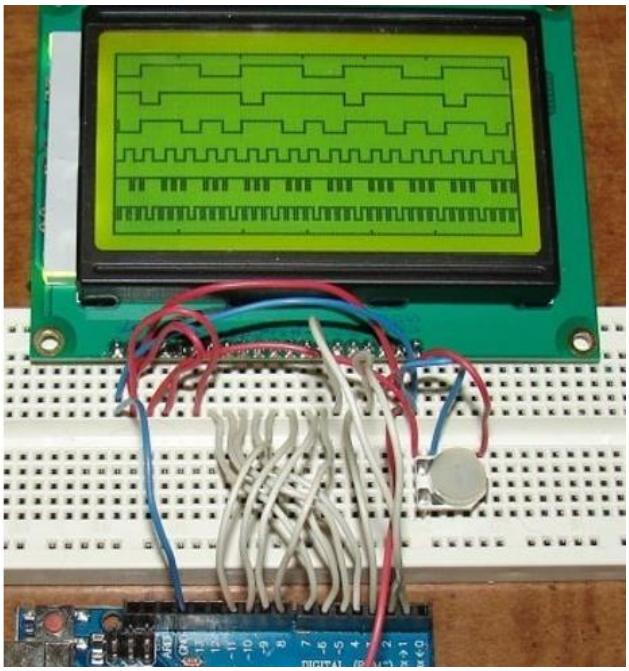
```
 u8g.drawPixel (100, 16);
 u8g.drawPixel (25, 32);
 u8g.drawPixel (50, 32);
 u8g.drawPixel (100, 32);
 u8g.drawPixel (25, 48);
 u8g.drawPixel (50, 48);
 u8g.drawPixel (100,48);
}

void sample_data (void) {
// wait for a trigger of a positive going input
 while (Input < OldInput) {
 OldInput = analogRead (A0);
 Input = analogRead (A0);
}
// collect the analog data into an array
// do not do division by 10.24 here, it makes it slower!
 StartSample = micros ();
 for (int xpos=0; xpos<100; xpos++) {
 Sample [xpos] = analogRead (A0);
}
 EndSample = micros();
}

void draw (void) {
 char buf [12];
 u8g_prepare ();
 DrawMarkers ();
// display the collected analog data from array
// Sample/10.24 because 1024 becomes 100 = 5 volts
 for (int xpos=1; xpos<99; xpos++) {
 u8g.drawLine (xpos, Sample[xpos]/10.24, xpos+1, Sample[xpos+1]/10.24);
}
 SampleTime = EndSample-StartSample;
 u8g.drawStr (100, 8, "Time");
 u8g.drawStr (100, 18, itoa(SampleTime, buf, 10));
}

void setup(void) {
// assign default color value
 if ( u8g.getMode () == U8G_MODE_R3G3B2)
 u8g.setColorIndex (255); // RGB = white
 else if (u8g.getMode () == U8G_MODE_GRAY2BIT )
 u8g.setColorIndex (3); // max intensity
 else if (u8g.getMode () == U8G_MODE_BW)
 u8g.setColorIndex(1); // pixel on, black
}

void loop (void) {
// collect the data
 sample_data ();
// show      collected data
 u8g.firstPage();
 do { draw(); }
 while (u8g.nextPage ());
```

```
// rebuild the picture after some delay
 delay(100);
}
```

Следующий проект – это логический анализатор на шесть каналов. Нам нужно сделать доступными все шесть аналоговых входов, так что нам нужно «перебросить» управляющие сигналы LCD на другие выводы. Выводы D1, D2 и D3 доступны с этим LCD, так что я переместил их сюда. Вот обновлённая схема с перемещёнными управляющими выводами. Это изменение оставляет все шесть аналоговых входов свободными для использования в качестве логических входов.



Ниже картинка логического анализатора в действии. Картинка получена при наблюдении за сигналами от 100 КС до 1 МС выходов 74LS390, сдвоенного делителя на 10 с входным тактированием 10 МС. Результат отображения весьма интересен.

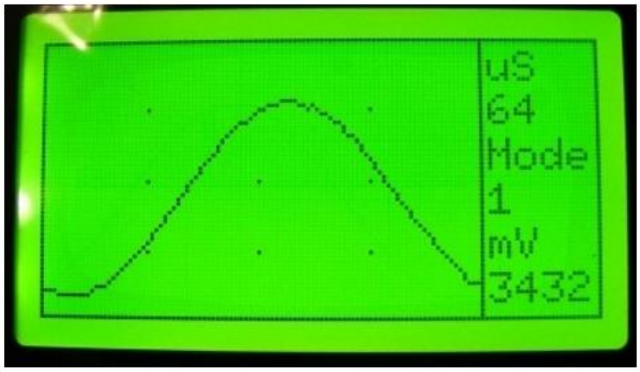


А вот код скетча, который позволит шестиканальному логическому анализатору работать.

```
/**********************************
128 by 64 six channel LCD Logic Analyzer
By Bob Davis
Uses Universal 8bit Graphics Library, http://code.google.com/p/u8glib/
Copyright (c) 2012, olikraus@gmail.com All rights reserved.
**********************************************/
#include "U8glib.h"
// 8Bit Com: D0..D7: 8,9,10,11,4,5,6,7 en=18, di=17, rw=16
// U8GLIB_ST7920_128X64_4X    u8g (8, 9, 10, 11, 4, 5, 6, 7, 18, 17, 16);
// ****     NOTE  **** I Moved the three control pins !!!
U8GLIB_ST7920_128X64_4X u8g (8, 9, 10, 11, 4, 5, 6, 7, 1, 2, 3);

int Sample [128];
int Input = 0;
int OldInput = 0;

void setup (void) {
pinMode (A0, INPUT);
pinMode (A1, INPUT);
pinMode (A2, INPUT);
pinMode (A3, INPUT);
pinMode (A4, INPUT);
pinMode (A5, INPUT);
// assign the default color value
if (u8g.getMode () == U8G_MODE_R3G3B2)
 u8g.setColorIndex (255); //  RGB=white
else if (u8g.getMode () == U8G_MODE_GRAY2BIT)
 u8g.setColorIndex(3); //max intensity
else if (u8g.getMode () == U8G_MODE_BW)
 u8g.setColorIndex(1); // pixel on, black
}

void u8g_prepare (void) {
 u8g.setFont (u8g_font_6x10);
 u8g.setFontRefHeightExtendedText ();
 u8g.setDefaultForegroundColor ();
 u8g.setFontPosTop();
}

void DrawMarkers (void) {
 u8g.drawFrame     (0, 0, 128, 64);
 u8g.drawPixel (20, 1);
 u8g.drawPixel (40, 1);
 u8g.drawPixel (60, 1);
 u8g.drawPixel (80, 1);
 u8g.drawPixel (100, 1);
 u8g.drawPixel (20, 62);
 u8g.drawPixel (40, 62);
 u8g.drawPixel (60, 62);
 u8g.drawPixel (80, 62);
 u8g.drawPixel (100, 62);
}

void draw (void) {
 u8g_prepare ();
```

```
 DrawMarkers ();
//wait for a trigger of a positive going input
 Input = digitalRead (A0);
while (Input != 1) {
 Input = digitalRead (A0);
}

// collect the analog data into an array
for (int    xpos = 0; xpos<128; xpos++) {
 Sample [xpos] = PINC;
}
// display the collected analog data from the array
for (int    xpos = 0; xpos<128; xpos++) {
 u8g.drawLine (xpos, ((Sample [xpos] & B00000001)*4) + 4, xpos+1,
((Sample[xpos+1]&B00000001)*4)+4);
 u8g.drawLine (xpos, ((Sample [xpos] & B00000010)*2) + 14, xpos+1,
((Sample[xpos+1]&B00000010)*2)+14);
 u8g.drawLine (xpos, ((Sample [xpos] & B00000100)*1) + 24,xpos+1,
((Sample[xpos+1]&B00000100)*1)+24);
 u8g.drawLine (xpos, ((Sample [xpos] & B00001000)/2) + 34, xpos+1,
((Sample[xpos+1]&B00001000)/2)+34);
 u8g.drawLine (xpos, ((Sample [xpos] & B00010000)/4) + 44,xpos+1,
((Sample[xpos+1]&B00010000)/4)+44);
 u8g.drawLine (xpos, ((Sample [xpos] & B00100000)/8) + 54, xpos+1,
((Sample[xpos+1]&B00100000)/8)+54);
}
} // end of draw routine

void loop (void) {
// this is the picture loop
// u8g.firstPage();
do { draw(); }
 while(u8g.nextPage());
// rebuild the picture after some delay
 delay(100);
}
```

Вы можете продолжить с шестиканальным анализатором, добавив внешний АЦП CA3306. Схема CA3306 в третьей главе этой книги. Вы также можете добавить переключатель скорости, используя подключение D12 и D13 к земле. Вот картинка экрана LCD, работающего с осциллографом 5 MSPS.



Вы также можете использовать следующий код скетча для логического анализатора с пятью миллионами выборок в секунду или для осциллографа.

```
/**********************************
128 by 64 LCD Oscilloscope ext AtoD speed select
By Bob Davis
Uses Universal 8bit Graphics Library, http://code.google.com/p/u8glib/
Copyright (c) 2012, olikraus@gmail.com All rights reserved.
**********************************************/
#include "U8glib.h"
// 8Bit Com: D0..D7: 8,9,10,11,4,5,6,7 en=18, di=17, rw=16
//U8GLIB_ST7920_128X64_4X u8g (8, 9, 10, 11, 4, 5, 6, 7, 18, 17, 16);
// NOTE that the pins have bee rearranged
U8GLIB_ST7920_128X64_4X u8g (8, 9, 10, 11, 4, 5, 6, 7, 1, 2, 3);

Byte Sample [100];
//int Sample[100];
int Input = 0;
int OldInput = 0;
long StartSample = 0;
long EndSample = 0;
long SampleTime = 0;
int MaxSample = 0;
int MinSample = 0;
int SampleSize = 0;
int dTime = 0;
int mode = 0;

void u8g_prepare (void) {
 u8g.setFont (u8g_font_6x10);
 u8g.setFontRefHeightExtendedText ();
 u8g.setDefaultForegroundColor ();
 u8g.setFontPosTop();
}

void DrawMarkers (void) {
 u8g.drawFrame (0, 0, 128, 64);
 u8g.drawFrame     (100, 0, 128, 64);
 u8g.drawPixel (25, 16);
 u8g.drawPixel (50, 16);
 u8g.drawPixel (75, 16);
 u8g.drawPixel (25, 32);
 u8g.drawPixel (50, 32);
 u8g.drawPixel (75, 32);
 u8g.drawPixel (25, 48);
 u8g.drawPixel (50, 48);
 u8g.drawPixel (75, 48);
}

void get_mode (void) {
if (digitalRead (13) == 0) mode++;
if (digitalRead (12) == 0) mode--;
if (mode>9) mode = 0;
if (mode<0) mode = 9;
// Select delay times for loop modes
```

```
if (mode == 0) dTime = 0;
if (mode == 1) dTime = 0;
if (mode == 2) dTime = 1;
if (mode == 3) dTime = 5;
if (mode == 4) dTime = 10;
if (mode == 5) dTime = 50;
if (mode == 6) dTime = 100;
if (mode == 7) dTime = 500;
if (mode == 8) dTime = 1000;
if (mode == 9) dTime = 5000;
}

void sample_data (void) {
// wait for a trigger of a positive going input
while (digitalRead (A0) == 0) {     }
// collect the analog data into an array
// mode 0 will use verbose method
if (mode == 0) {
StartSample = micros();
Sample[0]=PINC;
Sample[1]=PINC;   Sample[2]=PINC;   Sample[3]=PINC;
Sample[4]=PINC;   Sample[5]=PINC;   Sample[6]=PINC;
Sample[7]=PINC;   Sample[8]=PINC;   Sample[9]=PINC;
Sample[10]=PINC;  Sample[11]=PINC;  Sample[12]=PINC;
Sample[13]=PINC;  Sample[14]=PINC;  Sample[15]=PINC;
Sample[16]=PINC;  Sample[17]=PINC;  Sample[18]=PINC;
Sample[19]=PINC;  Sample[20]=PINC;  Sample[21]=PINC;
Sample[22]=PINC;  Sample[23]=PINC;  Sample[24]=PINC;
Sample[25]=PINC;  Sample[26]=PINC;  Sample[27]=PINC;
Sample[28]=PINC;  Sample[29]=PINC;  Sample[30]=PINC;
Sample[31]=PINC;  Sample[32]=PINC;  Sample[33]=PINC;
Sample[34]=PINC;  Sample[35]=PINC;  Sample[36]=PINC;
Sample[37]=PINC;  Sample[38]=PINC;  Sample[39]=PINC;
Sample[40]=PINC;  Sample[41]=PINC;  Sample[42]=PINC;
Sample[43]=PINC;  Sample[44]=PINC;  Sample[45]=PINC;
Sample[46]=PINC;  Sample[47]=PINC;  Sample[48]=PINC;
Sample[49]=PINC;  Sample[50]=PINC;  Sample[51]=PINC;
Sample[52]=PINC;  Sample[53]=PINC;  Sample[54]=PINC;
Sample[55]=PINC;  Sample[56]=PINC;  Sample[57]=PINC;
Sample[58]=PINC;  Sample[59]=PINC;  Sample[60]=PINC;
Sample[61]=PINC;  Sample[62]=PINC;  Sample[63]=PINC;
Sample[64]=PINC;  Sample[65]=PINC;  Sample[66]=PINC;
Sample[67]=PINC;  Sample[68]=PINC;  Sample[69]=PINC;
Sample[70]=PINC;  Sample[71]=PINC;  Sample[72]=PINC;
Sample[73]=PINC;  Sample[74]=PINC;  Sample[75]=PINC;
Sample[76]=PINC;  Sample[77]=PINC;  Sample[78]=PINC;
Sample[79]=PINC;  Sample[80]=PINC;  Sample[81]=PINC;
Sample[82]=PINC;  Sample[83]=PINC;  Sample[84]=PINC;
Sample[85]=PINC;  Sample[86]=PINC;  Sample[87]=PINC;
Sample[88]=PINC;  Sample[89]=PINC;  Sample[90]=PINC;
Sample[91]=PINC;  Sample[92]=PINC;  Sample[93]=PINC;
Sample[94]=PINC;  Sample[95]=PINC;  Sample[96]=PINC;
Sample[97]=PINC;  Sample[98]=PINC;  Sample[99]=PINC;
// Sample[100]=PINC;
EndSample = micros();
```

```
}
// mode 1 will use loop with no delay
if (mode == 1) {
StartSample = micros();
for (int    xpos = 0; xpos<100; xpos++) {
 Sample[xpos] = PINC;
// delayMicroseconds(dTime);
}
EndSample = micros();
}
// mode 2 or more will use loop with delay
if (mode >= 2) {
 StartSample = micros();
for (int    xpos = 0; xpos<100; xpos++) {
 Sample[xpos] = PINC;
 delayMicroseconds(dTime);
}
EndSample = micros();
}
}
void draw (void) {
char buf[12];
u8g_prepare();
DrawMarkers();
// display the collected analog data from array
for (int xpos=1; xpos<99; xpos++) {
// For Oscope  more use this line
 u8g.drawLine (xpos, Sample[xpos], xpos+1, Sample[xpos+1]);
// For logic analyzer use the next 6 lines instead
//u8g.drawLine (xpos, ((Sample [xpos] & B00000001)*4) + 4, xpos,
((Sample[xpos+1]&B00000001)*4)+4);
//u8g.drawLine (xpos, ((Sample [xpos] & B00000010)*2) + 14, xpos,
((Sample[xpos+1]&B00000010)*2)+14);
// u8g.drawLine (xpos, ((Sample [xpos] & B00000100)*1) + 24, xpos,
((Sample[xpos+1]&B00000100)*1)+24);
// u8g.drawLine (xpos, ((Sample [xpos] & B00001000)/2) + 34, xpos,
((Sample[xpos+1]&B00001000)/2)+34);
// u8g.drawLine (xpos, ((Sample [xpos] & B00010000)/4) + 44,xpos,
((Sample[xpos+1]&B00010000)/4)+44);
// u8g.drawLine (xpos, ((Sample [xpos] & B00100000)/8) + 54,xpos,
((Sample[xpos+1]&B00100000)/8)+54);
}
SampleTime = EndSample-StartSample;
if (SampleTime    < 9999) u8g.drawStr (102, 2, "uS");
if (SampleTime    > 9999) {
SampleTime = SampleTime/1000;
u8g.drawStr (102, 2, "mS");
}
u8g.drawStr (102, 12, itoa (SampleTime, buf, 10));
u8g.drawStr (102, 22, "Mode");
u8g.drawStr (102, 32, itoa (mode, buf, 10));
// Determine sample voltage peak to peak
MaxSample = Sample[10];
MinSample = Sample[10];
for (int    xpos = 0; xpos<100; xpos++) {
```

```
// OldSample[xpos] = Sample[xpos];
if (Sample[xpos] > MaxSample) MaxSample = Sample[xpos];
if (Sample[xpos] < MinSample) MinSample = Sample[xpos];
}
// Range of 0 to 64 * 78 = 4992 mV
SampleSize = (MaxSample - MinSample)*78;
u8g.drawStr (102, 42, "mV");
u8g.drawStr (102, 52, itoa (SampleSize, buf, 10));
}

void setup (void) {
// set up input pins
 pinMode (12, INPUT);
 digitalWrite (12, HIGH);
 pinMode (13, INPUT);
 digitalWrite (13, HIGH);
// assign default color value
if (u8g.getMode() == U8G_MODE_R3G3B2)
u8g.setColorIndex(255); // RGB=white
else if (u8g.getMode() == U8G_MODE_GRAY2BIT)
u8g.setColorIndex(3);  // max intensity
else if (u8g.getMode() == U8G_MODE_BW )
u8g.setColorIndex(1); // pixel on, black
}

void loop (void) {
// Set up the mode
get_mode();
// collect the data
sample_data();
// show      collected data
u8g.firstPage();
do { draw(); }
while (u8g.nextPage());
// rebuild the picture after some delay
delay(500);
}
// end of program
```

