УДК 681.3

ЭКОНОМИЧНЫЙ ИНТЕРПРЕТАТОР ДЛЯ УЗЛОВ СЕНСОРНОЙ СЕТИ

Д.В. Рагозин

Нижегородский Государственный Университет,
лаборатория физических основ и технологий беспроводной связи,
603000, Российская Федерация, Нижний Новгород, проспект Гагарина, 23, корп. 1
тел.:+7 (8312) 65 60 35
Ragozin@wl.unn.ru

Рассматривается специализированный интерпретатор «Метеор-Форт» для узлов сенсорной сети, предназначенный для использования в узлах серии «Ант-3», разработанной Лабораторией физических основ и технологий беспроводной связи Нижегородского государственного университета. Интерпретатор унаследовал лучшие характеристики от интерпретаторов байт-кодов и системы программирования Forth, потребляет от 1 кБт до 5 кБт постоянной памяти и позволяет удалённо расширять функциональность узла сети. В сравнении с существующими интерпретаторами, Метеор-Форт занимает малый объём памяти, оптимизирован по быстродействию и включает в себя среду программирования. Интерпретатор оптимизирован для серии процессоров MSP430 фирмы Texas Instruments.

The paper describes a “Meteor-Forth” interpreter, which is intended for sensor network nodes (motes). It is developed for “Ant-3” motes line, which are produced in Laboratory of Physical Fundamentals and Technologies of Wireless Communications in Nizhny Novgorod State University. The interpreter inherits its best features from byte-code interpreters and Forth programming system, requires from 1 KB up to 5 KB ROM and allows to extend mote functionally remotely. If compared to state-of-art interpreters, Meteor-Forth memory requirements are low; it is optimized by speed and includes programming system. The interpreter is optimized for Texas Instruments MSP430 microcontroller line.

Введение

Идея использования сенсорных сетей для задач мониторинга и управления становится всё более популярным в течении последних лет. Но, несмотря на то, что цена узлов сенсорной сети уменьшается с каждым годом, использование сенсорной сети для решения промышленных задач пока не является выгодным из-за слишком большой цены решения и неотработанности программных решений на практике. Основной ограничивающий фактор – цена узлов сенсорной сети. Если учитывать локальную специфику (например, суровый континентальный климат промышленно развитых стран СНГ), то соответствующее климатическое исполнение узлов (климатическое исполнения корпуса, элементов питания и часто вынужденная неремонтопригодность узлов) ещё более удорожает решение. Поддержка функционирования сети (особенно в рамках концепции ремонтонепригодности узлов сети) также является очень дорогой.

Цена одного узла сети (наиболее известной марки Crossbow [1]) составляет на данный момент 100–150 долларов США, и это пока основной ограничивающий фактор их широкого использования. Возможность уменьшения стоимости узла связана в первую очередь с уменьшением вычислительной мощности узла и выполнения иных компонентов узла в соответствии с задачами пользователя. Например, часто неприемлемым является использование «стандартного» трансивера на узле: как минимум, есть три характеристики его выбора - стоимость, потребляемая энергия, дальность передачи, то же самое относится и к процессору. Но, уменьшение вычислительной мощности процессора приводит к повышению требований к качеству программного обеспечения процессора. В режиме реального времени эти параметры становятся критичными. Отдельную задачу представляет собой управление узлами и мониторинг их деятельности.

В работе описывается реализация экономичного интерпретатора, который используется как интерпретатор байт-кода и как удалённый терминал для управления узлом. Интерпретатор оптимизирован для работы с платформой «Ант-3» [2], разработанной в Лаборатории физических основ и технологий беспроводной связи Нижегородского государственного университета [3]. Интерпретатор предназначался для интеграции в операционную среду «Метеор» для узлов серии «Ант-3», он легко интегрируется и в TinyOS. Основная цель создания интерпретатора – создание для узлов с минимальными ресурсами механизма удалённого управления, например внедрение интерпретатора и виртуальной машины в узлы с 4–8 Кбт ПЗУ. Такие узлы (например, младшие моделями процессоров семейства MSP430 фирмы Texas Instruments [4]) способны контролировать не менее десяти датчиков, трансивер, имеют достаточную вычислительную мощность для реализации алгоритмов шифрования средней сложности, что делает их практически идеальными для создания на их базе ультранизкопотребляющих узлов сети. Узлы, производящие маршрутизацию пакетов в сети, должны иметь несколько большее количество ресурсов, но в большинстве случаев для 16-разрядных процессоров размер ПЗУ ограничен 40–48 Кбт, а размер ОЗУ – 16 Кбт.

