Уроки C/C++ 33
Уроки Arduino 32
Введение
Базовые уроки
Продвинутые уроки
Интерфейс SPI
Интерфейс I2C (Wire)
EEPROM память
PROGMEM память
Внешние прерывания

Мы уважаем вашу конфиденциальность

Мы используем файлы cookie, чтобы улучшить ваш опыт просмотра, показывать персонализированную рекламу или контент и анализировать наш трафик. Нажимая «Принять все», вы соглашаетесь на использование наших файлов cookie.

Настроить Отклонить Принять все

Справочник 35
Программирование 21
Робототехника 8
Модули и компоненты 55
Обучающие проекты 1

Работа с голым камнем, ATtiny

Оглавление

UART и ISP
Подключение программатора
Ставим "ядро"
Программирование
"Загрузить через программатор"
Проект на голом МК
Минимальная обвязка и полезные советы
Перенос МК с Arduino на свою плату
ео

Данный урок применим к AVR Arduino и опирается на информацию из предыдущего урока о программаторах, обязательно изучите сначала его

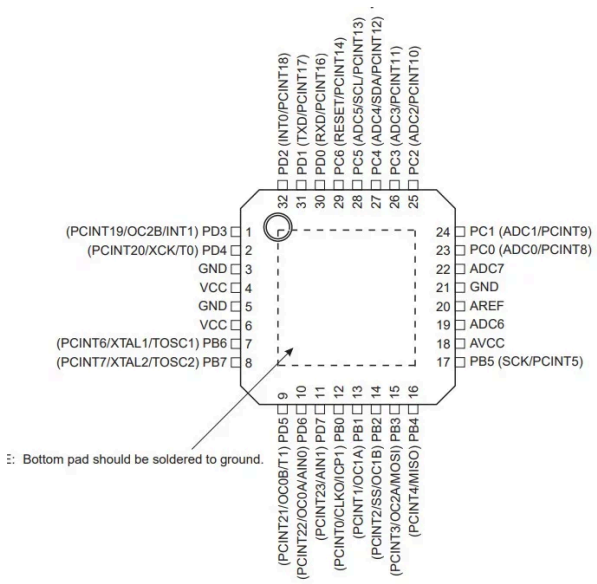
ошлом уроке я рассказывал о том, как загрузить прошивку в Ардуино, используя внешние устройства: USB-TTL бразователь и ISP программатор. Как нетрудно догадаться, при помощи этих же инструментов можно загрузить прошивку и в й чип, будь то ATmega328 или ATtiny85. МК серии ATtiny являются младшими братьями ATмег, у них меньше ног, меньше ти, меньше интерфейсов и таймеров, но зато они дешевле и хорошо подходят для мелких проектов. Давайте сравним олько популярных МК:

MK/spec	ATmega328	ATtiny85	ATtiny13
Flash	32k	8k	1k
SRAM	2k	512b	64b
EEPROM	1k	512b	64b
Цифр. Ног	23	6	6
Аналог. Ног	8	4	4
Таймеры	3	2	1
SPI	+	+	-
UART	+	-	-
I2C	+	+	-
Цена	95р	70р	20р

Примечание: большинство функций объединены на одних и тех же пинах. Как вы можете видеть, чем дешевле МК, тем меньше у него возможностей. Полное подробное сравнение можно посмотреть здесь.

UART и ISP

Для подключения прошивателей к голому чипу нам нужно будет изучить распиновку (pinout) на нужный микроконтроллер. Распиновки бывают цветные и красивые (часто с ошибками), а бывают более серьёзные и правильные. Лучше всего открыть даташит на нужный МК и на второй же странице найти 100% правильную распиновку. Например для ATmega328, ATtiny85 и ATtiny13:



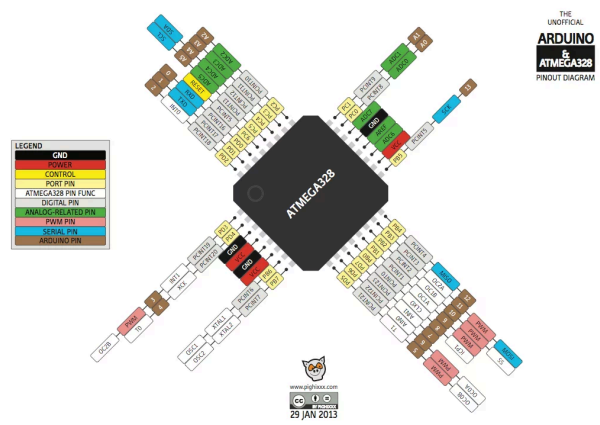
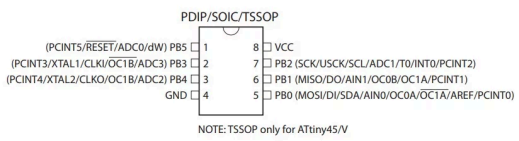
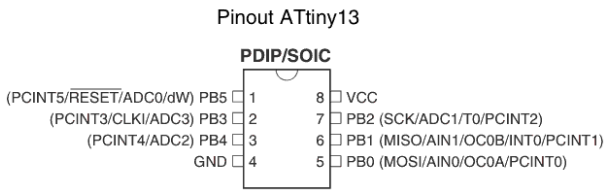


Figure 1-1. Pinout ATtiny25/45/85





На данных "схемах" подписаны все функции пинов МК. Чтобы загрузить прошивку через USB-TTL, то есть при помощи "живущего в памяти" загрузчика (bootloader), МК должен иметь на борту аппаратный UART, то есть пины **RX** и **TX**. Если таких пинов нет - прошивку можно загрузить только через ISP программатор. Вы спросите, а как же Digispark? Там стоит МК ATtiny85, у которого нет UART, но прошивка загружается через USB! Верно, но там хитрые разработчики сделали не менее хитрый загрузчик, который имитирует USB, и прошивка на Digispark загружается при помощи специальной программы, которая запускается в фоне, когда вы нажимаете кнопку "Загрузить" в Arduino IDE. Резюмируя для общего случая:

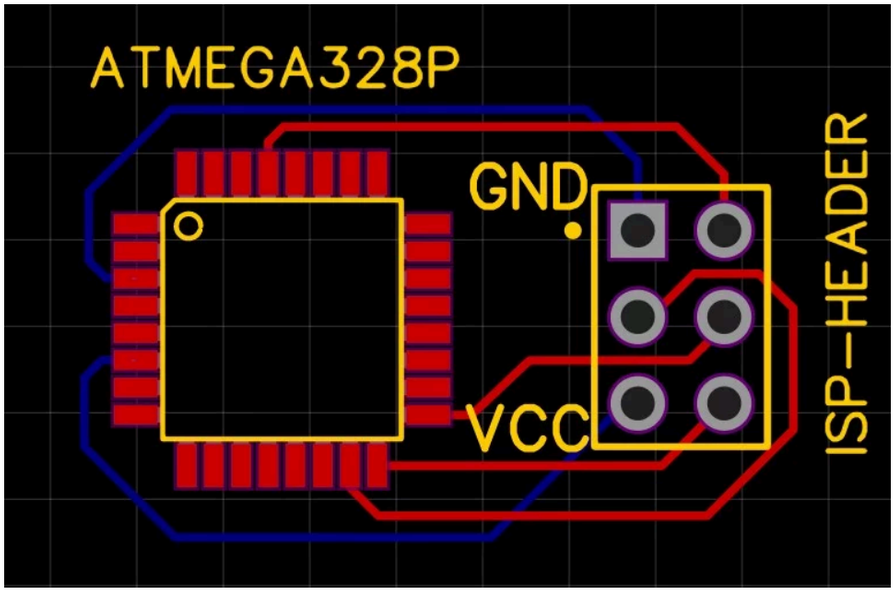
- Если в МК прошит загрузчик (bootloader) и на борту имеется аппаратный UART (пины RX TX), прошивку можно загрузить через USB-TTL "загружатор", также через него можно заниматься отладкой кода при помощи Serial.
- Если в МК нет пинов RX TX, значит прошивку можно загрузить только при помощи ISP программатора, да и о загрузчике в целом можно забыть, не нужен он. Отладкой всё ещё можно пользоваться, подключив USB-TTL и подняв на МК "программный" UART. Например в ядре для ATtiny85 (об этом ниже) уже идёт встроенный SoftwareSerial и можно им пользоваться.

