

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК
КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ОТЧЕТ ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Возможности построения реальных сенсорных сетей

Зав. кафедрой: Ю.Б. НЕЧАЕВ, д. ф-м н., проф.

Студент: А.А. ВАЛИКОВ, 1 курс маг.

Руководитель: А.В. СТРОМОВ

Воронеж 2018

Содержание

1	Введение	3
2	Цель	3
3	Теоретическое обоснование	3
3.1	Топология	3
3.2	Компоненты сенсорного узла	4
4	Описание возможностей модуля	4
4.1	Режимы функционирования	4
4.2	Команды	5
4.3	Рекомендации по использованию модуля	7
5	Описание программы	7
5.1	Прошивка для arduino	7
5.2	Бек-энд на Java Spring	9
5.3	Фронт-энд на Angular	10
6	Вывод	10
7	Заключение	10
8	Список литературы	11

1 Введение

Всё большее распространение получают системы умного дома. Перед инженерами встаёт всё больше технических препятствий на пути реализации подобных систем, например такие как автономность устройств, процессом сбора данных с большого числа датчиков, обработка собранной информации и принятие требуемых решений.

Для больших распределённых систем с большим количеством обрабатываемой информации предпочтительным способом реализации сети сенсоров может являться беспроводная сенсорная сеть. При этом могут рассматриваться как случаи мобильных датчиков (так называемые MANET) и стационарные сети.

2 Цель

Исследование характеристик реальных сенсорных сетей. Конкретнее, ведётся работа по следующим направлениям:

- Увеличение время жизни сети (снижение энергопотребления). Доведение в идеале одного сенсорного юнита до такого энергопотребления, чтобы имелась возможность использовать автономные источники питания при нечастой подзарядке.
- Оценить численно, какое количество информации может «снять» сеть из окружающего мира без потерь данных. Также проварьировать число участников сети. Разработать алгоритмы, позволяющие максимально приблизиться к теоретическому пределу таких сетей.

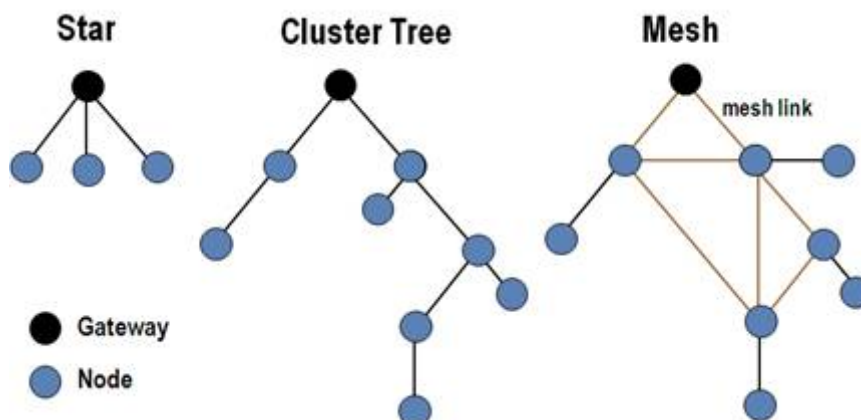
3 Теоретическое обоснование

3.1 Топология

Важнейшей характеристикой сети является её **топология**. Для сетей как правило используются:

- Звезда
- Кластерное дерево
- Меш

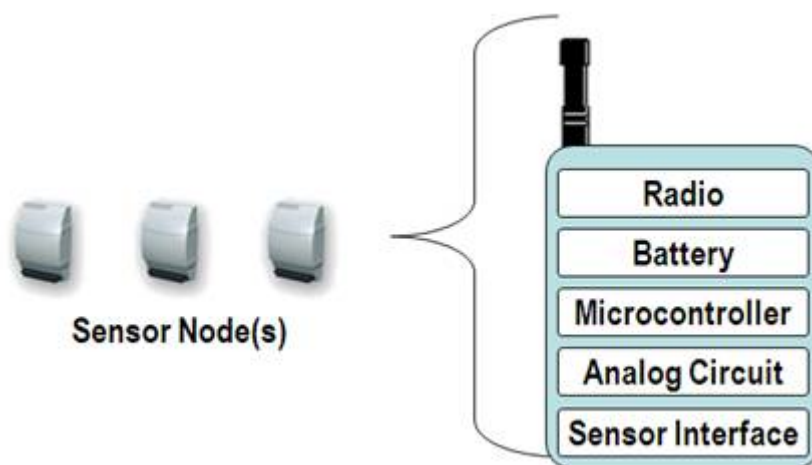
В этой работе предполагается использование топологии меш, где каждый узел связан с несколькими своими соседями, и в сети присутствует один узел сбора информации.



