custos
17 дек 2015 в 12:03

ATtiny85: прототип беспроводного сенсора

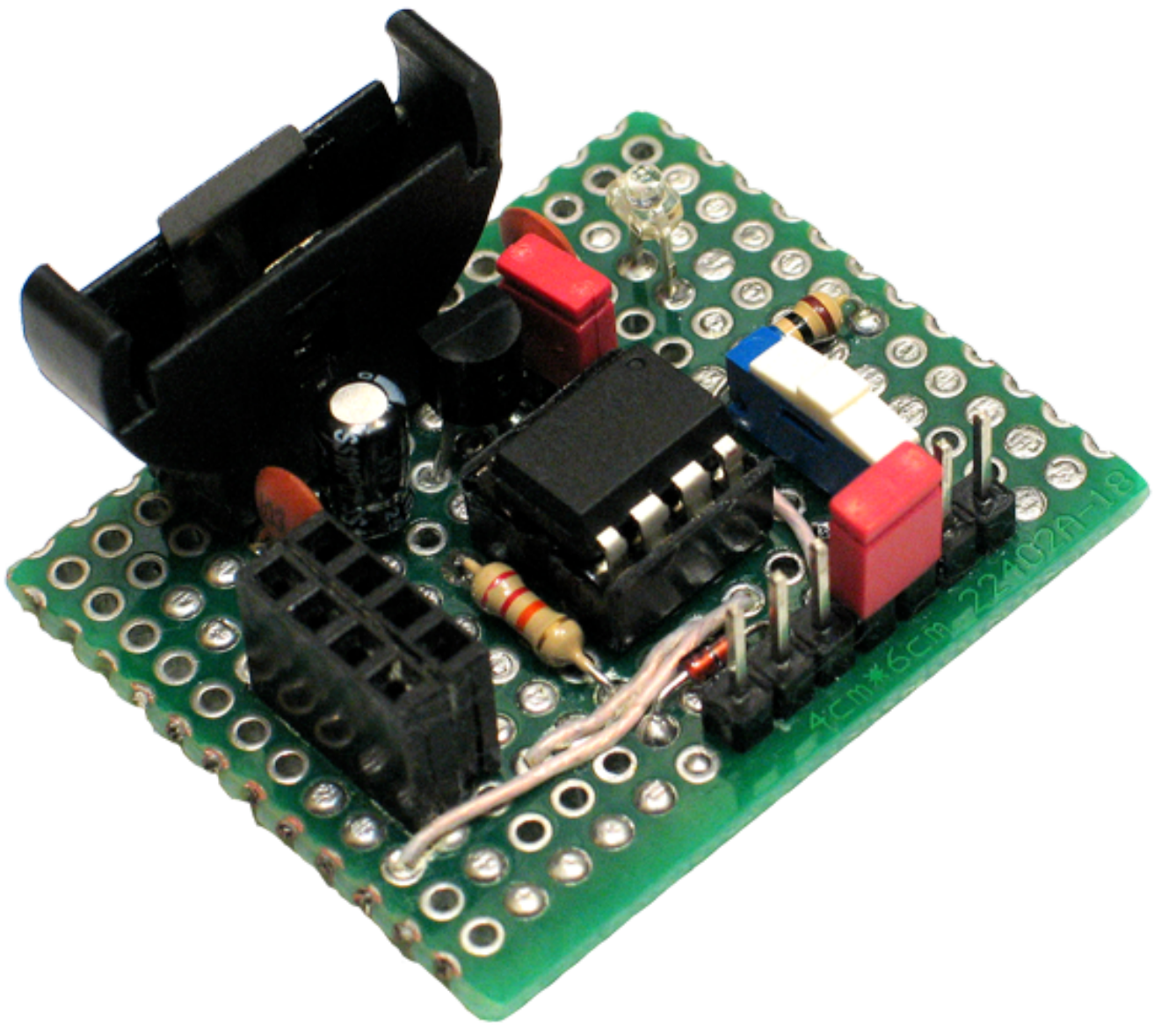
14 мин 64K

DIY или Сделай сам, Электроника для начинающих

Туториал

Обычно, для перехода от идеи к реализации, необходим прототип устройства, удобный для проверки и отладки на месте, что особенно важно для мобильного устройства. Далее

+14 166 22



Цель — создать сенсор работающий, условно говоря, в коробке с искусственным освещением и передающий температуру и статус освещения с немедленной реакцией на изменение освещения: включилось, отключилось, мигнуло. Сенсор решено было сделать мобильным и питать от элемента CR2032, иначе говоря, при разряде до 2.7V (предел для датчика TMP36), можно рассчитывать на 200mAh.

Микроконтроллер ATtiny85 имеет всего 5 портов ввода/вывода и возможность отключить RESET в пользу дополнительного порта. Данный бюджет был распределён следующим образом:

- **3 порта** — радиомодуль NRF24L01+, спецификация требует пять портов, но в данном случае это не приемлемо и будет использована 3-х пиновая конфигурация;
- **1 порт** — датчик освещения на базе фототранзистора BPW17N;
- **2 порта** — температурный датчик на базе TMP36, второй порт нужен для подачи питания, чтобы иметь возможность отключать датчик при необходимости.

Элементная база определена, можно приступать к проектированию.

