Обновление программного обеспечения

удаленных узлов в ZigBee-сетях

Спецификация ZigBee никак не регламентирует вопросы обновления встраиваемого программного обеспечения (ПО) ZigBee-узлов. Система может не предоставлять возможностей «перепрошивки» удаленных узлов по радиоканалу и при этом полностью соответствовать спецификации ZigBee-альянса. Тем не менее, необходимость заложить механизмы удаленного программирования продиктована здравым смыслом. Ведь в отсутствие таковых для обновления ПО каждый узел в сети нужно деинсталлировать с места установки и обеспечить кабельное соединение с персональным компьютером. Такая процедура в больших ZigBee-сетях может оказаться весьма трудоемкой. Именно поэтому компания Ember, один из ведущих поставщиков аппаратных и программных средств для сетей ZigBee, разработала ZigBee-совместимые механизмы удаленного обновления программного обеспечения и предлагает их к использованию в сетях конечных заказчиков. Подробному описанию этих механизмов и посвящена данная статья.

Александра Дмитриенко
dav@efo.ru

Введение

Система удаленного программирования, предлагаемая Ember, базируется на использовании самостоятельного программного модуля — загрузчика (bootloader), размещенного в специально зарезервированной области флэш-памяти микроконтроллера удаленного узла. Загрузочный модуль выполняет задачи по получению новой «прошивки» через последовательный порт или по радиоканалу и записи полученного образа программы на место старого приложения.

Стратегия использования загрузочного программного модуля для обновления основного программного обеспечения не нова. Такая тактика с успехом используется, к примеру, для обновления программного обеспечения микроконтроллеров по последовательному интерфейсу или для загрузки во флэш-память альтернативного образа программы, хранящегося в EEPROM. Именно такие хорошо известные методы и легли в основу предлагаемой Ember технологии. Уникальная особенность этого решения — способность загрузчика работать с радиоканалом.

В общем случае технологию обновления программного обеспечения с использованием загрузочного модуля можно применять для любого типа микроконтроллера, на базе которого реализован ZigBee-узел. Конкретная же реализация компании Ember предназначена для однокристального решения — микросхемы EM250, объединяющей приемопередатчик стандарта 802.15.4 и микропроцессорное ядро, выполняющее обработку стека протоколов ZigBee и задачи основного приложения.

Сейчас компания Ember предлагает два механизма обновления по радиоканалу, различающиеся как возможностями, так и требованиями к аппаратным ресурсам ZigBee-узлов. Первый, так называемый «автономный» загрузчик (standalone bootloader), уже описывался ранее в статье [1]. Его возможности оказались недостаточными для эффективного применения в больших сетях, поэтому разработчики Ember предложили еще один механизм, дающий бóльшую гибкость, но требующий дополнительных аппаратных средств (внешней памяти EEPROM). Новый механизм базируется на загрузчике иного типа, который называют «прикладным» (application bootloader).

Итак, рассмотрим оба варианта более подробно.

Автономный загрузчик

Программа автономного загрузчика располагается в младших адресах адресного пространства флэш-памяти микроконтроллера EM250 и занимает 10 кбайт. Микроконтроллер может функционировать как под управлением основной прикладной программы, так и под управлением программы-загрузчика.

Взаимодействие программы автономного загрузчика с основным приложением осуществляется только в момент передачи ему управления от основной программы, при вызове определенной API-функции. Далее его работа не зависит от кода основного приложения. Именно поэтому данный тип загрузчика и называется автономным.

Получив управление, программа автономного загрузчика начинает опрашивать вход последовательного порта, и если получает символ “Enter”, то переходит в режим загрузки по проводному каналу. Если на вход последовательного порта никаких данных не поступает,

включается режим работы с беспроводным приемопередатчиком.
Загружаемый файл «прошивки» имеет бинарный формат, при его передаче используется протокол XModem, обеспечивающий разбивку передаваемого файла на блоки размером 128 байт и подсчет контрольных сумм для проверки целостности данных.
При передаче файла «прошивки» по радиоканалу каждый блок данных размером 128 байт + CRC-сумма фрагментируются и передаются по радиоканалу при помощи двух пакетов, так как максимальный размер полезной нагрузки одного пакета, передаваемого по радиоканалу, недостаточен для передачи нефрагментированного блока.
Сразу же после получения первых 128 байт файла «прошивки» через последовательный порт или радиоканал загрузчик осуществляет копирование полученных данных во флэш-память на место старого приложения (рис. 1). С этого момента старое программное обеспечение уже не может быть восстановлено, так как часть информации потеряна безвозвратно.