Дополнительные услуги, в том числе и удалённое управление, не должны значительно увеличивать сложность системы, чтобы избежать перехода на пока более дорогостоящие 32-разрядные платформы.

Предлагается «поэтапное» решение для удалённого контроля, когда только лишь часть функциональности может быть интегрирована в узел, в зависимости от требований задачи. В случае «малой» системы (4 Кбт ПЗУ) может быть задействован только интерпретатор байт-кода. На системе со «средними» объёмами памяти (8 Кбт ПЗУ) пользователь может расширять функциональность интерпретатора и посылать текстовые команды. Для более мощных систем (более 8 Кбт ПЗУ и 2 Кбт ОЗУ) можно предлагать удалённое программирование и текстовую сессию передачи команд. Таким образом, каждый узел сети может содержать виртуальную машину для исполнения байт-кода.

Во 2 части обсуждаются конкурирующие подходы к построению интерпретаторов и систем удалённого управления. В части 3 обсуждаются требования к виртуальной машине узла сети и реализация виртуальной машины. Часть 4 посвящена реализации «Метеор-Форт», размеру кода, быстродействию.

1. Конкурирующие разработки

Существует множество интерпретаторов и виртуальных машин (ВМ), предназначенных для мобильных устройств. Одним из наиболее популярных является ВМ Java (J2ME), но обычно на узлах сети нет столько ресурсов, сколько требует эта система. В данном случае мы планируем предложить эффективное решение для встраиваемых систем с объёмом памяти 4-8 Кбт. Были рассмотрены различные системы, например, Mate[5], Bombilla[6], Scylla [7], FORTH [8] и даже экономичные интерпретаторы BASIC [9]. Интерпретаторы с очень большим потреблением ресурсов нами не рассматривались.

Mate и Bombilla являются адекватными решениями для операционной системы TinyOS [10], так как TinyOS ориентирована на аппаратную платформу, имеющую 60 – 128 КБт ПЗУ [11] – что достаточно для размещения Mate. Большинство узлов от Crossbow [10] – кроме платформы Telos-A/B – реализованы на базе контроллеров Atmel серии AVR [12]. В данном случае TinyOS занимает 4 КБт ПЗУ, интерпретатор Mate занимает 1 КБт ОЗУ и 16 КБт ПЗУ [5]. Существует несколько интерпретаторов, основанных на Mate: TinyScript (императивный интерпретирующий язык, похожий на BASIC), использующий 1.2 КБт ОЗУ и 26 кБт ПЗУ, и Mottle (язык с чертами Scheme, с динамической типизацией), использующий 1.9 КБт ОЗУ и 29 КБт ПЗУ – при том, что 1 КБт ОЗУ доступен для программ пользователя [5].

Основная цель разработки ВМ Mate и Bombilla – широковещательная передача управляющих программ от шлюза по сети, с возможностью удалённого управления и изменения программного обеспечения на узлах сети. С другой стороны, расширение функциональности интерпретатора возможно только на уровне капсул, каждая из которых содержит до 26 байт-кодов, без возможности расширения набора байт-кодов (кроме перекомпиляции исходного кода). Поэтому, возможности, например, Bombilla по расширению функциональности узла выглядят «игрушечными», но, безусловно, удалённо контролировать узел таким образом можно. Для обновления программного обеспечения в рамках TinyOS разработан пакет Deluge [13], который содержит исключительно функциональность для передачи и выбора прошивок ПЗУ и не предоставляет средства удалённого мониторинга и контроля.

Наиболее интересным решением Bombilla является оптимизация байт-кода для обработки сетевых пакетов TinyOS. Функциональность байт-кода может быть изменена пользователем на уровне исходного кода интерпретатора с последующей его перекомпиляцией. Например, существуют специальные коды для доступа к информации пакета сети и добавления информации к пакету.

