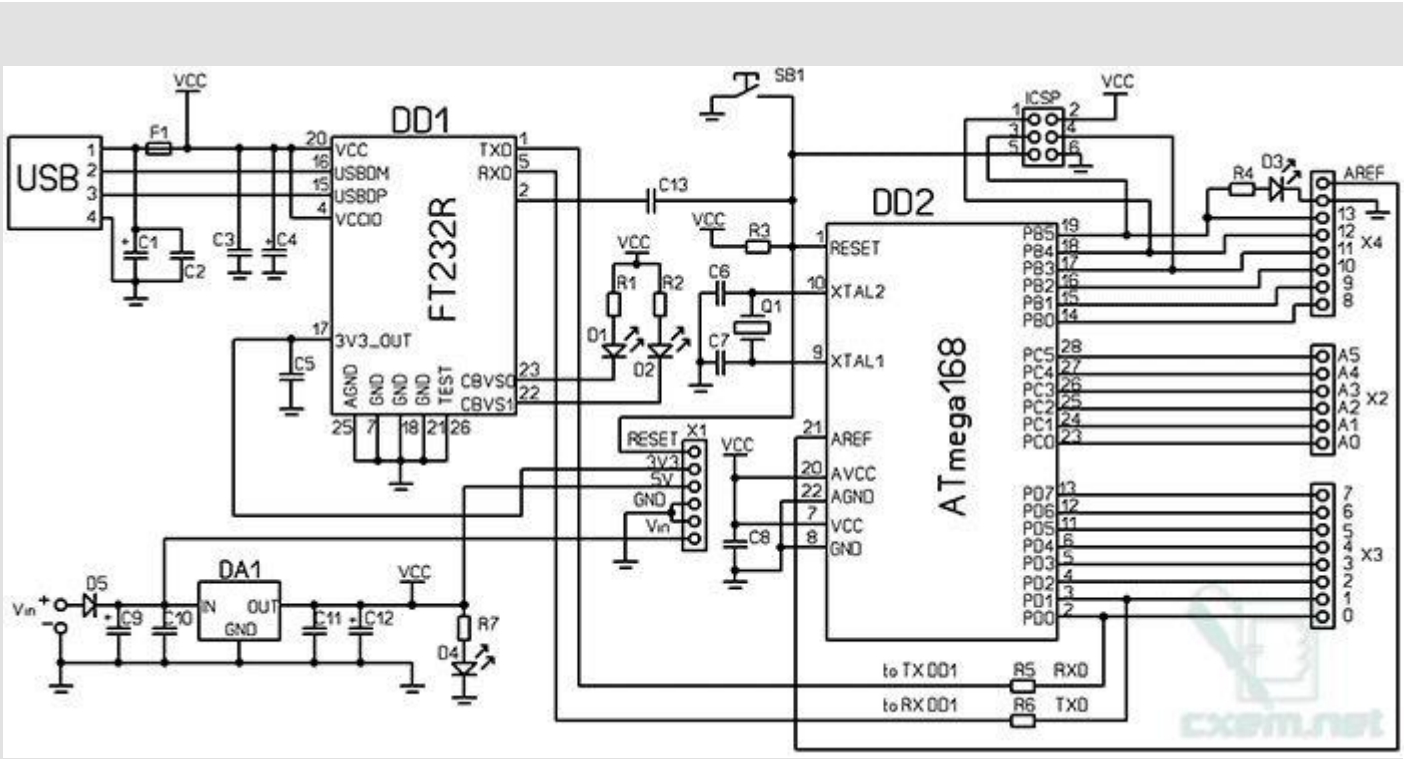
alex-day › Блог › Arduino без Arduino: работаем с микроконтроллерами напрямую

alex-day был 10 часов назад
42 года
Я ездил на BMW 3 series E21 Бэшка
Киев, Украина

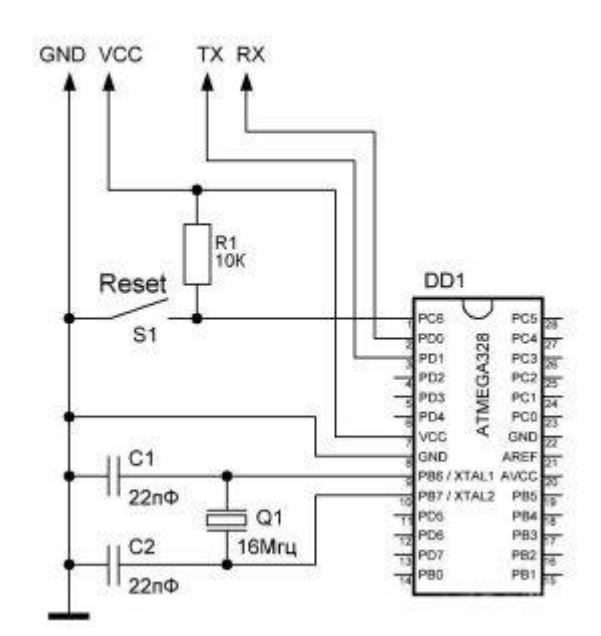
Если вспомнить историю создания Arduino (www.drive2.ru/b/2520138/), то Arduino стало популярно благодаря трем вещам, составляющим ее основу: Среды программирования Arduino IDE (на самом деле это среда языка Processing), Языка программирования Wiring (На самом деле такого языка не существует — то что мы видим это самый обычный C, дополненный большим числом библиотек) и Плат Arduino.
Я уже писал ранее, что без каждой из этих трех составляющих можно обойтись и приводил пример того, как можно обойтись без знания C — www.drive2.ru/b/2729013/. Как отказаться от Arduino IDE написано здесь — www.visualmicro.com/page/…what_is_visual_micro.html, а сегодня я хотел бы написать о том, как отказаться от "плат Arduino".
Итак, что же собой представляет плата, получившая такой коммерческий успех?