Глава 5. Последовательный цветной LCD 1.8TFT SPI

Это LCD с последовательным доступом, поэтому дисплей будет занимать меньше выводов Arduino, оставляя больше свободных выводов для других целей. Этот дисплей требует только пять I/O выводов для операций, включая питание и землю. Программы для этого LCD будут использоваться вместе с новыми встроенными TFT драйверами, которые можно найти в версии 1.0.5 или более поздней драйверов Arduino. Похоже, что существует две версии 1.8-дюймового TFT LCD. Одна версия имеет 10 выводов, а другая 16. Определение выводов подписано в нижней части платы, так что не составит труда записать их пред тем, как перевернуть и подключить дисплей.

Вот картинка обратной стороны 1.8-дюймового LCD. Как можно видеть, выводы чётко промаркированы.

Скетчи для этого LCD используют новые встроенные TFT драйверы из набора драйверов Arduino 1.0.5 или выше. LCD не будет работать без правильно установленного TFT драйвера. Этот встроенный TFT драйвер тот же, что и Adafruit ST7735 драйвер, он только переименован в «TFT».

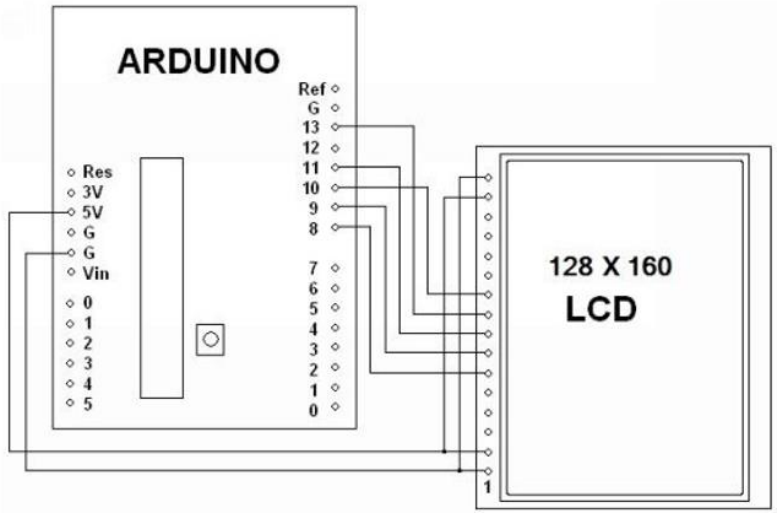
Вот таблица, показывающая соединение Arduino Uno с LCD дисплеем. Я использовал переключатели для соединения двух выводов GND и 5V вместе на плате.

Arduino Uno	1.8 SPI TFT
GND	Pin 01 (GND)
5V (VCC)	Pin 02 (VCC)
Not used	Pin 03
Not used	Pin 04
Not used	Pin 05
D8	Pin 06 (RESET)
D9	Pin 07 (A0)
D11 (MOSI)	Pin 08 (SDA)
D13 (SCK)	Pin 09 (SCK)
D10 (SS)	Pin 10 (CS)
Not used	Pin 11 SD Card
Not used	Pin 12 SD Card
Not used	Pin 13 SD Card
Not used	Pin 14 SD Card

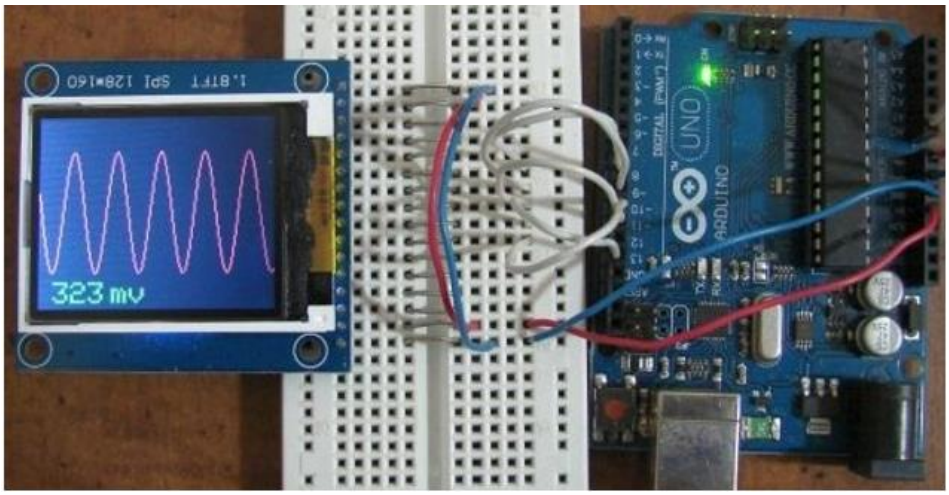
5V (VCC)	Pin 15 (LED+)
GND	Pin 16 (LED-)

Заметьте, что первый вывод LCD находится справа, если смотреть с верхней части LCD экрана.

Ниже схема того, как соединяется LCD. Заметьте, что D0-D7 и шесть аналоговых выводов A0-A6 теперь все свободны для использования в этом и других проектах.



Ниже картинка этого LCD, работающего в качестве экрана простого осциллографа, с демонстрационным скетчем. Это достаточно простой осциллограф, использующий A0, как аналоговый вход. Схему такого подключения можно посмотреть в предыдущей главе этой книги.



Этот скетч даст возможность получить простой осциллограф, который работает до 1 кГц. Осциллограмма отображается красным, а текст, отмечающий напряжение от пика до пика, отображён зелёным.

```
/**********************************
TFT Oscilloscope Reads the value of analog input on A0,
and shows the value on the screen.
Created     15 January 2014 by Bob Davis
**********************************/

#include <TFT.h>  // Arduino LCD library
```

```
#include <SPI.h>
// pin definition for the Uno
#define  rst  8
#define  dc  9
#define  cs   10
TFT TFTscreen = TFT (cs, dc, rst);

// set up variables
int xPos = 0;
int value = 0;
int maxvalue = 100;
int minvalue = 100;
int sensor[160];
char buf[12];

void setup() {
// initialize the display
 TFTscreen.begin ();
// clear the screen
 TFTscreen.background (0, 0, 0);
// Set the font size
 TFTscreen.setTextSize (2);
}

void loop () {
// quickly collect the data
 for (int   xpos = 0; xpos<160; xpos++) {
 sensor [xpos] = analogRead(A0);
}
// determine the peak to peak voltage
 maxvalue = sensor[1];
 minvalue = sensor[1];
for (int xpos = 0; xpos<160; xpos++) {
 if (sensor[xpos] >maxvalue) maxvalue = sensor[xpos];
 if (sensor[xpos] <minvalue) minvalue = sensor[xpos];
}
value = maxvalue-minvalue;
// erase the screen to start again
 TFTscreen.background (0, 0, 0);
// display the collected data
for (int xpos = 0; xpos<159; xpos++) {
// select the color = B,G,R
 TFTscreen.stroke (50, 50, 255);
// draw the line  (xPos1,     yPos1, xPos2, yPos2);
 TFTscreen.line (xpos, sensor[xpos]/8, xpos+1, sensor[xpos+1]/8);
// Set font color to green
 TFTscreen.stroke (0, 255, 0);
// Write the text value of the sensor
if (xpos == 0) TFTscreen.text (itoa(value/2, buf, 10), 10, 110);
if (xpos == 0) TFTscreen.text ("mv", 50, 110);
}
}
```

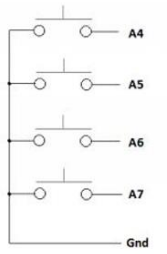
Теперь давайте добавим АЦП на выводы D0-D7. В программе команда параллельного ввода «PINC», которая использовалась в некоторых других проектах, теперь заменена на «PIND». Теперь

мы переключимся на АЦП TLC5510. TLC5510 – это восьмибитовый быстрый конвертор. Он есть в SOP корпусе, так что, вам придётся использовать адаптер для подключения его к макетной плате.

Заметьте, что выводы TLC5510 теперь идут от D0 до D7 Arduino. Вам понадобится либо отключить D0 и D1 в плане обновления программы Arduino, либо использовать два сопротивления по 1 кОм, как показано на схеме для TLC5510. Те же самые выводы D0 и D1 всегда используются для коммуникации Arduino через порт USB. Однако это единственный быстрый путь, чтобы получить доступ к 8 параллельным битам Arduino Uno.

Теперь поставим некоторые из тех свободных аналоговых входов «на службу». Вы можете подключить четыре выключателя без фиксации к A2, A3, A4 и A5 для выбора *вверх*, *вниз*, *вправо* и *влево*. Вверх и вниз будут выбирать изменение разделов, а вправо и влево будут менять значения выбранных разделов. Вы может также подключить все четыре выключателя к одному аналоговому вводу, переключая их к разным напряжениям резисторного делителя.

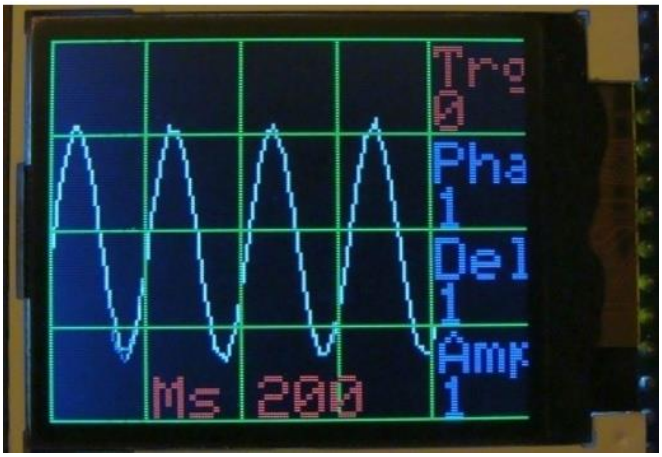
Ниже схема выключателей.



Наконец, есть множество программных улучшений. Мы можем теперь переключаться между множеством разделов подобных задержке между выборками (как мы это делали раньше), равно как и уровнями триггера и фазой переключения триггера.

Другое программное улучшение в том, что триггер теперь очень «плавный». Вначале он проверяет переход сигнала к положительным значениям, затем к отрицательным. Иначе он может только где-то «поймать» положительный уровень и приступить к сбору выборок. Триггер также может обрести возможность использования времени ожидания в тех случаях, когда триггер не срабатывает.

Вот картинка дисплея 1.8 inch SPI TFT LCD при работе PIND программы осциллографа.