Иным возможным решением является система программирования Форт [8]. Она включает интегрированную среду программирования (интерпретатор и компилятор в байт-код) и может быть размещена в 4 КБт ПЗУ, но не является традиционным решением для встраиваемых систем из-за языка программирования, основанного на обратной польской записи. Форт широко использовался для программирования встраиваемых систем 20 лет назад, используется сейчас, отдельно отметим поддержку многопоточности и пригодность для использования на узлах сенсорной сети.

Как вариант были рассмотрены урезанные интерпретаторы с языка BASIC [9], которые использовались ещё на 8-разрядных микрокомпьютерах. Минимальный BASIC-интерпретатор поддерживает минимальный набор операторов BASIC, 26 переменных, 1 массив и произвольный доступ к памяти. Конечно, урезанный BASIC может быть использован для наших целей, но неэффективное внутреннее представление и низкая скорость интерпретации не оставляют BASIC шансов на применение в сенсорных сетях.

2. Цель создания новой ВМ для узлов сенсорной сети

Платформа Ант-3. В отличие от семейства Mica (от Crossbow) [11], основанного на серии контроллеров Atmel AVR [12], серия «Ant-3» (как и узлы Telos-A/B) реализована на базе семейства контроллеров MSP430 от Texas Instruments [4]. Текущая серия MSP430 имеет энергопотребление 250 мкА на 1 МГц в активном режиме (220 мкА в поставках после 2-го квартала 2006 года), 1.5 мкА в режиме ожидания и 0.1 мкА в режиме сна.

В случае рассмотрения типичных решений от Crossbow видно, что при энергопотреблении трансивера в 15-20 мА уменьшение энергопотребления процессора является вторичной целью, и

оптимизация программной части ценности не имеет. В случае серии узлов «Ант-3» это не совсем так, так как ультранизкопотребляющие узлы могут иметь передатчик на основе нелинейного рассеивателя [14], позволяющие уменьшить ток, потребляемый при передаче информации до нескольких микроампер, а проблема низкого энергопотребления процессора становится актуальной.

Серия узлов «Ант-3» включает, как минимум, два исполнения узлов сети: «упрощённый» узел, с низким энергопотреблением и без поддержки маршрутизации пакетов; и «базовая станция» - низкопотребляющий узел с поддержкой маршрутизации пакетов по нескольким интерфейсам. Для «упрощённых» узлов используются контроллеры TI MSP430F133 с 8 Кбайт ПЗУ, для «базовой станции» используется контроллер MSP430F1611 с 48 Кбайт ПЗУ и 16 Кбайт ОЗУ. Учитывая, что потребление памяти описанными ранее интерпретаторами составляет 20 Кбайт ПЗУ, использовать их крайне невыгодно для 16-разрядной платформы.

2.2. Использование интерпретатора в рамках внешней операционной системы . Каждый микро-интерпретатор имеет в своём составе машинно-зависимую часть («примитивы»), часть, зависимую от операционной системы и машинно-независимую часть. В ряде реализаций машинно-независимая часть реализована на языке высокого уровня или используется другой интерпретатор байт-кода. Реализация части интерпретатора, зависимой от операционной системы (доступ к драйверам, доступ к файловой системе, доступ к интерфейсам коммуникационных интерфейсов) полностью зависит от имеющихся интерфейсов операционной системы. В конкретном случае, реализация части интерпретаторов, зависимой от операционной системы сильно зависят от реализации поддержки пакетов операционной системой.

В нашем случае – возможность выбора между мини-операционной системой TinyOS [10] и операционной средой «Метеор», разрабатываемой для поддержки «упрощённых» узлов серии «Ант-3». Обе эти системы являются библиотеками, предоставляющими набор стандартных функций для управления узлом и поддержки функционирования сенсорной сети.

Мини операционная система TinyOS может быть использована как «стандартизированная» основа для построения достаточно эффективной операционной системы для узла сети. Однако, для промышленного использования, система должна быть доработана, например, добавлением уровня поддержки качества услуг. В данном случае доработка TinyOS обозначает добавление новых модулей (исследовательская компонента) и упорядочение имеющегося кода (техническая компонента), а также специализацию ПО сенсорной сети под заказчика.