Как можно увидеть на плате находятся микроконтроллер AtMega 168 или 328, микросхема питания — DA1, контроллер виртуального com порта — DD1 и кварц 16 МГц — Q1. В общем то на первый взгляд ничего лишнего, но это только на первый: Используемая микросхема питания позволяет питать плату от напряжения от 5 до 12В или кратковременно до 30В, т.е. для авто с его 14,5В не пригодна и нужно делать свой источник питания. Контроллер COM порта используется в основном только для заливки программ и не является обязательным (в плате Arduino Pro Micro и ей подобных

он отсутствует). Кварц, несомненно, позволяет точно работать с временем, но если погрешность в несколько милисикунд для вас не критична, то можно вспомнить о том, что микроконтроллеры фирмы Atmel, к которым относится и Atmega168/328, содержат внутренний кварц и могут отсчитывать такты сами себе.
Так что же эта плата лишняя? В общем то да. В большинстве случаев без нее действительно можно обойтись и сейчас мы поговорим как.

Поддержка средой программирования
Находим где у вас установлена Arduino и открываем папочку hardware
По умолчанию это здесь — *C:\Program Files\Arduino\hardware\arduino\avr*
В эту папку мы будем распаковывать архивы с библиотеками, которые будем качать отсюда:
1) Для микроконтроллеров
ATmega8, ATmega8A,
ATmega88, ATmega88A, ATmega88P, ATmega88PA, ATmega88PB
ATmega168, ATmega168A, ATmega168P, ATmega168PA, ATmega168PB
ATmega328, ATmega328P, ATmega328PB
ATmega48, ATmega48A, ATmega48P, ATmega48PA, ATmega48PB
~~качаем ATmega8 Series (8/48/88/168/328) отсюда — github.com/sleemanj/optib…ob/master/dists/README.md~~
См. отдельную статью О бедном AtMega замолвите слово
Данные библиотеки позволяют запустить МК на 3 частотах: 1MHz, 8MHz или 16MHz (Для работы требуется внешний кварц 16МГц).
Тут необходимо понимать, что внешний кварц увеличивает быстродействие и стабильность работы (1 миллисекунда выполнения программы всегда будет равняться 1 миллисекунде реального времени), но увеличивает, пусть и ненамного, стоимость конструкции и снижает надежность за счет большего числа деталей. Лично мое мнение, что для большинства конструкций, проектируемых для автомобиля, можно смело обойтись и встроенным кварцем. Для схем зажигания, тахометра можно использовать внешний кварц, подключенный по схеме ниже, но дешевле взять готовую платку типа Arduino Pro Micro.
Достаточно ценное замечание от *alexfrance*
Был печальный опыт при использовании внутреннего генератора МК тини2313. На морозе при -20 контроллеры зависали, глючили. Установка внешнего кварца помогла. Поскольку внутренний генератор представляет из себя RC цепь, то он очень термозависим

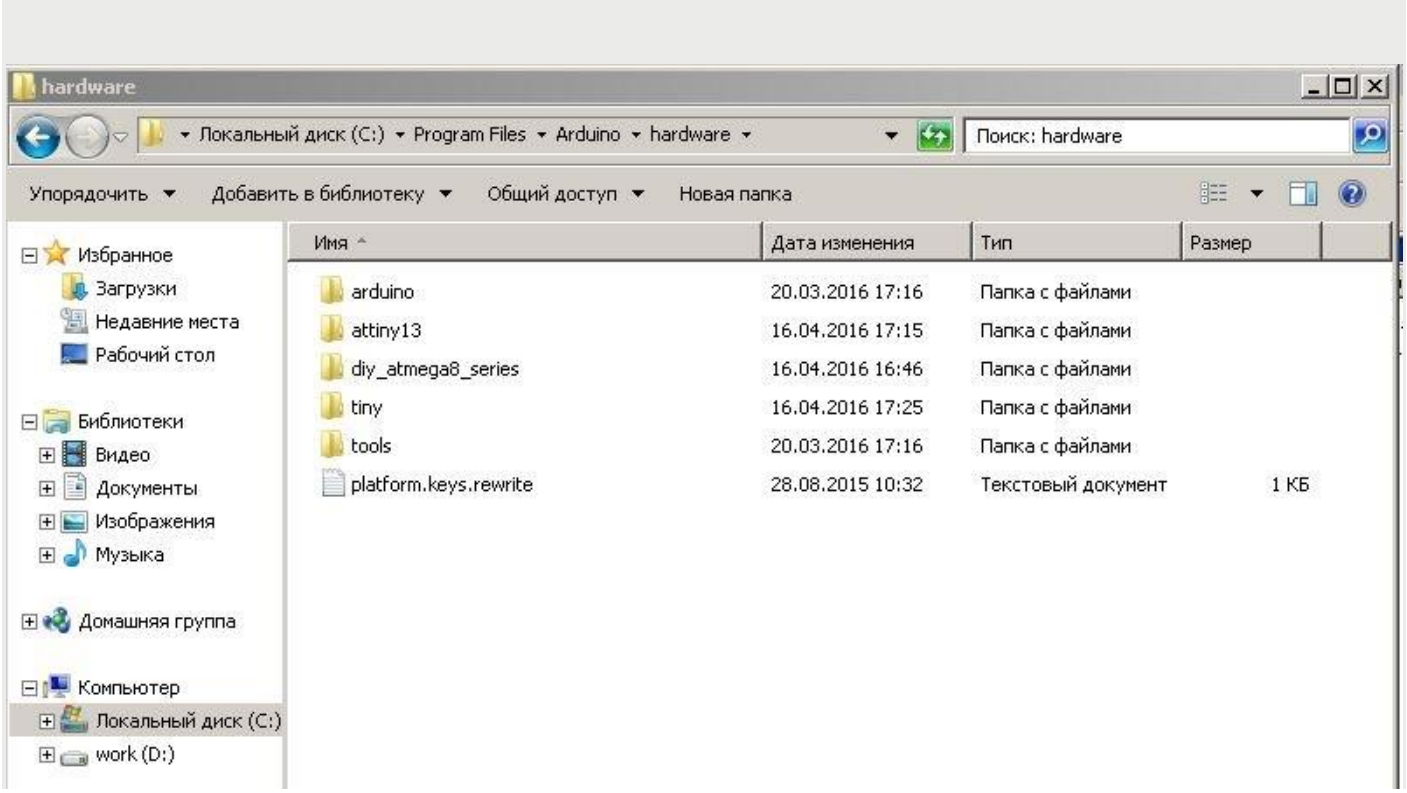


2) Микроконтроллеры ***Attiny13 (A)***
Библиотеки и файлы для поддержки "Тинек" можно скачать по ссылочке выше, а можно взять