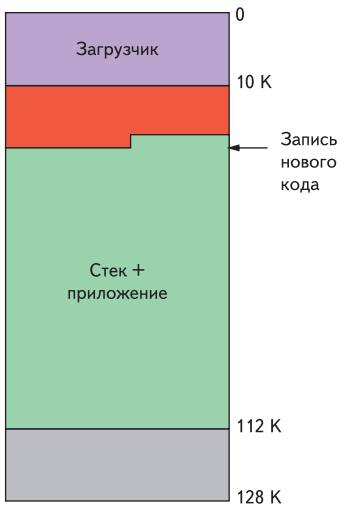

Рис. 1. Копирование нового программного кода основного приложения во флэш-память микроконтроллера поверх старой прошивки

Как видно на рис. 1, стек ZigBee находится в той же области памяти, что и основное приложение. Программа-загрузчик, расположенная в другой секции, не имеет доступа к стеку ZigBee. Поэтому при работе с радиоканалом используются не ZigBee-пакеты, а протокол более низкого уровня — 802.15.4. Все механизмы маршрутизации и ретрансляции сообщений в ZigBee-сети реализуются на более высоком,

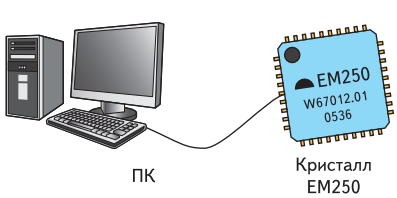

Рис. 2. Загрузчик: последовательный режим

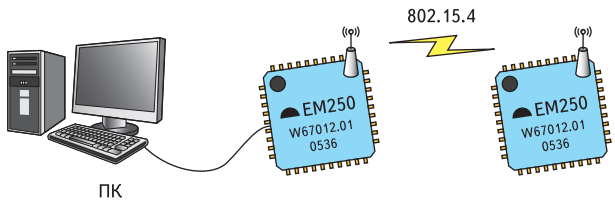

Рис. 3. Автономный загрузчик: режим транзитной пересылки

сетевом уровне стека протоколов. По этой причине при использовании автономного загрузчика файл-образ новой программы не может быть ретранслирован роутерами. Это значит, что узел-источник должен находиться в зоне досягаемости узла-приемника.
Для автономного загрузчика предусмотрено несколько режимов работы.

Последовательный режим (serial)

Вариант последовательной загрузки предполагает обновление программного обеспечения узла через UART. При использовании данного режима следует обеспечить проводное соединение между компьютером и устройством (рис. 2).
Необходимость обеспечения проводного соединения является недостатком этого режима. Тем не менее, его можно с успехом применять на этапе разработки ZigBee-сети, когда узлы еще не установлены на объекте.

Режим транзитной пересылки (pass through)

Способность устройства получать файл «прошивки» по последовательному интерфейсу, характерная для последовательного режима, стала основой для более развитых режимов удаленного программирования, использующих беспроводное соединение. Так, в режиме транзитной пересылки узел-источник также получает код новой программы через последовательный канал. Далее образ полученной программы транслируется узлу-приемнику по беспроводному протоколу 802.15.4 (рис. 3). При этом узел-источник должен находиться в радиусе действия узла-приемника.
Здесь стоит отметить, что узел-источник, который ретранслирует код «прошивки», не передает управление программе-загрузчику, а работает под управлением основного приложения. Для того чтобы обеспечить формирование пакетов формата 802.15.4, код приложения использует специальные, разработанные компанией Ember, утилиты.
Режим транзитной пересылки можно использовать для обновления программного обеспечения ZigBee-узлов, установленных на объекте. Для «перепрошивки» узла нет необходимости деинсталлировать его с места установки, достаточно приблизиться в зону действия устройства с узлом-источником, подключенным к ноутбуку или карманному персональному компьютеру (КПК).
Благодаря использованию радиоканала, режим транзитной пересылки значительно упрощает процесс обновления ПО удаленных узлов.

Режим клонирования (clone)

В режиме клонирования узел-источник считывает собственное программное обеспечение из флэш-памяти, преобразует его в бинарный файл «прошивки» и транслирует целевому устройству (рис. 4).
При использовании режима клонирования можно записать образец нового программного обеспечения в одно устройство-эталон, а далее — обойти территорию, копируя образ новой программы в узлы системы, установленной на объекте. При этом отпадает необходимость носить с собой по территории ноутбук или КПК. Ноутбук потребуется только для записи программного обеспечения в узел-эталон. При отсутствии ноутбука или КПК у конечного заказчика узел-эталон с новым программным обеспечением может быть создан в сервисном центре.