Ниже полный листинг программы.

```
/***********************************
1.8 SPI PIND TFT Oscilloscope
Reads the D0-D7 pins    using PIND,
and shows the value on the screen.
Created      7 July 2015 by Bob Davis
***********************************/
#include <TFT.h> // Arduino LCD library
#include <SPI.h>

// pin definition for the Uno LCD
#define        rst   8
#define        dc   9
#define   cs    10
TFT TFTscreen = TFT (cs, dc, rst);

// set up variables
int xPos = 0;
char buf [12];
int Input = 0;
byte Sample [160];
int StartSTime = 0;
int EndSTime = 0;
int STime = 0;
int trigger = 64;
int trigphase = 1;
int tdelay = 1;
int select = 1;
int gain = 1;

void setup () {
// initialize the display
 TFTscreen.begin();
// clear the screen
 TFTscreen.background (0, 0, 0);
// Set the font size
 TFTscreen.setTextSize (2);
// A to D input pins
 pinMode (0, INPUT);
 pinMode (1, INPUT);
 pinMode (2, INPUT);
 pinMode (3, INPUT);
 pinMode (4, INPUT);
 pinMode (5, INPUT);
 pinMode (6, INPUT);
 pinMode (7, INPUT);
// Push button switches
 pinMode (16, INPUT_PULLUP);
 pinMode (17, INPUT_PULLUP);
 pinMode (18, INPUT_PULLUP);
 pinMode (19, INPUT_PULLUP);
}

void loop () {
// wait for a positive going trigger
if (trigphase == 1) {
```

```
 for (int   timeout=0; timeout < 1000; timeout++) {
 Input = PIND;
if (Input < trigger) break;
}
for (int    timeout = 0; timeout < 1000; timeout++) {
 Input = PIND;
if (Input > trigger) break;
}
}
// wait for a negative going trigger
if (trigphase == 0) {
for (int    timeout = 0; timeout < 1000; timeout++) {
 Input = PIND;
if (Input > trigger) break;
}
for (int    timeout = 0; timeout < 1000; timeout++) {
 Input = PIND;
if (Input < trigger) break;
}
}
// quickly collect the data with no delay
if (tdelay == -1) {
 StartSTime = micros();
      Sample[0]=PIND;                    Sample[1]=PIND;
      Sample[2]=PIND;                    Sample[3]=PIND;
      Sample[4]=PIND;                    Sample[5]=PIND;
      Sample[6]=PIND;                    Sample[7]=PIND;
      Sample[8]=PIND;                    Sample[9]=PIND;
      Sample[10]=PIND;                   Sample[11]=PIND;
      Sample[12]=PIND;                   Sample[13]=PIND;
      Sample[14]=PIND;                   Sample[15]=PIND;
      Sample[16]=PIND;                   Sample[17]=PIND;
      Sample[18]=PIND;                   Sample[19]=PIND;
      Sample[20]=PIND;                   Sample[21]=PIND;
      Sample[22]=PIND;                   Sample[23]=PIND;
      Sample[24]=PIND;                   Sample[25]=PIND;
      Sample[26]=PIND;                   Sample[27]=PIND;
      Sample[28]=PIND;                   Sample[29]=PIND;
      Sample[30]=PIND;                   Sample[31]=PIND;
      Sample[32]=PIND;                   Sample[33]=PIND;
      Sample[34]=PIND;                   Sample[35]=PIND;
      Sample[36]=PIND;                   Sample[37]=PIND;
      Sample[38]=PIND;                   Sample[39]=PIND;
      Sample[40]=PIND;                   Sample[41]=PIND;
      Sample[42]=PIND;                   Sample[43]=PIND;
      Sample[44]=PIND;                   Sample[45]=PIND;
      Sample[46]=PIND;                   Sample[47]=PIND;
      Sample[48]=PIND;                   Sample[49]=PIND;
      Sample[50]=PIND;                   Sample[51]=PIND;
      Sample[52]=PIND;                   Sample[53]=PIND;
      Sample[54]=PIND;                   Sample[55]=PIND;
      Sample[56]=PIND;                   Sample[57]=PIND;
      Sample[58]=PIND;                   Sample[59]=PIND;
      Sample[60]=PIND;                   Sample[61]=PIND;
      Sample[62]=PIND;                   Sample[63]=PIND;
```

```
      Sample[64]=PIND;                    Sample[65]=PIND;
      Sample[66]=PIND;                    Sample[67]=PIND;
      Sample[68]=PIND;                    Sample[69]=PIND;
      Sample[70]=PIND;                    Sample[71]=PIND;
      Sample[72]=PIND;                    Sample[73]=PIND;
      Sample[74]=PIND;                    Sample[75]=PIND;
      Sample[76]=PIND;                    Sample[77]=PIND;
      Sample[78]=PIND;                    Sample[79]=PIND;
      Sample[80]=PIND;                    Sample[81]=PIND;
      Sample[82]=PIND;                    Sample[83]=PIND;
      Sample[84]=PIND;                    Sample[85]=PIND;
      Sample[86]=PIND;                    Sample[87]=PIND;
      Sample[88]=PIND;                    Sample[89]=PIND;
      Sample[90]=PIND;                    Sample[91]=PIND;
      Sample[92]=PIND;                    Sample[93]=PIND;
      Sample[94]=PIND;                    Sample[95]=PIND;
      Sample[96]=PIND;                    Sample[97]=PIND;
      Sample[98]=PIND;                    Sample[99]=PIND;
      Sample[100]=PIND;                   Sample[101]=PIND;
      Sample[102]=PIND;                   Sample[103]=PIND;
      Sample[104]=PIND;                   Sample[105]=PIND;
      Sample[106]=PIND;                   Sample[107]=PIND;
      Sample[108]=PIND;                   Sample[109]=PIND;
      Sample[110]=PIND;                   Sample[111]=PIND;
      Sample[112]=PIND;                   Sample[113]=PIND;
      Sample[114]=PIND;                   Sample[115]=PIND;
      Sample[116]=PIND;                   Sample[117]=PIND;
      Sample[118]=PIND;                   Sample[119]=PIND;
      Sample[120]=PIND;                   Sample[121]=PIND;
      Sample[122]=PIND;                   Sample[123]=PIND;
      Sample[124]=PIND;                   Sample[125]=PIND;
      Sample[126]=PIND;                   Sample[127]=PIND;
      Sample[128]=PIND;                   Sample[129]=PIND;
      Sample[130]=PIND;
      EndSTime = micros();
}
// Collect the data with a no delay
// It will not allow a delay of 0 so this is here to do that
if (tdelay == 0) {
 StartSTime = micros();
for (int xpos = 0; xpos <130; xpos++) {
 Sample[xpos]=PIND;
}
 EndSTime = micros();
}
// Collect the data with a variable delay
if (tdelay > 0) {
 StartSTime = micros();
for (int xpos = 0; xpos <130; xpos++) {
 Sample [xpos] = PIND;
 delayMicroseconds (tdelay);
}
 EndSTime = micros();
}
 STime = EndSTime - StartSTime;
```

```
// display the collected data
for (int xpos = 0; xpos <160; xpos++) {
// select the color black = B,G,R
 TFTscreen.stroke (0, 0, 0);
// erase the old line
 TFTscreen.line (xpos+1, 0, xpos+1, 160);
// select the color white = B,G,R
 TFTscreen.stroke (255, 255, 255);
// draw the trace line (xPos1, yPos1, xPos2, yPos2); 0
// if (xpos <128) TFTscreen.line (xpos, (Sample[xpos]/2), xpos+1,
Sample[xpos+1]/2);
if (xpos <128) {
if (gain==1) {
 TFTscreen.line (xpos, (128-Sample[xpos]/2), xpos+1, 128-Sample[xpos+1]/2);
}
if (gain==2) {
 TFTscreen.line (xpos, (192-Sample[xpos]*1), xpos+1, 192-Sample[xpos+1]*1);
}
if (gain == 3) {
 TFTscreen.line (xpos, (320-Sample[xpos]*2), xpos+1, 320-Sample[xpos+1]*2);
}
if (gain == 4) {
 TFTscreen.line (xpos, (448-Sample[xpos]*3), xpos+1, 448-Sample[xpos+1]*3);
}
if (gain == 5) {
 TFTscreen.line (xpos, (576-Sample[xpos]*4), xpos+1, 576-Sample[xpos+1]*4);
}
}
// Set font color to green
 TFTscreen.stroke (0, 255, 0);
// draw the green lines
 TFTscreen.line (xpos, 0, xpos+1, 0);
 TFTscreen.line (xpos, 32, xpos+1, 32);
 TFTscreen.line (xpos, 64, xpos+1, 64);
 TFTscreen.line (xpos, 96, xpos+1, 96);
 TFTscreen.line (xpos, 127, xpos+1, 127);
// draw top to bottom green lines
if (xpos == 0) TFTscreen.line (xpos, 0, xpos,160);
if (xpos == 32) TFTscreen.line (xpos, 0, xpos,160);
if (xpos == 64) TFTscreen.line (xpos, 0, xpos,160);
if (xpos == 96) TFTscreen.line (xpos, 0, xpos,160);
if (xpos == 127) TFTscreen.line (xpos, 0, xpos,160);
}

// check the status of push button switches
// now circular so only two switches are required
if (digitalRead (17) == 0) select++;
if (digitalRead (16) == 0) select--;
if (select >4) select=1;
if (select <1) select=4;
// update the trigger level
if (select == 1) {
if (digitalRead (18) == 0) trigger++;
if (digitalRead (19) == 0) trigger--;
if (trigger > 128) trigger = 1;
```

```
if (trigger < 1) trigger=128;
}
// change the trigger phase
if (select == 2) {
if (digitalRead (18) == 0) trigphase++;
if (digitalRead (19) == 0) trigphase--;
if (trigphase < 0) trigphase = 1;
if (trigphase > 1) trigphase=0;
}
// Change the amount of delay
if (select == 3) {
// increase delay
if (digitalRead (18) == 0) {
if (tdelay == 500) tdelay = -1;
if (tdelay == 200) tdelay = 500;
if (tdelay == 100) tdelay = 200;
if (tdelay == 50) tdelay = 100;
if (tdelay == 20) tdelay = 50;
if (tdelay == 10) tdelay = 20;
if (tdelay == 5) tdelay = 10;
if (tdelay == 2) tdelay = 5;
if (tdelay == 1) tdelay = 2;
if (tdelay == 0) tdelay = 1;
if (tdelay == -1) tdelay = 0;
}
if (digitalRead (19) == 0) {
// decrease delay
if (tdelay == -1) tdelay = 500;
if (tdelay == 0) tdelay = -1;
if (tdelay == 1) tdelay = 0;
if (tdelay == 2) tdelay = 1;
if (tdelay == 5) tdelay = 2;
if (tdelay == 10) tdelay = 5;
if (tdelay == 20) tdelay = 10;
if (tdelay == 50) tdelay = 20;
if (tdelay == 100) tdelay = 50;
if (tdelay == 200) tdelay = 100;
if (tdelay == 500) tdelay = 200;
}
}
// Change the amount of gain
if (select == 4) {
if (digitalRead (18) == 0) gain++;
if (digitalRead (19) == 0) gain--;
if (gain > 5) gain=1;
if (gain < 1) gain=5;
}

// Update the text on the right side
// Set font color to bright blue
 TFTscreen.stroke (255, 100, 100);
// if selected set font color to red
if (select == 1) TFTscreen.stroke (0, 0, 255);
 TFTscreen.text ("Trg", 129, 3);
 TFTscreen.text (itoa(64 - trigger, buf, 10), 129, 18);
```

```
// Set font color to bright blue
 TFTscreen.stroke (255, 100, 100);
// if selected set font color to red
if (select == 2) FTscreen.stroke (0, 0, 255);
 TFTscreen.text ("Pha", 129, 35);
 TFTscreen.text (itoa(trigphase, buf, 10), 129, 50);
// Set font color to bright blue
 TFTscreen.stroke (255, 100, 100);
// if selected set color to red
if (select == 3) TFTscreen.stroke (0, 0, 255);
 TFTscreen.text ("Del", 129, 67);
 TFTscreen.text (itoa(tdelay, buf, 10), 129, 82);
// Set font color to bright blue
 TFTscreen.stroke (255, 100, 100);
// if selected set font color to red
if (select == 4) TFTscreen.stroke (0, 0, 255);
 TFTscreen.text ("Amp", 130, 98);
 TFTscreen.text (itoa(gain, buf, 10), 130, 113);
// display the sample time
 TFTscreen.stroke (0, 0, 255);
 TFTscreen.text ("Ms", 35, 113);
 TFTscreen.text (itoa(STime, buf, 10), 70, 113);
}
// End of program
```

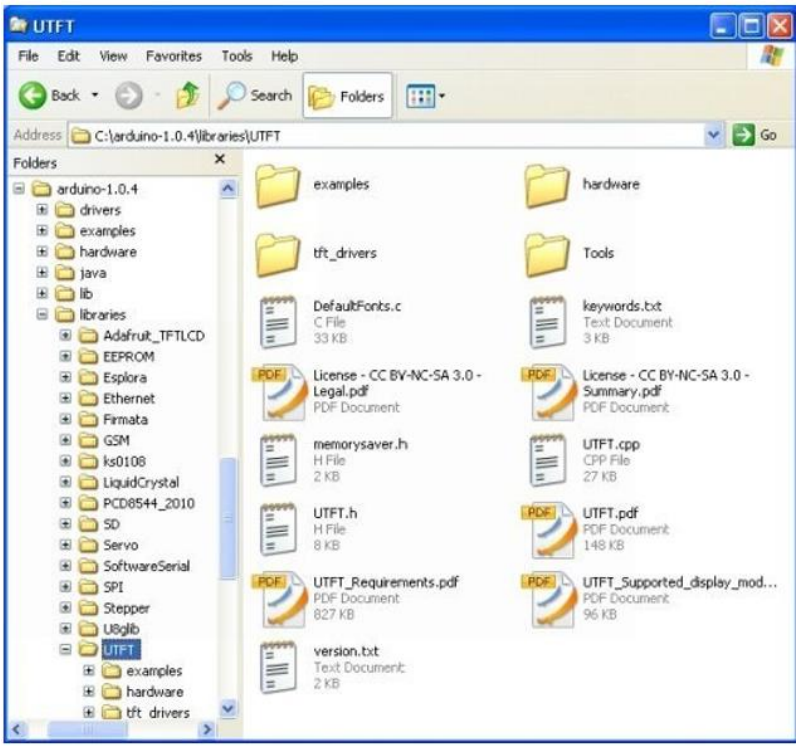
Глава 6. Параллельный цветной LCD TFT240_262K

При переходе к LCD с большим разрешением есть из чего выбрать. Размеры есть от 1.8 до 3.2 дюймов. И есть дисплеи ещё большего размера. Как выбрать, с каким LCD работать? Во-первых, взгляните на количество выводов разъёма LCD. Те, что больше, имеют двухрядный соединитель по 20 выводов в каждом ряду, поскольку дисплеи – 16-битовые устройства. Это не даст их включить в большинство макетных плат, и это слишком много выводов для Arduino Uno. Однако, если у вас Arduino Mega или тот модуль, что называют «Funduino», тогда у вас достаточно выводов для подключения 16-битового LCD.