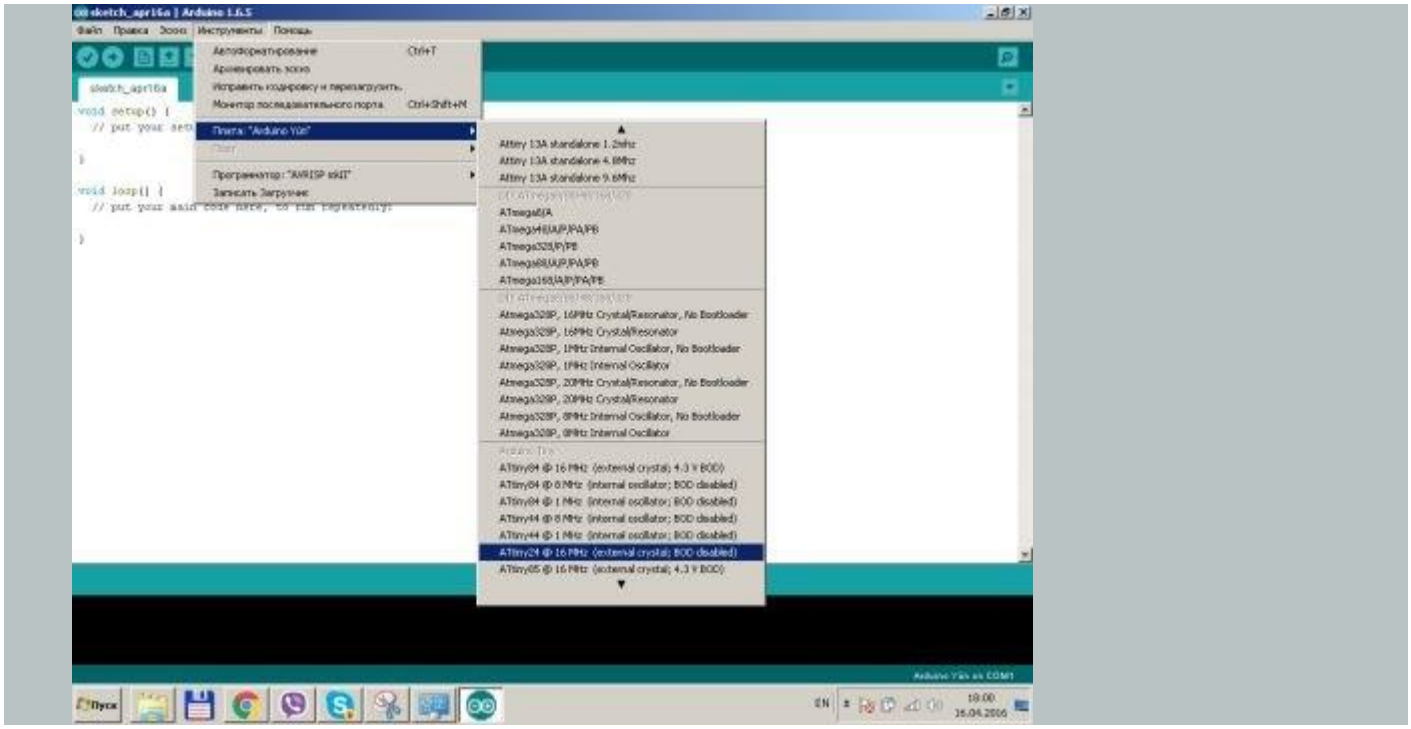
версию от разработчика — sourceforge.net/projects/ard-core13/files/
Скачанный файл также кладем в папку hardware

3) Для микроконтроллеров
ATtiny84, ATtiny44, ATtiny24,
ATtiny85, ATtiny45, ATtiny25,
ATtiny2313, ATtiny4313
ссылка для скачивания -code.google.com/p/arduino-tiny/
После распаковки заходим в папку tiny и переименовываем файл C:\Program Files\Arduino\hardware\tiny\avr\Prospective Boards.txt в C:\Program Files\Arduino\hardware\tiny\avr\Boards.txt

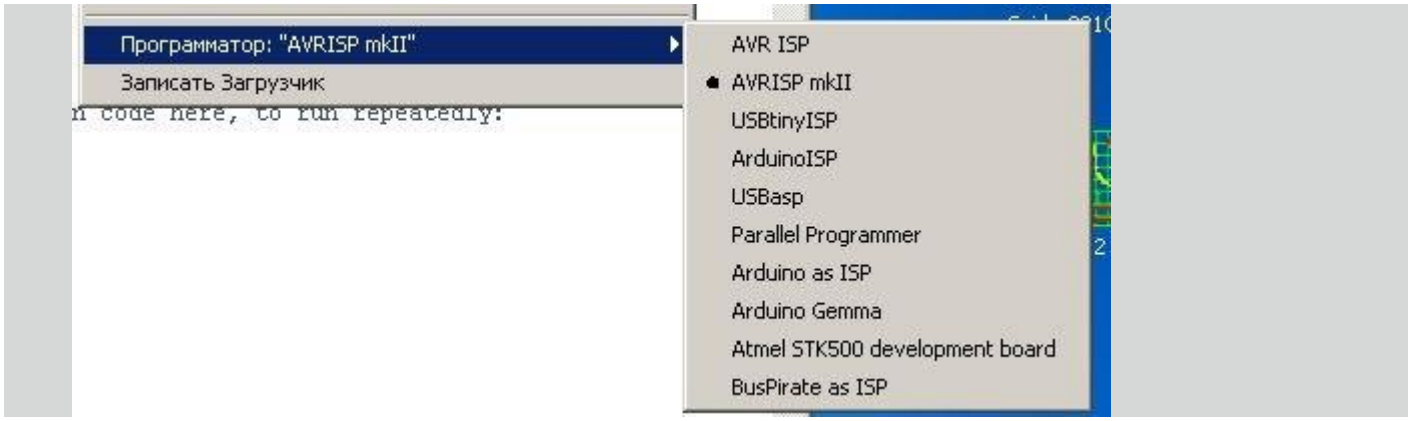
После всех скачиваний и распаковок получим папку hardware с таким содержимым