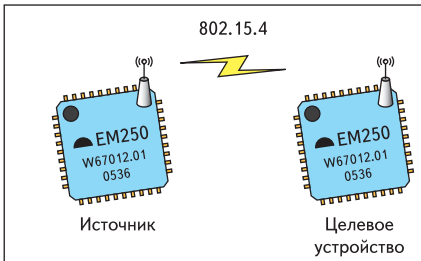

Рис. 4. Загрузчик: режим клонирования

Доработка режима клонирования позволит автоматизировать процесс обновления программного обеспечения. Можно удаленно управлять процедурой клонирования, отправляя ZigBee-команду какому-либо узлу в сети скопировать собственную «прошивку» своему соседу. Тем не менее, полностью автоматизировать процесс удастся только лишь в том случае, если все узлы в сети имеют одинаковое программное обеспечение, что в большинстве случаев неверно. Как правило, в сети ZigBee функционируют узлы различных типов, под управлением разного программного обеспечения, к примеру, концентраторы, роутеры и «спящие» конечные устройства. При этом и роутеры, и конечные устройства могут иметь различное функциональное назначение и, соответственно, разное ПО.
Но если в сети имеется большое количество однотипных узлов, можно автоматизировать процесс обновления ПО хотя бы частично. Следует иметь в виду, что такой автоматизированный режим фактически подходит только для обновления ПО роутеров. Ведь для клонирования «прошивки» «спящего» узла его узлом-источником должен быть такой же «спящий» конечный узел. А два

конечных узла, даже однотипных, не могут взаимодействовать друг с другом напрямую, без участия родительского узла, даже если они находятся в зоне радиовидимости: это особенность топологии многоячеистой (mesh) ZigBee-сети.

Режим восстановления (recovery)

Как уже упоминалось ранее, после старта процедуры записи нового образа программы во флэш-память старое программное обеспечение уже не может быть восстановлено (рис. 1). Если в процессе трансляции «прошивки» произошел сбой, что может случиться при загрузке нового образа программы как по радиоканалу, так и через UART, устройство оказывается без прикладного программного обеспечения. Узел продолжает функционировать под управлением программы-загрузчика и имеет возможность общаться с другими узлами по радиоканалу, используя протокол обмена 802.15.4. Поэтому он остается на том же канале и ожидает повторения процедуры загрузки. Любой роутер, находящийся в радиусе действия такого узла, может начать заново процесс передачи файла «прошивки». Такой режим называется режимом восстановления. Возможен вариант режима восстановления *recovery clone*, где в качестве образа программы узел-источник копирует собственное программное обеспечение, а также *recovery pass through*, если образ программы передается узлу-источнику через последовательный канал.

Режим восстановления «по умолчанию» (default recovery)

Этот режим полностью аналогичен режиму *recovery*, с той лишь разницей, что информационный обмен осуществляется на 13-м частотном канале. Узел, оставшийся без прикладного программного обеспечения, переходит на 13-й канал после сброса питания, так как он «не помнит», на каком канале осуществлялся информационный обмен до этого.

Важная особенность автономного загрузчика, а именно невозможность ретрансляции файла «прошивки» ZigBee-роутерами, оборачивается существенным недостатком при использовании в сетях с большим количеством узлов. Чем больше ZigBee-сеть, тем более трудновыполнимой оказывается задача обхода территории объекта и обновления программного обеспечения каждого устройства (в качестве приблизительной оценки времени может служить следующее значение: 2 мин/узел). В больших сетях общее время обслуживания сети может оказаться настолько велико, что сделает использование автономного загрузчика просто неприемлемым. Именно поэтому возникла необходимость в разработке альтернативного механизма обновления программного обеспечения, который с успехом может быть использован в больших сетях.

Прикладной загрузчик

Для работы прикладного загрузчика требуется дополнительное аппаратное обеспечение — внешняя EEPROM-память, которая используется для предварительного сохранения файла «прошивки», полученного целевым узлом.

Возможность сохранения образа новой программы во внешней памяти позволила перенести функции работы с радиоканалом из загрузочного программного модуля в секцию основного программного обеспечения. В результате процесс получения нового программного обеспечения по радиоканалу происходит под управлением основного приложения, для которого стек ZigBee доступен, поэтому для пересылки используются ZigBee-пакеты, а не пакеты уровня 802.15.14. Благодаря этой особенности при использовании прикладного загрузчика узел-источник не обязан находиться в зоне действия узла-приемника, он может располагаться в другой части ZigBee-сети, при этом для доставки новой «прошивки» может использоваться серия ретрансляций.

Управление программному загрузочному модулю передается только после того, как новый образ успешно сохранен во внешнюю память EEPROM. Загрузочный модуль выполняет лишь задачу копирования данных из EEPROM во флэш-память по интерфейсу SPI или I²C (рис. 5) и не поддерживает функции работы с радиоканалом, поэтому его размер вдвое меньше загрузочного модуля автономного загрузчика и составляет всего 5 кбайт.