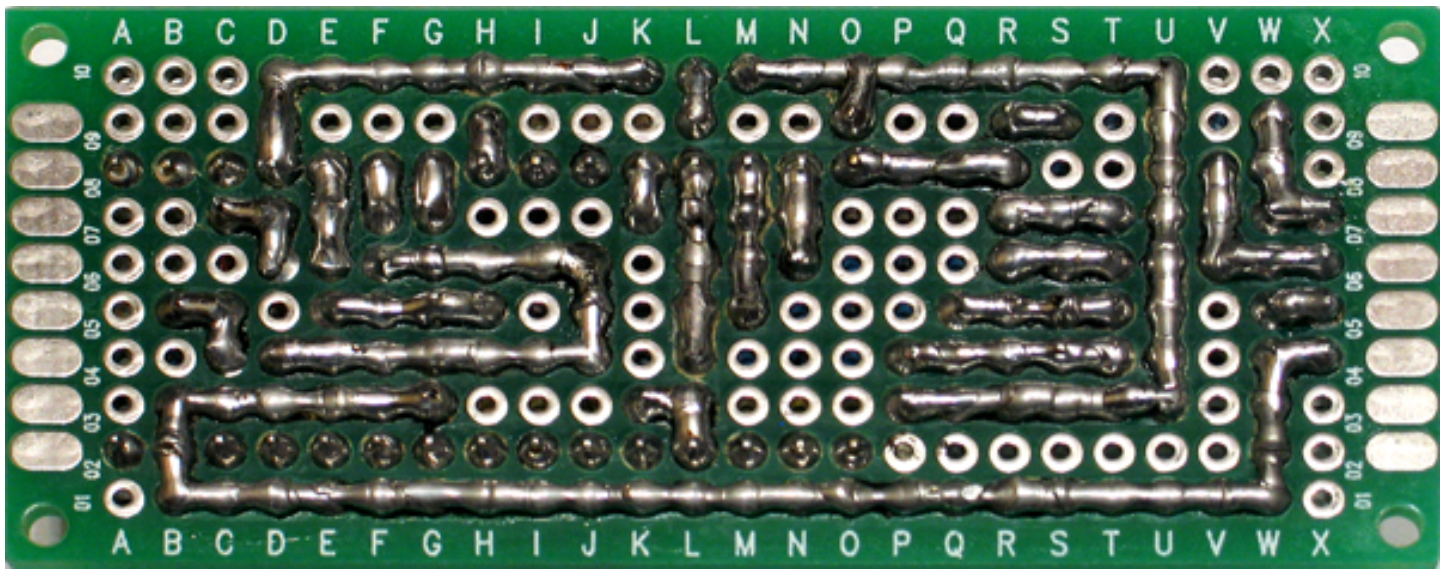
RESET в качестве порта ввода/вывода

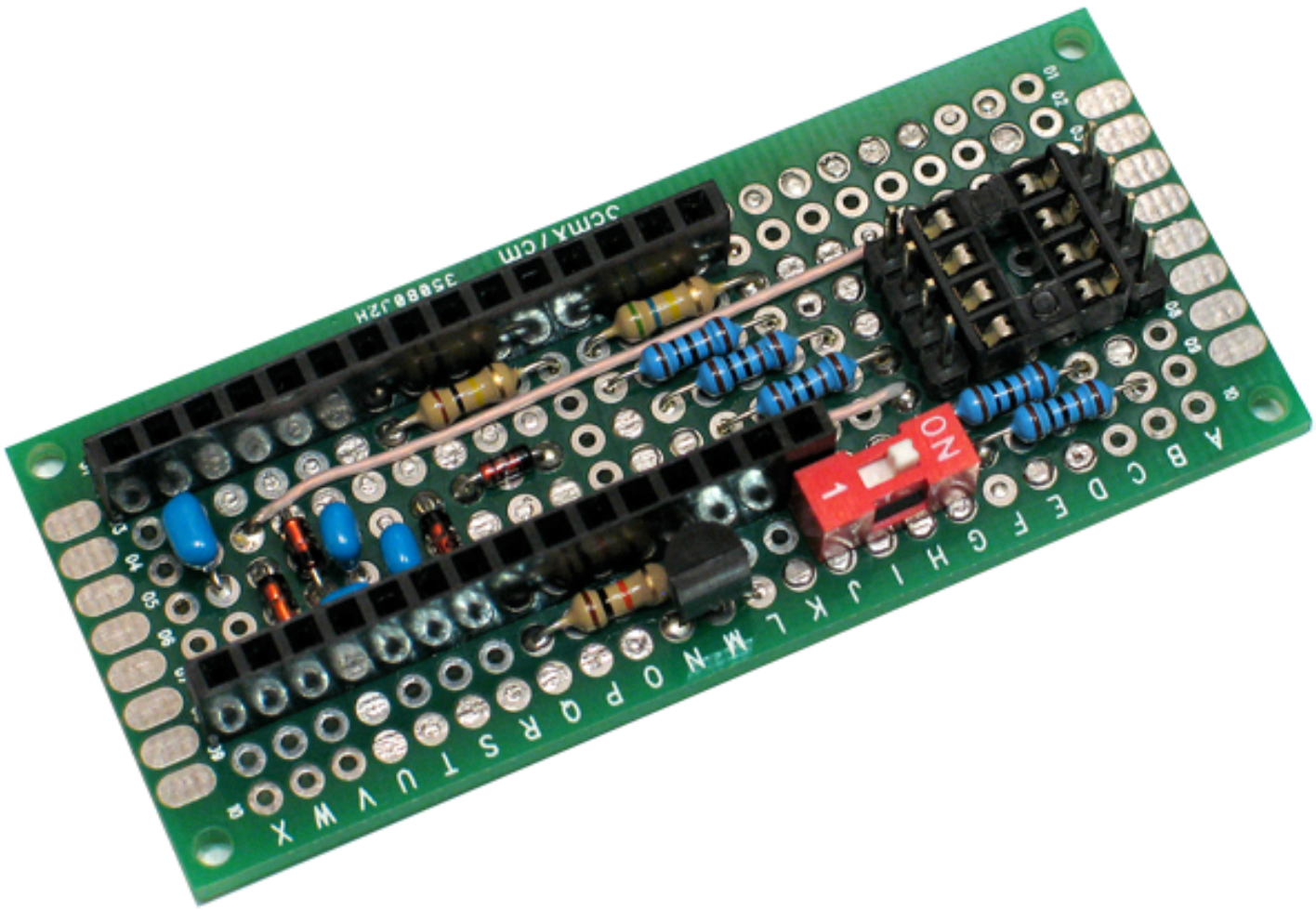
Ситуация требует использовать RESET в качестве порта ввода/вывода. Однако RESET необходим для программирования микроконтроллера, в том числе и для прошивки настроек (Fuses). Соответственно, после отключения RESET программировать устройство будет не возможно. Таким образом, чтобы не получить «кирпич» первым делом необходимо раздобыть высоковольтный (12V) программатор, для сброса Fuse-ов.

Подходящую схему и скетч для Arduino можно найти по ссылке ATTiny Fuse Reset with 12 Volt Charge Pump. Данная схема не требует дополнительного питания, и для получения 12V используется умножитель (Charge Pump).

- Описание работы схемы высоковольтного программатора

У меня получился следующий дизайн:





- Высоковольтный программатор в сборе (фото и комментарии)

Принципиальная схема

Разработка схемы вместе с макетной платой будет производится в KiCad EDA, это свободный пакет для автоматизации разработки электронных схем.

Исходные файлы KiCad проекта размещены на GitHub, в папке **attwlight_sensor.pcb**.

Существует множество хороших инструкций по работе с данным редактором, например стоит посмотреть ролик на YouTube KiCAD Quick-Start Tutorial.

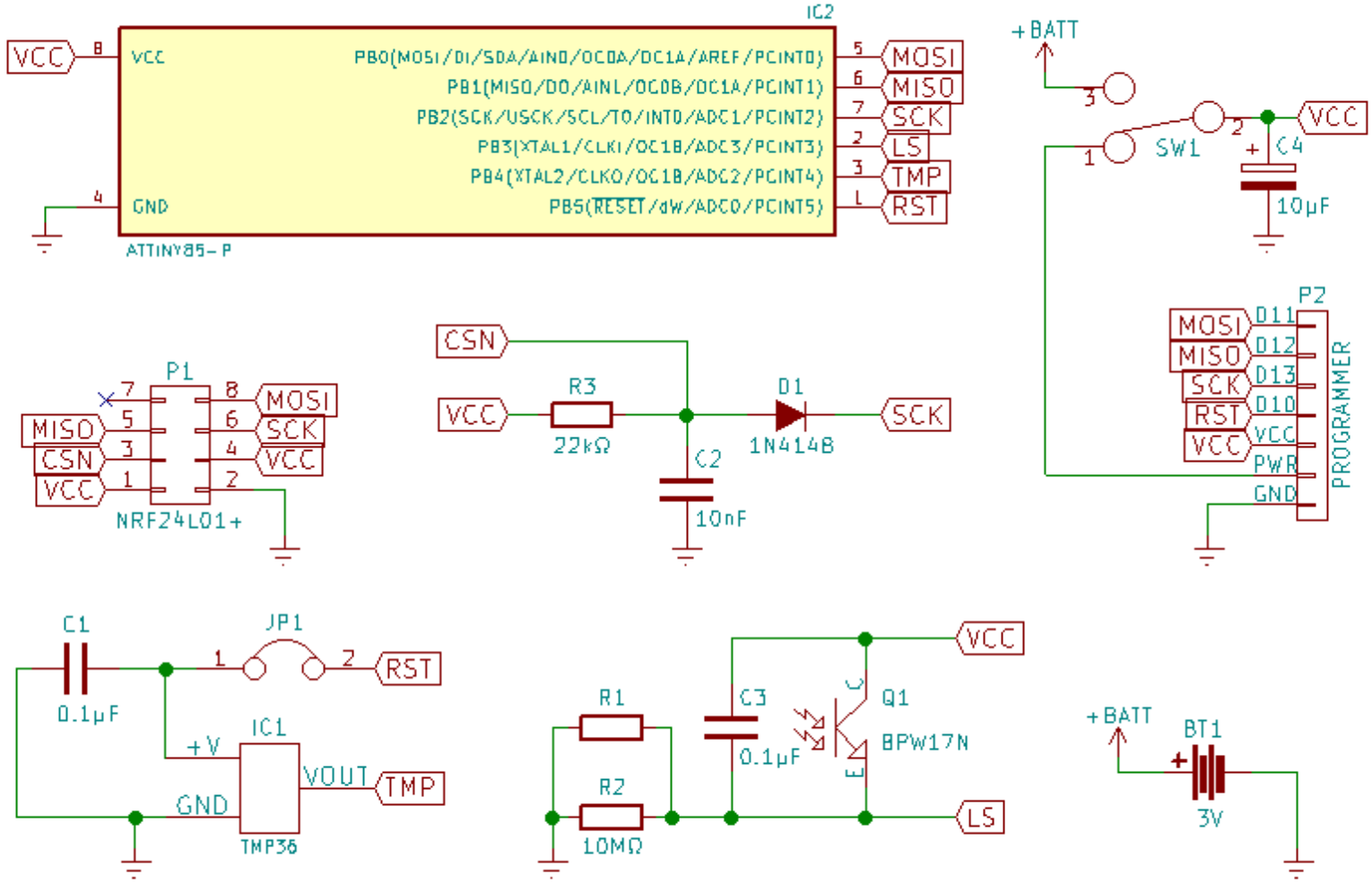
Если совсем кратко, то для начала работы над схемой в KiCad достаточно запустить редактор схемы и справа пройтись по иконкам, чтобы добавить: элементы , питание , провода и метки .

Метки нужны, чтобы не загромождать схему проводами. При разводке печатной платы все одинаковые метки необходимо будет соединить дорожками.

Если в доступных библиотеках нет необходимого элемента его можно создать с нуля либо изменить один из существующих. В последнем случае достаточно поставить курсор на нужный элемент и нажать Ctrl+E (редактировать элемент в редакторе) и в новом окне: создать элемент из текущего , отредактировать его внешний вид и свойства , затем сохранить в новую библиотеку .

Чтобы созданную библиотеку можно было использовать, её необходимо подключить в **Preferences** → **Component Libraries**, воспользовавшись кнопкой **Add** в верхней части. Для добавления в библиотеку других элементов, её нужно выбрать и после добавления сохранить .

Итоговая схема:



По сути, устройство состоит из трех независимых частей. Проверив их по отдельности вполне можно ожидать, что в сборе всё заработает как было задумано. Прототип как раз и нужен, чтобы проверить насколько данное ожидание соответствует действительности. Ниже, в комментариях к схеме, я остановлюсь на том, что пошло не так.

Датчик температуры TMP36

В рабочем режиме питание на датчик подается с порта PB5 микроконтроллера. В отладочной конфигурации подать питание на датчик можно установив перемычку между выводами VCC и D10 контактной площадки P2. Сигнал RESET активен при низком уровне и высокий уровень не мешает работе микроконтроллера и даже увеличивает стабильность по сравнению с висящим в воздухе RESET. Для съема данных с датчика используется аналоговый порт ADC2.

Засада ждала в виде ёмкости C1, которая была подключена непосредственно к RESET микроконтроллера. Понятно, что емкость на RESET снижает время активности сигнала или даже блокирует его, мешая программированию устройства, поэтому до сборки был проведен эксперимент, но как оказалось не достаточно чистый. Дело в том, что если на работающую схему добавить данный конденсатор, то процесс проходит успешно ровно один раз, если включить схему с ним — то не разу. Пока дело дошло до конденсатора, была впаяна перемычка между D1 и SCK, очевидно, что эта часть схемы не мешает установке на SCK низкого уровня, но подозреваемых больше не было. После установки перемычки между C1 и землей, программировать устройство стало можно, но только с питанием 5V. Однако в инструкции указано 1.8-5.5V и эксперименты с этим микроконтроллером и питанием 3V были успешными. Оказалось что на питании TMP36 есть встроенная ёмкость 10nF и этого достаточно, чтобы блокировать процесс при низком напряжении. В результате перемычка встала на своё место.

Радиомодуль NRF24L01+

Нумерация выводов у штыревых разъёмов в KiCad не соответствует принятой для данного модуля, поэтому нумерация здесь зеркальная относительно указанной в инструкции для NRF24L01+. Для работы радиомодуля в 3х пиновой конфигурации необходимо поменять местами выводы MISO и MOSI, и добавить часть схемы справа, состоящую из элементов: R3, D1 и C2. Схема данной конфигураци взята отсюда nRF24L01+ with ATtiny85 3 Pins.