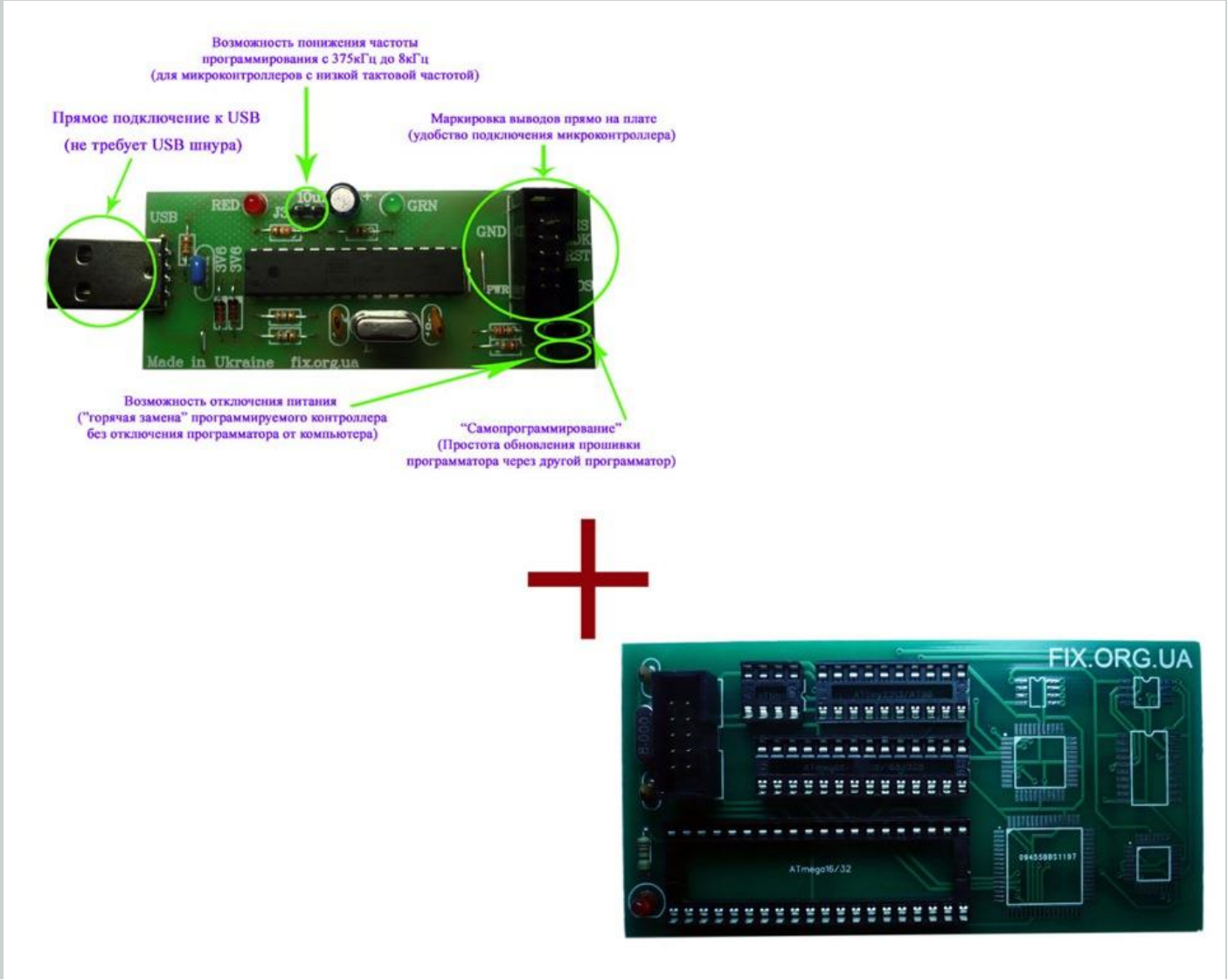
А запустив Arduino IDE увидим:



Подключение микроконтроллера
Прошить программы в МК можно 2 различными способами:
1) Классический вариант — прошивка при помощи программатора
Тут все просто: покупаем любой программатор из списка поддерживаемых



и вперед
Наиболее распространенным является USBasp, он производится активно китайцами, его несложно сделать и самому. В Украине могу порекомендовать производителя с таким наборчиком (увы, снят уже с производства, ребята делают только плату адаптора) — fix.org.ua/index.php/%D0%…%D1%80%D0%BE%D0%BC-detail