Наиболее интересным в TinyOS является компилятор nesC [10] с языка nesC – расширения стандартного C – с его помощью написана TinyOS. Компилятор nesC – удобный инструмент для определения интерфейсов модулей, расширяющих TinyOS и др. Специфическая особенность компилятора – поддержка интеграции в TinyOS, создания прошивки для определённых узлов, а также режим симуляции, когда код исходной программы компилируется в вид, пригодный для симуляции сенсорной сети с многими узлами на ПЭВМ IBM PC. Компилятор nesC – замечательная компонента, выделяющая TinyOS из ряда других операционных систем (реального времени), предназначенных для платформ Atmel AVR и TI MSP430. Компилятор преобразовывает код nesC в код на языке Си, преобразовывая пространство имён nesC в имена стандартного Си. Далее код на языке Си (дополненный кодами TinyOS для конкретной платформы) может быть откомпилирован компилятором (поддерживается *gcc*) для платформ MSP430, AVR или IBM PC и зашит в узел или промоделирован в системе моделирования для IBM PC.

Операционная среда «Метеор» – набор кодов, которые могут быть использованы как библиотека для узлов, где невозможно использовать ядро TinyOS. Как правило, это происходит в случаях: очень малого объёма ПЗУ (4 Кбайт); слишком развитого программного обеспечения, реализующего другие процессы (не связанные с коммуникациями) на узле. Предполагается использование «Метеор» на «упрощённых» узлах сети маршрутизатора типа «звезда», передающего пакеты только до ближайшей базовой станции сети.

2.3. Виртуальная машина «Метеор-Форт». Для серии узлов «Ант-3» была реализована ВМ и интерпретатор «Метеор-Форт». Текущая реализация «Метеор-Форт» обрабатывает 2 типа пакетов сенсорной сети: пакеты кодов ВМ (аналогичны капсулам Mate); текстовые команды, интерпретируемые интерпретатором Форт и используемые или для управления узлом через оператора, или для расширения функциональности узла. Использование пакетов второго типа в сенсорной сети предъявляет специальные требования к маршрутизатору – поддержку сессии (гарантированная двусторонняя доставка пакетов между управляющей ПЭВМ и узлом сети). В данной работе мы не рассматриваем детали реализации терминальной сессии.

2.4 Требования к реализации интерпретатора. Основное требование к реализации интерпретатора – минимальный объём его кода и кода приложений. Интерпретаторы, такие как Bombilla, интегрируют все необходимые для обработки пакетов функции в систему инструкций (байт-код), поэтому даже маленькие фрагменты кода (около 20 байт) могут производить полезную обработку пакетов. В данном случае система байт-кодов всегда основана на кодах 0-адресной стековой машине, как имеющей минимальный размер исполняемого кода [16]. Кроме арифметических операций, ВМ предлагает байт-коды для переходов, загрузки коротких констант, установки и загрузки глобальных переменных, а также доступа к коммуникационному каналу и сбору информации с датчиков.

Mate и Bombilla ограничивают объём пользовательского кода восемью пакетами (капсулами), в которых хранятся принятые из сети пакеты. Капсулы кода могут выполняться параллельно (как независимые потоки), или же вызывать друг друга как подпрограммы. Подчеркнём, что достаточно трудно расширить функциональность ВМ, исполняющей байт-код. Изменение функций байт-кодов возможно только для специальных реализаций ВМ, и создаёт проблемы совместимости в случае разных версий байт-кодов.

Рассматривая интерпретатор Форт как кандидата для реализации встраиваемого в узел интерпретатора. Форт – очень «старая» система программирования, использующая «шитый» код для большинства своих реализаций [17]. Форт-система занимает от 3–4 КБайт ПЗУ и включает систему программирования с языком, основанным на обратной польской записи. Базовая версия включает операторы структурного программирования и поддерживает полиморфизм. Согласно классификациям Форт-систем, байт-код является «косвенным шитым интерпретивным кодом» [18], так как все функциональные инструкции в байт-коде пронумерованы числами 0-255 (для 8-разрядных систем). Форт представляет каждую свою процедуру как цепочку адресов других процедур или примитивов. Для 16-разрядного процессора длина инструкций Форт равна 16 битам, что в два раза больше, чем длина байт-кода. Форт-система расширяема пользователем, включает среду программирования, достаточно проста и позволяет строить достаточно большие управляющие системы. Далее, система программируется через текстовый терминал, который может быть использован напрямую в случае сессии удалённого управления: в этом случае оператор может связаться с удалённым узлом сети и ввести команды в «ручном режиме», например, для управляющего воздействия или тестирования системы. Эта функциональность может быть востребована в случае больших сенсорных сетей для трубопроводного транспорта, железных дорог, линий электропередач, так как часто требуется отслеживать функционирование отдельных устройств, контролируемых узлами, а также анализировать протоколы работы системы. Кроме того, удалённо можно дополнять имеющийся код программы.