Таким образом, в отличие от автономного

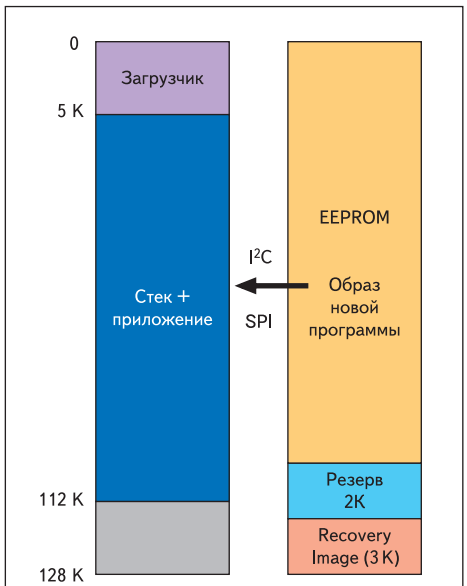

Рис. 5. Копирование нового образа программы из памяти EEPROM

загрузчика, функционал механизма обновления программного обеспечения загрузчика прикладного реализован как в загрузочном программном модуле, так и в основном приложении ZigBee-узла. Именно поэтому он и получил свое название — «прикладной».

Рассмотрим режимы работы прикладного загрузчика.

Режим последовательной загрузки (serial)

Функционально режим последовательной загрузки прикладного загрузчика аналогичен соответствующему режиму загрузчика автономного. Образ новой программы загружается через последовательный порт, поэтому предварительно необходимо обеспечить кабельное соединение целевого ZigBee-узла и компьютера, играющего роль узла-источника (рис. 2). Но реализован механизм прикладного загрузчика иначе. Процесс его работы разделен на три фазы. Первая фаза — это получение файла «прошивки», проверка контрольных сумм и сохранение образа в EEPROM. Вторая фаза — это считывание сохраненного образа и проверка его корректности. При успешном выполнении первых двух этапов следует третий, заключительный этап: управление передается загрузочному модулю, который выполняет копирование сохраненного образа программы из внешней памяти данных во флэш-память программ. При возникновении сбоя на первом этапе или при обнаружении ошибок на втором узел остается под управлением старого приложения и может либо повторить первый и второй этапы, либо выйти из режима загрузки и продолжить функционирование в рабочем режиме.

Режим транзитной пересылки (pass through)

Режим транзитной пересылки прикладного загрузчика также имеет свой аналог среди режимов автономного загрузчика. Есть компьютер, узел-источник, узел-приемник и радиоканал между ними (рис. 3). Однако благодаря тому, что файл «прошивки» передается при помощи ZigBee-пакетов, появляется возможность ретрансляции файла «прошивки» на пути от узла-источника к узлу-приемнику. Поэтому в данном случае режим транзитной пересылки — это частный случай режима транзитной пересылки с ретрансляциями, он используется тогда, когда узел-приемник находится в радиусе действия узла-получателя.

Режим транзитной пересылки с ретрансляцией (multihop pass through)

Схема режима транзитной пересылки с ретрансляцией представлена на рис. 6.

Процесс загрузки в этом режиме условно можно разделить на пять этапов. Первые два этапа полностью повторяют этапы режима последовательной загрузки. Первая фаза — это получение нового образа программы узлом-источником из последовательного порта и его копирование в EEPROM-память. На втором этапе осуществляется проверка корректности сохраненного образа. Третий этап — это считывание программы из EEPROM и ее пересылка целевому устройству. При этом узел-получатель сохраняет полученный образ программы в собственную EEPROM. Четвертый этап — проверка корректности сохраненного целевым узлом образа программы. И только при успешном выполнении четырех перечисленных этапов наступает заключительная фаза: целевое устройство передает управление загрузочному программному модулю, который выполняет задачу копирования программы из EEPROM во флэш-память программ.

Следует заметить, что при использовании данного режима процесс загрузки одного узла может оказаться достаточно продолжительным, так как на ретрансляцию пакетов роутерами

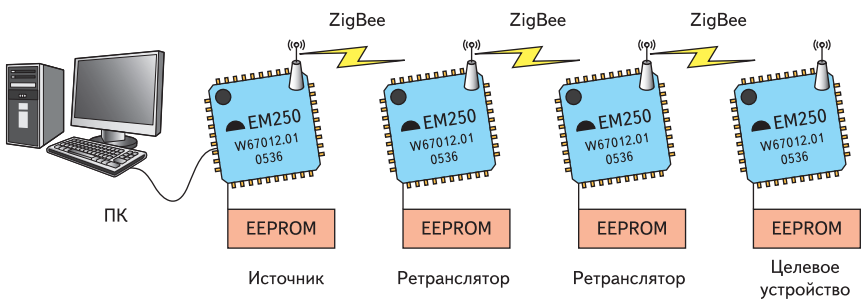

Рис. 6. Прикладной загрузчик: режим транзитной пересылки с ретрансляцией

затрачивается дополнительное время, а также в целях гарантированности доставки пакетов стек ZigBee использует подтверждение доставки сообщений уровня поддержки приложений, что дополнительно нагружает сеть. Для грубой оценки можно использовать следующие эмпирические данные: время обновления программного обеспечения одного узла при использовании одной ретрансляции на пути от узла-источника до узла-отправителя составляет 10–12 минут. Это в пять раз дольше, чем при использовании режима транзитной пересылки автономного загрузчика.