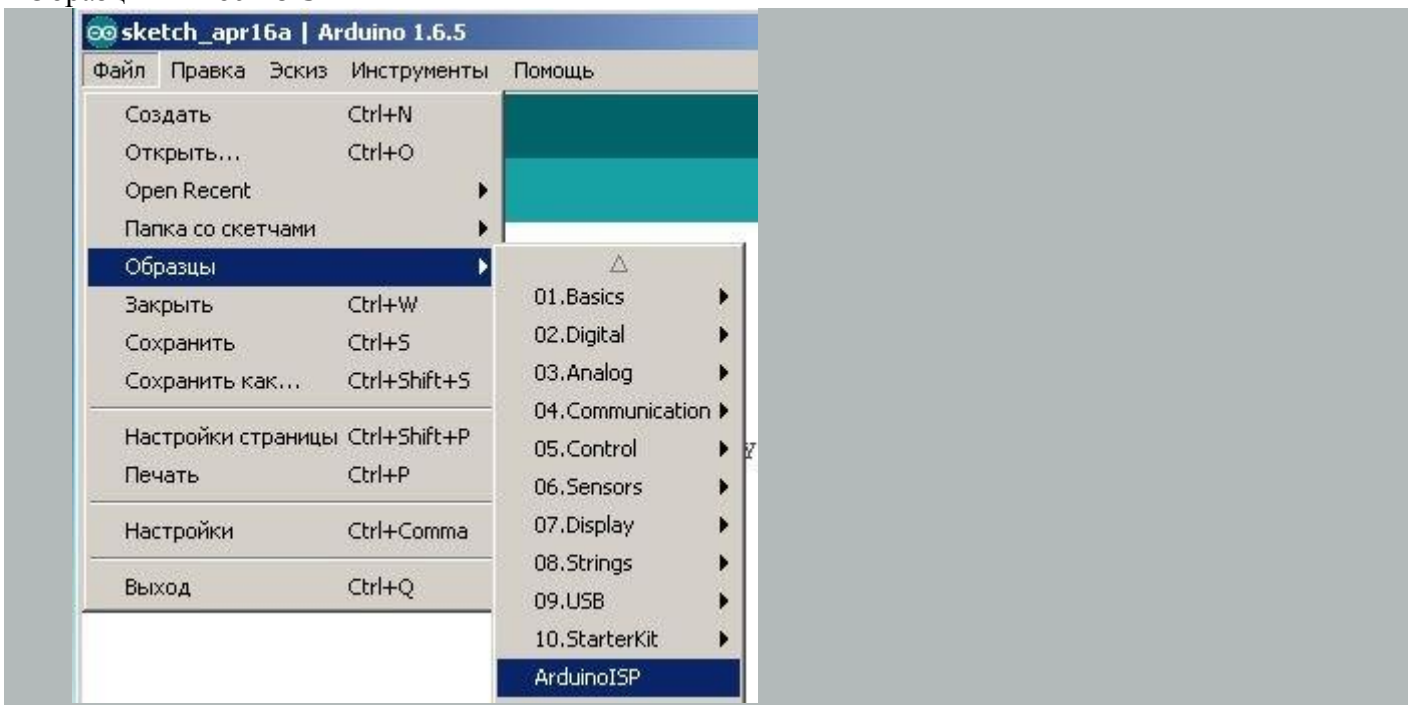


Прошивать им просто: проставил драйвера, вставил МК в нужный слот, выбрал программатор из списка (COM выбирать не нужно), указал микроконтроллер и его частоту и лей программу.
Вторым по популярности является USBtinyISP. В продаже я их не видел, а как его сделать самостоятельно подробно рассказано здесь — robocraft.ru/blog/2948.html
Профессиональные программаторы типа STK500 (Эх была у меня такая плата на предыдущей работе, классная вещь) явно не входят в рамки этой статьи. Поэтому идем дальше.

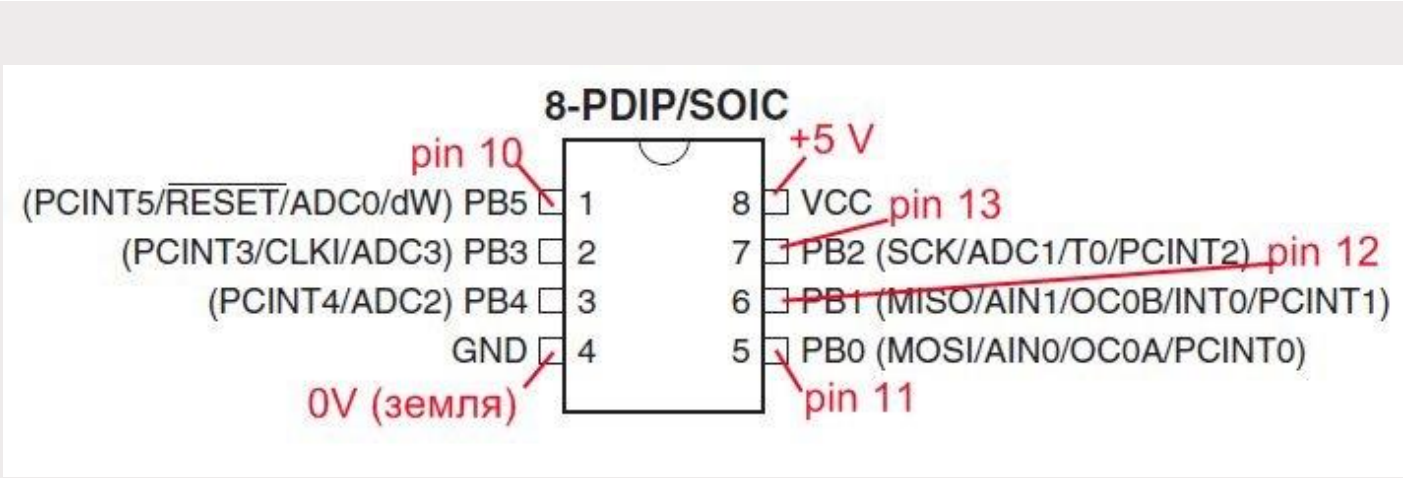
2) Прошивка при помощи заводской платы Arduino
Следует отметить, что далеко не все платы Arduino подходят для сего действия
Возьмите свою плату и проверьте перед тем как пытаться. Расположение выводов интерфейса SPI должно быть следующим:

pin name:	платы на основе mega168 или 328:	mega(1280 and 2560)
// MOSI:	11:	51
// MISO:	12:	50
// SCK:	13:	52