Есть также некоторое количество LCD на eBay, которые уже имеют подключенный плоский кабель. Если у вас нет разъёма для соединения с этим кабелем или переходной платы, вы не сможете использовать этот тип LCD.

Один из дисплеев, о которых я говорил, это HY-TFT240_262K. Он имеет 18 выводной соединитель для LCD данных. Также есть соединители для сенсорного экрана и для дополнительного модуля памяти. Мы предполагаем пока использовать только 18 выводной соединитель. Некоторые из этих LCD имеют переключатель для питания 3.3 или 5 вольт. Убедитесь, что переключатели установлены правильно, прежде чем подавать питающее напряжение на LCD.

Для этого проекта мы будем использовать «UTFT library». Это означает «Universal Thin Film Transistor, универсальный тонкоплёночный транзистор» библиотеку. Эти дисплеи относятся к «TFT» или «Thin Film Transistor» LCD дисплеям. Они поддерживают три цветовых операции, равно как и высокое разрешение. Они были разработаны для использования в сотовых телефонах и MP4 плеерах. UTFT библиотека загружается как «UTFT.rar». Поскольку это «rar» файл, а не «zip» файл, вам понадобится программа для распаковки подобных файлов, например, «7-Zip». Вновь, вы выкладываете распакованный файл в вашу директорию «Arduino\Libraries». Всё должно выглядеть так, как на картинке ниже, если всё установлено правильно.



Вот список UTFT команд:

```
UTFT (Model, RS, WR, CS, RST, ALE);       InitLCD(Orientation);
getDisplayXSize();                        getDisplayYSize();
lcdOff();                                 lcdOn();
setContrast(c);                           clrScr();
fillScr(r,   g, b);                       fillScr(color);
 setColor(r, g, b);                       setColor(color);
getColor();                               setBackColor(r, g, b);
setBackColor(color);                      getBackColor();
drawPixel(x, y);                          drawLine(x1, y1, x2, y2);
drawRect(x1, y1, x2, y2);                 drawRoundRect(x1, y1, x2, y2);
fillRect(x1, y1, x2, y2);                 fillRoundRect(x1, y1, x2, y2);
drawCircle(x, y, radius);                 fillCircle(x, y, radius);
print(st, x, y, deg);
printNumI(num, x, y, length, filler);
printNumF(num, dec, x, y, divider, length, filler);
setFont(fontname);                        getFont();
getFontXsize();                           getFontYsize();
```

Ниже приведена таблица для установок некоторых из множества контроллеров, которые поддерживаются библиотекой UTFT. Заметьте, что многие из них позволяют работать в 8-битовой, 16-битовой версиях и с последовательной передачей данных. Arduino Uno можно использовать с 8-битовой и последовательной передачей данных.

Controller	Model for UTFT()	Supported mode		
		8bit	16bit	Serial
HX8340-B(N)	HX8340B_S			✓
HX8340-B(T)	HX8340B_8	✓		
HX8347-A	HX8347A		✓	
HX8352-A	HX8352A		✓	
ILI9320	ILI9320_8	✓		
	ILI9320_16		✓	
ILI9325C	ILI9325C	✓		
ILI9325D	ILI9325D_8	✓		
	ILI9325D_16		✓	
ILI9327	ILI9327		✓	
ILI9481	ILI9481		✓	
PCF8833	PCF8833			✓
S1D19122	S1D19122		✓	
S6D1121	S6D1121_8	✓		
	S6D1121_16		✓	
SSD1289	SSD1289		✓	
	SSD1289_8	✓		
	SSD1289LATCHED[6]		LATCHED	
SSD1963	SSD1963_480		✓	
	SSD1963_800		✓	
	SSD1963_800ALT[7]		✓	
ST7735	ST7735			✓

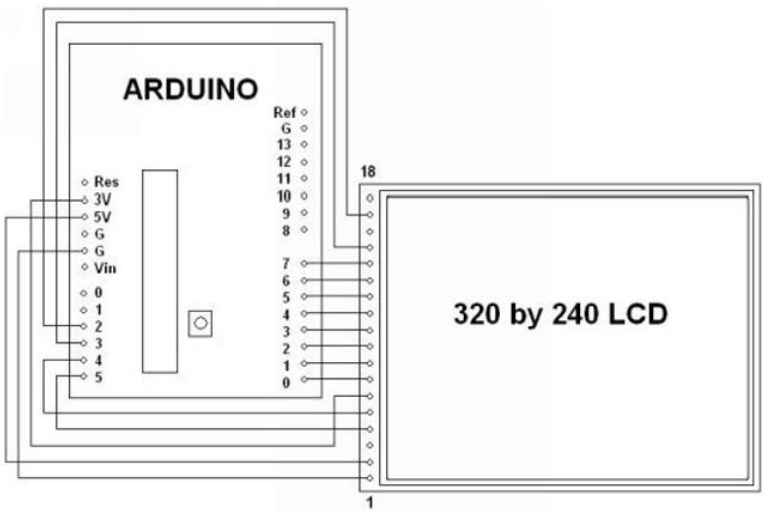
Согласно руководству к UTFT библиотеке вот то, как следует соединять поддерживаемые TFT LCD с Arduino. Заметьте, что DB0-DB7 становятся DB8-DB15 для 8-битового интерфейса. Этот список также показывает, как соединять 16-битовые устройства с Arduino Uno, но это использует все до последнего доступные выводы. Список выводов TFT не совпадает с нашим TFT дисплеем. Перечисленные подключения служат для 16-битовых дисплеев.

Signal	TFT pin	Arduino	
		2009/Uno/Leonardo	Mega/Due[2]
DB0[5]	21	D8	D37
DB1[5]	22	D9	D36
DB2[5]	23	D10	D35
DB3[5]	24	D11	D34
DB4[5]	25	D12	D33
DB5[5]	26	D13	D32
DB6[5]	27	A0 (D14)	D31
DB7[5]	28	A1 (D15)	D30
DB8	7	D0	D22
DB9	8	D1	D23
DB10	9	D2	D24
DB11	10	D3	D25
DB12	11	D4	D26
DB13	12	D5	D27
DB14	13	D6	D28
DB15	14	D7	D29
RS	4	Any free pin	
WR	5	Any free pin	
RD	6	Must be pulled high (3.3v)	
CS	15	Any free pin	
REST	17	Any free pin	

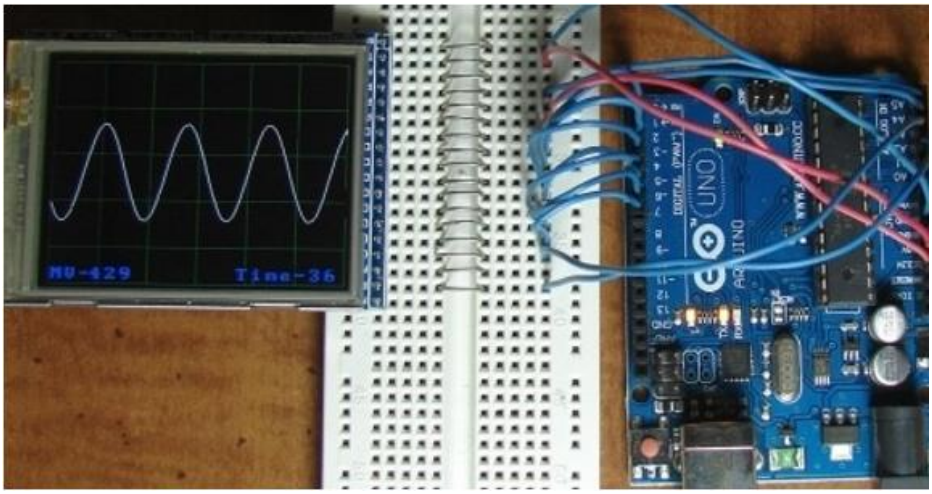
Посмотрите другой список для подключения 18 выводного LCD к Arduino Uno.

LCD	Name	Arduino
1	Gnd	Gnd
2	+5V	5V
3	NC	
4	RS	A5
5	RW	A4
6	RD	3.3V
7	DB0	D0
8	DB1	D1
9	DB2	D2
10	DB3	D3
11	DB4	D4
12	DB5	D5
13	DB6	D6
14	DB7	D7
15	CS	A3
16	NC	
17	Rst	A2
18	NC	

Вот и схема подключения LCD.



Далее картинка с работающим осциллографом. У него зелёные маркеры делений, синий текст, показывающий милливольты от пика до пика, и синий текст, показывающий время выборки. Выключатель, соединённый с A1, может переключать: ускорить выборки, когда подключён к земле, или замедлить выборки, когда подключён к 5В. Программный цикл триггера делает изображение достаточно стабильным. Схему с подключением аналогового входа можно найти в предыдущих главах этой книги.



Вот скетч для трёхцветного Arduino осциллографа.

```
//**************************************
// Three color Fast analog Oscilloscope
// By Bob Davis
// UTFT_(C)2012 Henning Karlsen
// web: http://www.henningkarlsen.com/electronics

#include <UTFT.h>
#ifdef cbi
#define cbi(sfr,  bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifdef sbi
#define sbi(sfr,  bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

// configure what fonts we will be using
```

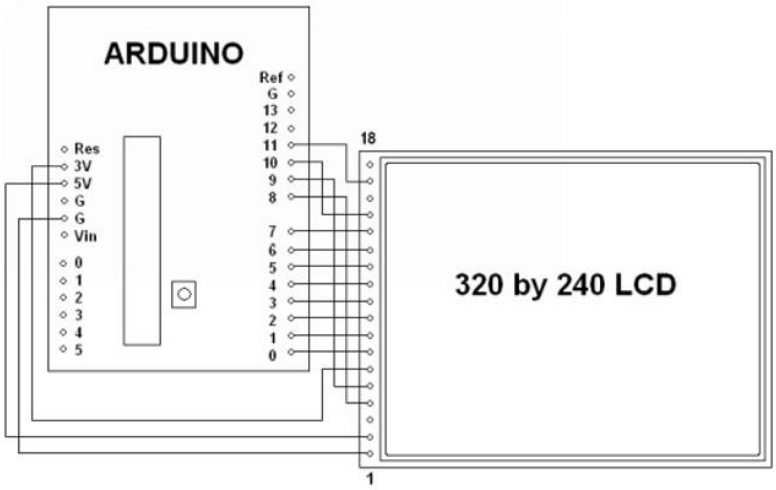
```
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];

UTFT myGLCD(ILI9325C,19,18,17,16);
int Input = 0;
int OldInput = 0;
int MaxSample = 0;
int MinSample = 0;
int Sample[320];
int OldSample[320];
int StartTime = 0;
int EndTime = 0;
int rate = 1;

void DrawMarkers() {
 myGLCD.setColor (0, 200, 0);
 myGLCD.drawLine (0, 0, 0, 240);
 myGLCD.drawLine (54,   0, 54, 240);
 myGLCD.drawLine (107, 0, 107, 240);
 myGLCD.drawLine (160, 0, 160, 240);
 myGLCD.drawLine (213, 0, 213, 240);
 myGLCD.drawLine (266, 0, 266, 240);
 myGLCD.drawLine (319, 0, 319, 240);
 myGLCD.drawLine (0, 0, 319, 0);
 myGLCD.drawLine (0, 50, 319, 50);
 myGLCD.drawLine (0, 100, 319, 100);
 myGLCD.drawLine (0, 150, 319, 150);
 myGLCD.drawLine (0, 200, 319, 200);
 myGLCD.drawLine (0, 239, 319, 239);
}

void setup() {
 myGLCD.InitLCD();
 myGLCD.clrScr();
 myGLCD.setBackColor (0, 0, 0);
 myGLCD.setFont(BigFont);
}

void loop() {
// set color (Red, Green, Blue) range = 0 to 255
char buf[12];
while(1) {
// Set sample speed according to switch on A1
if (analogRead(A1) < 500) {
 cbi(ADCSRA, ADPS2);
 rate = 0;
}
else {
 sbi(ADCSRA, ADPS2);
 rate = 1;
}
DrawMarkers();
// wait for a trigger of a positive going input
 OldInput = analogRead(A0);
```

```
 Input = analogRead(A0);
 while (Input < OldInput) {
 Input = analogRead(A0);
}
// collect the analog data into an array
// do not do division here, it makes it slower!
StartTime = micros();
for(int xpos=0; xpos<319; xpos++) {
 Sample[xpos] = analogRead(A0);
}
EndTime = micros();
// display the collected analog data from array
// Sample/4.1 because 1024 becomes 250 = 5 volts
for(int xpos=0; xpos<319; xpos++) {
 myGLCD.setColor(0, 0, 0);
 myGLCD.drawLine(xpos, OldSample[xpos]/4.1,    xpos+1,
OldSample[xpos+1]/4.1);
 myGLCD.setColor(255, 255, 255);
 myGLCD.drawLine(xpos, Sample[xpos]/4.1, xpos+1, Sample[xpos+1]/4.1);
}
// Determine sample voltage peak to peak
MaxSample = Sample[100];
MinSample = Sample[100];
for(int xpos=0; xpos<319; xpos++) {
 OldSample[xpos] = Sample[xpos];
if (Sample[xpos] > MaxSample) MaxSample = Sample[xpos];
if (Sample[xpos] < MinSample) MinSample = Sample[xpos];
}
// Display sample voltage and time
myGLCD.setColor(0, 0, 255);
int SampleSize = MaxSample - MinSample;
myGLCD.print("MV=", 1, 220);
myGLCD.print(itoa(SampleSize, buf, 10), 44, 220);
int SampleTime = EndTime - StartTime;
myGLCD.print("uS=", 161, 220);
// Adjust time according to sample speed
if (rate == 1) {
 SampleTime = (EndTime/1000 - StartTime/1000);
 myGLCD.print("mS=", 161, 220);
}
 myGLCD.print(itoa(SampleTime, buf, 10), 210, 220);
}
}
```

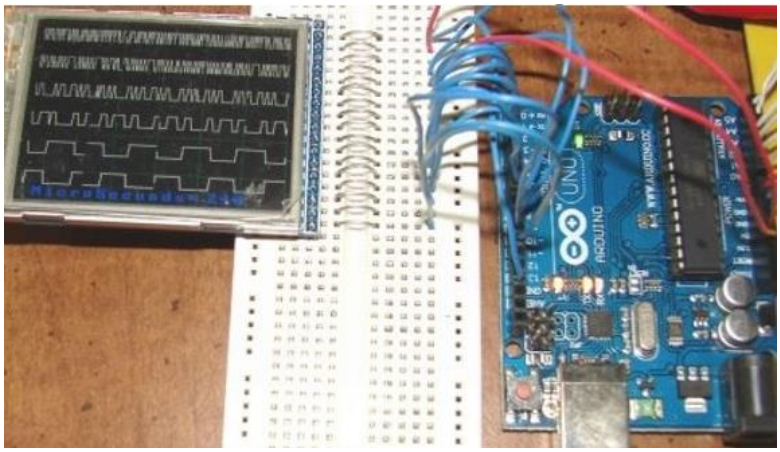
В следующем проекте я покажу вам, как сделать шестиканальный логический анализатор, использующий этот LCD дисплей. Вы можете использовать цикл для получения данных через «PINC» со скоростью до 1.33 миллиона выборок в секунду. Вы можете также использовать «Sample[1]=PINC, Sample[2]=PINC, Sample[3]=PINC» и т.д., и тогда вы достигнете скорости 5 миллионов выборок в секунду.