Например, при необходимости удалённого контроля напряжения питания «вручную» оператор может ввести команду

```
SVS
```

которая оставляет на стеке значение напряжения на конкретном узле в милливольтах и видеть на терминале ответ

```
3175
```

В ряде других случаев может быть необходимо произвести некоторые действия с управляющими механизмами (закрыть кран)

```
KRAN 2 OFF
```

Естественно, что оператору необязательно вводить эти данные вручную, это может делать автоматизированное рабочее место.

Минусы системы Форт – старый и нетрадиционный язык программирования; отсутствие сильной типизации интерпретатора. Первый недостаток может легко исправляться пользовательскими надстройками системы управления, второй – исправляется с помощью верификаторов кода стековой машины и с незначительными надстройками над интерпретатором непосредственно на узле сети. Отметим, что возможность удалённого перепрограммирования может быть полезной, но востребованность такой услуги сомнительна в реальных условиях, её надежность зависит от конфигурации сенсорной сети. Поэтому вопрос о передаче больших объёмов кода по сети сейчас не ставится.

ВМ, использующие байт-код, и Форт-интерпретаторы (интерпретаторы шитого интерпретивного кода) выглядят весьма привлекательно на фоне рассмотренных решений для применений в узлах с малыми ресурсами, поскольку их особенности весьма полезны для ряда применений в сенсорных сетях. Наше решение совмещает в себе достоинства и байт-кодовых реализаций и реализаций на базе Форт-систем.

2.5. Модель использования Метеор-Форт. Можно часто услышать, что Форт безнадёжно устарел. В сравнении с BASIC и Scheme, вариации которых реализованы для узлов Crossbow, Форт не столь дружествен к пользователю в его базовой реализации. Оператор сети возможно никогда не будет использовать Форт-терминал, а пользоваться только графическими интерфейсами систем управления. Программист, знакомый с Форт, может легко разработать программу для Метеор-Форт (так как его реализация близка к стандарту Forth [19]), да и для начинающего программиста не существует непреодолимых проблем в разработке управляющих программ (которые реально являются короткими сценариями, а не большими программами). Если сравнивать с Bombilla, то Форт практически эквивалентен

ему по сложности разработки программ. Но, Форт расширяем, и мы не ограничены количеством капсул. Даже в случае разных версий прошивок узлов сети, мы может легко управлять сетью, так как «связывание» (линковка) в Форт производится не по адресам, а по именам, и одинаковые команды могут производить действия на разных узлах сети в соответствии с конкретными прошивками узлов.

Мы не делаем попыток использовать компилятор времени исполнения (JIT) для Метеор-Форт, так как использование JIT может увеличивать объём кода программы (инструкции MSP430 занимают 16 бит плюс код JIT-компилятора, занимающий значительный объём ПЗУ). Так как узлы сенсорной сети не предназначены для обработки больших объёмов данных, то использование JIT не может в общем случае значительно повлиять на производительность сервисных приложений. В итоге, JIT-компилятор, достаточно простой (включая эффективное отображение стека на регистры RISC-процессора) и эффективный для большинства RISC-платформ, не оправдывает средства, затраченные на его реализацию для узла сенсорной сети.

Однако, часть операционной системы может быть описана в кодах Метеор-Форт, если этот код будет меньше по размеру, чем аналогичный код на Си, и некоторое снижение быстродействия не имеет значения для данной задачи.