Подключение программатора

Программатор, или Ардуину в качестве программатора, подключить очень просто. Смотрим распиновку и подключаем:

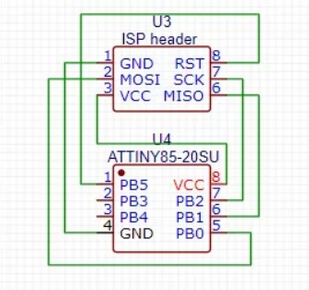
- Шину ISP: пины **MOSI**, **MISO** и **SCK**. Они есть на всех МК
- Сброс **RST**
- Землю **GND**. Любую из имеющихся, они соединены внутри МК
- Если МК не питается от своего источника, подключаем заодно **VCC**

Например ATmega328p подключаем к USB ASP (обсуждали в прошлом уроке) 6-пин вот так:

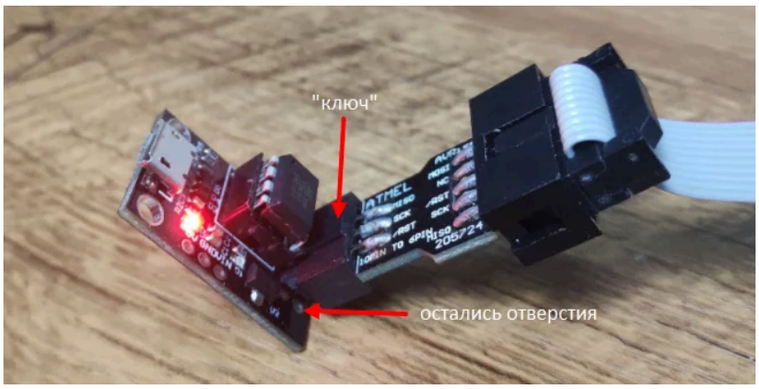


> Да, другие компоненты не нужны. Новый (из магазина) МК тактируется от внутреннего генератора на 8 МГц и может без проблем прошиваться прямо так как на схеме выше

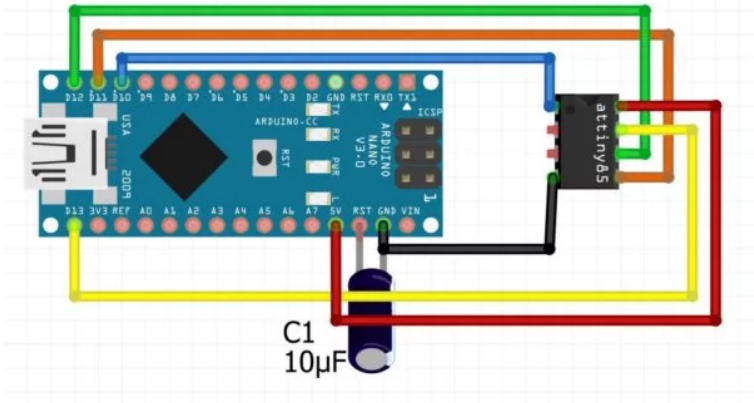
Тиньки к тому же USB ASP подключаются так:



Для удобства я использую макетку- дигиспарк, на которой разведены пины как раз под ISP 6-пин хэдер: втыкается выпирающим "ключом" в сторону МК. В плате выведены 8 пинов, нам нужны верхние 6 (на фото видно не запаянные пины ниже штекера):



Также можно прошивать МК через Arduino (Arduino as ISP, обсуждали в прошлом уроке). Схема для ATtiny85:



Конденсатор нужен обязательно!

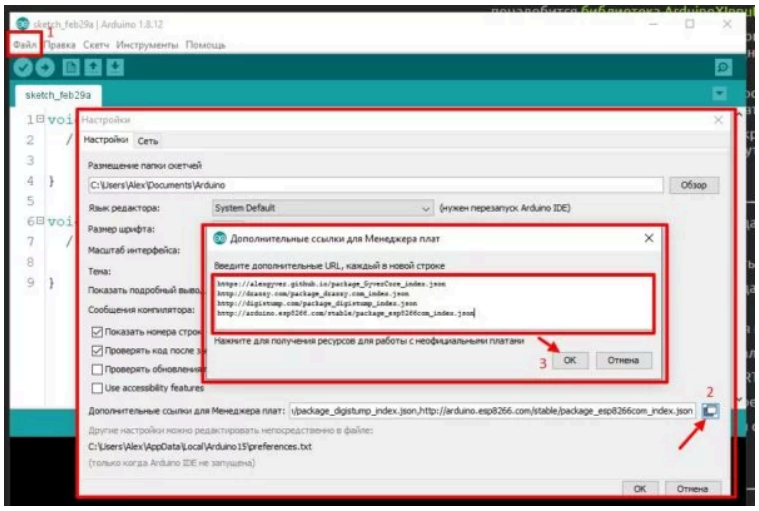
Подключили. Что дальше? Дальше мы уже можем работать с фьюзами через программу avrdudeprog (обсуждали в прошлом уроке), выбрав в списке соответствующий программатор и в списке МК - соответствующий МК. Также через эту программу можно загрузить скомпилированный "бинарник" - файл прошивки. Но нас всё-таки интересует работа через Arduino IDE.