Работа данной схема возможна по тому, что на CE (chip enable) можно сразу установить высокий уровень, а CSN (chip select) нужен только для передачи модулю сигнала о начале новой команды, что достигается сбросом высокого уровня в низкий. Соответственно, CSN можно объединить с одним из оставшихся выводов, в данном случае SCK (clock). Штатные импульсы SCK не успевают зарядить конденсатор C2 и на CSN остается низкий уровень, перед началом передачи команды нужно установить на SCK высокий уровень на более длительное время, чтобы получить высокий уровень на CSN. С чем связанна необходимость поменять местами MISO и MOSI я не понял, видимо так было удобно.

Для стабильной работы модуля желательно наличие ёмкости 1-10μF на выводах питания модуля, для этих целей на схеме предусмотрен конденсатор C4.

После устранения проблем с датчиком температуры, первым делом была проверена возможность прошить ATtiny85 с установленным радиомодулем. Процесс не пошёл, avrdude ругнулся на не верную сигнатуру, но я проявил упорство и запустил прошивку… и в итоге тоже ошибка. В результате эксперимента радиомодуль не пострадал, правда и шансов что-то в него прошить не было из-за модификаций приведенных выше. А вот микроконтроллер пришлось реанимировать с помощью высоковольтного программатора. Идея оказалась неудачной, но попробовать стоило.

Микроконтроллер ATtiny85

В процессе тестирования микроконтроллер успешно работал от внутреннего осциллятора без делителя на частоте 8MHz, однако с непосредственно подключенным радиомодулем его работа стала не стабильной. В том смысле, что если соединить радиомодуль с разъёмом проводами длиной 30cm, всё работает нормально, а с модулем, установленным в разъем, что-либо отправить невозможно. Все попытки стабилизировать систему дополнительными конденсаторами, ровно как и сделать её нестабильной, разместив вынесенный радиомодуль вплотную к микроконтроллеру, ни к чему не привели. В результате пришлось включить делитель, что вернуло микроконтроллер к заводским настройкам. Работа ATtiny85 на частоте 1MHz каких либо сложностей не вызвала, но заметно сократила ожидаемое время работы от батареи. Подробнее об этом в конце статьи.

Датчик освещения на BPW17N

С этим датчиком всё относительно просто, на него попадает только искусственное освещение, а регистрировать он должен только моменты его включения и отключения. Для максимально быстрой реакции на событие датчик генерирует прерывание на порту PCINT3 (Pin Change Interrupt).

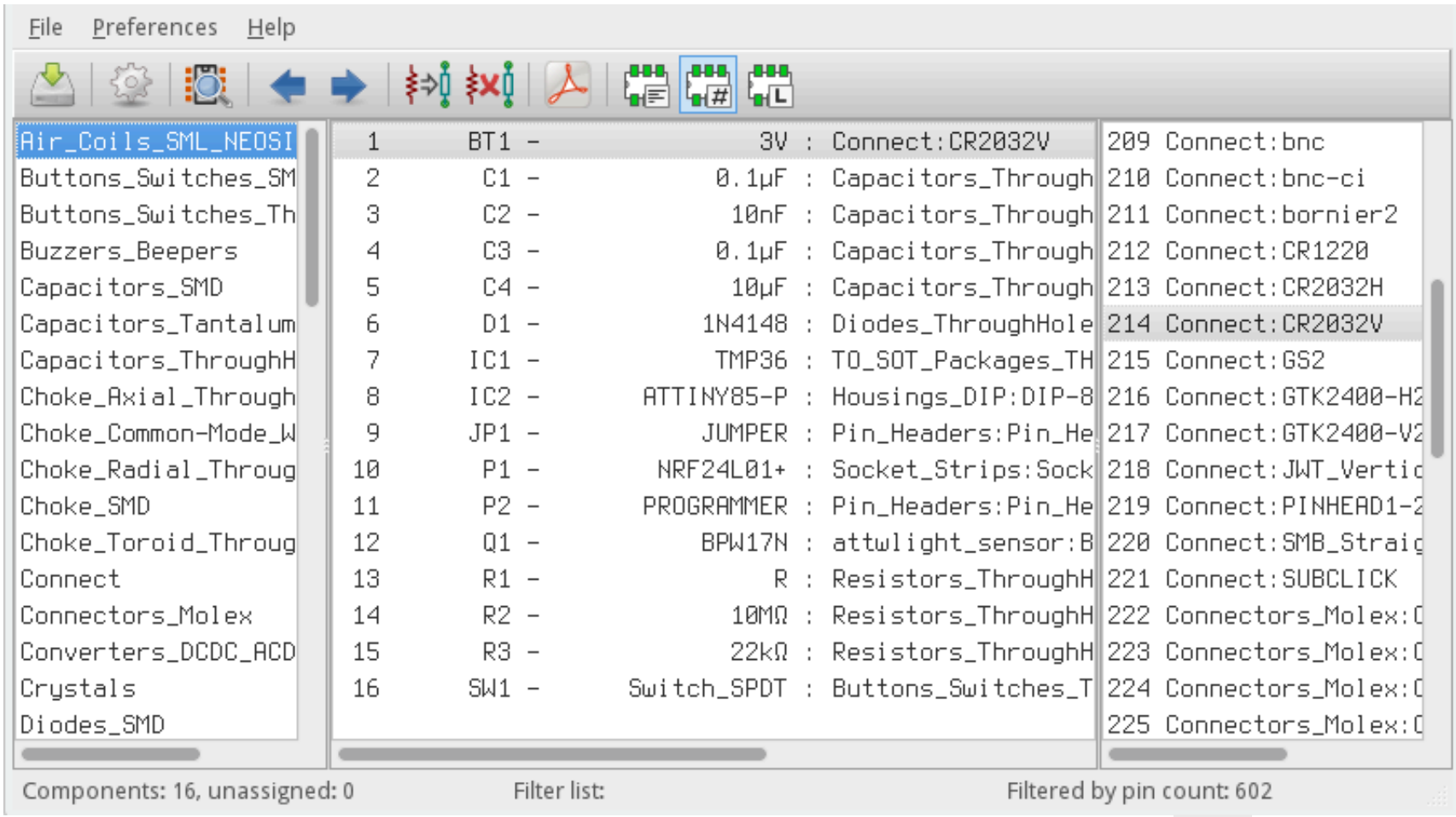
Элементы R1 и C3 подстроечные, для них предусмотрено место на плате, но они не распаяны. C3 параллельно с фототранзистором Q1 может понадобиться, чтобы исключить дребезг сигнала. Проблема в том, что стандартный для таких случаев конденсатор 0.1μF, разряжается через резистор 10MΩ около секунды, что не приемлемо для регистрации события — свет мигнул. Соответственно, можно установить конденсатор меньшей ёмкости и/или уменьшить сопротивление резистора R2, но сначала нужно проверить, есть ли в этом смысл.

Поскольку здесь используется прерывание может показаться, что в схеме нужен компаратор, но эти мысли сразу отпадают, как только становится понятно сколько эта штука потребляет. Правда и потребление микроконтроллера, при уровне сигнала вблизи центра рабочего диапазона, подскакивает с 7μA примерно до 260μA в режиме сна. Однако данную проблему проще решить программно, а именно, после срабатывания прерывания (PCINT3 или WDT), можно проверить напряжение на аналоговом входе ADC3, если оно неприемлемо, сообщить о проблеме и пропустить активацию PCINT3. В любом случае это нештатная ситуация, которая требует внимания.

Макетная плата

Готовую схему необходимо развести на плате. Это также можно сделать в KiCad. Для этого сначала требуется завершить работу со схемой:

1. Промаркировать элементы на схеме , все настройки можно оставить по умолчанию.
2. Сопоставить элементы схемы с посадочными местами на плате . Появится следующее окно:



   Здесь для наглядности следует включить просмотр выбранного элемента .

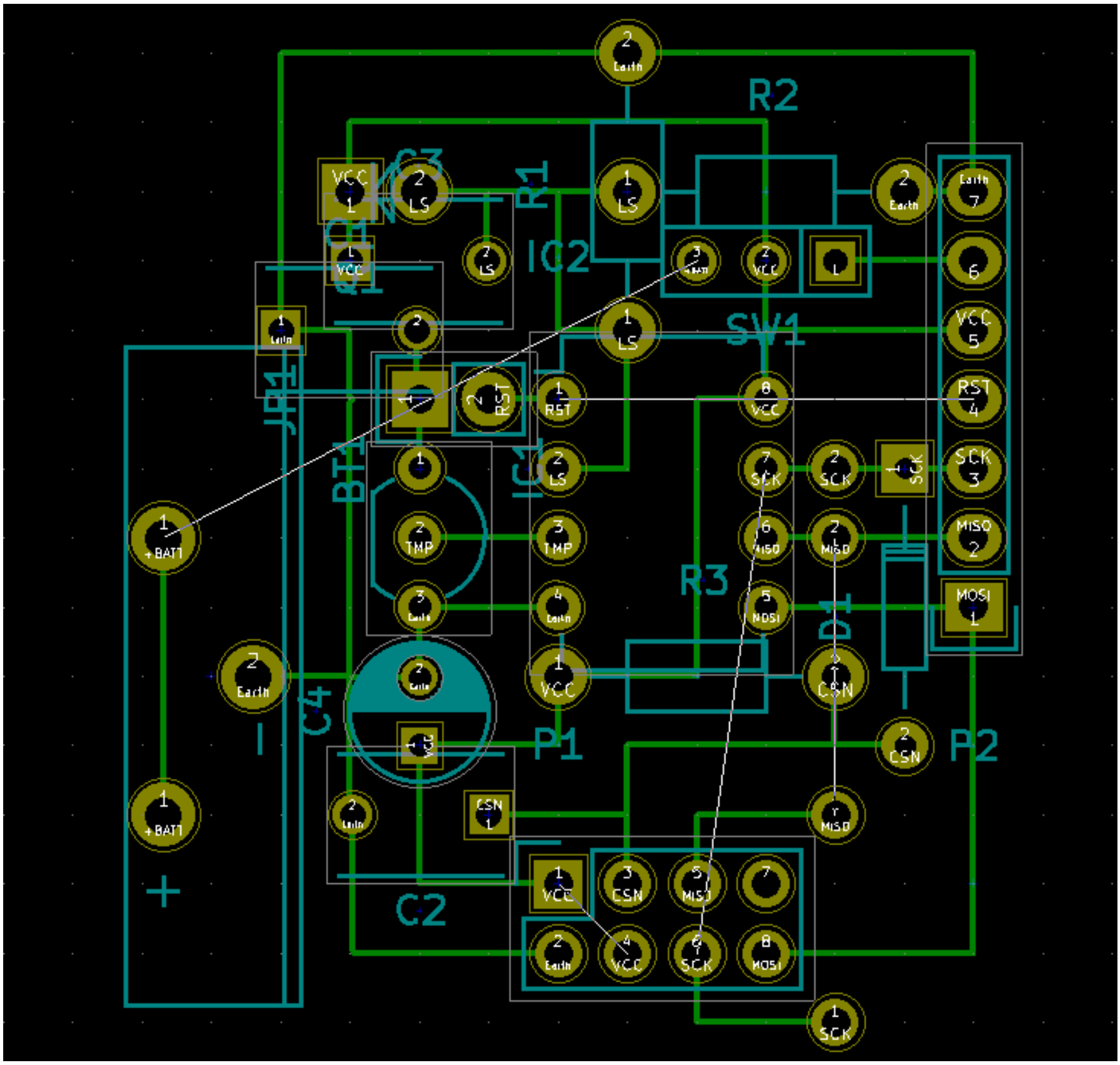
   Поскольку предстоит работать с макетной платой необходимо следить, чтобы все посадочные места укладывались в сетку 1/10" (2.54mm). Если требуемый элемент найти не удаётся, проще назначить похожий, затем, в редакторе печатной платы, изменить его (Ctrl+E) и записать в свою библиотеку.