Вы можете удалить булево выражение под «// draw new data», заменив его таким «myGLCD.drawLine(xpos, Sample[xpos], xpos+1, Sample[xpos+1])», и вы получите осциллограф, если подключите быстрый АЦП типа «flash».

Ниже модифицированная схема с изменениями, которые вам нужны для освобождения всех входных аналоговых выводов. В основном провода, идущие к A2-A5, перекинуты на D8-D11.



А далее картинка шестиканального логического анализатора. Она показывает шесть выходов быстрого АЦП AD775 с 20 кГц входным синусоидальным сигналом на его входе.



Вот код скетча для логического анализатора.

```
//**************************************
// Three color 6   channel Logic Analyzer
// By Bob Davis
// UTFT_(C)2012 Henning Karlsen
// web:     http://www.henningkarlsen.com/electronics

#include <UTFT.h>
// Declare which fonts we will be using
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];

// Note that control pins are now assigned to pins D8-D11
UTFT myGLCD(ILI9325C,8,9,10,11);
int Input = 0;
int Sample[320];
int StartSample = 0;
int EndSample = 0;
```

```
void DrawMarkers() {
myGLCD.setColor(0, 200, 0);
myGLCD.drawLine(0, 0, 0, 240);
myGLCD.drawLine(54, 0, 54, 240);
myGLCD.drawLine(107, 0, 107, 240);
myGLCD.drawLine(160, 0, 160, 240);
myGLCD.drawLine(213, 0, 213, 240);
myGLCD.drawLine(266, 0, 266, 240);
myGLCD.drawLine(319, 0, 319, 240);
myGLCD.drawLine(0, 0, 319, 0);
myGLCD.drawLine(0, 50, 319, 50);
myGLCD.drawLine(0, 100, 319, 100);
myGLCD.drawLine(0, 150, 319, 150);
myGLCD.drawLine(0, 200, 319, 200);
myGLCD.drawLine(0, 239, 319, 239);
}

 void setup() {
 myGLCD.InitLCD();
 myGLCD.clrScr();
 pinMode(14, INPUT);
 pinMode(15, INPUT);
 pinMode(16, INPUT);
 pinMode(17, INPUT);
 pinMode(18, INPUT);
 pinMode(19, INPUT);
}

void loop() {
// set color(Red, Green,       Blue) range = 0 to 255
 myGLCD.setBackColor(0, 0, 0);
 myGLCD.setFont(BigFont);
 char buf[12];
while(1) {
 DrawMarkers();
// wait for trigger of a positive input
while (Input == 0) {
 Input = digitalRead(A0);
}
// collect the analog data into an array
// Read analog port as a parallel port PINC
StartSample = micros();
for(int xpos=0; xpos <319; xpos++) {
 Sample[xpos] = PINC;
 EndSample = micros();
// display the collected analog data from array
for(int xpos=0; xpos <319; xpos++) {
// Erase old stuff
 myGLCD.setColor(0, 0, 0);
 myGLCD.drawLine(xpos+1, 1, xpos+1, 220);
// Draw new data
myGLCD.setColor(255, 255, 255);
myGLCD.drawLine(xpos,((Sample[xpos]&B00000001)*16)+2,xpos+1,
((Sample[xpos+1]&B00000001)*16)+2);
```

```
myGLCD.drawLine(xpos,((Sample[xpos]&B00000010)*8)+42,xpos+1,
((Sample[xpos+1]&B00000010)*8)+42);
myGLCD.drawLine(xpos,((Sample[xpos]&B00000100)*4)+82,xpos+1,
((Sample[xpos+1]&B00000100)*4)+82);
myGLCD.drawLine(xpos,((Sample[xpos]&B00001000)*2)+122,xpos+1,
((Sample[xpos+1]&B00001000)*2)+122);
myGLCD.drawLine(xpos,((Sample[xpos]&B00010000)/1)+162,xpos+1,
((Sample[xpos+1]&B00010000)/1)+162);
myGLCD.drawLine(xpos,((Sample[xpos]&B00100000)/2)+202,xpos+1,
((Sample[xpos+1]&B00100000)/2)+202);
}
// display the sample time
 myGLCD.setColor(0, 0, 255);
 int SampleTime = EndSample - StartSample;
 myGLCD.print("MicroSeconds=", 10, 220);
 myGLCD.print(itoa(SampleTime, buf, 10), 224, 220);
}
}
```

Мы можем даже ускорить работу осциллографа, если добавим внешний АЦП, как, например, CA3306. Это «flash» конвертор, что означает, что он имеет 64 компаратора, которые мгновенно переводят аналоговый вход в цифровой выход. Благодаря этому, он может выполнять более 15 миллионов преобразований в секунду. Схему CA3306 можно найти в предыдущих главах этой книги.

Вот скетч для создания этого быстрого осциллографа. Заметьте, что добавлены две связанные кнопки без фиксации. Одна идёт к D12, другая к D13; их вторые концы подключены к земле. Выключатели позволяют выбирать «scan rate, скорость сканирования» и «trigger level, уровень переключения».

```
//**************************************
// Three color 5msps  external AtoD Scope
// By Bob Davis
// UTFT_(C)2012 Henning Karlsen
// web:     http://www.henningkarlsen.com/electronics
// Switches on D12 & D13 determine sweep speed and   trigger level
//**************************************

#include <UTFT.h>
// Declare which fonts we will be using
extern uint8_t SmallFont[];
extern uint8_t BigFont[];
extern uint8_t SevenSegNumFont[];

// Note that the control pins are now assigned to 8-11
UTFT myGLCD(ILI9325C,8,9,10,11);
int Input = 0;
byte Sample[320];
byte OldSample[320];
int StartSample = 0;
int EndSample = 0;
int MaxSample = 0;
int MinSample = 0;
int mode = 0;
```

```
int dTime = 1;
int Trigger = 10;
int SampleSize = 0;
int SampleTime = 0;

void DrawMarkers() {
 myGLCD.setColor(0, 220, 0);
 myGLCD.drawLine(0, 0, 0, 240);
 myGLCD.drawLine(60, 0, 60, 240);
 myGLCD.drawLine(120, 0, 120, 240);
 myGLCD.drawLine(180, 0, 180, 240);
 myGLCD.drawLine(239, 0, 239, 240);
 myGLCD.drawLine(319, 0, 319, 240);
 myGLCD.drawLine(0, 0, 319, 0);
 myGLCD.drawLine(0, 60, 319, 60);
 myGLCD.drawLine(0, 120, 319, 120);
 myGLCD.drawLine(0, 180, 319, 180);
 myGLCD.drawLine(0, 239, 319, 239);
}

void setup() {
myGLCD.InitLCD();
myGLCD.clrScr();
pinMode(12, INPUT);
digitalWrite(12, HIGH);
pinMode(13, INPUT);
digitalWrite(13, HIGH);
pinMode(14, INPUT);
pinMode(15, INPUT);
pinMode(16, INPUT);
pinMode(17, INPUT);
pinMode(18, INPUT);
pinMode(19, INPUT);
}

void loop() {
// Set the background color(Red, Green, Blue)
 myGLCD.setBackColor(0, 0, 0);
 myGLCD.setFont(BigFont);
 char buf[12];
while(1) {
 DrawMarkers();
if (digitalRead(13) == 0) mode++;
if (mode > 10) mode = 0;
// Select delay times for scan modes
if (mode == 0) dTime = 0;
if (mode == 1) dTime = 0;
if (mode == 2) dTime = 1;
if (mode == 3) dTime = 2;
if (mode == 4) dTime = 5;
if (mode == 5) dTime = 10;
if (mode == 6) dTime = 20;
if (mode == 7) dTime = 50;
if (mode == 8) dTime = 100;
if (mode == 9) dTime = 200;
```

```
if (mode == 10) dTime = 500;
// Select trigger level
if (digitalRead(12) == 0) Trigger = Trigger + 10;
if (Trigger > 50) Trigger = 0;
// Wait for input to be greater than trigger
while (Input < Trigger) {
Input = PINC;
}

// Collect the analog data into an array
if (mode == 0) {
// Read analog port as a parallel port no loop
StartSample = micros();
      Sample[0]=PINC;                    Sample[1]=PINC;
      Sample[2]=PINC;                    Sample[3]=PINC;
      Sample[4]=PINC;                    Sample[5]=PINC;
      Sample[6]=PINC;                    Sample[7]=PINC;
      Sample[8]=PINC;                    Sample[9]=PINC;
      Sample[10]=PINC;                   Sample[11]=PINC;
      Sample[12]=PINC;                   Sample[13]=PINC;
      Sample[14]=PINC;                   Sample[15]=PINC;
      Sample[16]=PINC;                   Sample[17]=PINC;
      Sample[18]=PINC;                   Sample[19]=PINC;
      Sample[20]=PINC;                   Sample[21]=PINC;
      Sample[22]=PINC;                   Sample[23]=PINC;
      Sample[24]=PINC;                   Sample[25]=PINC;
      Sample[26]=PINC;                   Sample[27]=PINC;
      Sample[28]=PINC;                   Sample[29]=PINC;
      Sample[30]=PINC;                   Sample[31]=PINC;
      Sample[32]=PINC;                   Sample[33]=PINC;
      Sample[34]=PINC;                   Sample[35]=PINC;
      Sample[36]=PINC;                   Sample[37]=PINC;
      Sample[38]=PINC;                   Sample[39]=PINC;
      Sample[40]=PINC;                   Sample[41]=PINC;
      Sample[42]=PINC;                   Sample[43]=PINC;
      Sample[44]=PINC;                   Sample[45]=PINC;
      Sample[46]=PINC;                   Sample[47]=PINC;
      Sample[48]=PINC;                   Sample[49]=PINC;
      Sample[50]=PINC;                   Sample[51]=PINC;
      Sample[52]=PINC;                   Sample[53]=PINC;
      Sample[54]=PINC;                   Sample[55]=PINC;
      Sample[56]=PINC;                   Sample[57]=PINC;
      Sample[58]=PINC;                   Sample[59]=PINC;
      Sample[60]=PINC;                   Sample[61]=PINC;
      Sample[62]=PINC;                   Sample[63]=PINC;
      Sample[64]=PINC;                   Sample[65]=PINC;
      Sample[66]=PINC;                   Sample[67]=PINC;
      Sample[68]=PINC;                   Sample[69]=PINC;
      Sample[70]=PINC;                   Sample[71]=PINC;
      Sample[72]=PINC;                   Sample[73]=PINC;
      Sample[74]=PINC;                   Sample[75]=PINC;
      Sample[76]=PINC;                   Sample[77]=PINC;
      Sample[78]=PINC;                   Sample[79]=PINC;
      Sample[80]=PINC;                   Sample[81]=PINC;
      Sample[82]=PINC;                   Sample[83]=PINC;
```

```
Sample[84]=PINC;                Sample[85]=PINC;
Sample[86]=PINC;                Sample[87]=PINC;
Sample[88]=PINC;                Sample[89]=PINC;
Sample[90]=PINC;                Sample[91]=PINC;
Sample[92]=PINC;                Sample[93]=PINC;
Sample[94]=PINC;                Sample[95]=PINC;
Sample[96]=PINC;                Sample[97]=PINC;
Sample[98]=PINC;                Sample[99]=PINC;
Sample[100]=PINC;               Sample[101]=PINC;
Sample[102]=PINC;               Sample[103]=PINC;
Sample[104]=PINC;               Sample[105]=PINC;
Sample[106]=PINC;               Sample[107]=PINC;
Sample[108]=PINC;               Sample[109]=PINC;
Sample[110]=PINC;               Sample[111]=PINC;
Sample[112]=PINC;               Sample[113]=PINC;
Sample[114]=PINC;               Sample[115]=PINC;
Sample[116]=PINC;               Sample[117]=PINC;
Sample[118]=PINC;               Sample[119]=PINC;
Sample[120]=PINC;               Sample[121]=PINC;
Sample[122]=PINC;               Sample[123]=PINC;
Sample[124]=PINC;               Sample[125]=PINC;
Sample[126]=PINC;               Sample[127]=PINC;
Sample[128]=PINC;               Sample[129]=PINC;
Sample[130]=PINC;               Sample[131]=PINC;
Sample[132]=PINC;               Sample[133]=PINC;
Sample[134]=PINC;               Sample[135]=PINC;
Sample[136]=PINC;               Sample[137]=PINC;
Sample[138]=PINC;               Sample[139]=PINC;
Sample[140]=PINC;               Sample[141]=PINC;
Sample[142]=PINC;               Sample[143]=PINC;
Sample[144]=PINC;               Sample[145]=PINC;
Sample[146]=PINC;               Sample[147]=PINC;
Sample[148]=PINC;               Sample[149]=PINC;
Sample[150]=PINC;               Sample[151]=PINC;
Sample[152]=PINC;               Sample[153]=PINC;
Sample[154]=PINC;               Sample[155]=PINC;
Sample[156]=PINC;               Sample[157]=PINC;
Sample[158]=PINC;               Sample[159]=PINC;
Sample[160]=PINC;               Sample[161]=PINC;
Sample[162]=PINC;               Sample[163]=PINC;
Sample[164]=PINC;               Sample[165]=PINC;
Sample[166]=PINC;               Sample[167]=PINC;
Sample[168]=PINC;               Sample[169]=PINC;
Sample[170]=PINC;               Sample[171]=PINC;
Sample[172]=PINC;               Sample[173]=PINC;
Sample[174]=PINC;               Sample[175]=PINC;
Sample[176]=PINC;               Sample[177]=PINC;
Sample[178]=PINC;               Sample[179]=PINC;
Sample[180]=PINC;               Sample[181]=PINC;
Sample[182]=PINC;               Sample[183]=PINC;
Sample[184]=PINC;               Sample[185]=PINC;
Sample[186]=PINC;               Sample[187]=PINC;
Sample[188]=PINC;               Sample[189]=PINC;
Sample[190]=PINC;               Sample[191]=PINC;
Sample[192]=PINC;               Sample[193]=PINC;
```

```
      Sample[194]=PINC;                  Sample[195]=PINC;
      Sample[196]=PINC;                  Sample[197]=PINC;
      Sample[198]=PINC;                  Sample[199]=PINC;
      Sample[200]=PINC;                  Sample[201]=PINC;
      Sample[202]=PINC;                  Sample[203]=PINC;
      Sample[204]=PINC;                  Sample[205]=PINC;
      Sample[206]=PINC;                  Sample[207]=PINC;
      Sample[208]=PINC;                  Sample[209]=PINC;
      Sample[210]=PINC;                  Sample[211]=PINC;
      Sample[212]=PINC;                  Sample[213]=PINC;
      Sample[214]=PINC;                  Sample[215]=PINC;
      Sample[216]=PINC;                  Sample[217]=PINC;
      Sample[218]=PINC;                  Sample[219]=PINC;
      Sample[220]=PINC;                  Sample[221]=PINC;
      Sample[222]=PINC;                  Sample[223]=PINC;
      Sample[224]=PINC;                  Sample[225]=PINC;
      Sample[226]=PINC;                  Sample[227]=PINC;
      Sample[228]=PINC;                  Sample[229]=PINC;
      Sample[230]=PINC;                  Sample[231]=PINC;
      Sample[232]=PINC;                  Sample[233]=PINC;
      Sample[234]=PINC;                  Sample[235]=PINC;
      Sample[236]=PINC;                  Sample[237]=PINC;
      Sample[238]=PINC;                  Sample[239]=PINC;
      Sample[240]=PINC;
EndSample = micros();
}
if (mode == 1) {
// Read analog port as a parallel port with no delay
 StartSample = micros();
for(int xpos=0; xpos <240; xpos++) {
 Sample[xpos]=PINC;
}
EndSample = micros();
}
if (mode >= 2) {
// Read analog port as a parallel port variable delay
 StartSample = micros();
for(int xpos=0; xpos <240; xpos++) {
 Sample[xpos] = PINC;
 delayMicroseconds(dTime);
}
EndSample = micros();
}

// Display the collected analog data from array
for(int xpos=0; xpos <239; xpos++) {
// Erase the old stuff
 myGLCD.setColor(0, 0, 0);
 myGLCD.drawLine(xpos+1,255-OldSample[xpos+1]*4,xpos+2,
255-OldSample[xpos+2]*4);
if (xpos == 0) myGLCD.drawLine(xpos+1,1,xpos+1,239);
// Draw the new data
 myGLCD.setColor(255, 255, 255);
 myGLCD.drawLine(xpos, 255-Sample[xpos]*4,xpos+1,
255-Sample[xpos+1]*4);
```

```
}

//Determine sample voltage peak to peak
 MaxSample = Sample[100];
 MinSample = Sample[100];
for(int xpos = 0; xpos <240; xpos++) {
 OldSample[xpos] = Sample[xpos];
if (Sample[xpos] > MaxSample) MaxSample=Sample[xpos];
if (Sample[xpos] < MinSample) MinSample = Sample[xpos];
}
// display the sample time, delay time and trigger level
 myGLCD.setColor(0, 0, 255);
 SampleTime = EndSample - StartSample;
 myGLCD.print("uSec.", 240, 10);
 myGLCD.print("      ", 240, 30);
 myGLCD.print(itoa(SampleTime, buf, 10), 240, 30);
 myGLCD.print("Delay", 240, 70);
 myGLCD.print("     ", 240, 90);
 myGLCD.print(itoa(dTime, buf, 10), 240, 90);
 myGLCD.print("Trig.", 240, 130);
 myGLCD.print(itoa(Trigger, buf, 10), 240, 150);
// Range of 0 to 64 * 78 = 4992    mV
 SampleSize=(MaxSample-MinSample)*78;
 myGLCD.print("mVolt", 240, 190);
 myGLCD.print(itoa(SampleSize, buf, 10), 240, 210);
}
}
// end of program
```