Ставим "ядро"

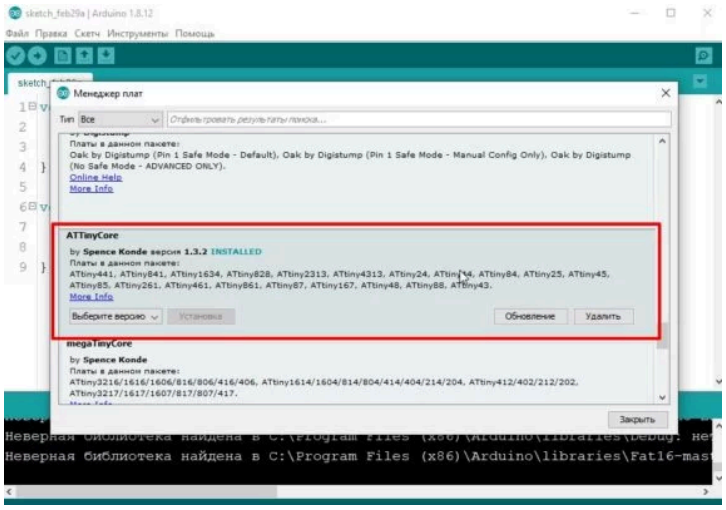
Для того, чтобы работать с Attiny через Arduino IDE, нам нужно установить так называемое ядро, или как оно называется в самой IDE - плату. Для ATmega328 у нас уже есть стандартное ядро, например плата Arduino NANO. Но тут есть нюанс: внутренние настройки "платы" NANO рассчитаны на работу с загрузчиком (bootloader) и с внешним тактированием 16 Мгц, то есть лучше не рисковать и установить ядро, которое поддерживает работу без загрузчика и с возможностью выбора частоты, чтобы иметь полный контроль над платой. Могу посоветовать вот эти:

- GyverCore - ядро для ATmega328, которое разработали мы с коллегой. Лёгкое и быстрое, возможность работы без загрузчика и широкий выбор вариантов тактирования. Ссылка для менеджера плат: `https://alexgyver.github.io/package_GyverCore_index.json`
- MiniCore – ядро для поддержки и расширенной настройки МК ATmega328, ATmega168, ATmega88, ATmega48 и ATmega8. Ссылка для менеджера плат: `https://mcudude.github.io/MiniCore/package_MCUdude_MiniCore_index.json`
- ATTinyCore – ядро для поддержки и расширенной настройки МК ATtiny 441/841, 44/84, 45/85, 461/861, 48/88, 828, 1634, 87, 167. Ссылка для менеджера плат: `http://drazzy.com/package_drazzy.com_index.json`
- MicroCore – ядро для поддержки и расширенной настройки МК ATtiny13, ATtiny13A и ATtiny13V. Ссылка для менеджера плат: `https://mcudude.github.io/MicroCore/package_MCUdude_MicroCore_index.json`
- megaTinyCore – ядро для поддержки и расширенной настройки МК ATtiny 3217, 1617, 817, 417, 3216, 1616, 816, 416, 1614, 814, 414, 214, 412, 212, 1607, 807, 1606, 806, 406, 1604, 804, 404, 204, 402, 202. Новые модели шьются по UPDI, по этой теме читайте отличную статью на русском языке. Можно сделать программатор из обычной Ардуины - читай тут. Данное ядро можно установить через менеджер плат, ищите по названию `Arduino megaAVR board package`, никаких ссылок вставлять не нужно.
- MightyCore - ядро для поддержки и расширенной настройки МК ATmega8535, ATmega16, ATmega32, ATmega164, ATmega324, ATmega644 и ATmega1284. Ссылка для менеджера плат: `https://mcudude.github.io/MightyCore/package_MCUdude_MightyCore_index.json`
- MegaCore - ядро для поддержки и расширенной настройки МК ATmega64, ATmega128, ATmega640, ATmega1280, ATmega1281, ATmega2560, ATmega2561, AT90CAN32, AT90CAN64 и AT90CAN128. Ссылка для менеджера плат: `https://mcudude.github.io/MegaCore/package_MCUdude_MegaCore_index.json`