3. Реализация экономичного интерпретатора Метеор-Форт

3.1. ВМ, использующая байт-код. При разработке интерпретатора не преследовалась цель создания сложной специализированной ВМ для программирования узлов, но была цель создания простого и стабильного решения, которое может использоваться для достаточно больших сенсорных сетей. В основе решения лежит «скрещивание» ВМ, исполняющей байт-коды и шитый код Форт. В итоге, интерпретатор должен поддерживать удалённую терминальную сессию, исполнение капсул, запуск нескольких капсул и поддержку абстрактного уровня доступа к оборудованию.

Основная идея реализации Метеор-Форт – одновременное использование байт-кода и 16-разрядного шитого интерпретивного кода (ШИК), используемого в Форт. Исходные реализации обоих виртуальных машин (для байт-кода и ШИК) очень эффективны, но общая реализация менее эффективна. В частности, процессоры серии MSP430 имеют доступ к слову памяти только по выровненным адресам, поэтому переменение в коде байт-кодов и адресов слов Форт может требовать вставки байт-кодов «нет операции».

Было выбрано 80 наиболее употребительных слов Форт [16], которые реализованы как байт-коды (включая 6 инструкций доступа к драйверам внешних устройств и 9 инструкций для обработки пакетов). Часть инструкций по обработке пакетов унаследована от Bombilla (доступ к полям пакета и обработка пакетов). 24 инструкции (байт-кода) используются для быстрого доступа к полям пакета, 16 байт-кодов используются для доступа к глобальным переменным, 8 – для вызова капсул, 32 – для ближних условных и безусловных переходов, 32 – для загрузки констант. ВМ байт-кодов поддерживает 192 различных предопределённых байт-кода. Байт-коды со значением выше 191 используются для указания на первую часть адреса слова Форт. Такая реализация позволяет иметь пространство в 16 Кбайт для слов Форт, в том числе 12 Кбайт адресуемой памяти для пользовательских программ. В зависимости от конфигурации памяти контроллера и операционной системы эти 16 Кбайт адресов могут быть по разному распределены между ОЗУ и ПЗУ.

Представление в виде байт-кодов Метеор-Форт [20] так же компактно, как аналогичные коды Mate и Bombilla. Кроме того, размер пользовательского кода не ограничен объёмом 8 капсул. Это полезно для узлов, на которые могут загружаться достаточно большие сценарии для обработки накопленной информации или тестирования. Метеор-Форт позволяет переписывать пользовательские программы из ПЗУ в ОЗУ для последующего сохранения.

Отметим, что машина байт-кодов может быть использована отдельно от интерпретатора и компилятора Форт. Это позволяет разместить интерпретатор байт-кода в малые узлы, где свободен всего один 1 Кбайт ПЗУ.

3.2. Особенности реализации системы Форт для узла сети на базе контроллера MSP430. Переработка стандартной системы Форт (стандарта 1983 года [21]) была одной из наиболее сложных этапов разработки. Основной перечень слов системы взят из стандартной реализации Форт-83. Все особенности парадигмы Форт были сохранены. Каждый поток Метеор-Форт содержит пользовательский контекст, отдельный стек возвратов и отдельный стек данных. Это позволяет исполняться нескольким капсулам кода, полученным из сети, одновременно; кроме того, несколько копий Метеор-Форт могут исполняться одновременно (например, сессия управления и обработка данных датчиков). Конечно, только одна копия Метеор-Форт может пополнять словарь системы, но остальные могут исполнять коды без ограничений

Для уменьшения размера команд, передаваемых по сети, использует набор символов Radix-50 [22] вместо стандартного набора ASCII. Код Radix-50 (RAD-50) позволяет поместить 3 символа в 16-разрядном слове, при этом экономится 33% объёма данных и в стандартный пакет сети помещается до 40 символов (для сессии управления). Radix-50 позволяет использовать только 40 символов: буквы “A”–“Z”, цифры “0”–“9”, знак минуса “–“, запятая “,”, знак доллара “$” и пробел. Этот набор символов вполне достаточен для программирования узла сети. Длина имени в Метеор-Форт ограничена 12 символами. Часть имён Форт изменена с учётом набора символов Radix-50.

Использование в случае Метеор-Форт набора символов Radix-50 является видимо единственным способом сократить количество данных, передаваемых по сети, так как стандартные методы архивирования потока (кодирование Хаффмена, LZ77, LZW [23]) не работает на коротких строках.