Глава 7. Последовательный цветной LCD - 2.2 до 2.8, SPI

Мой следующий LCD для этой книги – последовательный LCD на базе драйвера ILI9341. Эти дисплеи 249x320 пикселей, но всё ещё используют последовательный интерфейс, так что они не нуждаются в большом количестве выводов Arduino. К сожалению, эти LCD требуют либо 3, либо 2.5-вольтовой логики. Эти логические уровни могут быть легко получены, если использовать два резистора по 1 кОм, чтобы поделить 5-вольтовую логику, превратив её в 2.5-вольтовую. Вы можете также использовать резисторы 1 кОм и 2.2 кОм, получив 3-вольтовый логический уровень. Я пытался использовать один 1 кОм или 2.2 кОм резистор последовательно, но LCD в таком варианте не работал.

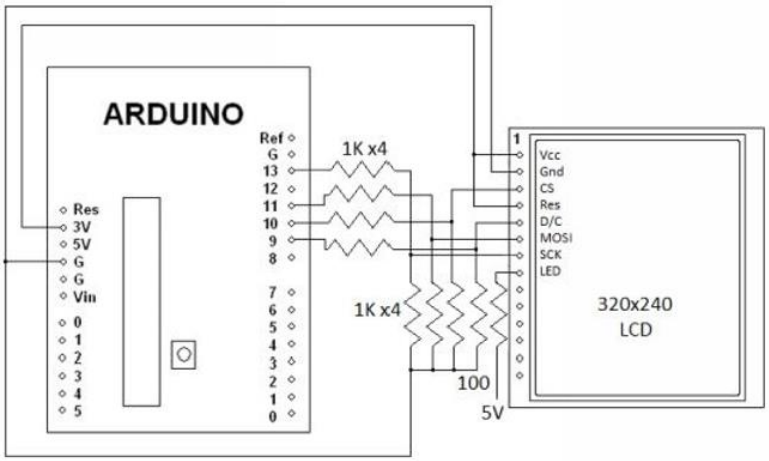
Adafruit версия этого LCD имеет микросхему 4050, которая конвертирует логический уровень 5 вольт в 3 вольта. В Интернете есть также схема с 4050 в качестве интерфейса LCD.

Когда я впервые включил подсветку LCD, он был подключён к напряжению 3.3 вольта. Через несколько минут работы стабилизатор 3.3 вольт у Arduino начал выключаться. Вместо этого лучше подключать подсветку LED к 5 вольтам через резистор 100 Ом. В любом случае удостоверьтесь, что есть последовательно включённый резистор, поскольку в LCD нет токоограничительного резистора.

Ниже вы видите, какие выводы используются для подключение этого LCD.

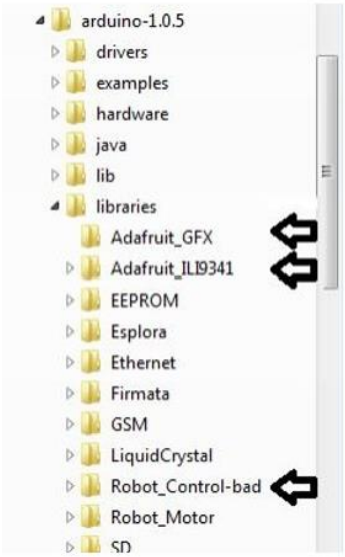
1. VCC	Connect to 3.3 volts
2. Ground	Connect to ground
3. CS	Connect to D10 via resistor divider
4. Reset	Connect to 3.3 volts
5. D/C	Connect to D9 via resistor divider
6. MOSI/SDI	Connect to D11 via resistor divider
7. SCK	Connect to D13 via resistor divider
8. LED	Connect to 5V through 100 ohm resistor
9. MISO	Not needed unless you have a touch screen.

Далее схема, показывающая делитель с 1 кОм для получения логических уровней 2.5 вольт.



Adafruit имеет самый лучший программный драйвер для использования с этим LCD. Вы должны будете загрузить, распаковать и переименовать оба драйвера от Adafruit и для ILI9341, и для графики (GFX).

Затем, когда вы сделаете всё это, вы можете получить сообщение об ошибке с «Robot Control» драйверами при попытке загрузить вашу программу в Arduino. Я знаю, что это не имеет значения, но, если это случится, вы можете просто переименовать мешающую директорию «Robot_Control» так, чтобы она перестала быть «valid». Это приведёт к сообщению об ошибке при загрузке интерфейса Arduino, но вы можете игнорировать его. Вы можете видеть все эти изменения в директории библиотек Arduino, которые вам нужны, на следующей картинке.



Также драйвер Adafruit слегка медлит. Кто-то написал более быструю версию, но она существуют под тем же именем, так что, трудно их различить. Но быстрая версия работает почти вдвое быстрее!

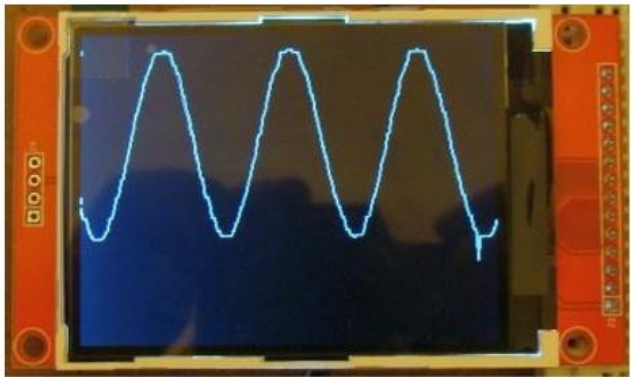
Вот список некоторых команд, поддерживающих этот драйвер.

```
tft.drawLine (x1, y1, x2, y2, Color);
tft.drawPixel (x, y, Color);
tft.setCursor (x, y);
tft.println ("text");
tft.setTextColor (ILI9341_Color);
tft.setTextSize (2);
tft.begin();
tft.fillScreen (ILI9341_Color);
tft.setRotation(1);
```

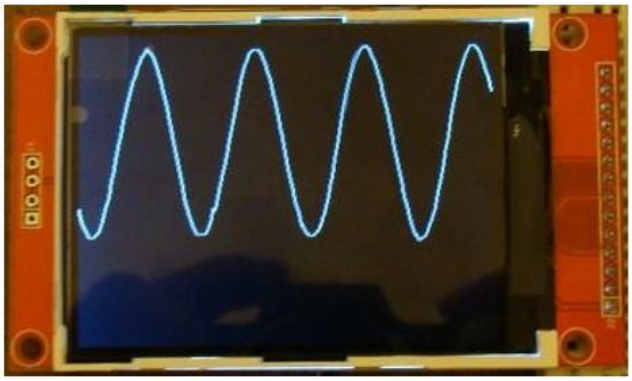
А это список цветов, включённых в драйвер.

RED	GREEN	BLUE	BLACK
YELLOW	WHITE	CYAN	BRIGHT_RED
GRAY1	GRAY2		

Ниже картинка базового осциллографа, который использует встроенный АЦП, работающий в 16 раз медленнее относительно его нормальной скорости. Программное ускорение приводит к некоторой неточности, что заметно в верхней и нижней части изображения. Вот синусоидальный сигнал 1000 Гц. Без ускоряющей программы сигнал 100 Гц будет выглядеть почти идентично, но будет чуть равномернее.



Ниже картинка синусоидального сигнала 100 Гц с конвертором, работающим на нормальной скорости. Отметьте плавность синусоиды.

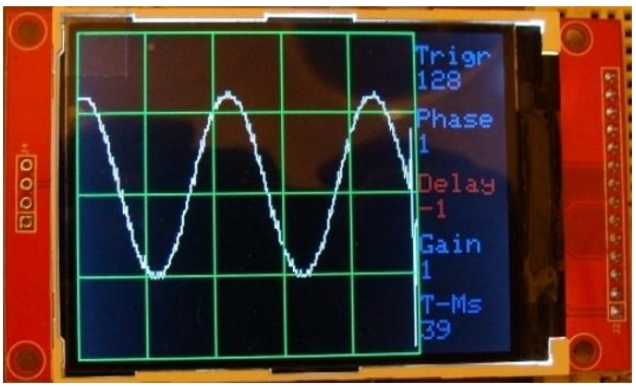


Далее листинг программы для простого осциллографа, использующего внутренний АЦП.

```
/***********************************
2.8 SPI A0 TFT Oscope Simple
Reads the A0 analog input,
and shows the value on the screen.
Created      27 July 2015 by Bob Davis
***********************************/
#include <SPI.h>
#include "Adafruit_ILI9341.h"
#include "Adafruit_GFX.h"

// pin definition for the Uno LCD
#define TFT_DC    9
#define      TFT_CS     10
// Use hardware SPI (on Uno, #13=clk, #11=mosi) and the above for CS/DC
//Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
Adafruit_ILI9341 tft = Adafruit_ILI9341();

//// set up variables
int Input = 0;
byte Sample[320];
int trigger=64;

void setup() {
// initialize rotate and clear the display
 tft.begin();
 tft.fillScreen(ILI9341_BLACK);
 tft.setRotation(1);
```

```
// Set the font size
 tft.setTextSize(2);
}

void loop() {
// Speed up the converter.
// Clear bit 2 of ADC pre-scalier from 125KHz to 2 MHz
 ADCSRA &= ~(1    <<ADPS2);
// wait for a positive going trigger
for (int timeout = 0; timeout < 1000; timeout++) {
 Input = analogRead(A0);
if (Input < trigger) break;
}
for (int timeout = 0; timeout < 1000; timeout++) {
 Input = analogRead(A0);
if (Input > trigger) break;
}
// quickly collect the data with no delay
for (int xpos=0; xpos <320; xpos++) {
 Sample[xpos] = analogRead(A0);
}
// display the collected data
for (int xpos=0; xpos <319; xpos++) {
// erase the old and draw new line
 tft.drawLine(xpos+1, 0, xpos+1, 240, ILI9341_BLACK);
 tft.drawLine(xpos, (Sample[xpos]*2), xpos+1, Sample[xpos+1]*2,
ILI9341_WHITE);
}
}
// End of program
```

А это картинка LCD, работающего с программой осциллографа, который использует внешний АЦП.