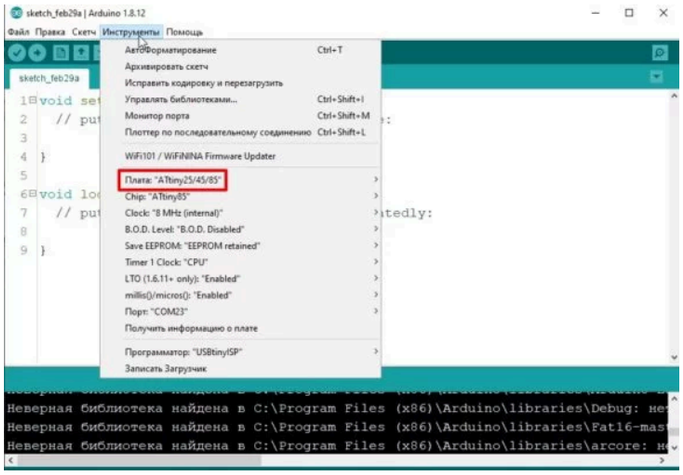
Как установить ядро: идём в `Файл/Настройки` и вставляем ссылку в окно дополнительных ссылок для менеджера плат:



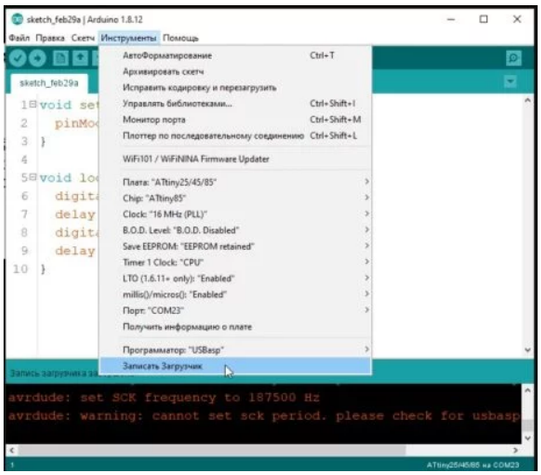
Далее идём в `Инструменты/Плата/Менеджер плат...` и находим нужное ядро. Устанавливаем:



После этого в списке плат появится новое семейство плат/МК на выбор. Я буду работать с ATtiny85:



Нас интересует меню **Clock** - тактирование. ATtiny85 интересна тем, что может тактироваться от внутреннего источника на 8 МГц, но также может его "умножить" на 2 и получится 16 МГц, этот вариант называется 16 MHz (PLL). Это очень круто, потому что голый чип будет работать с почти максимальной скоростью, что делает его ничуть не хуже той же Arduino Nano. Выбираем программатор из списка (я буду прошивать при помощи USB ASP). Однократно выполним `Инструменты/Записать загрузчик`, чтобы применить настройки тактирования:



Про остальные менюшки и варианты можно догадаться из их названия, или почтить подробное описание на GitHub по ссылкам выше.

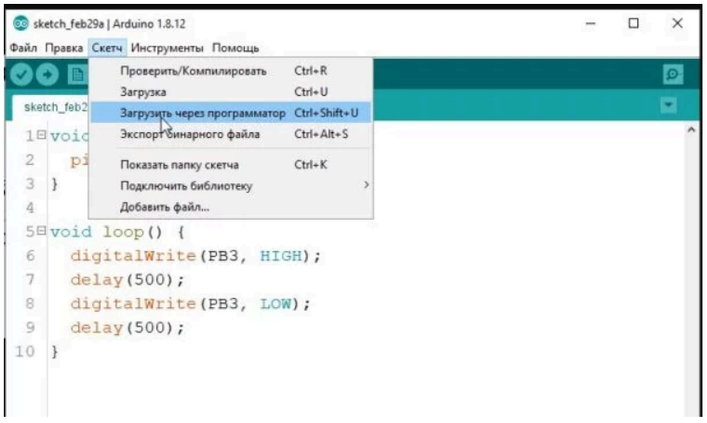
Программирование

Итак, что же даёт нам ядро помимо выбора настроек МК? Можно программировать МК всё теми же командами, что и раньше! Мигать светодиодами через `digitalWrite`, измерять напряжение через `analogRead` и прочее прочее. Давайте напишем классический Blink:

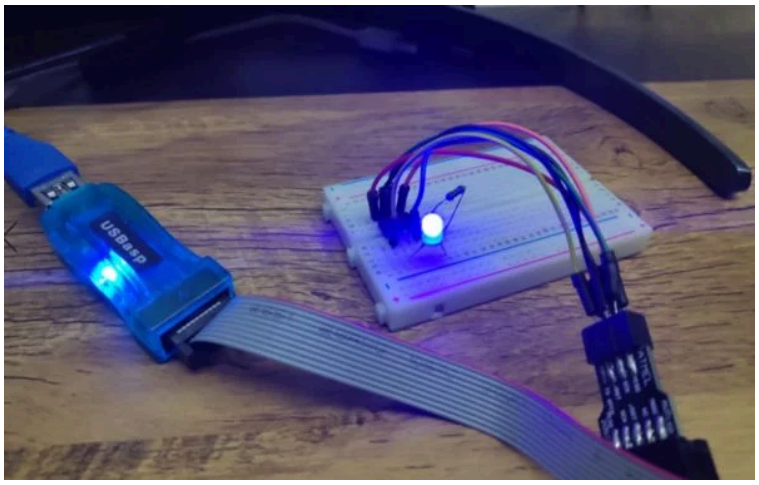
```
void setup() {
  pinMode(PB3, OUTPUT);
}

void loop() {
  digitalWrite(PB3, HIGH);
  delay(500);
  digitalWrite(PB3, LOW);
  delay(500);
}
```

> PB3 - это номер пина, прямо как на распиновке

Всё! Осталось загрузить прошивку. Для этого нажимаем `Скетч/Загрузить через программатор`:



Я подключил светодиод через резистор на 220 Ом и он мигает два раза в секунду, всё как написано:



Что следует помнить при работе с тиньками: у них мало памяти, а все вот эти Ардуино-функции являются кошмаром индуса и занимают очень много места в памяти. Если тини85 ещё как-то переживёт такие издевательства и сможет уместить в себе вполне интересный проект из Ардуино-функций, то в тини13 уже сложно уместить что-то серьёзное. Напомню: всего 64 байта оперативной памяти и 1 кб флэша!

Важный момент по работе с ядрами для других МК, у которых больше 8 ног. Для сохранения удобства работы с IO функциями ядра (digital/analog/Read/Write) к пину можно обращаться как `PIN_ + имя_ноги` на распиновке, например подадим высокий сигнал на пин `PC3`: `digitalWrite(PIN_PC3, HIGH)`. Все остальные нюансы расписаны на странице ядер по ссылкам выше.

Я думаю вы поняли, что в целом работа с голыми МК не особо то и отличается от работы с обычной платой Arduino, и теперь можно переходить к сложным самоделкам на базе своей платы, в центре которой будет стоять микроконтроллер. Давайте поделюсь парой советов по минимальной обвязке.

"Загрузить через программатор"

Одноимённый пункт в меню "Скетч" в Arduino IDE загружает скетч через выбранный программатор, а также затирает загрузчик! Если в проекте было решено использовать загрузчик - не используйте загрузку через программатор после его прошивки!

Проект на голом МК

Зачем делать проект на своей плате и голом микроконтроллере?

- Конечно же размер, своя плата получится компактнее в большинстве случаев.
- Энергопотребление. Ни для кого не секрет, что всякие удобные штуки на плате ардуино потребляют огромный по меркам энергосбережения ток, и для автономного устройства лучше взять голый камушек.
- Работая с голым камнем мы можем выбрать микроконтроллер под свои задачи. Например для мелкого проекта необязательно брать 328 мегу, если с задачей справится аттини13, которая стоит 20 рублей.
- Возможная экономия. Ардуино нано стоит условно 170 рублей без доставки. Голая 328 мега на том же алиэкспресс – 100 рублей. Если брать десяток. В дорогущем российском чип и дип – 190 рублей. То есть если очень приспичит – можно и тут купить без особых потерь, потому что оригинальная нано стоит 3 тысячи рублей в том же ЧИДе.
- Удобство разводки платы. Микросхема в корпусе под поверхностный монтаж находится на одном слое и не мешает дорожкам на другом слое.
- Удобство монтажа. Даже если делать свою плату прототип лутом, то припаять на неё чип в 100 раз удобнее и быстрее, чем сверлить три десятка отверстий под ардуину, а потом их запаивать.
- Мелкосерийное производство готовых или почти готовых плат, для себя если нужно несколько или на продажу. На том же jlcpcb можно заказать изготовление плат с распайкой внимание smd компонентов. То есть ардуину вам никто не припаяет, а вот ту же 328 мегу припаяют за 100 рублей, тиньку 13ю – за 35 рублей, а чем больше партия – тем дешевле. К этому вопросу мы вернёмся ближе к концу этого урока.