   Как только все посадочные места будут назначены нужно сохранить результат и вернуться в редактор схемы.
3. Сгенерировать netlist , настройки по умолчанию вполне приемлемы.

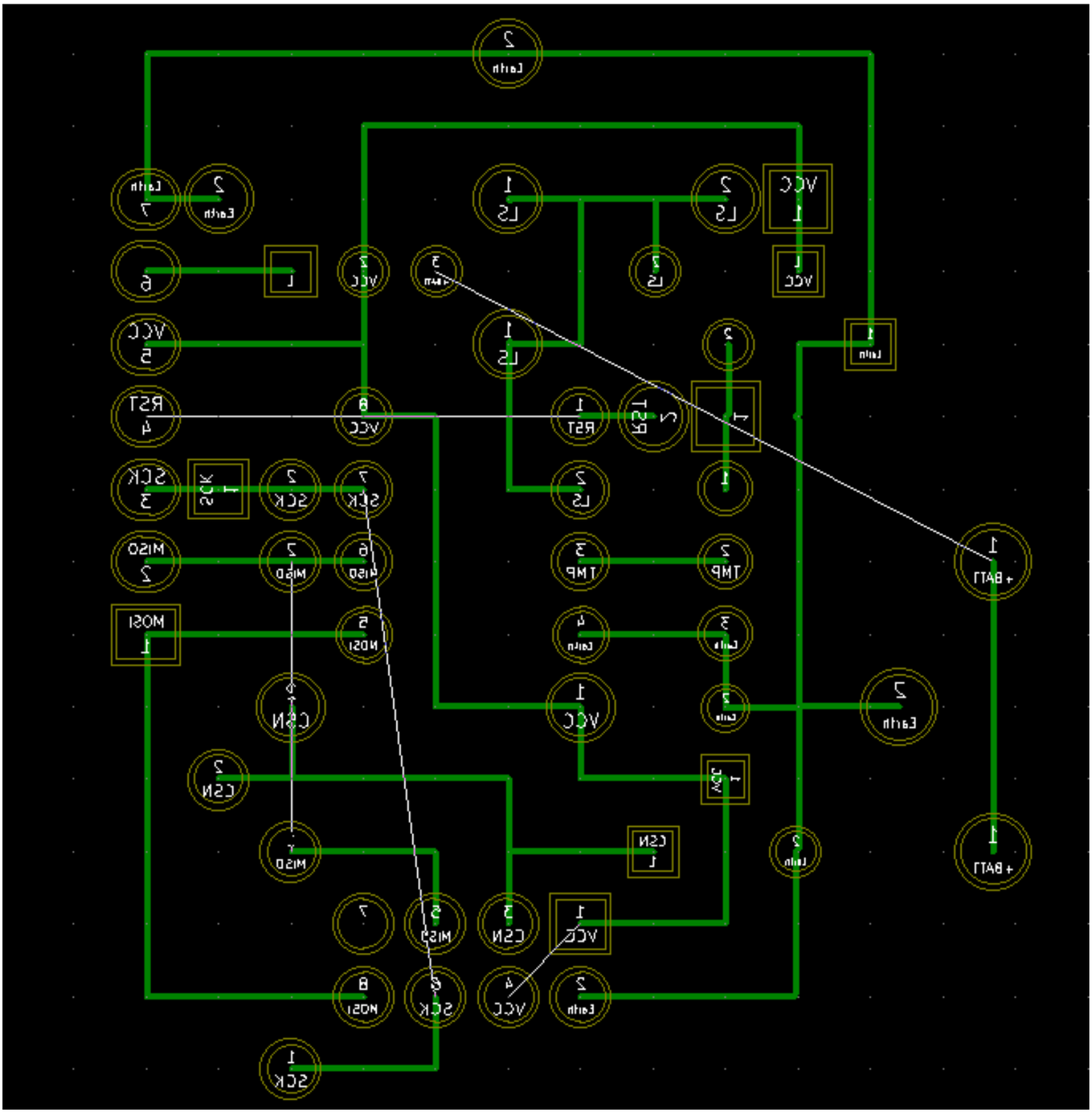
Редактор схемы можно закрыть и перейти к редактору печатной платы . Здесь нажать , затем загрузить ранее созданный netlist (Read Current Netlist), убедиться, что не было ошибок и закрыть окно. Если схема была изменена или элементам были назначены другие посадочные места, необходимо обновить netlist и загрузить его здесь снова.

При первой загрузке netlist все посадочные места окажутся стопкой в одной куче. Сначала необходимо переключить сетку на 1/10" (2.54mm), растащить элементы (используя кнопку m), убрать лишние надписи **Visibles** → **Render** → **Values** и, пользуясь подсказками редактора, соединить элементы дорожками.

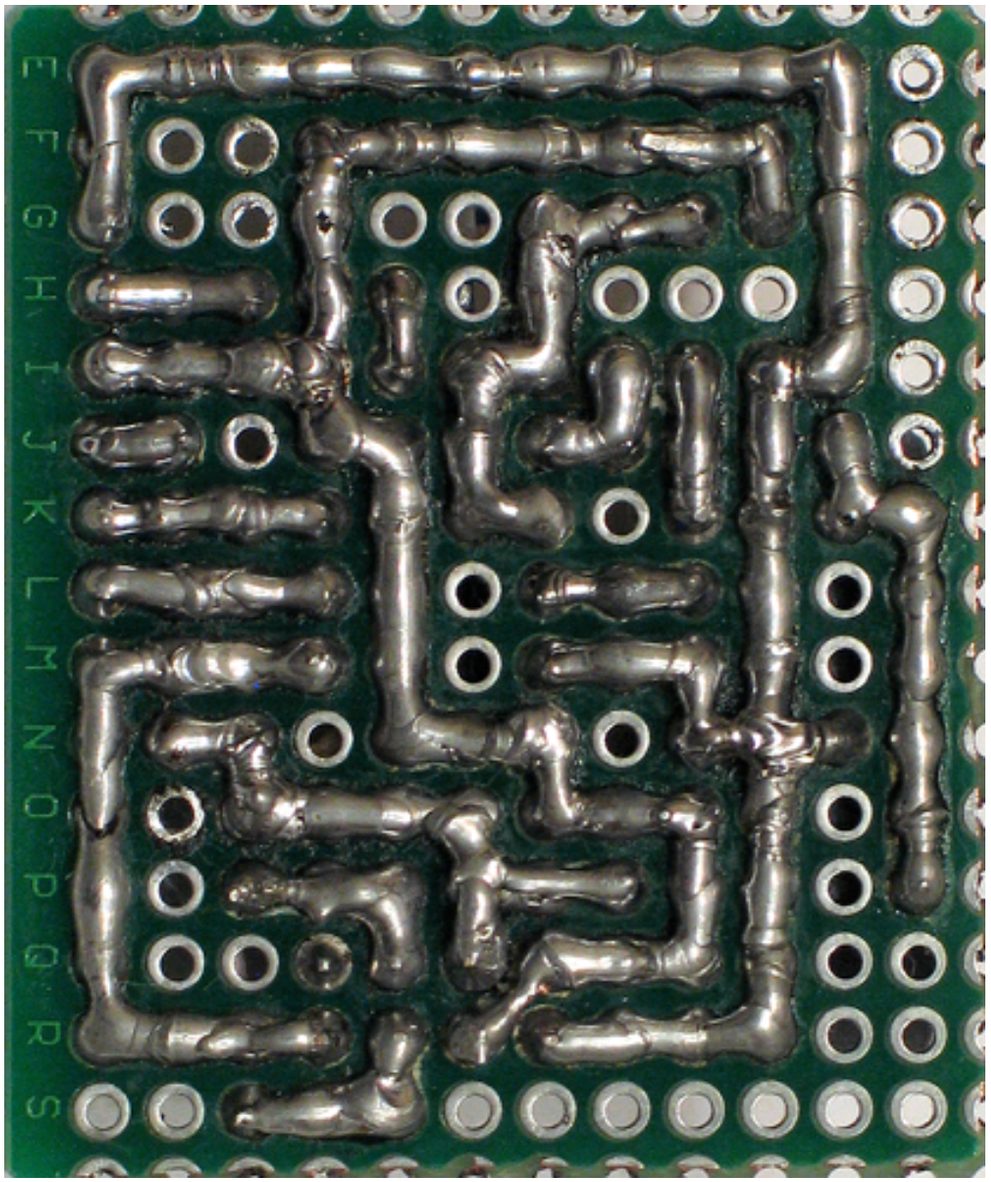
Результат будет выглядеть примерно так:



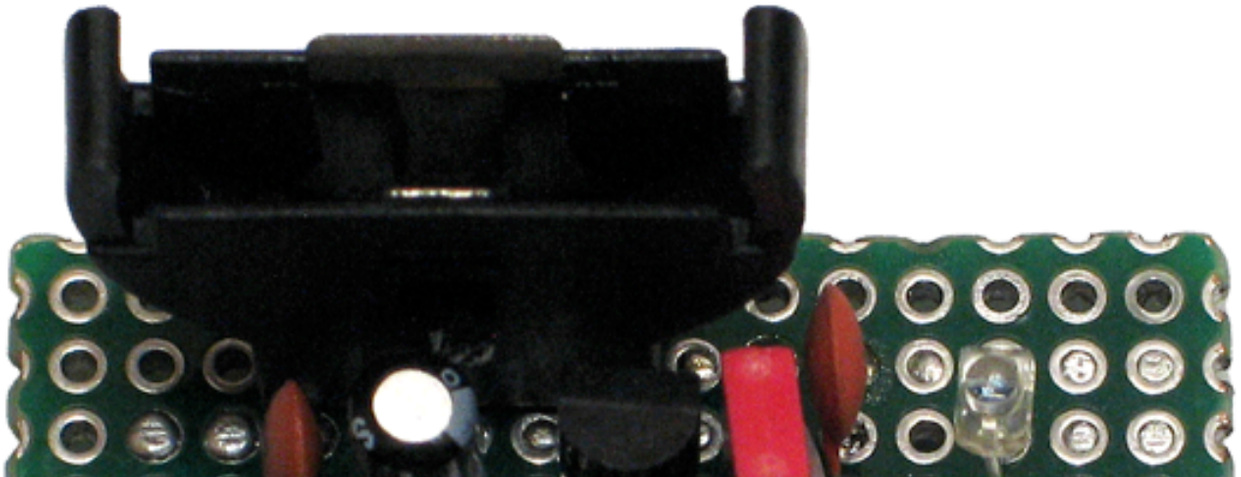
Оставшиеся белые линии, указывают какие посадочные места необходимо соединить проводами. К сожалению в KiCad нет способа вывести результат, так чтобы его было удобно воспроизвести на макетной плате. Но в данном случае макетная плата получилась не большой, и с ней можно поступить просто: отключить лишние надписи
Visibles → **Render** → **Footprints Front**, снять скриншот и зеркально отразить его по вертикали.

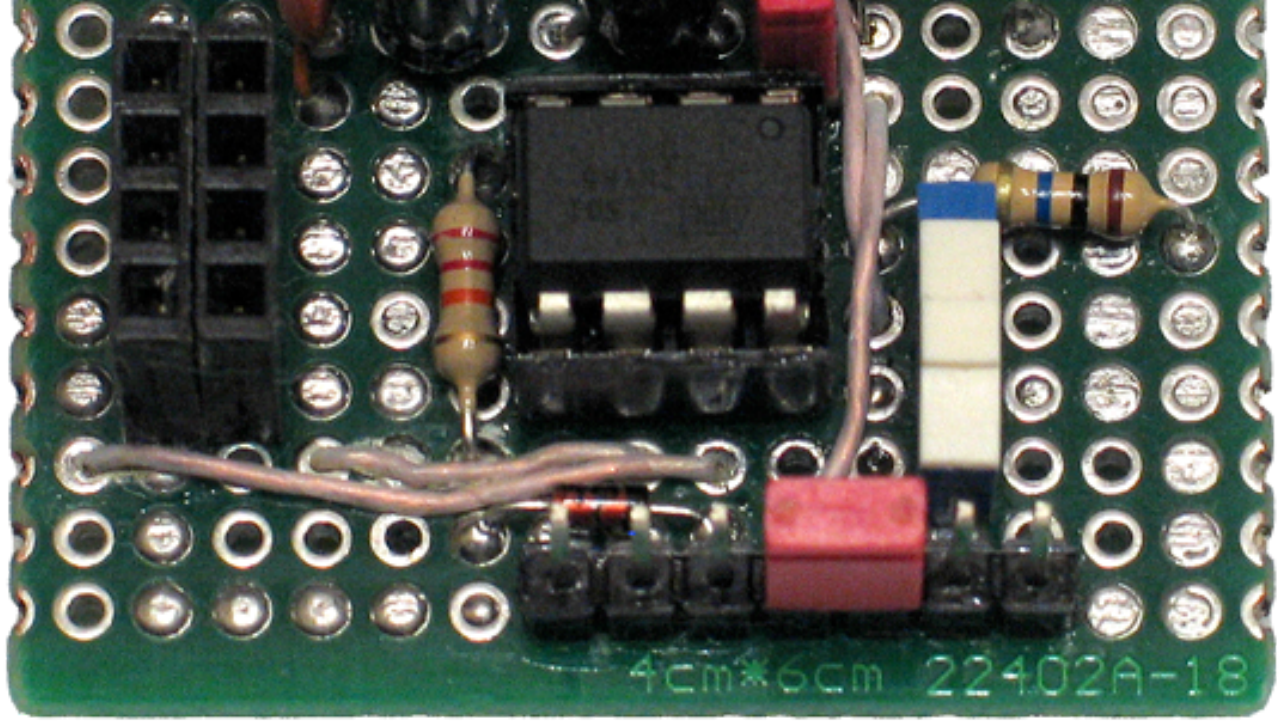


Готовый макет:



Тут два несоответствия со схемой выше: не распаяны подстроечные элементы R1,C3 и диод короче на одно отверстие.





Плата немного побита жёсткой отладкой, но на то она и макетная. Большинство других вариантов, после всех модификаций, выглядело бы хуже.

Подготовка IDE и программатора

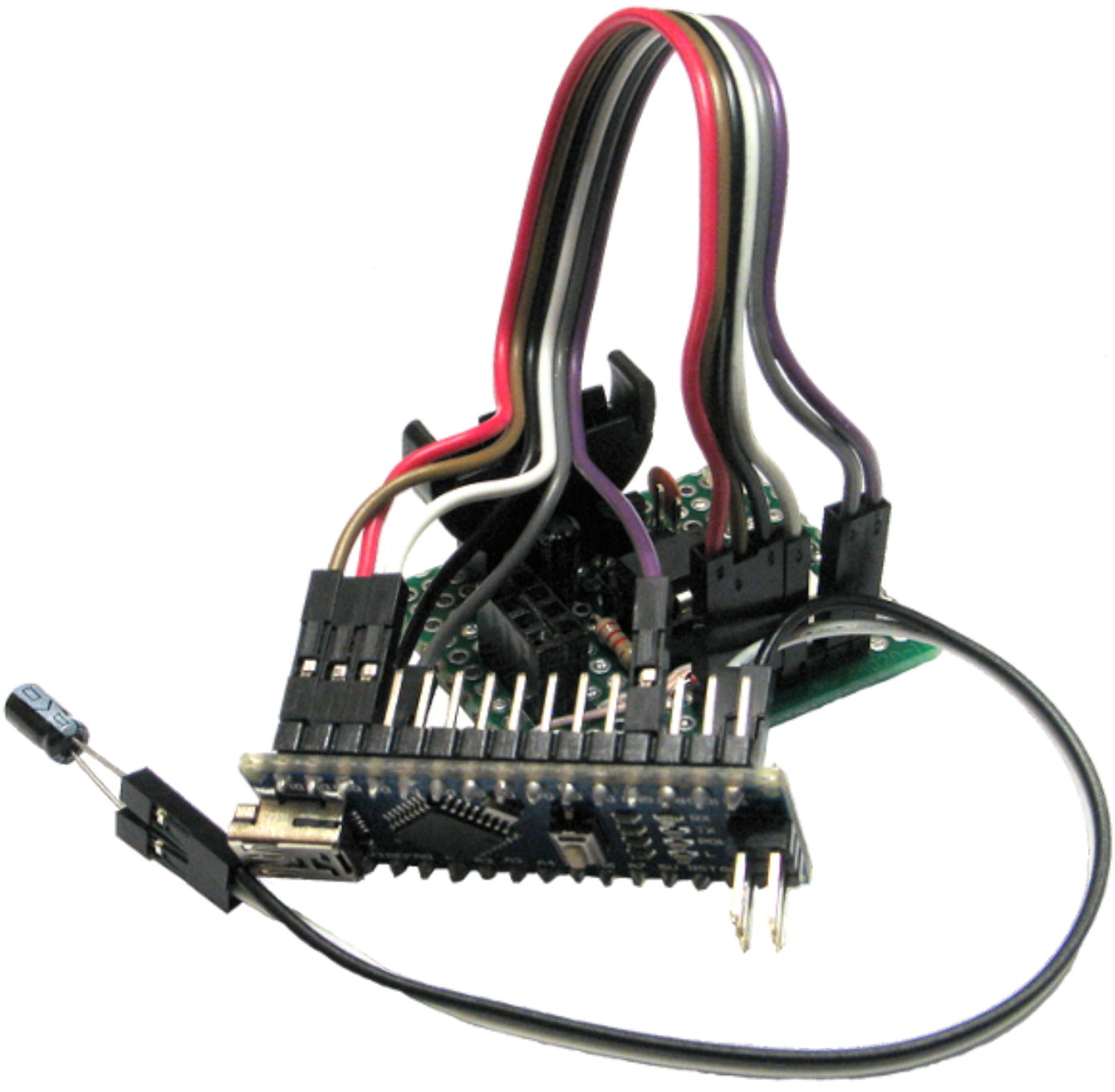
Arduino IDE наиболее доступное кроссплатформенное программное обеспечение для программирования AVR микроконтроллеров и вполне удобное для небольших проектов.