Вот код для осциллографа с внешним 8-битовым АЦП. Эта программа также использует четыре выключателя без фиксации, подключенных к A3-A5 с общим проводом, идущим на землю. Выбранное значение подсвечивается красным, тогда правая и левая кнопки меняют его величину.

```
/**********************************
2.8 SPI PIND TFT Oscilloscope
Reads the D0-D7 pins using PIND,
and shows the value on the screen.
```

```
Created 7 July 2015 by Bob Davis
************************************/
 //#include <TFT.h>  // Arduino LCD library
#include <SPI.h>
#include "Adafruit_ILI9341.h"
#include "Adafruit_GFX.h"

// pin definitions for the ILI9341 2.8 LCD
#define TFT_DC    9
#define TFT_CS    10
// Use hardware SPI (on Uno, #13=clk, #12=nc, #11=mosi) and the above for
CS/DC
// Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
Adafruit_ILI9341 tft = Adafruit_ILI9341();

// set up the variables
int xPos = 0;
char buf[12];
int Input = 0;
byte Sample[250];
int StartSTime = 0;
int EndSTime = 0;
int stime = 0;
int trigger = 128;
int trigphase = 1;
int tdelay = 1;
int select = 1;
int gain = 1;

void setup() {
// Initialize the display
 tft.begin();
// Rotate and clear the screen
 tft.fillScreen(ILI9341_BLACK);
 tft.setRotation(1);
// Set the font size
 tft.setTextSize(2);
// A to D input pins
      pinMode(0, INPUT); pinMode(1, INPUT); pinMode(2, INPUT);
      pinMode(3, INPUT); pinMode(4, INPUT); pinMode(5, INPUT);
      pinMode(6, INPUT); pinMode(7, INPUT);
// Push button switches
      pinMode(16, INPUT_PULLUP); pinMode(17, INPUT_PULLUP);
      pinMode(18, INPUT_PULLUP); pinMode(19, INPUT_PULLUP); }

void loop() {
// wait for a positive going trigger
if (trigphase == 1) {
for (int timeout = 0; timeout < 1000; timeout++) {
 Input = PIND;
if (Input < trigger) break;
}
for (int timeout = 0; timeout < 1000; timeout++) {
 Input = PIND;
if (Input > trigger) break;
```

```
}
}
// wait for a negative going trigger
if (trigphase == 0) {
for (int timeout = 0; timeout < 1000; timeout++) {
 Input = PIND;
if (Input > trigger) break;
}
for (int timeout = 0; timeout < 1000; timeout++) {
 Input = PIND;
if (Input < trigger) break;
}
}
// quickly collect the data with no delay
if (tdelay == -1) {
StartSTime = micros();
      Sample[0]=PIND;                    Sample[1]=PIND;
      Sample[2]=PIND;                    Sample[3]=PIND;
      Sample[4]=PIND;                    Sample[5]=PIND;
      Sample[6]=PIND;                    Sample[7]=PIND;
      Sample[8]=PIND;                    Sample[9]=PIND;
      Sample[10]=PIND;                   Sample[11]=PIND;
      Sample[12]=PIND;                   Sample[13]=PIND;
      Sample[14]=PIND;                   Sample[15]=PIND;
      Sample[16]=PIND;                   Sample[17]=PIND;
      Sample[18]=PIND;                   Sample[19]=PIND;
      Sample[20]=PIND;                   Sample[21]=PIND;
      Sample[22]=PIND;                   Sample[23]=PIND;
      Sample[24]=PIND;                   Sample[25]=PIND;
      Sample[26]=PIND;                   Sample[27]=PIND;
      Sample[28]=PIND;                   Sample[29]=PIND;
      Sample[30]=PIND;                   Sample[31]=PIND;
      Sample[32]=PIND;                   Sample[33]=PIND;
      Sample[34]=PIND;                   Sample[35]=PIND;
      Sample[36]=PIND;                   Sample[37]=PIND;
      Sample[38]=PIND;                   Sample[39]=PIND;
      Sample[40]=PIND;                   Sample[41]=PIND;
      Sample[42]=PIND;                   Sample[43]=PIND;
      Sample[44]=PIND;                   Sample[45]=PIND;
      Sample[46]=PIND;                   Sample[47]=PIND;
      Sample[48]=PIND;                   Sample[49]=PIND;
      Sample[50]=PIND;                   Sample[51]=PIND;
      Sample[52]=PIND;                   Sample[53]=PIND;
      Sample[54]=PIND;                   Sample[55]=PIND;
      Sample[56]=PIND;                   Sample[57]=PIND;
      Sample[58]=PIND;                   Sample[59]=PIND;
      Sample[60]=PIND;                   Sample[61]=PIND;
      Sample[62]=PIND;                   Sample[63]=PIND;
      Sample[64]=PIND;                   Sample[65]=PIND;
      Sample[66]=PIND;                   Sample[67]=PIND;
      Sample[68]=PIND;                   Sample[69]=PIND;
      Sample[70]=PIND;                   Sample[71]=PIND;
      Sample[72]=PIND;                   Sample[73]=PIND;
      Sample[74]=PIND;                   Sample[75]=PIND;
      Sample[76]=PIND;                   Sample[77]=PIND;
```

```
Sample[78]=PIND;                Sample[79]=PIND;
Sample[80]=PIND;                Sample[81]=PIND;
Sample[82]=PIND;                Sample[83]=PIND;
Sample[84]=PIND;                Sample[85]=PIND;
Sample[86]=PIND;                Sample[87]=PIND;
Sample[88]=PIND;                Sample[89]=PIND;
Sample[90]=PIND;                Sample[91]=PIND;
Sample[92]=PIND;                Sample[93]=PIND;
Sample[94]=PIND;                Sample[95]=PIND;
Sample[96]=PIND;                Sample[97]=PIND;
Sample[98]=PIND;                Sample[99]=PIND;
Sample[100]=PIND;               Sample[101]=PIND;
Sample[102]=PIND;               Sample[103]=PIND;
Sample[104]=PIND;               Sample[105]=PIND;
Sample[106]=PIND;               Sample[107]=PIND;
Sample[108]=PIND;               Sample[109]=PIND;
Sample[110]=PIND;               Sample[111]=PIND;
Sample[112]=PIND;               Sample[113]=PIND;
Sample[114]=PIND;               Sample[115]=PIND;
Sample[116]=PIND;               Sample[117]=PIND;
Sample[118]=PIND;               Sample[119]=PIND;
Sample[120]=PIND;               Sample[121]=PIND;
Sample[122]=PIND;               Sample[123]=PIND;
Sample[124]=PIND;               Sample[125]=PIND;
Sample[126]=PIND;               Sample[127]=PIND;
Sample[128]=PIND;               Sample[129]=PIND;
Sample[130]=PIND;               Sample[131]=PIND;
Sample[132]=PIND;               Sample[133]=PIND;
Sample[134]=PIND;               Sample[135]=PIND;
Sample[136]=PIND;               Sample[137]=PIND;
Sample[138]=PIND;               Sample[139]=PIND;
Sample[140]=PIND;               Sample[141]=PIND;
Sample[142]=PIND;               Sample[143]=PIND;
Sample[144]=PIND;               Sample[145]=PIND;
Sample[146]=PIND;               Sample[147]=PIND;
Sample[148]=PIND;               Sample[149]=PIND;
Sample[150]=PIND;               Sample[151]=PIND;
Sample[152]=PIND;               Sample[153]=PIND;
Sample[154]=PIND;               Sample[155]=PIND;
Sample[156]=PIND;               Sample[157]=PIND;
Sample[158]=PIND;               Sample[159]=PIND;
Sample[160]=PIND;               Sample[161]=PIND;
Sample[162]=PIND;               Sample[163]=PIND;
Sample[164]=PIND;               Sample[165]=PIND;
Sample[166]=PIND;               Sample[167]=PIND;
Sample[168]=PIND;               Sample[169]=PIND;
Sample[170]=PIND;               Sample[171]=PIND;
Sample[172]=PIND;               Sample[173]=PIND;
Sample[174]=PIND;               Sample[175]=PIND;
Sample[176]=PIND;               Sample[177]=PIND;
Sample[178]=PIND;               Sample[179]=PIND;
Sample[180]=PIND;               Sample[181]=PIND;
Sample[182]=PIND;               Sample[183]=PIND;
Sample[184]=PIND;               Sample[185]=PIND;
Sample[186]=PIND;               Sample[187]=PIND;
```

```
      Sample[188]=PIND;                  Sample[189]=PIND;
      Sample[190]=PIND;                  Sample[191]=PIND;
      Sample[192]=PIND;                  Sample[193]=PIND;
      Sample[194]=PIND;                  Sample[195]=PIND;
      Sample[196]=PIND;                  Sample[197]=PIND;
      Sample[198]=PIND;                  Sample[199]=PIND;
      Sample[200]=PIND;                  Sample[201]=PIND;
      Sample[202]=PIND;                  Sample[203]=PIND;
      Sample[204]=PIND;                  Sample[205]=PIND;
      Sample[206]=PIND;                  Sample[207]=PIND;
      Sample[208]=PIND;                  Sample[209]=PIND;
      Sample[210]=PIND;                  Sample[211]=PIND;
      Sample[212]=PIND;                  Sample[213]=PIND;
      Sample[214]=PIND;                  Sample[215]=PIND;
      Sample[216]=PIND;                  Sample[217]=PIND;
      Sample[218]=PIND;                  Sample[219]=PIND;
      Sample[220]=PIND;                  Sample[221]=PIND;
      Sample[222]=PIND;                  Sample[223]=PIND;
      Sample[224]=PIND;                  Sample[225]=PIND;
      Sample[226]=PIND;                  Sample[227]=PIND;
      Sample[228]=PIND;                  Sample[229]=PIND;
      Sample[230]=PIND;                  Sample[231]=PIND;
      Sample[232]=PIND;                  Sample[233]=PIND;
      Sample[234]=PIND;                  Sample[235]=PIND;
      Sample[236]=PIND;                  Sample[237]=PIND;
      Sample[238]=PIND;                  Sample[239]=PIND;
      Sample[240]=PIND;                  Sample[241]=PIND;
      Sample[242]=PIND;                  Sample[243]=PIND;
      Sample[244]=PIND;                  Sample[245]=PIND;
      Sample[246]=PIND;                  Sample[247]=PIND;
      Sample[248]=PIND;                  Sample[249]=PIND;
      Sample[250]=PIND;
EndSTime = micros();
}
// Collect the data with a no delay
// It will not allow a delay of 0 so this is here to do that
if (tdelay == 0) {
 StartSTime = micros();
for (int xpos=0; xpos <250; xpos++) {
 Sample[xpos]=PIND;
}
 EndSTime = micros();
}
// Collect the data with a variable delay
if (tdelay > 0) {
 StartSTime = micros();
for (int xpos=0; xpos <250; xpos++) {
 Sample[xpos]=PIND;
 delayMicroseconds(tdelay);
}
 EndSTime = micros();
}
stime = EndSTime - StartSTime;
// fix a bug in mode -1 that the displayed time is not correct
if (tdelay == -1) stime = 49;
```