Тем не менее, режим транзитной пересылки с ретрансляциями является единственно приемлемым для больших сетей, так как при его использовании отпадает необходимость физического обхода территории объекта. Весь процесс обновления можно контролировать с единого рабочего места. И, хотя обновление всех узлов в сети может занять продолжительное время, при автоматизации процесса вмешательство обслуживающего персонала будет необходимо только для запуска процесса и проверки результатов выполнения, остальные фазы могут протекать автоматически, в фоновом режиме, и даже без вывода сети из эксплуатации.

При наличии большого количества однотипных узлов в сети для сокращения времени обновления программного обеспечения можно рекомендовать использование широковещательной или групповой рассылки файла-образа с новым программным обеспечением. Однако при принятии решения об использовании широковещательных режимов адресации следует учитывать, что они сильно нагружают сеть. Это может привести к тому, что не останется достаточной полосы пропускания для обслуживания основных задач сети, то есть ее функционирование в рабочем режиме окажется невозможным на время обновления программного обеспечения. Но в ряде случаев такой подход может оказаться востребованным, если целевое назначение системы позволяет использовать режимы сервисного обслуживания сети с временной остановкой выполнения основных задач.

Режим восстановления

Казалось бы, при использовании прикладного загрузчика нет необходимости в режимах восстановления приложения. Ведь копирование нового образа программы во флэш-память осуществляется только после того, как весь файл успешно получен и его целостность подтверждена корректностью контрольных сумм. Но успешно полученный образ программы может оказаться неправильным, например, в случае, если пользователь, контролирующий процесс обновления ПО, выберет совершенно неподходящий для данного устройства файл. Тогда новое прикладное ПО, скопированное во флэш-память, может оказаться неработоспособным. Загрузочный модуль прикладного загрузчика не способен работать ни с последовательным портом, ни с радиоканалом, ведь его задача — копирование данных из EEPROM во флэш-память. Поэтому для восстановления программного обеспечения предусмотрен запасной механизм. Для этой цели используется мини-программа, сохраненная в зарезервированном пространстве EEPROM-памяти в области Recovery Image. Так как размер этой программы составляет всего 3 кбайт, она с легкостью умещается в RAM-памяти кристалла EM250. В случае обнаружения аварийной ситуации загрузчик подгружает ее в память RAM и передает ей управление. Задача этой мини-программы — обновление программного обеспечения через последовательный порт. Поэтому при возникновении необходимости в использовании режима восстановления необходимо деинсталлировать устройство с места установки и обеспечить между ним и компьютером кабельное соединение.

Демонстрационные проекты Ember

Компания Ember предоставляет демонстрационные проекты, иллюстрирующие возможности обоих типов загрузчиков. Они написаны на языке Си для компилятора Ember xIDE for EM250 и входят в состав программного обеспечения, поставляемого с отладочными комплектами EM250 Ember Jump Start Kit. Проекты предоставляются в исходных кодах, что позволяет использовать готовые процедуры механизмов загрузки приложениях конечных разработчиков сетей ZigBee.

Демонстрация режимов автономного загрузчика

Проект, посвященный автономному загрузчику, носит название standalone-bootloader-demo-v2. После загрузки откомпилированного кода в ZigBee-узел пользователь получает доступ к пользовательскому меню, выводимому через нулевой последовательный порт устройства (serial port 0). Меню печатается в результате выполнения команды help и содержит следующие пункты: *help, form, leave, join, serial, query network, query_neighbor, recover, default, clone, remote.* Для работы с функциями загрузчика используются команды *serial, remote, clone, recover* и *default.* Таблица 1 содержит описание этих команд. Для выполнения команды пользователь должен ввести в последовательный порт название команды и ее параметры.

Команда *serial* инициирует загрузку образа новой программы через последовательный порт. В результате ввода этой команды выводится уведомление “Entering serial bootloader. BYE!”, сигнализирующее о том, что узел передал управление программе-загрузчику.

Программный модуль загрузчика сконфигурирован на работу с последовательным портом номер 1 (serial port 1), поэтому при получении упомянутого сообщения необходимо при помощи программы Hyper Terminal подключиться к первому последовательному порту устройства. При успешном установлении связи появляется меню загрузчика следующего вида:

```
EM250 Bootloader v20 b09
1. upload ebl
2. run
3. ebl info
```

В первой строчке отображается информация о текущей версии загрузчика. При выборе пункта меню 3 отображается информация об аппаратной платформе устройства. При помощи пункта 2 можно вернуть управление

Таблица 1. Команды для работы с функциями загрузчика

Команда	Параметры	Описание
serial	нет	Начать загрузку нового программного обеспечения через последовательный порт
remote	EUI адрес целевого узла*	Получить новую «прошивку» через последовательный порт и загрузить в целевой узел, находящийся в радиусе действия (режим транзитной пересылки)
clone	EUI адрес целевого узла*	Скопировать собственное программное обеспечение целевому узлу
recover	EUI адрес целевого узла*, режим (1 —passthru, 2 — clone)	Начать загрузку нового программного обеспечения в режиме восстановления
default	Режим (1 — passthru, 2 — clone)	Начать загрузку нового программного обеспечения в режиме восстановления по умолчанию (на 13-м частотном канале, широковещательный режим)

* Адрес EUI указывается в фигурных скобках, например: {FF 16 46 29 08 11 43}

основному приложению. Для начала процесса обновления программного обеспечения необходимо выбрать пункт 1.