Не смотря на то, что Arduino IDE официально не поддерживает ATtiny85, существует штатный способ добавить данный микроконтроллер в список поддерживаемых. Как это сделать, подробно описано по ссылке Programming an ATtiny w/ Arduino 1.6. Однако данный способ годится только для непосредственного программирования ATtiny85 и не позволяет использовать Arduino код. С одной стороны это хорошо, код чище будет, но задача в данном случае — разработать скетч демонстрирующий способность устройства функционировать как было задумано, и такая оптимизация здесь преждевременна.

Значит нужно Arduino ядро для ATtiny85, его можно взять здесь arduino-tiny, версия для Arduino 1.5 подойдет. Доступны будут не все функции, но базовый набор, включая программный последовательный порт, для отладки, можно будет использовать. К сожалению в актуальной, на данный момент, Arduino IDE v1.6.6 что-то сломали, и с этим ядром необходимо использовать версию 1.6.5.

- Подключение ядра к Arduino IDE v1.6.5

Осталось приготовить программатор. Для этого совсем не обязательно использовать специальное устройство, можно в ближайшую Arduino прошить скетч из **File** → **Examples** → **ArduinoISP**, затем соединить с сенсором, как указанно на принципиальной схеме, и не забыть про конденсатор 10μF, который необходимо установить между RST и GND выводами Arduino.



Программировать устройство нужно без установленных радиомодуля и перемычек. В меню **Tools** → **Board** нужно выбрать конфигурацию микроконтроллера, в данном случае: ATtiny85 @ 1 MHz (internal oscillator; BOD disabled), в меню **Tools** → **Port** выбрать USB порт программатора и в меню **Tools** → **Programmer** указать тип программатора, в данном случае: Arduino as ISP, затем воспользоваться кнопкой Upload.

Программное обеспечение

Тестовый скетч для работы с сенсором получился довольно интересным, в нем можно найти широкий набор примеров, а именно:

- Работа с радиомодулем NRF24L01+;
- Измерение напряжения питания средствами микроконтроллера;
- Перевод микроконтроллера в режим сна;
- Пробуждение по таймеру WDT (WatchDog Timer);
- Пробуждение по Pin Change прерыванию PCINT;
- Работа с аналоговыми и цифровыми портами.

Чтобы скомпилировать скетч необходимы сторонние библиотеки:

- TMRh20/RF24
- TMRh20/RF24Network

Это оптимизированные версии библиотек для работы с радиомодулем NRF24L01+, которые подходят для микроконтроллера ATtiny85. Их необходимо скачать и распаковать в папку libraries в директории со скетчами Arduino.

Ниже будут приведены наиболее интересные куски кода с подробными комментариями. Полностью скетч можно найти на GitHub в папке **attwlight_sensor**, также в папке **attwlight_rx** находится скетч для приёмника.

Перевод микроконтроллера в режим сна

Функция void sleep() вызывается в главном цикле void loop() и останавливает его до тех пор пока не сработает одно из прерываний PCINIT3 либо WDT. Затем выполняется полезная работа и снова вызывается данная функция.

Функция содержит обращения к регистрам микроконтроллера, и что при этом происходит не всегда очевидно. Чтобы с этим разобраться нужно заглянуть в документацию на микроконтроллер ATtiny85 datasheet раздел «23. Register Summary», в нем приведен полный список регистров с прямой ссылкой на страницу с подробным описанием для чего предназначен тот или иной бит.

```
#include <avr/wdt.h>
#include <avr/sleep.h>
#include <avr/interrupt.h>

#define WDTO_INFINITE 255
#define SLEEP_PERIOD WDTO_8S
#define SKIP_WDT_WAKEUPS 14 // x SLEEP_PERIOD + SLEEP_PERIOD (14 ~= 120 sec)

// Вектор (функция) прерывания PCINT, у ATtiny85 один порт (B), следовательно только один в
ISR(PCINT0_vect)
{
  light_pcint = true;
}

// Вектор прерывания таймера WDT
ISR(WDT_vect)
{
  wakeup_counter++;
}

void sleep()
{
  GIMSK = _BV(PCIE); // Включить Pin Change прерывания
  if (SLEEP_PERIOD == WDTO_INFINITE || payload.light != LIGHT_FUZZY)
    PCMSK |= _BV(LIGHT_PIN); // PCINT3; включить если нет проблем с освещением или иначе не
  ADCSRA &= ~_BV(ADEN); // отключить ADC; уменьшает энергопотребление

  if (SLEEP_PERIOD != WDTO_INFINITE) {
    wdt_enable(SLEEP_PERIOD); // установить таймер
    WDTCR |= _BV(WDIE); // включить прерывания от таймера; фикс для ATtiny85
  }

  // Установить режим сна Power-down; MCUSR &= ~_BV(SM0); MCUSR |= _BV(SM1);
  set_sleep_mode(SLEEP_MODE_PWR_DOWN);
  sleep_enable(); // разрешить режим сна; MCUSR |= _BV(SE);
```

```
  sei(); // включить прерывания

  sleep_cpu(); // заснуть

  cli(); // отключить прерывания; для безопасного отключения PCINT3
  PCMSK &= ~_BV(LIGHT_PIN); // PCINT3; отключить
  sleep_disable(); // запретить режим сна; MCUSR &= ~_BV(SE);
  ADCSRA |= _BV(ADEN); // включить ADC
  sei(); // включить прерывания; иначе таймеры не будут работать
}
```

Опрос датчика освещения

Особенности работы с данным датчиком уже были разобраны в разделе «Принципиальная схема». Отмечу только, что параметры здесь выбраны так, чтобы потребление микроконтроллера в режиме сна не превышало 20μA.

```
#define FUZZY_VOLTAGE 0.4 // ширина зоны неприемлемого напряжения в середине, включая концы

const unsigned int fuzzy_start = int(1024.0 * (1.0 - FUZZY_VOLTAGE) / 2.0);
const unsigned int fuzzy_stop = 1023 - fuzzy_start;

light_t readLightStatus(unsigned int ms)
{
  delay(ms); // если нужно время на стабилизацию
  int input = analogRead(LIGHT_PIN); // ADC3

  // оценка надёжности результата
  if (input >= fuzzy_start && input <= fuzzy_stop)
    return LIGHT_FUZZY;
  if (input < fuzzy_start)
    return LIGHT_OFF;
  if (input > fuzzy_stop)
    return LIGHT_ON;

  // заглушка; на случай ошибки в коде
  return LIGHT_UNKNOWN;
}
```

Измерение напряжения питания

Если не использовать вывод AREF микроконтроллера, который предназначен для задания опорного напряжения, все измерения на аналоговом порту будут привязаны к напряжению питания. Батарея не стабилизированный источник, следовательно для корректного считывания показаний датчика температуры, необходим способ измерять напряжение питания в процессе работы. Напряжение питания также является хорошим индикатором состояния батареи.

Ниже приведен код, который позволяет произвести требуемые измерения полагаясь на стабильность внутреннего опорного напряжения Vbg (Bandgap reference voltage) равного 1.1V. Согласно спецификации Vbg может находится в пределах 1.0-1.2V, поэтому необходимо убедиться в точности измерений и при необходимости скорректировать константу 1125300.

```
long readVcc()
{
  // Установка референса в Vcc и измерение внутреннего Vbg == 1.1V
  ADMUX = _BV(MUX3) | _BV(MUX2);
  delay(2); // требуется минимум 1ms для стабилизации
  ADCSRA |= _BV(ADSC); // ADC Start conversion - начать опрос
  while (bit_is_set(ADCSRA, ADSC)); // ADSC == 1 пока опрос активен

  // Чтение ADCL блокирует ADCH, чтение ADCH разблокирует оба регистра => ADCL читать первы
  uint8_t low  = ADCL; // младший регистр данных ADC
  uint8_t high = ADCH; // старший регистр данных ADC

  long result = (high << 8) | low;

  result = 1125300L / result; // 1125300 = 1.1*1023*1000
  return result; // Vcc в милливольтах
}
```

Измерение температуры

Получив напряжение питания, легко вычислить точное напряжение на аналоговом входе микроконтроллера и перевести его в температуру.

```
float readTmp36(long vcc)
{
  int input = analogRead(TMP36_PIN); // напряжение относительно refVolts
  float refVolts = float(vcc) / 1000.0;
  float voltage = float(input) * refVolts / 1024.0; // фактическое напряжение
  float temperatureC = (voltage - 0.5) * 100.0; // смещение 500mV & 10mV == 1°C
  return temperatureC;
}
```

Остальная часть скетча нужна, чтобы собрать данные вместе и отправить их наиболее оптимальным образом, с точки зрения энергопотребления. Ничего интересного там нет, обычная логика, надеюсь, комментариев в коде достаточно, чтобы разобраться как всё работает.