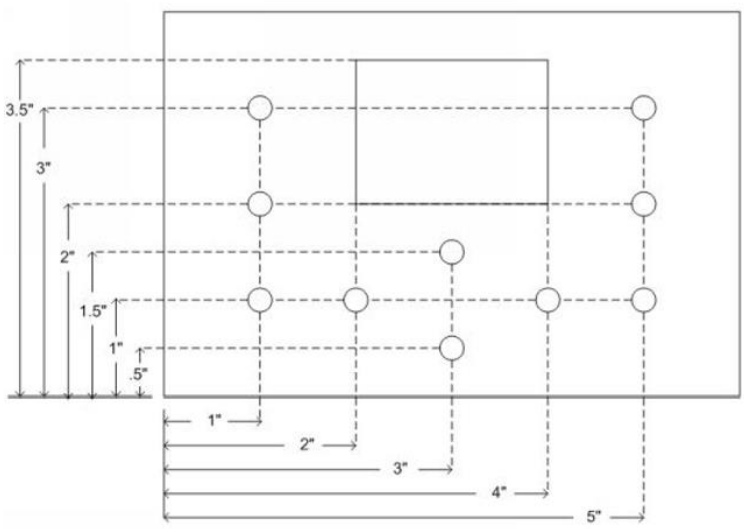
```
// display the collected data
for (int xpos = 0; xpos <320; xpos++) {
// erase the old line
 tft.drawLine(xpos+1, 0, xpos+1, 240, ILI9341_BLACK);
// draw the trace line (xPos1, yPos1, xPos2, yPos2, color);
// 256-inverts the data so it is right side up
if (xpos<250) {
if (gain==1) {
 tft.drawLine(xpos, (256-Sample[xpos]),  xpos+1,    256-Sample[xpos+1],
ILI9341_WHITE);
}
if (gain == 2) {
 tft.drawLine(xpos, (384-Sample[xpos]*2), xpos+1, 384-Sample[xpos+1]*2,
ILI9341_WHITE);
}
if (gain == 3) {
 tft.drawLine(xpos, (512-Sample[xpos]*3), xpos+1, 512-Sample[xpos+1]*3,
ILI9341_WHITE);
}
if (gain == 4) {
 tft.drawLine(xpos, (640-Sample[xpos]*4), xpos+1, 640-Sample[xpos+1]*4,
ILI9341_WHITE);
}
if (gain == 5) {
 tft.drawLine(xpos, (768-Sample[xpos]*5), xpos+1, 768-Sample[xpos+1]*5,
ILI9341_WHITE);
}
}
// draw the horizontal green lines as dots
if (xpos<250) {
 tft.drawPixel(xpos, 0, ILI9341_GREEN);
 tft.drawPixel(xpos, 60, ILI9341_GREEN);
 tft.drawPixel(xpos, 120, ILI9341_GREEN);
 tft.drawPixel(xpos, 180, ILI9341_GREEN);
 tft.drawPixel(xpos, 239, ILI9341_GREEN);
}
// draw top to bottom green lines
if (xpos == 0) tft.drawLine(xpos, 0, xpos, 240, ILI9341_GREEN);
if (xpos == 50) tft.drawLine(xpos, 0, xpos, 240, ILI9341_GREEN);
if (xpos == 100) tft.drawLine(xpos, 0, xpos, 240, ILI9341_GREEN);
if (xpos == 150) tft.drawLine(xpos, 0, xpos, 240, ILI9341_GREEN);
if (xpos == 200) tft.drawLine(xpos, 0, xpos, 240, ILI9341_GREEN);
if (xpos == 250) tft.drawLine(xpos, 0, xpos, 240, ILI9341_GREEN);
}

//*** check the status of push button switches
// They are now circular, so only 2 switches are needed
if (digitalRead(17) == 0) select++;
if (digitalRead(16) == 0) select--;
if (select > 4) select = 1;
if (select < 1) select = 4;
// update the trigger level
if (select == 1) {
if (digitalRead(18) == 0) trigger++;
if (digitalRead(19) == 0) trigger--;
```

```
if (trigger > 240) trigger = 1;
if (trigger < 1) trigger = 240;
}
// change the trigger phase
if (select == 2) {
if (digitalRead(18) == 0) trigphase++;
if (digitalRead(19) == 0) trigphase--;
if (trigphase < 1) trigphase = 1;
if (trigphase > 1) trigphase = 0;
}
// Change the amount of delay
if (select == 3) {
// increase delay
if (digitalRead(18) == 0) {
if (tdelay == 500) tdelay = -1;
if (tdelay == 200) tdelay = 500;
if (tdelay == 100) tdelay = 200;
if (tdelay == 50) tdelay = 100;
if (tdelay == 20) tdelay = 50;
if (tdelay == 10) tdelay = 20;
if (tdelay == 5) tdelay = 10;
if (tdelay == 2) tdelay = 5;
if (tdelay == 1) tdelay = 2;
if (tdelay == 0) tdelay = 1;
if (tdelay == -1) tdelay = 0;
}
if (digitalRead(19) == 0) {
// decrease delay
if (tdelay == -1) tdelay = 500;
if (tdelay == 0) tdelay = -1;
if (tdelay == 1) tdelay = 0;
if (tdelay == 2) tdelay = 1;
if (tdelay == 5) tdelay = 2;
if (tdelay == 10) tdelay = 5;
if (tdelay == 20) tdelay = 10;
if (tdelay == 50) tdelay = 20;
if (tdelay == 100) tdelay = 50;
if (tdelay == 200) tdelay = 100;
if (tdelay == 500) tdelay = 200;
}
}
// Change the amount of gain
if (select == 4) {
if (digitalRead(18) == 0) gain++;
if (digitalRead(19) == 0) gain--;
if (gain > 5) gain = 1;
if (gain < 1) gain=5;
}

//*** Update the text on the right side
 tft.setTextColor(ILI9341_BLUE);
// if selected set font color to red
if (select == 1) tft.setTextColor(ILI9341_RED);
 tft.setCursor(252, 10);
 tft.println("Trigr");
```

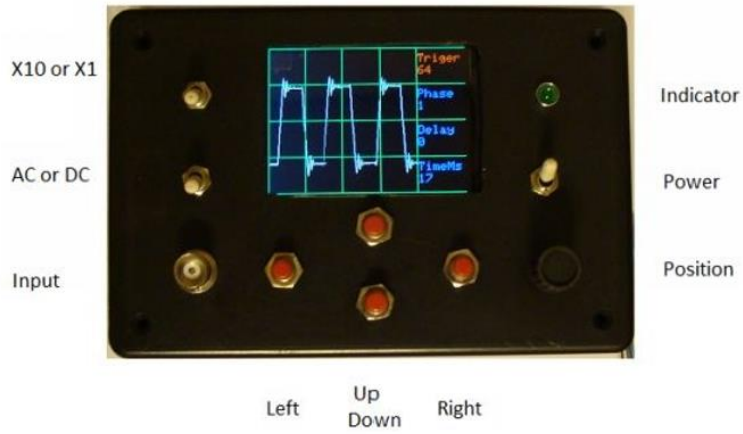
```
 tft.setCursor(252, 29);
 tft.println(itoa(128-trigger, buf, 10));
// if selected set font color to red
 tft.setTextColor(ILI9341_BLUE);
// if selected set font color to red
if (select == 2) tft.setTextColor(ILI9341_RED);
 tft.setCursor(252, 58);
 tft.println("Phase");
 tft.setCursor(252, 77);
 tft.println(itoa(trigphase, buf, 10));
// if selected set color to red
 tft.setTextColor(ILI9341_BLUE);
// if selected set font color to red
if (select == 3) tft.setTextColor(ILI9341_RED);
 tft.setCursor(252, 106);
 tft.println("Delay");
 tft.setCursor(252, 125);
 tft.println(itoa(tdelay, buf, 10));
 tft.setTextColor(ILI9341_BLUE);
// if selected set font color to red
if (select == 4) tft.setTextColor(ILI9341_RED);
 tft.setCursor(252, 152);
 tft.println("Gain");
 tft.setCursor(252, 171);
 tft.println(itoa(gain, buf, 10));
 tft.setTextColor(ILI9341_BLUE);
// if selected set font color to red
if (select == 5) tft.setTextColor(ILI9341_RED);
 tft.setCursor(252, 196);
 tft.println("T-Ms");
 tft.setCursor(252, 215);
 tft.println(itoa(stime, buf, 10));
}
// End of program
```

Глава 8. Помещаем осциллограф в корпус

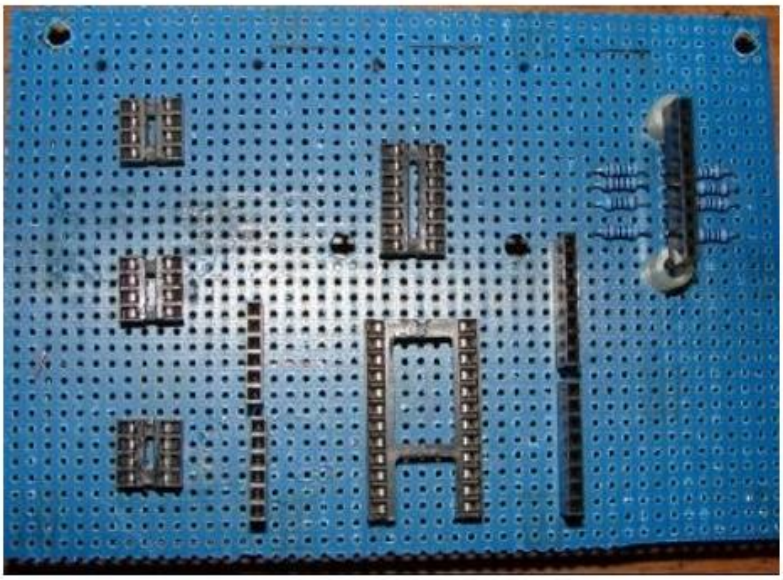
Если вы подошли со своей любимой разработкой к этой главе, то пришло время поместить осциллограф на базе Arduino в корпус. Я использовал пластиковый корпус 6”x4”x2” из радиомагазина. Я остановился на внешнем АЦП TLC5510. Вот внешний вид пластикового корпуса. Показанное окно для LCD предназначено для 2.8-дюймового дисплея. Вам понадобится это окно приспособить к вашему дисплею. Другие отверстия имеют ¼ дюйма в диаметре. Некоторые выключатели потребуют чуть больших отверстий.



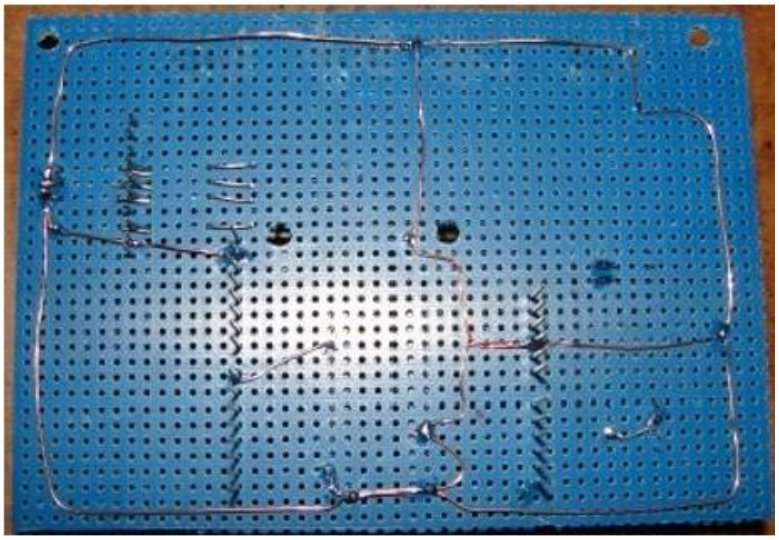
Вот картинка передней панели осциллографа с надписями.



Постарайтесь соединить всё на плате такими короткими проводами, какими это будет возможно. Ниже картинка того, как это сделано у меня. Отметьте две ¼ дюймовые распорки, чтобы приподнять соединитель LCD чуть выше.

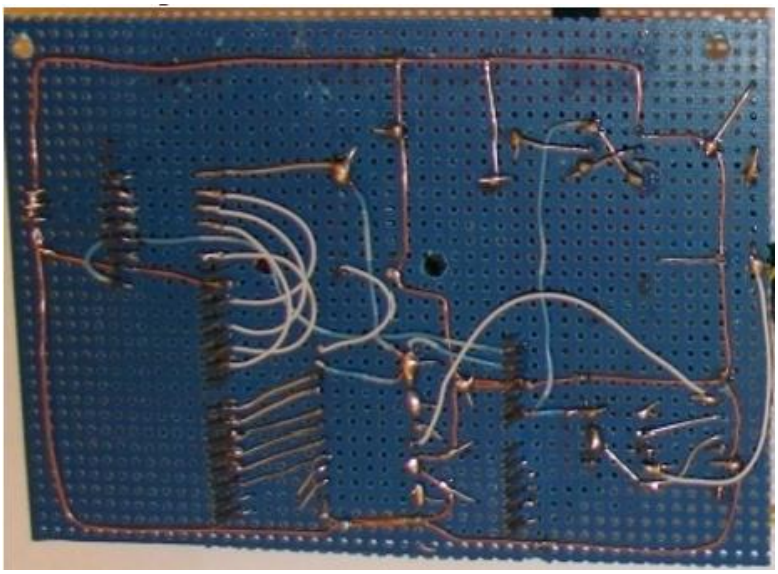


Первое, что вам следует соединить – это все точки, требующие соединения с землёй. Они должны быть «избыточны», поскольку соединяются в кольцо. Заземляющие провода должны быть выполнены проводом калибра 24-20.

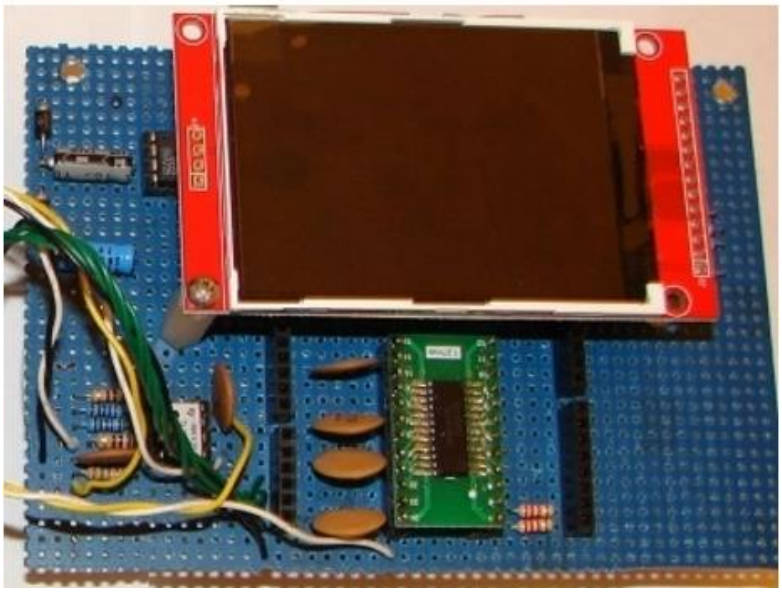


Затем соедините все провода питания – это +5 и +9 вольт. Они тоже должны быть выполнены проводом калибра 20-24, но в этот раз используйте изолированный провод. Затем выполните короткие прямые соединения, которые можно выполнить проводом без изоляции, как, например, соединение TLC5510 с выводами D0-D7. Также соедините все выводы, которые замыкаются. Затем проведите оставшиеся соединения, цепь за цепью, и убедитесь, что монтаж выполнен полностью.

Ниже картинка полностью смонтированной платы. Она выглядит относительно простой. Передняя панель снята при фотографировании этой платы.



А вот картинка верхней части собранной платы. Провода слева идут к передней панели осциллографа.



Библиография

Programming Arduino
Getting Started With Sketches
By Simon Mark
Copyright 2012 by the McGraw-Hill Companies

Эта книга даёт подробное объяснение программного кода для Arduino. Однако проекты в книге самые базовые. Есть LCD и Ethernet адаптеры.

Getting Started with Arduino
By Massimo Banzi
Copyright 2011 Massimo Banzi

Автор со-основатель проекта Arduino. Это книга – быстрое введение в программирование с несколькими простыми проектами.

Arduino Cookbook
by Michael Margolis
Copyright © 2011 Michael Margolis and Nicholas Weldin. All rights reserved.
Printed in the United States of America.
Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA.

В этой книге много больших проектов, каждый из которых очень хорошо объясняется.

Practical Arduino: Cool Projects for Open Source Hardware
Copyright © 2009 by Jonathan Oxer and Hugh Blemings
ISBN-13 (pbk): 978-1-4302-2477-8
ISBN-13 (electronic): 978-1-4302-2478-5
Printed and bound in the United States of America

На странице 197 этой книги есть то, как перегрузить АЦП для ускорения выборок в осциллографе и логическом анализаторе.