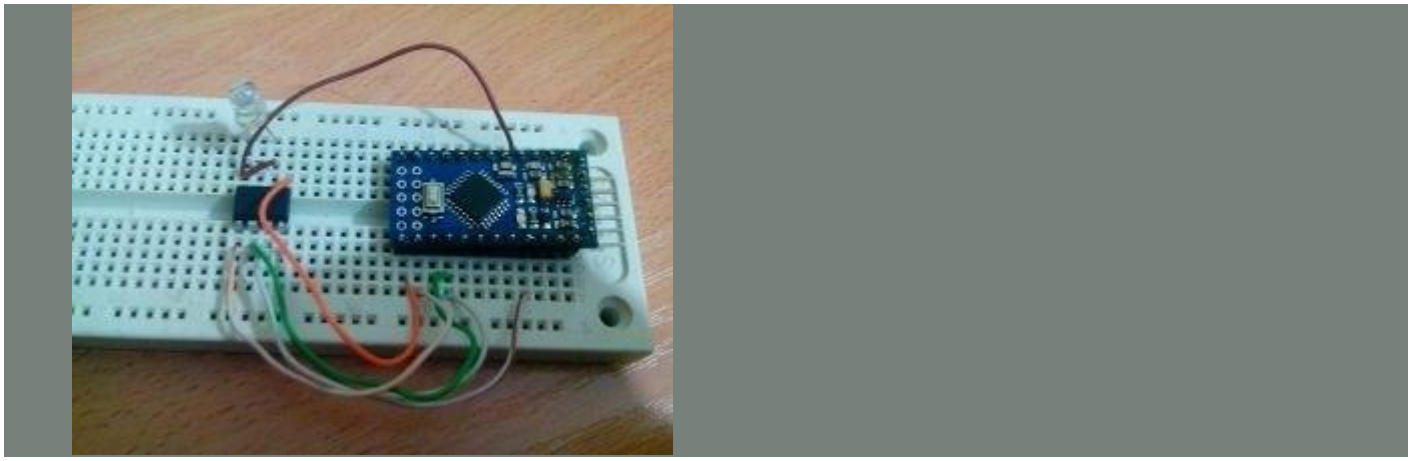
Если совпадает, то все ок, если нет, то лучше найти другую плату
Итак, последовательность действий:
а) Готовим плату. Для этого подключаем плату Arduino к компу, выбираем пункт меню Файл->Образцы-> ArduinoISP



и "вгружаем" код на плату. Все мы превратили нашу плату в программатор.
б) Подключаем. Схема подключения простая: нужно подключить выводы интерфейса SPI платы к соответствующим им выводам интерфейса SPI микроконтроллера, а вывод Reset микроконтроллера к 10 пину платы (или 53 для плат на основе mega1280 и 2560). На схеме ниже пример подключения Attiny13 (Attiny85 и Attiny45 аналогично, остальные МК смотрим распиновку) к плате на основе микроконтроллера Atmega 168/328



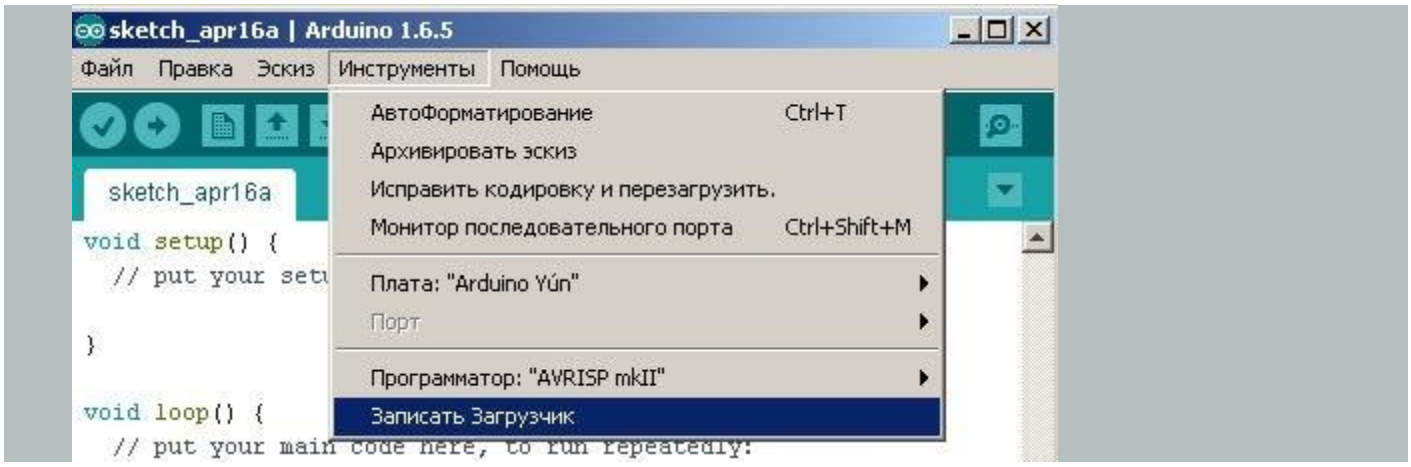
У меня получилось как то так:



Заливаем загрузчик. А нужно ли?
Итак сначало определимся что такое загрузчик и зачем это нужно.
Установка загрузчика дает возможность напрямую, через последовательный порт прошивать микроконтроллер (только имеющие аппаратный последовательный порт).Например так прошиваются пустые ATMEGA328P, которые потом можно использовать вместо установленной штатно микросхемы на Arduino UNO и устанавливать далее на самодельные платы.
Т.е. на ту же Attiny13 заливать загрузчик просто не имеет смысла — у нее нет аппаратного порта (выводов Tx, Dx), хотя некоторые "умельцы", пишущие обучающие статьи это делают (см UPD ниже). А вот для Atmega8 это можно сделать — она при этом потеряет 1кБайт из 8 своей памяти, но зато залить прошивку уже можно будет не через SPI, а подключив ее к адаптору COM порта (как это сделать рассказано здесь — www.drive2.ru/b/2642464/ на примере Arduino Pro Mini, смотрим раздел "Подключаемся").

UPD. *Добавка написаная много познее*
С прошедшим временем понимаю, что на теме загрузчика стоит остановиться отдельно, заодно рассказать подробно, что такое фьюзы.
Но сейчас вкратце: В каждый микроконтроллер, помимо записи прошивки, необходимо записывать биты конфигурации — так называемые фьюзы и локбиты. С завода они идут записанные среднепотолочно и в реале скорее всего вам не подойдут — от этого и происходит неправильный расчет времени, например.
Но записью загрузчика можно обойти этот момент — как раз при записи загрузчика в микроконтроллере прописываются правильные фьюзы. Поэтому, даже если загрузчик не нужен, его можно прописать, чтобы прописались фьюзы, а потом при записи самой прошивки ("скетча") поставить галочку, что он не нужен и его можно удалить.