После выбора первого пункта меню пользователь видит символы 'C', которые печатает устройство в знак готовности загрузить новое программное обеспечение. Пользователь в течение 30 секунд должен выбрать файл с новой «прошивкой» и установить в опциях, что протокол передачи файла — XModem. Если в течение 30 секунд файл не выбран, появится сообщение "Serial upload aborted" и загрузчик вернется к основному меню.

Если пользователь выбрал файл вовремя, стартует процесс обновления программного обеспечения. При успешном его завершении узел перезагружается и запускается под управлением основного приложения. При любом сбое на этапе передачи файла загрузчик выдает сообщение об ошибке и возвращается в меню, после чего процесс загрузки можно повторить.

Команда *remote* запускает процесс обновления программного обеспечения целевого узла в режиме транзитной пересылки. Узел, подключенный к последовательному порту, является источником, а роль целевого узла играет устройство, адрес которого указал пользователь в качестве параметра.

Все начинается с процедуры аутентификации. Если аутентификация прошла успешно, целевой узел передает управление программе-загрузчику, в то время как узел-источник продолжает работать под управлением основного приложения. При успешном завершении аутентификации пользователю выводится приглашение начать трансляцию файла «прошивки» по последовательному каналу: сообщение "Please, start bin upload image...". После получения такого приглашения необходимо подключиться к первому последовательному порту узла-источника при помощи программы Hyper Terminal. Если связь установлена успешно, пользователь видит печатаемые устройством символы 'C'. При своевременном выборе загружаемого файла пользователем (в течение 30 секунд) процесс обновления программного обеспечения продолжается, при успешном его завершении пользователь видит уведомление "Bootload complete!", выведенное через нулевой последовательный порт. После вывода данного сообщения узел-источник продолжает свою работу в обычном режиме. Удаленный узел-приемник по завершении процесса перезагружается и включается под управлением нового прикладного программного обеспечения.

Для демонстрации режима клонирования автономного загрузчика используется команда *clone*, с которой в качестве параметра указывается адрес целевого узла.

Как и в режиме транзитной пересылки, основному процессу загрузки предшествует проверка полномочий узла-источника. При успешном завершении процедуры аутентификации узел-приемник передает управление программе-загрузчику, а узел-источник, оставаясь под управлением прикладного программного обеспечения, начинает процесс считывания собственного программного обеспечения из флэш-памяти и его трансляцию целевому узлу.

При успешном завершении загрузки узел-приемник передает управление новому прикладному программному обеспечению. Узел-источник также выходит из режима загрузки, предварительно выдав сообщение "Bootload Complete!".

Для демонстрации режимов восстановления необходимо сымитировать возникновение сбоя на этапе передачи файла «прошивки». Например, можно выключить питание узла-источника во время трансляции файла с новой программой при использовании режимов клонирования или транзитной пересылки. В такой ситуации процесс загрузки прерывается, и целевой узел остается без прикладного программного обеспечения. Он продолжает работу на том же самом канале под управлением программы-загрузчика и находится в ожидании, готовый в любой момент времени начать загрузку нового программного обеспечения по радиоканалу.

Для того чтобы начать процесс восстановления утраченного программного обеспечения, необходимо ввести команду *recover*, указав соответствующие параметры. В качестве адреса узла-получателя можно указать как действительный EUI-адрес устройства, так и широковещательный адрес {FF FF FF FF FF FF FF FF}. Еще один параметр, который необходимо указать, — это режим загрузки. Опция 1 соответствует режиму восстановления с транзитной пересылкой, опция 2 — режиму восстановления с клонированием.

Процедуре восстановления не предшествует проверка прав узла-источника, так как процесс аутентификации поддерживается только прикладным программным обеспечением. Программа-загрузчик, под управлением которой функционирует целевой узел, не в состоянии провести аутентификацию, поэтому сразу стартует процедура загрузки. Если процесс завершен успешно, узел-получатель перезагружается и передает управление новому прикладному программному обеспечению.

Команда *default* инициирует аналогичный процесс, с той лишь разницей, что сначала осуществляется проверка номера канала, на котором работает устройство. Если номер канала отличен от 13-го, выдается сообщение об ошибке следующего вида: "Error: invalid channel, 0x0B. Should be on 0x0D". Если номер используемого канала — 13-й, процесс продолжается, в точности повторяя этапы режима *recovery*.