Метеор-Форт поддерживает создание новых слов, переменных и констант, новые определяющие слова, поддерживается стандартная система словарей, управляющие и циклические конструкции. Метеор-Форт практически не отличается по функциональности от стандартной системы Форт-83.

Процесс Метеор-Форт поддерживает работу с двумя пакетами сети: входящим и исходящим. В случае необходимости поддержки более чем одного входящего пакета, предусмотрены дополнительные слова доступа к пакетам. На вход Форт могут подаваться 2 типа пакетов: последовательность байт-кодов, текстовые команды интерпретатора. Пакет первого типа обрабатывается интерпретатором байт-кода и ШИК, пакет второго типа – «стандартнй ввод» для текстового интерпретатора Форт и содержит текст, который в зависимости от режима будет исполнен или скомпилирован в код. При необходимости послать информацию обратно оператору используются сервисные слова, передающие пакет с содержимым стека данных. По умолчанию, содержимое стека высылается обратно оператору после исполнения команды, пришедшей от оператора, что позволяет видеть результаты исполнения команд без дополнительного программирования (см. примеры выше).

3.3. Эффективность текущей реализации Метеор-Форт. Производительность интерпретатора была измерена для контроллеров TI MSP430. Результаты приведены в табл. 1 как количество машинных циклов, необходимое для исполнения примитива и потери (в процентах времени исполнения) на интерпретацию кода.

Таблица 1. Производительность интерпретатора Метеор-Форт

Тип инструкции	Кол-во циклов без учёта циклов ВМ	Кол-во циклов с учётом циклов ВМ	% потерь из-за работы ВМ
Загрузка константы (16/8 бит)	8/6	24/28	67 %/79 %
Загрузка длинной константы	5	27	81 %
Арифметические и стековые операции	1 – 6	23 – 28	79 % - 96 %
Короткие переходы	7	34	79 %
Переходы (взят/не взят)	7/8	29/30	76 %/73 %
Доступ к памяти (загрузка/сохранение)	2/8	24/30	92 %/73 %
Загрузка/сохранение локальной переменной	9	36	75 %
Доступ к полям пакета	8	35	77 %

Для вызова слова в интерпретаторе Форт потери при вызове обычно равны 75%-80% [16], поэтому быстродействие нашей реализации ВМ неплохое. Для сложных операций, например доступа к пакету, потери составляют всего 5%-15%. Остальные важные характеристики приведены в табл. 2.

Таблица 2. Характеристики производительности ВМ

Характеристика ВМ Метеор-Форт	Циклы
Время вызова реализации байт-кода	22
Время вызова слова Форт	17
Время вызова ВМ	2

Если рабочая частота MSP430 составляет 800 кГц, он способен исполнить до 31000 кодов/слов в секунду, если частота составляет 4 МГц – ВМ может исполнить до 150000 кодов/слов в секунду.
Размеры кода для различных компонентов Метеор-Форт приведены в таблице 3.

Таблица 3. Размеры компонентов Метеор-Форт

Компонент	ПЗУ	ОЗУ	Комментарий
Основная ВМ	1 Кбт	100 Бт	ВМ и реализации байт-кодов
Имена для ВМ	1.5 Кбт	-	Таблица имён для байт-кодов Метеор-Форт
Форт-система	2 – 2.5 Кбт	60 Бт	Без ВМ и имён ВМ
Метеор-Форт	4.5 – 5 Кбт	160-200 Бт	Вся система

При сравнении с другими интерпретаторами (например, Bombilla), Метеор-Форт (вместе с внешней операционной системой – TinyOS или Метеор) может быть использован на узлах сети с размером ПЗУ от 4 до 12 КБт.

Заключение

Описан новый интерпретатор для узлов сенсорной сети «Метеор-Форт», специально разработанный для узлов сети серии Ант-3. Виртуальная машина интерпретатора исполняет как байт-код, так и шитый интерпретивный код. Основанный на системе программирования Форт, интерпретатор производителен, умещается в маленьком объёме постоянной памяти, может быть использован для узлов сети с

ограниченными ресурсами. Интегрированная система программирования позволяет пользователю удалённо расширить функциональность узла и представляет удалённую сессию управления узлом – для тестирования или проведения иных действий оператором сети.