Если считаете, что вам это нужно, то устанавливаем микроконтроллер в программатор (подключаем к плате-программатору) и нажимаем "Записать загрузчик"



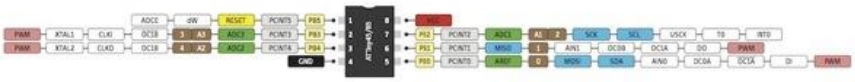
Прошивка программ
а) Пишем программу. Тут все как обычно, нужно только учесть, что для Attiny поддерживаются не все возможности "языка Arduino", а только
pinMode()
digitalWrite()
digitalRead()
analogRead()
analogReference(INTERNAL) / (EXTERNAL)
shiftOut()
pulseIn()
analogWrite()
millis()
micros()
delay()
delayMicroseconds()

б) Прошиваем.
Если мы используем плату Ардуино, то выбираем в качестве программатора "Arduino as ISP", в разделе "платы" наш микроконтроллер и частоту на которой он будет в дальнейшем работать, в разделе "порт" виртуальный COM нашей платы-программатора и нажимаем "вгрузить".
Вариант записи через программатор описан здесь — О бедном AtMega замолвите слово

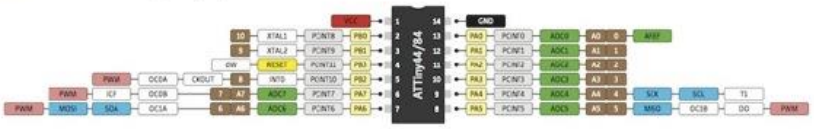
в) проверяем работоспособность. Тут есть некоторые разногласия что делать с ногой RESET. Кто-то считает, что в процессе работы ее можно оставлять в воздухе, кто-то, что ее нужно подтянуть через резистор 10 кОм к питанию. Работает и так и так, тут больше вопрос религии :)