Режим *default* необходимо использовать тогда, когда в процессе загрузки программного обеспечения у узла-получателя произошел сброс питания. В этой ситуации при включении питания программа-загрузчик стартует на 13-м канале.

При работе с функциями автономного загрузчика зачастую необходимо знать EUI-адрес узла-получателя, версию прикладного программного обеспечения и версию программы загрузочного модуля. Узнать эту информацию можно, используя команды *query_network* и *query_neighbour*. В результате выполнения команды *query_network* всем узлам в сети отправляется запрос информации о версии прикладного программного обеспечения. Команда *query_neighbour* инициирует запрос ближайшего окружения о версии программы-загрузчика. Узлы, получившие такие запросы, отправляют запрашиваемую информацию, а также добавляют свой EUI-адрес в тело сообщения.

Остальные команды основного меню демонстрационного приложения, *form, leave* и *join*, используются для управления сетевыми функциями ZigBee-узла. Команда *form* инициирует создание новой сети. Узел, выполнивший такую команду, становится координатором. Результат выполнения команды *join* зависит от того, является ли в данный момент узел членом сети. Если является, то команда разрешает подключение новых узлов к сети в течение одной минуты. Если узел к сети еще не подключен, команда *join* инициирует процесс его подключения в качестве роутера. Команда *leave* служит сигналом покинуть ZigBee-сеть.

Описанный демонстрационный проект позволяет ознакомиться с реализованными механизмами автономного загрузчика и составить представление о его возможностях для последующего сравнения с загрузчиком прикладным.

Демонстрация режимов прикладного загрузчика

Для демонстрации возможностей прикладного загрузчика компания Ember предоставляет два демонстрационных проекта. Они реализуют идентичные функции, разница между ними заключается в используемом интерфейсе для связи внешней памяти и кристалла EM250.

Первый проект называется app-bootloader-demo-spi. В качестве внешней памяти с SPI-интерфейсом разработчики Ember использовали микросхему Serial Flash AT45DB011B компании Atmel. Во втором проекте под названием app-bootloader-demo-i2c реализован интерфейс I^2C для микросхемы 24AA1025 Serial EEPROM компании Microchip. В каждом проекте реализованы функции по работе с соответствующим

Таблица 2. Описание команд и их параметры

Команда	Параметры	Описание
xmodem	–	Загрузить файл новой «прошивки» через последовательный порт
bootload	Сетевой адрес узла*	
validate	Сетевой адрес узла*	Проверить корректность образа программы, сохраненного во внешней памяти
update	Сетевой адрес узла*	Передать управление загрузчику для копирования сохраненного в EEPROM образа программы во флэш-память

* Сетевой адрес узла указывается в следующем формате: традиционный шестнадцатеричный префикс 0x и адрес в шестнадцатеричном виде, например 0xAF17.

Таблица 3. Различия прикладного и автономного загрузчика

	Автономный загрузчик	Прикладной загрузчик
Загрузка через последовательный порт	Загрузочный программный модуль	Основное приложение
Загрузка нового образа программы по радиоканалу	Загрузочный программный модуль. Уровень 802.15.4, без ретрансляций	Основное приложение. Уровень ZigBee, с поддержкой ретрансляций
Дополнительные требования к аппаратному обеспечению	нет	Нужна дополнительная EEPROM
Размер загрузочного модуля	10 K	5 K
Последовательный режим загрузки (serial)	+	+
Режим транзитной пересылки (pass through)	+	+
Режим транзитной пересылки с ретрансляцией (multihop pass through)	–	+
Режим клонирования (clone)	+	–
Режим восстановления (recovery):		
serial recovery	+	+
recovery pass through	–	+
recovery clone	–	+
default recovery pass through	–	+
default recovery clone	–	+

типом памяти, связанные с чтением, записью данных и тестированием.

Для начала работы с демонстрационной программой пользователь должен загрузить откомпилированный код в ZigBee-узел и подключиться к первому последовательному порту устройства (serial port 1). При успешной установке соединения в результате ввода команды *help* отображается пользовательское меню, состоящее из следующих пунктов: *form, join, join_sleepy, join_mobile, join_end, leave, help, query, xmodem, bootload, validate, update.*

Для управления процессом загрузки предназначены команды, завершающие этот список: *xmodem, bootload, validate* и *update.* Описание этих команд и их параметры приведены в таблице 2.

В результате выполнения команды *xmodem* устройство печатает приглашение начать загрузку: “Please, upload image (to EEPROM) via XModem”. Сразу же после вывода этой строчки появляются символы ‘C’. Это означает, что устройство готово к принятию нового файла. Пользователь должен выбрать отправляемый файл, не забыв при этом указать протокол передачи сообщения — XModem. После успешной загрузки файла выводится сообщение, сигнализирующее об успехе операции: “XModem (to EEPROM) Complete!”.