Если скомпилировать данный код, можно увидеть, что он свободно умещается в 8KiB и даже есть куда развернуться:

```
Sketch uses 7,120 bytes (86%) of program storage space. Maximum is 8,192 bytes.
Global variables use 227 bytes (44%) of dynamic memory, leaving 285 bytes for local variabl
```

Осталось проверить, что всё работает как было задумано, и можно прошивать Fuses переключающие RESET в порт ввода/вывода. Готовой конфигурации в Arduino IDE для этого нет, поэтому нужно воспользоваться командной строкой:

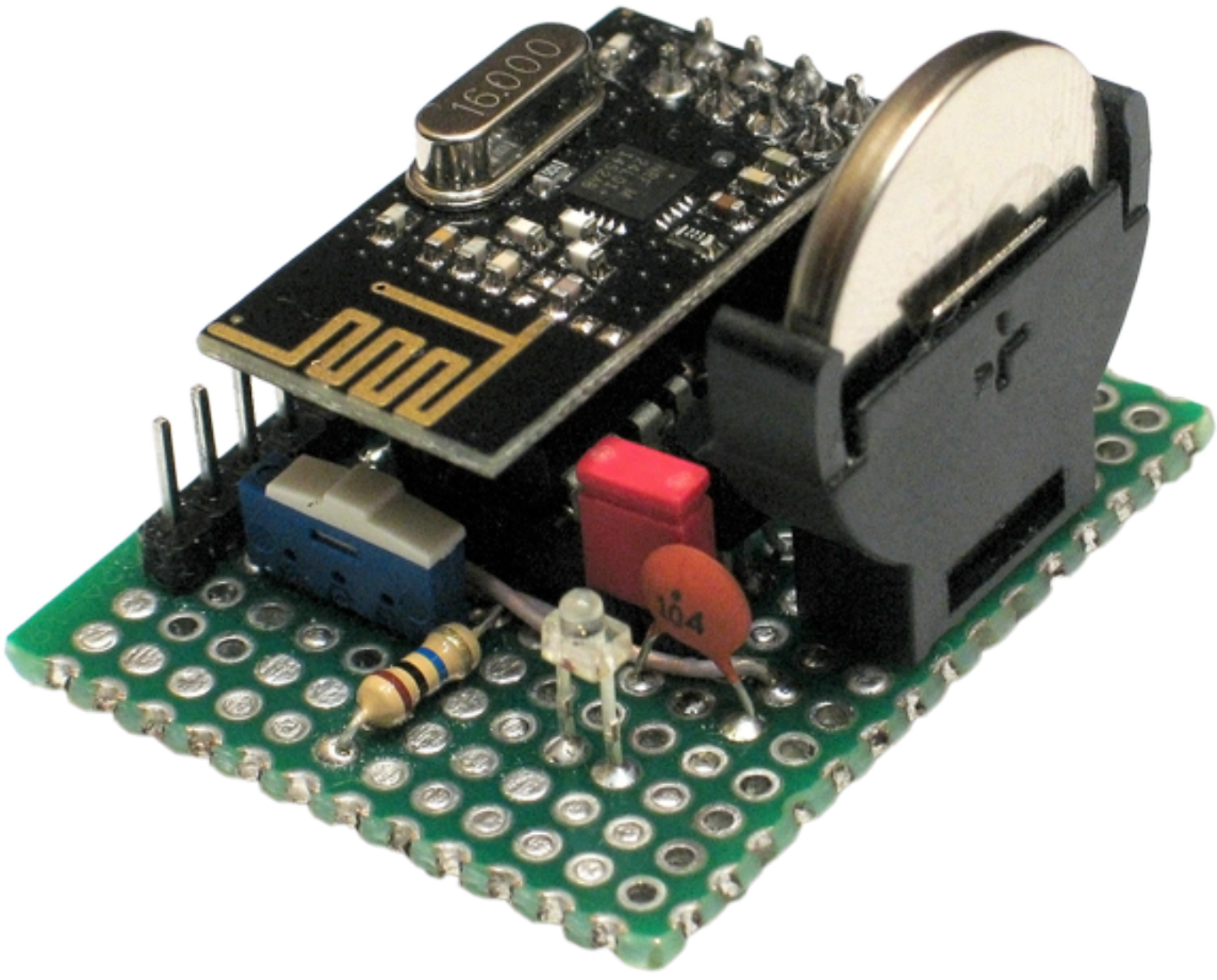
```
avrdude -c stk500v1 -p attiny85 -P /dev/ttyUSB0 -v -C /etc/avrdude.conf -b 19200 -U lfuse:w
```

- **avrdude** — команда из пакета Arduino IDE находящаяся в папке **hardware/tools/avr/bin/**;
- **/dev/ttyUSB0** — устройство программатора;
- **/etc/avrdude.conf** — ссылка на файл находящийся в директории с Arduino IDE по адресу

hardware/tools/avr/etc/avrdude.conf.

Что из себя представляют данные Fuses, можно посмотреть в калькуляторе Engbedded Atmel AVR® Fuse Calculator.

Если кратко, то эти настройки предписывают: использовать встроенный осциллятор с делителем частоты на 8 (итоговая частота 1MHz); использовать стандартные задержки при старте, необходимые для стабилизации осциллятора; отключить BOD (Brown-out Detector) — перезагрузка при низком напряжении питания для защиты памяти от записи мусора; включить возможность считывать прошивку; и собственно отключить RESET.



Энергопотребление

Пришло время измерить энергопотребление получившегося устройства. Вводные следующие: данные передается каждые 120sec, если пришло прерывание от датчика освещения, то немедленно; событиями от датчика освещения можно пренебречь; ёмкость батареи 200mAh.

Частота	1MHz	8MHz
Радиомодуль (потребление)	15.7mA	18.3mA
Микроконтроллер (потребление)	1.1mA	4.3mA
Режим сна (потребление)	7μA	7μA
Радиомодуль (активен за цикл)	51ms	16ms
Микроконтроллер (активен за цикл)	15ms	4ms
Оценка времени работы (дней)	513	739

Как видно, тут было ради чего бороться. К сожалению, стабильности на 8MHz добиться не удалось. Видимо без внешнего осциллятора, это не реально.

Все исходные коды проекта доступны на GitHub

Теги: attiny85, прототип, sleep mode, прерывания, nrf24l01+, kicad, датчик, arduino, температура, освещение

Хабы: DIY или Сделай сам, Электроника для начинающих

Редакторский дайджест

Присылаем лучшие статьи раз в месяц

Электропочта

Оставляя свою почту, я принимаю Политику конфиденциальности и даю согласие на получение рассылок

21 Карма

0 Рейтинг

Александр Борисенко @custos

Инженер-программист компьютерных систем и сетей

Подписаться

Комментарии 22

dimao79
17 дек 2015 в 13:47

Разводить макетку в трассировщике? Использовать ардуину как программатор? Меня сложно удивить, но тут — получилось.

Интересно, когда ардуину начнут использовать как блок питания?

Ответить

custos
17 дек 2015 в 14:16

> Разводить макетку в трассировщике?

Высоковольтный программатор я на бумаге разводил, не понравилось, а обвесить макетку пучком проводов, эстетика не позволяет, а так всё относительно быстро получается и выглядит неплохо.

> Интересно, когда ардуину начнут использовать как блок питания?

Смешно конечно, но в процессе работы с этим проектом я всего два раза доставал блок питания, в остальное время питался от Arduino UNO, и от батарейки в самом конце. Не люблю лишние сущности, если то что есть справляется, зачем покупать/паять дополнительное железо.

Ответить

dimao79
17 дек 2015 в 14:33

> Высоковольтный программатор я на бумаге разводил, не понравилось, а обвесить макетку пучком проводов, эстетика не позволяет, а так всё относительно быстро получается и выглядит неплохо.

Ну так сделайте уже следующий шаг — сделайте нормальную печатку либо сами, либо закажите где нибудь. Тем более, насколько я понял — это не разовое изделие.

Ответить

custos
17 дек 2015 в 14:59

Данное изделие разовое, как раз посмотреть нужно оно или нет. В остальном Вы правы, с макетками надо завязывать. Вероятно следующие изделие буду ЛУТ-ом разводить. Спасибо за замечание, мотивирует.

Видимо тут нужно пояснение… Электроника для меня скорее развлечение, очень давно, ещё в школе, разбирался с этим подробно, в итоге периодически накатывает, но не достаточно сильно, чтобы развернуться «по взрослому».

Ответить

KonstantinSoloviov
19 дек 2015 в 00:09

Первый раз вижу такую аккуратную пайке на макетке. Интересная техника.

ЛУТ, конечно, попробуйте, но используйте то что Вам удобнее.
Макетки, бредборды, SMD все имеет свои достоинства и недостатки, не стоит зацикливаться

Ответить

gorbln
17 дек 2015 в 16:38

> Цель — создать сенсор работающий, условно говоря, в коробке с искусственным освещением и передающий температуру и статус освещения с немедленной реакцией на изменение освещения

Извините, я настолько испорчен, или это действительно монитор для гроубокса?

Ответить

custos
17 дек 2015 в 17:04

> это действительно монитор для гроубокса

Предполагалось какая-то автоматизация инкубатора для птиц/яиц, но там всё не серьезно, я сказал что могу, мне сказали — хотим… вряд-ли от этого будет польза, ну кроме этой статьи, разумеется.

Ответить

gorbln
17 дек 2015 в 18:11

Блин. Это было второй идеей. Всё-таки, я **настолько** испорчен. =)

Ответить

HWman
17 дек 2015 в 22:15

Я когда-то писал про восстановление фьюзов для ATtiny
geektimes.ru/post/255396

Ответить

custos
17 дек 2015 в 23:24

У вас, в конце статьи, ссылка на схему с умножителем (Charge Pump), которая была тут реализована… первые две фото после КДПВ, плюс две под спойлером.

Ответить

shadows
17 дек 2015 в 22:57

а можете пояснить почему время активности радиомодуля увеличелось в 3 раза по сравнению с 8мгц? в моем представлении в радиомудуль для отправки передается около 32 баит, и через SPI они должны уити за пару ms.
а также почему время активности микроконтролера изменилось не в 8 раз?

Ответить

custos
17 дек 2015 в 23:17

> почему время активности радиомодуля увеличелось в 3 раза по сравнению с 8мгц?

Время активности радиомодуля, это вот этот кусок кода:

```
radio.powerUp();
network.write(header,&payload,sizeof(payload));
radio.powerDown();
```

Основные задержки от powerUp/Down, теоретически он активен чуть меньше указанного времени, но не на много.

> а также почему время активности микроконтролера изменилось не в 8 раз?

Наличие delay в коде, на 2ms микроконтроллер спать ведь не отправишь.

Ответить

shadows
18 дек 2015 в 01:29

а какой тогда период времени понимается под «цикл»?
и как delay может влиять? он же в обоих случаях должен спать именно 2ms и не больше, и не меньше.
самое главное не ясно почему время работы радиомодуля 51ms, а контролера 15? контроллер же не может спать пока идет передача данных, он же ей собственно и управляет.
В целом интересно было бы узнать методику измерения/расчета тока и времени.