Я на личном опыте убедился, что МК способен работать вообще без какой-либо обвязки. Это был проект "Читалка файлов с SD карты" на базе ATmega328. Проект достаточно непростой: МК читал текстовые файлы с карты памяти microSD и выводил их на OLED дисплей. Никаких лишних компонентов на плате нет, МК тактируется от внутренних 8 МГц и всё работает отлично. Даже карта памяти подключена напрямую к МК =) Но в надёжных устройствах делать так не рекомендуется!

Минимальная обвязка и полезные советы

Соединять все ноги питания (GND, VCC) максимально толстыми и короткими дорожками между собой:

- **GND** обычно делают полигоном на всю поверхность платы
- Я где то читал, что **VCC** рекомендуется соединять дорожками "внутри шелкографии", то есть под МК. Оно и логично, так они получатся короче всего

Поставить керамический конденсатор ~100 нФ/nF (0.1 мкФ/uF) по питанию МК: между VCC и GND для сглаживания микропульсаций напряжения:

- Располагать максимально близко к пинам питания МК

- Также параллельно ему можно поставить электролит (алюминиевый цилиндр) или танталовый конденсатор (компактный SMD чип) на 10-47 мкФ для сглаживания более серьёзных пульсаций напряжения. Особенно если МК питается от одного источника с другими потребителями, или используется некачественный источник питания

Если нужна кнопка сброса (reset), подключаем её к **RST** и **GND**, т.е. кнопка должна подать на RST **низкий** сигнал. Внутри МК уже присутствует подтягивающий резистор на пин RST:

- При наличии кнопки и дорожки от пина RST рекомендуется подтягивать пин к VCC резистором на 10 кОм, потому что внутренний подтягивающий резистор имеет довольно большое сопротивление и наведённые на дорожку помехи (она выступает в роли антенны) могут привести к сбросу МК

Если нужен внешний тактовый генератор - подключаем его между пинами **XTAL1** и **XTAL2** (см. распиновку своего МК), и оба пина подключаем к GND через керамические конденсаторы на 18-22 pF/пФ. Более точный номинал конденсатора можно узнать из даташита на генератор, но в 99% это 22 пФ

- Существуют генераторы со встроенными конденсаторами, например вот такие. На принципиальной схеме эти конденсаторы видно
- Не забываем в настройках платы или напрямую во фьюзах поставить внешнее тактирование. ВНИМАНИЕ! Если фьюзы настроены на внешнее тактирование и внешний генератор не подключен к пинам - вы не сможете прошить МК даже при помощи ISP программатора!

Если нужна удобная прошивка по ISP - выводим штекер для подключения. Вот тут можно открыть мой модуль такого штекера для EasyEDA:

Если нужна прошивка по UART - выводим **RX**, **TX**, **GND**. Для автоматического сброса МК перед началом прошивки у USB-TTL преобразователя должен быть выведен пин DTR. Его подключаем к RST через конденсатор на 0.1 мкФ. Также подключаем, VCC если нужно запитать МК от UART преобразователя. Прошивать можно любым USB-TTL преобразователем с выходом DTR. В показанной схеме подключение к преобразователю будет **RX->TX**, **TX->RX**!

Я думаю теперь вы готовы к созданию проекта на своей плате!