3.2 Компоненты сенсорного узла

Необходимыми компонентами сенсорного узла являются:

- Микроконтроллер
- Сенсорные датчики
- Источник питания (как правило батарея)
- Радиопередатчик



4 Описание возможностей модуля, используемого в работе

- НС-11 - беспроводной модуль, работающий на частоте 434 МГц.
- Пользователю не нужно программировать модуль, только четыре режима отвечают за прием и отправку данных последовательного порта.
- Низкое потребление тока: 80 мкА, 3.5 мА, 22 мА, в зависимости от выбранного режима.
- Неограниченное число байтов, передаваемое одновременно (но часть может быть потеряна)
- Все параметры сохраняются в памяти даже в случае отключения питания.
- Рабочее напряжение: 3.3 - 5В

4.1 Режимы функционирования

- **FU1 Стандартный режим**
Потребление тока: $3.4mA$.
- **FU2 Максимальное энергосбережение**
Задержка сигнала: $400ms$
Скорость передачи (симв.): 1200, 2400, 4800
Потребление тока: $80\mu A$
- **FU3 Максимальная скорость**
Задержка сигнала: $< 10ms$
Потребление тока: $23mA$.

- **FU4 Максимальное расстояние**

Задержка сигнала: $> 300ms$

Скорость передачи (симв.): 9,600

Потребление тока: $22mA$.

4.2 Команды

В скобках указаны переменные

1. AT

Тестовая команда. Если всё нормально, модуль возвращает ОК.

Отправляем: AT

Получаем: ОК

2. AT+A(000-255)

Меняет адрес модуля в указанном диапазоне.

Отправляем: AT+A012

Получаем: ОК-A012

3. AT+B(1200 | 2400 | 4800 | 9600 | 19200 | 38400 | 57600 | 115200)

Меняет скорость (baud rate) на одно из допустимых значений (указаны выше).

Отправляем: AT+B19200

Получаем: ОК-B19200

4. AT+C(001-127)

Меняет канал беспроводного соединения. Нули перед числами обязательны. Если значение слишком большое, данные могут быть потеряны. Так что, на самом деле доступны каналы 001 – 020.

Отправляем: AT+C015

Получаем: ОК-C015

5. AT+E(A | B | C)(соответствующие числа)

Эта команда используется для управления удалённым модулем.

- Первый параметр - три выше описанные команды (адрес (A), скорость (B) и канал (C) соответственно)
- Второй - значение, которое мы хотим присвоить выбранному параметру

Возвращаемые значения:

Успех:

E(A | B | C)R успех

Ошибки:

E(A | B | C)E ошибка параметра

E(A | B | C)F ошибка команды

Fail ошибка соединения

Пример 1

Устанавливаем адрес удалённого модуля в 050

Отправляем: AT+EA050

Получаем: ОК-EBR

Пример 2

Устанавливаем скорость удалённого модуля в 4800

Отправляем: AT+EB4800

Получаем: OK-EBR

6. AT+FC(M | S)(F | T)

Устанавливает модуль в режим IO управления.

Первый параметр:

- M - управляющий
- S - управляемый

Второй

- F - повторяющий
- T - обратный

Один модуль отправляет: AT+FCMT, чтобы стать управляющим

Другой отправляет: AT+FCST, чтобы стать управляемым

7. AT+FU(1-4)

Переключает в один из режимов, описанных выше.

Отправляем: AT+FU1

Получаем: OK-FU1

8. AT+GDPCxAx

Устаревшая команда.

9. AT+P(1-8)

Устанавливает мощность передатчика

Отправляем: AT+P6

Получаем: OK-P6

10. AT+R(A | B | C | P)

Получает значение одного из параметров

Отправляем: AT+RB

Получаем: B9600

Отправляем: AT+RA

Получаем: A001

11. AT+U(N | O | E)(1 | 2 | 3)

Устанавливает бит проверки данных и бит завершения последовательного порта.

Первый параметр:

- N - нет проверки
- O - нечётный
- E - чётный

Второй параметр:

- 1 - стоп-бит
- 2 - 2 бита
- 3 - 1.5 бита

Отправляем: AT+UO2

Получаем: UO2

12. **AT+RX**

Получает значения всех параметров модуля: режим последовательного порта / скорость / канал / адрес / мощность

Отправляем: AT+RX

Получаем: U1 B9600 C001 A000 P8

13. **AT+V**

Получает версию прошивки.

Отправляем: AT+V

Получаем: HC-11_V1.3

14. **AT+SLEEP**

Переводит модуль в спящий режим после выхода из AT. В этом режиме не возможна передача данных. Чтобы выйти из спящего режима, нужно войти в AT ещё раз. Эта команда доступна с версии 1.8.

Потребление тока: 20μA.

Отправляем: AT+SLEEP

Получаем: OK

15. **AT+RESET**

Сбрасывает последовательный порт, канал и адрес в стандартные значения.