Ответить

shadows
18 дек 2015 в 01:43

посмотрел код и похоже библиотека rf24 использует программный SPI соответсвенно и стандартный digitalWrite который в силу свой универсальности тратит много лишних тактов на простое дерганье битов на порте, тогда задержки становятся понятными. Может можно адаптировать под digitalWriteFast или прямую работу с портами?

Ответить

custos
18 дек 2015 в 10:18

Если до оптимизации дойдет дело, библиотеки, конечно, нужно будет перетряхнуть основательно, а вероятнее всего, сделать необходимое «руками».

Ответить

custos
18 дек 2015 в 10:12

Всё верно, измерения были не прямые, из-за накладных расходов, польза их была-бы не большой либо измерять пришлось бы слишком много. Согласен что получилось грязно, но для грубой оценки вполне пригодно. Конкретно алгоритм был следующий.

Энергопотребление: радиомодуля — return в начале loop; микроконтроллера — return перед sleep() (после powerDown радиомодуля); режим сна — return после sleep().

Время работы: [1] uptime полного цикла сбора данных и отправки — millis в начале каждого цикла; [2] 10 циклов buildStat + один полный buildStat с отправкой данных; [3] рабочий режим — полный uptime за 120s включая всю промежуточную работы и финальный сбор данных и отправку.

В итоге было получено время активности радиомодуя как [1] — ([2] — [1])/10 и микроконтроллера как [3] — [активность радиомодуля].

Ответить

custos
17 дек 2015 в 23:59

Спасибо за сообщение, у меня была ошибка, активность микроконтроллера на 8MHz — 4ms, ожидаемое время изменилось несущественно 733 → 739 дня.

Ответить

monah_tuk
20 дек 2015 в 19:07

Не совсем в тему, но какое сечение МГФТ, что на макетке заметен?

Ответить

custos
20 дек 2015 в 19:19

0.12 кв.мм, что-то вроде стандарта по умолчанию.

Ответить

smart
21 дек 2015 в 09:21

Я правильно разглядел, что у Вас это все напрямую от батарейки питается? Можно присобачить step up конвертер и вытягивать из батарейки побольше. Я недавно вот такие штуки купил – очень мелкие и делают 3.3V из практически любого меньшего напряжения.

P.S. А вообще, дешевле и проще было бы собрать это все на ESP8266 через wifi. Только по энергопотреблению, вероятно, чуть хуже получилось бы (хотя она тоже умеет глубоко спать).

Ответить

custos
21 дек 2015 в 10:32

В данном DC-DC конвертере не видно, что за чип использован, но думаю это не важно, схема должна что-то потреблять и вряд-ли она целесообразна, чтобы отобрать у батарейки оставшиеся 40-50mAh. ESP8266 спит не плохо, но в режиме передачи потребляет 170mA, CR2032 это не вытянет в принципе.

Ответить

custos
21 дек 2015 в 10:31

сообщение выше

Ответить

Зарегистрируйтесь на Хабре, чтобы оставить комментарий

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

gmtd
15 часов назад

Вайбовый панк

Простой 1 мин 5.7K

+88 14 34

OyminiRole1776
18 часов назад

VS Code: 5 интересных функций и расширений, меняющих подход к разработке

Простой 4 мин 13K

Обзор

+52 167 26

frog
23 часа назад

Аппаратное демо

3 мин 2.6K

+40 25 5

MaFrance351
10 часов назад

Оживляем дисплей из кабины лифта

Средний 6 мин 1.5K

Кейс

+32 9 10

PaskalEnotov
6 часов назад

Как я научился обманывать холодильники — и сделал бизнес на 1,3 млн в месяц

Простой 11 мин 5.9K

Кейс

+30 10 54

inetstar
9 часов назад

Оптимизация производительности серверов на базе Ерус

Средний 8 мин 1.5K

Туториал

+29 15 1

ntsaplin
5 часов назад

Нас не заменят! Почему сисадмины не боятся конкуренции с ИИ

Простой 5 мин 1.4K

Мнение

+25 6 4

AlexxIT
10 часов назад

Всё о стриминге музыки на Яндекс Станции и не только

Простой 5 мин 2.6K

+20 21 0

Catx2
4 часа назад

Французская зона отчуждения

4 мин 1.6K

+17 5 14

HWQA-IP
7 часов назад

Ломать не строить: как устроено хардверное тестирование

Простой 15 мин 655

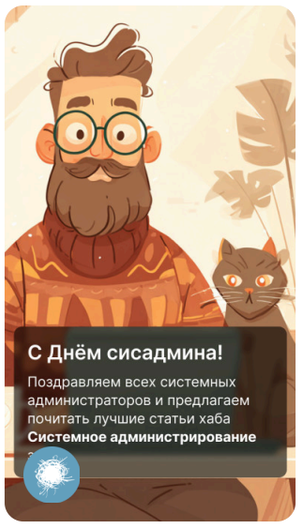
FAQ

+16 8 1

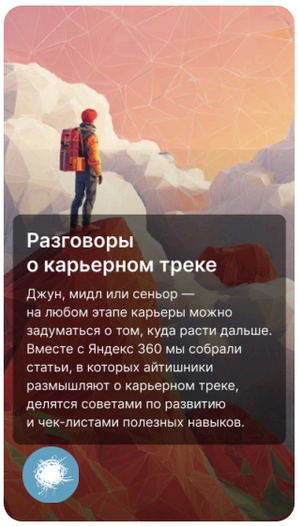
Показать еще

ИСТОРИИ

История Яндекс Почты



С Днём сисадмина!



Как расти в ИТ: советы, гайды и опыт сеньоров

1 055 450 очков в облаке – попробуй обогнать

Годнота из блогов компаний

Напиш
котора
настоя

КУРСЫ

C++ для робототехников

По факту набора · Skillbox

Разработка ТЗ на информационную систему по ГОСТ и SRS

28 июля 2025 · BABOKSchool

Курс «Специалист центра мониторинга инцидентов информационной безопасности (SOC)»

28 июля 2025 · Академия Codeby

Паттерны проектирования микросервисной архитектуры и нотация C4

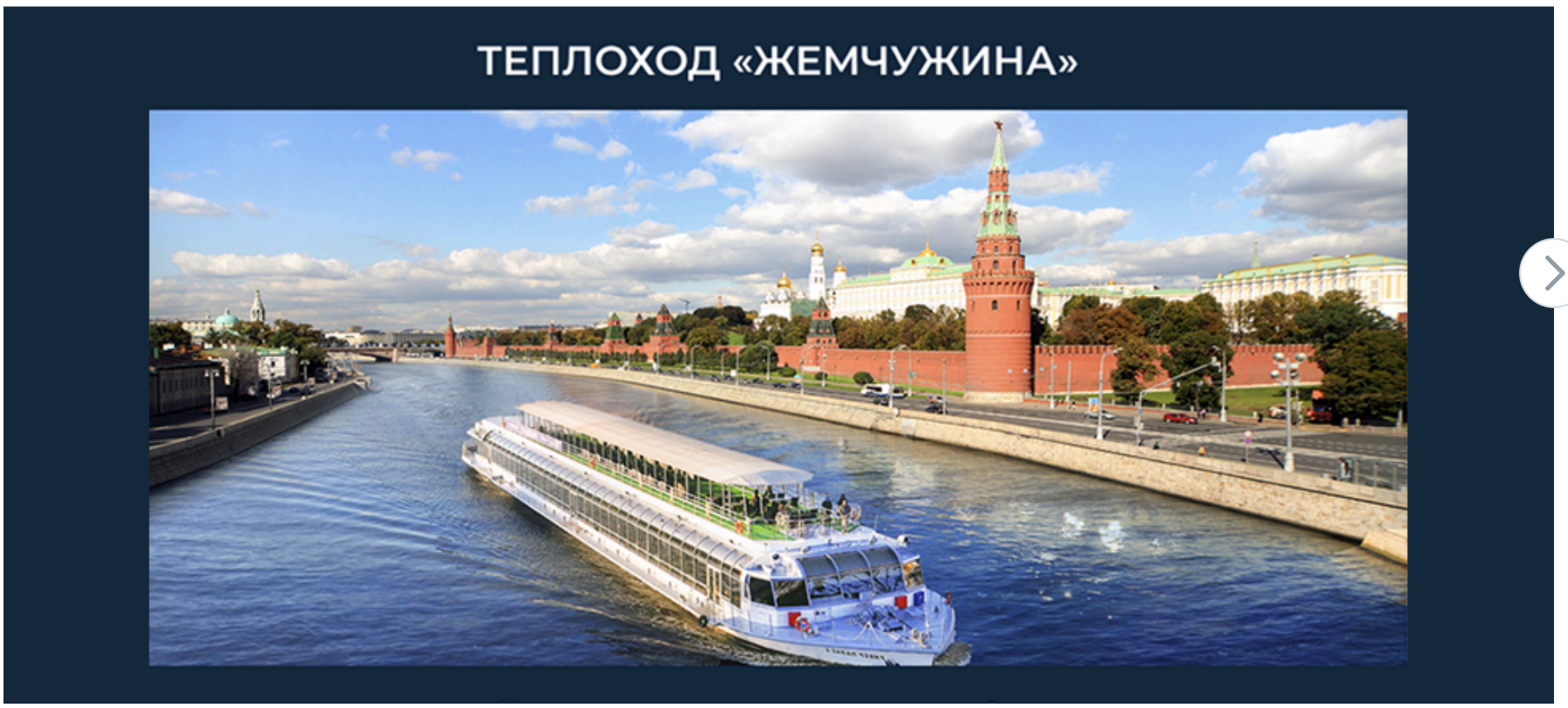
2 августа 2025 · Systems Education

Основы архитектуры и интеграции информационных систем

4 августа 2025 · BABOKSchool

Больше курсов на Хабр Карьере

БЛИЖАЙШИЕ СОБЫТИЯ



25 июля

Маркетинговый баттл за бюджеты клиник — на борту теплохода по Москве-реке

Москва

Другое

Больше событий в календаре

Хабр

Настройка языка

Техническая поддержка

© 2006–2025, Habr