Справочный раздел

Attiny13/45/85

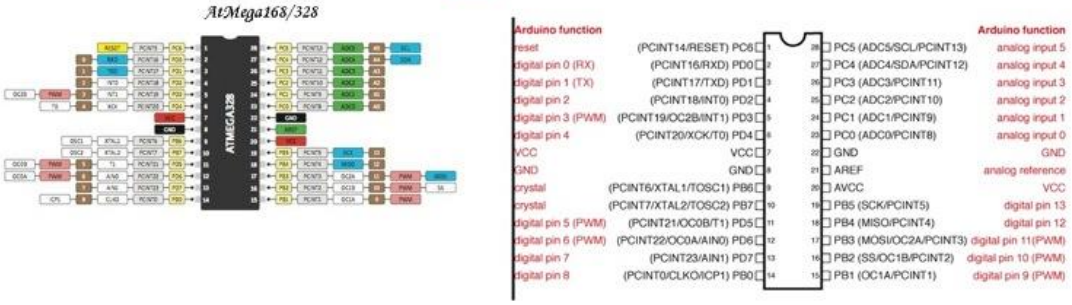


Attiny44/84

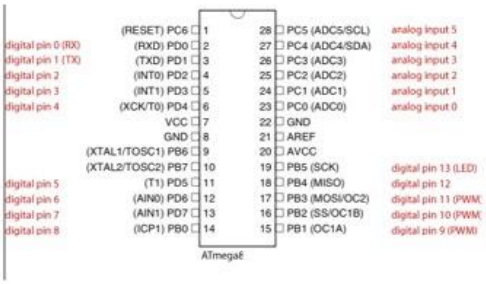


Attiny 2313/4313

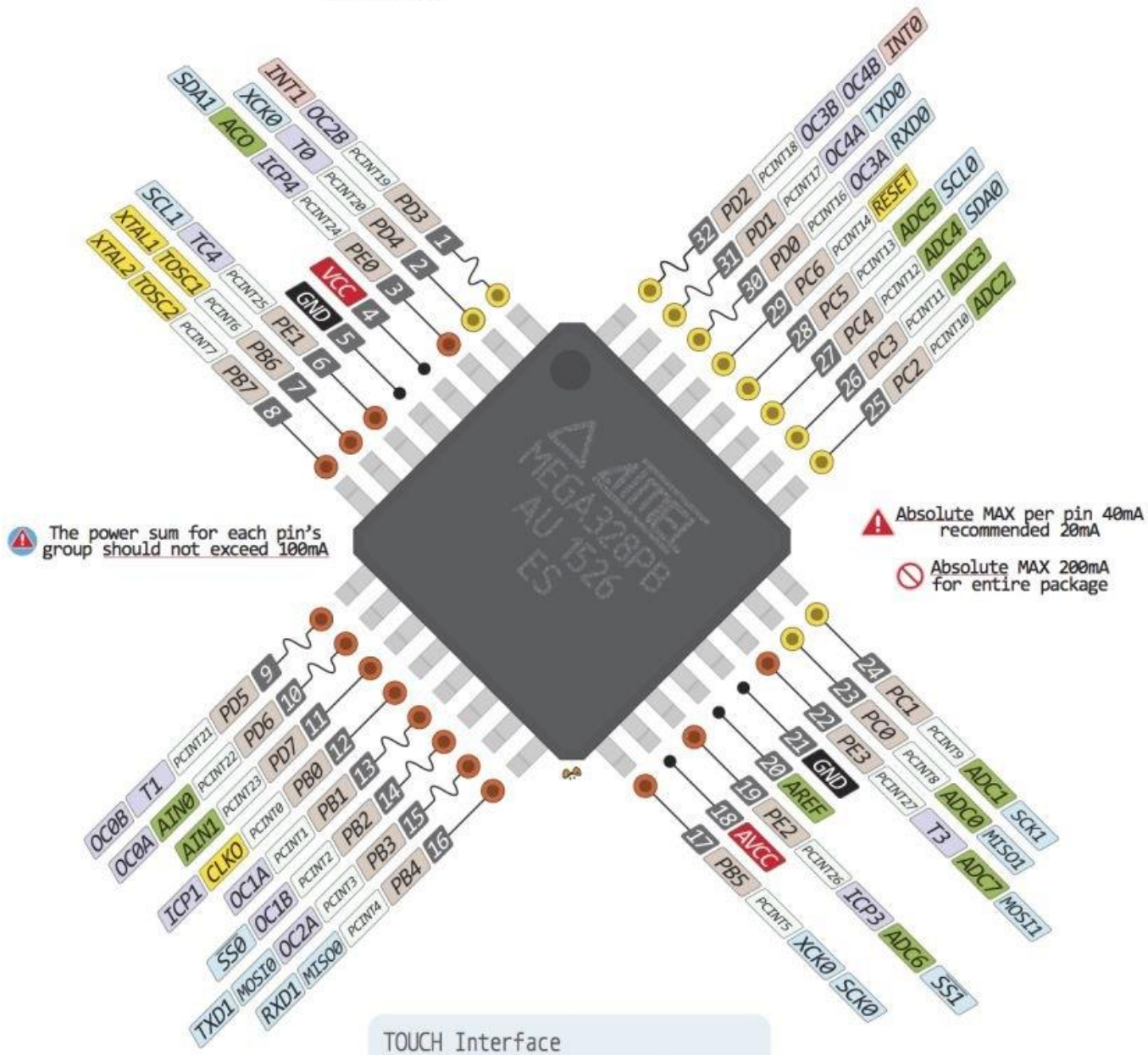
AtMega168/328

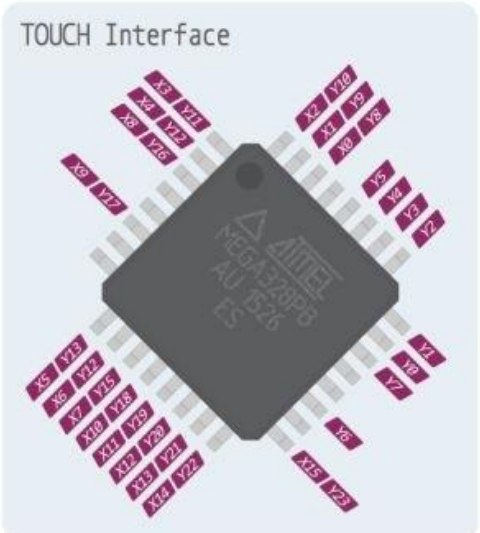


AtMega8



ATMEGA328PB
PINOUT

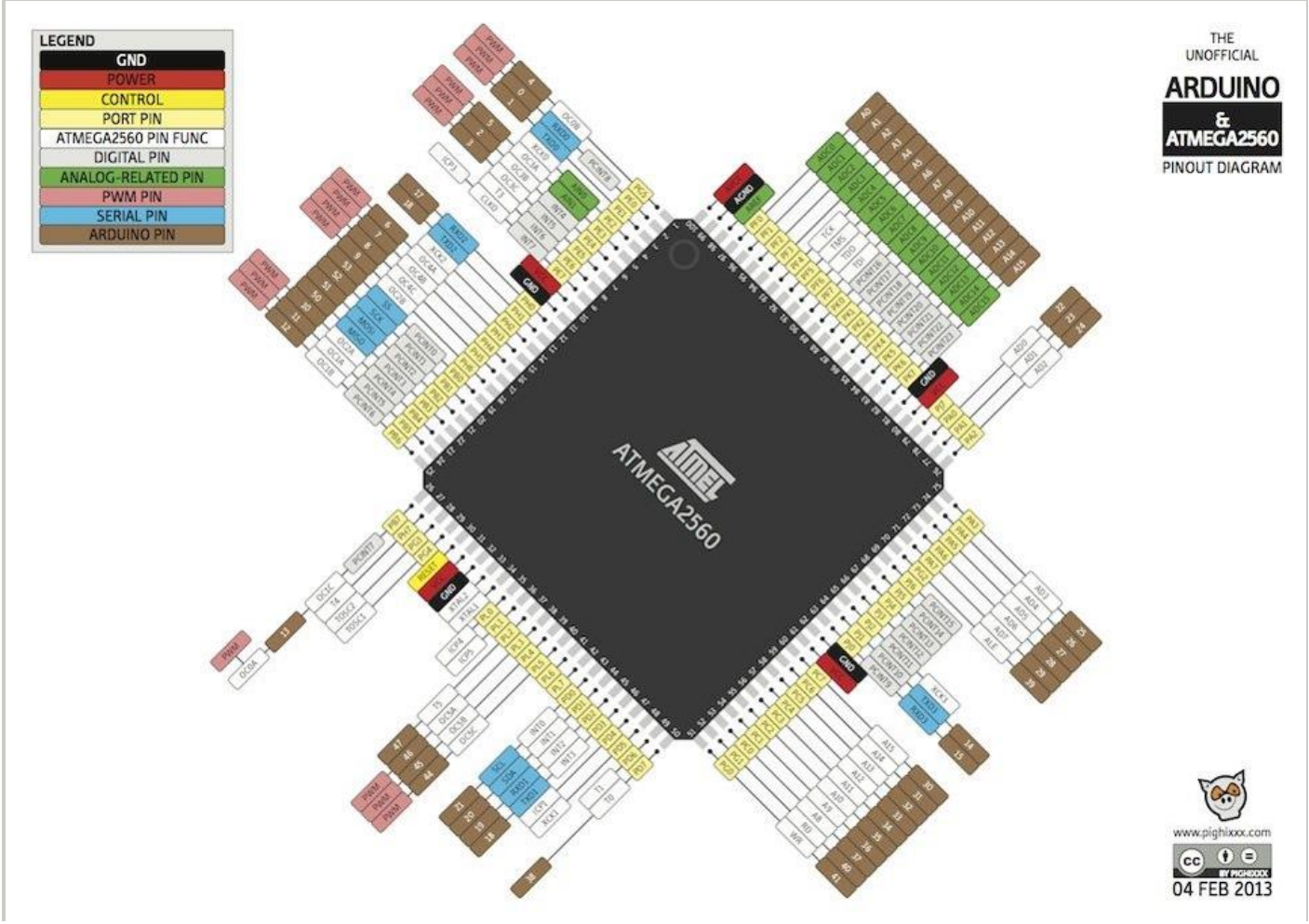




- Power
- GND
- Serial Pin
- Analog Pin
- Control
- INT
- Physical Pin
- Port Pin
- Pin function
- Interrupt Pin
- PWM Pin
- Port Power

bq

27 JAN 2016
ver 1 rev 0



2 года
https://www.drive2.ru/b/2955382/