Перенос МК с Arduino на свою плату

Напомню, что источник тактирования играет важную роль при загрузке прошивки. Микроконтроллер может быть настроен на тактирование от внутреннего генератора на 8 МГц, либо на тактирование от внешнего:

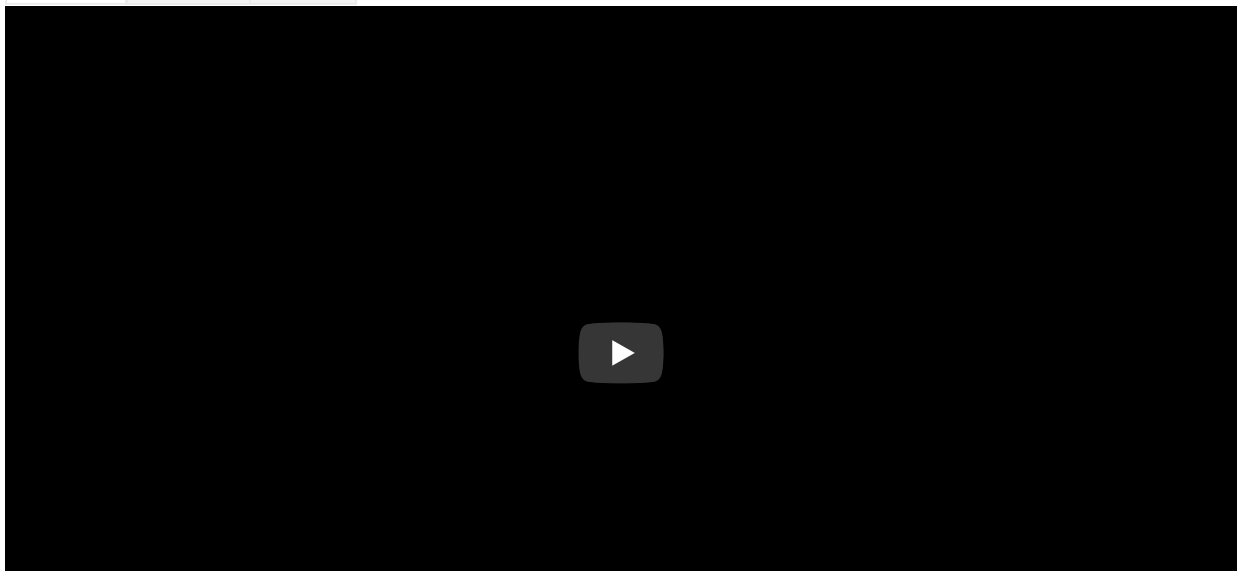
- При работе от **внутреннего** генератора МК запускается и работает просто при подаче напряжения, готов прошиваться как через UART (при наличии загрузчика), так и через ISP
- При настройке на работу от **внешнего** генератора МК сможет работать и прошиваться только при подключении внешнего генератора. То есть если генератор физически не подключен к МК - его нельзя будет прошить даже через ISP

Это же касается переноса микроконтроллера с платы Ардуино на свою плату: на Ардуино стоит кварц. Если на вашей плате есть кварц для МК - всё будет работать сразу. Если на вашей плате нет кварца - перепаянный с Ардуино МК не будет работать и прошиваться. Для переноса МК с платы Ардуино на свою плату (без кварца) нужно настроить МК на внутреннее тактирование, об этом мы говорили в прошлом уроке. Для этого нужно подключить к плате программатор (USB-ASP или Arduino as ISP) и прошить фьюзы:

- Вручную через Avrdudeprog, выставив нужное во встроенном калькуляторе фьюзов
- Вручную через Arduino IDE, сконфигурировав и выставив фьюз-байт вручную в boards.txt
- Автоматически через конфигурацию ядра. Для Arduino Nano это может быть GyverCore или miniCore, там в меню платы есть пункт `Clock -> Internal 8 MHz`. В Arduino IDE жмём "Записать загрузчик" и прошиваются новые фьюзы, после чего можно выпаивать МК и запускать/прошивать его уже без кварца

Видео

YouTube | VK Video | Rutube



[redacted]

Полезные страницы

- Набор GyverKIT – наш большой стартовый набор Arduino, продаётся в России
- Каталог ссылок на дешёвые Ардуины, датчики, модули и прочие железки с AliExpress
- Обратная связь – сообщить об ошибке в уроке или предложить дополнение по тексту (alex@alexgyver.ru)
- Поддержать автора за работу над уроками

5

Рейтинг статьи

Обновлено 28.04.2025

← **Программаторы, ISP, фьюзы**

Различные трюки →

Подписаться

Оставьте первый комментарий!

0 КОММЕНТАРИЕВ

Copyright © 2025 AlexGyver Technologies