Выполнение команды *bootload* сперва запускает вторую фазу процесса загрузки: проверку сохраненного образа программы в EEPROM. По итогам проверки выводится сообщение либо об ошибке, либо о корректности сохраненного файла-образа с указанием размера программы. Если проверка завершилась успешно, автоматически стартует третья фаза процесса — трансляция файла прошивки удаленному узлу. Удаленный узел сохраняет полученный образ программы в собственную EEPROM. При успешном завершении процедуры выводится сообщение: “Bootload complete!”.

Для осуществления проверки корректности сохраненного удаленным узлом образа программы используется команда *validate.* В ходе ее выполнения узел-источник генерирует пакет-запрос и направляет его узлу-приемнику. Целевой узел вызывает функцию проверки корректности образа, имеющегося в EEPROM, и генерирует ответ с информацией о полученном результате.

Завершительный этап процесса загрузки, а именно копирование файла-образа из EEPROM в память программ, инициируется вызовом команды *update.* Эта команда имеет параметр — адрес узла, сохранившего ранее новое программное обеспечение в собственной EEPROM. Узел, получивший команду обновить программное обеспечение, передает управление загрузочному модулю, который и осуществляет копирование нового программного обеспечения во флэш-память.

Последовательное выполнение ряда команд по работе с загрузчиком позволяет получить различные режимы его функционирования. Последовательная загрузка осуществляется при комбинировании команд *xmodem, validate* и *update.* При этом в качестве параметров команд *validate* и *update* необходимо указать локальный адрес ZigBee-узла.

Режим транзитной пересылки с ретрансляцией реализуется при поочередном вызове команд *xmodem, bootload, validate* и *update.*

Для выяснения адресов узлов, находящихся в сети, и получения информации о версиях их программного обеспечения используется команда *query.*

Команды *form, join, join_sleepy, join_mobile, join_end* и *leave* используются для управления сетевыми функциями ZigBee-узла. Команды *form, join* и *leave* работают аналогично соответствующим командам из проекта автономного загрузчика. При вызове команд *join_mobile, join_sleepy* и *join_end* устройство подключается к ZigBee-сети как конечный узел. В зависимости от типа выбранной команды оно будет либо мобильным, либо спящим, либо не спящим соответственно.

Мы рассмотрели все команды, реализованные в демонстрационных проектах, посвященных прикладному загрузчику. Описанное программное обеспечение предоставляет возможности для детального знакомства с функциями прикладного загрузчика и их тестирования.

Заключение

Итак, подведем итоги. В таблице 3 указаны различия прикладного и автономного загрузчиков. Каждый имеет свои особенности. Прикладной загрузчик требует использования дополнительной EEPROM-памяти, но зато имеет режим транзитной пересылки с ретрансляциями, рекомендованный к использованию для обновления ПО в больших сетях. Загрузочный модуль прикладного загрузчика занимает меньше места, что позволяет высвободить дополнительные 5 кбайт памяти программ для основного приложения. В небольших сетях с успехом можно применять автономный загрузчик, но для крупномасштабных сетей рекомендуется использование загрузчика прикладного.

Демонстрационные проекты, которые предоставляет компания Ember, позволяют ознакомиться с предлагаемыми механизмами загрузки. По итогам сравнения прикладного и автономного загрузчиков можно сделать осознанный выбор наиболее подходящего механизма обновления ПО для любой ZigBee-сети.

В заключение хотелось бы отметить, что стоимость дополнительной EEPROM-памяти невысока и не сильно увеличивает цену одного ZigBee-узла. Тем не менее, зачастую даже в больших сетях ZigBee для обновления программного обеспечения используется автономный загрузчик. Это происходит потому, что на этапе проектирования системы разработчики редко задумываются о механизмах обновления программного обеспечения. Этот вопрос, как правило, возникает значительно позже, тогда, когда аппаратное обеспечение уже готово и в его составе EEPROM-памяти не предусмотрено. Повторная разводка плат требует дополнительного времени и средств, поэтому фактически выбора не остается, приходится использовать загрузчик, не имеющий дополнительных требований к аппаратным ресурсам ZigBee-узла, то есть автономный.

Именно по этой причине, по нашему мнению, предварительное знакомство потенциальных разработчиков с особенностями существующих механизмов обновления программного обеспечения не только желательно, но даже необходимо, что и явилось побудительной причиной написания данной статьи. Надеемся, что своевременное информирование позволит вовремя обратить внимание разработчиков на вопросы обновления программного обеспечения в беспроводных сетях ZigBee. ■

Литература

1. Кривченко Т. Обновление программного обеспечения по радиоканалу в беспроводных сетях Ember // Электронные компоненты. 2007. № 3.
2. EmberZNet Application Developer’s Reference Manual. 13 February 2008, 120-3021-00B.