Отправляем: AT+RESET

Получаем: RESET_OK

16. **AT+IV**

Получает версию внутреннего кода управления. Команда доступна с версии 1.9

Отправляем: AT+IV

Получаем: I1

17. **AT+UPDATE**

Переводит модуль в режим ожидания обновления. После отправки команды модуль не будет отвечать на команды до перезагрузки. После того как контроллер последовательного порта отправит команду, закройте последовательный порт, выберите файл в помощнике обновления HC-11, откройте последовательный порт. Затем модуль сможет обновиться

4.3 Рекомендации по использованию модуля

Если расстояние между модулями слишком маленькое (меньше 50 см), стоит установить мощность передачи небольшой (P1 - P3). Иначе, передача будет перенасыщена и соединение не удастся. Если расстояние составляет несколько сантиметров, передача, вероятно, не получится.

5 Описание программы

5.1 Прошивка для arduino

Программа представляет собой код, ожидающий поступления команд с серийного порта, и действующий по двум сценариям:

- Команды config.enable и config.disable соответственно включают и отключают режим конфигурирования для модуля arduino.
- Любая другая команда отправляется напрямую в модуль. Команды для модуля имеют синтаксис AT+..., где ... один из допустимых параметров модуля.

Код программы

```
#include <SoftwareSerial.h>
SoftwareSerial radio(3, 4);
int setPin = 2;

String command = "";

void setup() {

    Serial.begin(9600);
    radio.begin(9600);

    pinMode(setPin, OUTPUT);
    digitalWrite(setPin, HIGH);

    getConfigStatus();
}

void loop() {

    command = readSerial();

    if (command.length() > 0) {
        if (command == "config.enable") {
            enableConfig();
        }

        else if (command == "config.disable") {
            disableConfig();
        }

        else if (command == "config.status") {
            getConfigStatus();
        }

        else {
            radio.print(command);
            Serial.println("MESSAGE: " + command);
        }
    }

    readRadio();

    delay(500);

    command = "";
}

void enableConfig() {
```



```

    digitalWrite(setPin , LOW);
    Serial.println(" Config enabled ");
}

void disableConfig() {
    digitalWrite(setPin , HIGH);
    Serial.println(" Config disabled ");
}

void getConfigStatus() {
    if (digitalRead(setPin) == 0) {
        Serial.println("CONFIG=TRUE");
    } else {
        Serial.println("CONFIG=FALSE");
    }
}

String readSerial() {
    String command = "";

    while (Serial.available() > 0) {
        command += char(Serial.read());
    }

    return command;
}

String readRadio() {
    String response = "";

    while (radio.available() > 0) {
        response += char(radio.read());
    }

    if (response.length() > 0) {
        response = "RESPONSE: " + response;
        Serial.print(response);
        return response;
    }
}

```

5.2 Бек-энд на Java Spring

Бэк-энд является прослойкой между фронт-эндом, где планируется размещение основной логики и собственно Arduino с модулем. Код, описывающий контроллер, обрабатывающий запросы к веб-серверу.

```

package somerpackage;
import arduino.*;
import somerpackage.Flower;
import somerpackage.FlowerRepository;

```

```

import javax.annotation.PostConstruct;

@Controller
@RequestMapping(path="/arduino_radio")
public class FlowerController {

    private Arduino arduino;

    @PostConstruct
    void afterInit() {
        arduino = new Arduino("ttyUSB1", 9600);
        arduino.openConnection();
    }

    @GetMapping(path="/disable_config")
    public @ResponseBody void disableConfig() {
        arduino.serialWrite("config.disable");
    }
}

```

5.3 Фронт-энд на Angular

Фронт-энд предполагает наличие UX элементов для управления параметрами модуля, такими как канал, дальность и другие. На данный момент фронт-энд находится в разработке, также возможно в будущем изменится цель проекта, что приведёт к переписыванию модуля.

6 Вывод

Большое количество настроек (адрес, канал, скорость) и четыре режима позволяют предположить широкие возможности модуля НС-11 для создания сенсорных сетей:

- Распределённых на большое расстояние (режим FU4)
- Имеющих большое кол-во участников на единицу площади (20 каналов и 255 адресов)
- Требующих высокой скорости передачи (режим FU3)
- Сильно ограниченных по запасу энергии (режим FU2 и спящий режим)

7 Заключение

8 Список литературы

- [1] National Instruments / What Is a Wireless Sensor Network?
<http://www.ni.com/white-paper/7142/en/>
- [2] Wireless sensor network
https://en.wikipedia.org/wiki/Wireless_sensor_network
- [3] Internet of Things: Wireless Sensor Networks
<http://www.iec.ch/whitepaper/pdf/iecWP-internetofthings-LR-en.pdf>
- [4] HC-11 Wireless Serial Port Module
<https://www.elecrow.com/download/HC-11.pdf>