1. *Crossbow* Technology Inc. http://www.xbow.com
2. *Умнов А.Л., Головачев Д.А., Ковалев П.А., Шишалов И.С.* Система сбора информации с узлов сенсорной сети // Радиотехника. Нелинейный мир, 2005. – Т.2, N 4 С. 249–253
3. *Лаборатория* Физических Основ и Технологий Беспроводной Связи Нижегородского государственного университета. Доступно по http://www.wl.unn.ru
4. *Семейство* микроконтроллеров MSP430. Рекомендации по применению. – М.: Компэл, 2005, – 544 С.
5. *Levis P. and Culler D.*. Mate: A tiny virtual machine for sensor networks. In ASPLOS, San Jose, CA, October 2002, pp. 85–95.
6. *Levis P., Gay D., Culler D.* Bridging the Gap: Programming Sensor Networks with Application Specific Virtual Machines // Sixth Symposium on Operating Systems Design and Implementation, Under Submission, 2004.
7. *Stanley-Marbell P. and Iftode L.* Scylla : A Smart Virtual Machine for Mobile Embedded Systems. In 3rd IEEE Workshop on Mobile Computing Systems and Applications, December 2000. – P. 41-50,
8. *Келли М., Спайс Н.* Язык программирования Форт. — М.: Радио и связь, 1993. – 320 С., ISBN 5-256-00438-7
9. *Tiny-BASIC* interpreter WikiPedia links. http://en.wikipedia.org/wiki/Tiny_BASIC_programming_language
10. *Levis P., Madden S., Polastre J., Szewczyk R., Whitehouse K., Woo A., Gay D., Hill J., Welsh M., Brewer E., and Culler D.*, "TinyOS: An operating system for wireless sensor networks," in Ambient Intelligence. New York, NY: Springer-Verlag, To Appear.
11. *Polastre J., Szewczyk R., Sharp C., Culler D.* The Mote Revolution: Low Power Wireless Sensor Network Devices // In Proc. Hot Chips 16: A Symposium on High Performance Chips. Stanford, August 22-24, 2004.
12. *Евстифеев А.В.* Микроконтроллеры AVR семейств Tiny и Mega фирмы Atmel. – М.: Додэка, 2004. – 250 С.
13. *Chlipala A., Hui J., Tolle G.* Deluge: Dissemination Protocols for Network Reprogramming at Scale. Technical Report CS262, Berkeley University, 13 p. http://www.cs.berkeley.edu/~jwhui/research/deluge/cs262/cs262a-report.pdf
14. *Умнов А.Л., Головачев Д.А., Филимонов В.А., Шишалов И.С.* Использование нелинейных рассеивателей в задачах связи и локации в беспроводных сенсорных сетях // Радиотехника. Нелинейный мир, 2004. – Т.2 N 5-6 С. 200 – 203.
15. *Koopman, Philip J.*, Stack Computers: the new wave, Ellis Horwood Ltd., Chichester, England (1989).
16. Gay *D., Levis P., Behren R., Welsh M., Brewer E., Culler D.* The nesC Language: A Holistic Approach to Networked Embedded Systems. SIGPLAN Not., 38(5):1-11, 2003. http://nescc.sourceforge.net
17. *Bell J. R.* Threaded code // In Programming Techniques, R. Morris ed., Communications of the ACM, June 1973. – Vol. 16, N 6, P. 370–372.
18. *Loeliger R. G.* Threaded Interpretive Languages: their Design and Implementation. Peterborough, NH: Byte Books, 1981. – 251 P.
19. *American* National Standards Institute, Inc. Technical Committee X3J14. American National Standard for Information Systems - Programming Languages - Forth. ANSI X3.215 – 1994. – 250 p.
20. *Meteor-Forth* Web Page. http://www.wl.unn.ru/~ragozin/meteor-forth.html
21. *Forth-83* standard. Mountain View Press, Mountain View, CA, 1983. http://forth.sourceforge.net/standard/fst83/
22. *Compaq* Computer Corporation. "Compaq Fortran 77 Language Reference Manual, Appendix B.3: Radix-50 Constants and Character Set". 1999.
23. *Ватолин Д., Ратушняк А., Смирнов М., Юкин В.* Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2002. – 384 С.