Porting Contiki OS to VSN

Erik Pertovt, Miha Smolnikar

Jozef Stefan Institute

AgroSense

Outline

Contiki Operating System

Introduction

Features

Communication

Porting Contiki OS to VSN platform

Requirements - Contiki vs. TinyOS

Environment and porting

Testbed

Contiki - Introduction

- Å Lightweight OS for sensor network nodes
- Å Swedish Institute of Computer Science (http://www.sics.se/)
- Å Open Source (BSD license)
  - ï Contiki 1.0 - 2003, Contiki 2.0 - 2007, …, Contiki 2.4 - 2010
- Å Implementation
  - ï C programming language
  - ï Footprint size
    - Å Bigger then TinyOS (event-driven)
    - Å Smaller than Mantis (preemptive multi-thread)

Contiki - Features

Å Ported to several platforms

  ï MSP430, AVR, HC12, Z80, 6502, x86

Å Simulators

  ï COOJA, MSPsim, netsim

Å File System – Coffee

  ï Flash based file system

Å Memory management

  ï Allocation at loading time (both ROM and RAM)

Contiki - Features

Å Modular image

  ï Core + loadable programs

  ï Dynamic loading and replacement of individual programs and services

  ï Core cannot be modified after node’s deployment

Å Over the air programming

Å No power save mechanisms

  ï Lets application specific parts of the system to implement such mechanisms (by exposing the size of the queue)

Contiki - Features

- Å Contiki system consists of
  - ï Kernel (CPU multiplexing and has no platform-specific code)
    - Å does not provide a hardware abstraction layer, but lets device drivers and applications communicate directly with the hardware
  - ï Program loader
  - ï Libraries
  - ï Set of processes (program or service)
    - Å service is a process implementing functionality used by more than application process (e.g. communication)
    - Å communication between processes always goes through the kernel

Contiki - Features

Å Contiki uses a hybrid model of event-driven kernel and the support for preemptive multi-threading

- ï processes in event-driven systems are implemented as event handlers that run to completion (cannot block)
- ï preemptive multi-threading can be used with individual processes (e.g. long computations) and is implemented as a library
- ï this allows the threaded programs to run on top of an event-based kernel without the overhead of multiple stacks

Contiki - Features

Å Event / Thread Hybrid model

- ï Event-driven kernel
  - Å No preemption – only by interrupts
- ï Preemptive multi-threading
  - Å On a per-process basis
  - Å Implemented as a library that can be explicitly linked with programs that require multi-threading
  - Å Memory management functions - library
- ï Protothreads
  - Å Thread-like construct on top of the event-driven Contiki kernel (no need of one stack per thread)

Contiki - Communication

Å Implemented as a service

ï Multiple communication stacks can be loaded simultaneously

ï Run-time replacement of individual parts of a stack

Å Supported protocol stacks

ï Rime

ï lwIP

ï µIP

ï µIPv6

Contiki - Communication

Å RIME

ï Extremely thin layers

ï Low overhead

ï Not a fully modular structure

Å Only the lowest and upper layer can be replaced

ï 2kB ROM, few 10kB RAM

Å lwIP – lightweight IP - 2000

ï IPv4 Compatible

ï Implemented protocols: UDP, TCP, ICMP and IP

ï Modular design – allows extension with additional protocols

ï 40kB ROM, 40kB RAM

Contiki - Communication

Å µIP – “the world’s smallest TCP/IP stack” – 2001

- ï IPv4 compliant
- ï 6kB ROM, 1kB RAM
- ï Minimal set of features
- ï Implemented protocols: TCP, ICMP, IP
  - Å No UDP support

Å µIPv6 - 2008

- ï IPv6 extension of µIP
- ï 11.5kB ROM and 1.8kB RAM
- ï Implemented protocols: TCP, UDP, ICMP, IP

Outline

Contiki Operating System

- Introduction
- Features
- Communication

Porting Contiki OS to VSN platform

- Requirements - Contiki vs. TinyOS
- Environment and porting
- Testbed

Requirements

Å Large scale, heterogeneous sensor networks

- ï Platform portability
- ï C programming language
- ï IP communication stack
- ï Remote reprogramming

Requirements - Contiki vs. TinyOS

Contiki

Å Written in C programming language (ported to Texas Instruments MSP430 and Atmel AVR)

Å Event-driven OS with optional preemptive multi-threading

Å Dynamic linking

TinyOS

Å Written in nesC programming language (ported to Atmel AVR)

Å Event-driven OS with non-preemptive multi-threading

Å Statically linked

Environment and porting

Å Contiki directory structure

- ï `/core/`
    - Å `/dev/` - device drivers (`CC1101/CC2500 on SPI`)
    - Å `/net/` - network drivers
- ï `/cpu/` - common code to all platforms with the same microcontroller (`stm32f103`)
- ï `/platform/` - platform specific code (`VSN v1.2`)
- ï `/apps/` - applications (`test applications`)

Environment and porting

- Å **ST microcontroller with ARM Cortex-M3 core**

- Å **µVision IDE from Keil Software**
    - ï Project Management
    - ï Source Code Editing
    - ï C/C++ Compiler
    - ï Program Debugging

- Å **OS Contiki version 2.4**
    - ï kernel
    - ï communication protocol stack Rime
    - ï system clock and event timer files for ARM
        - Å stm32f103 (from Contiki version 2.x-20100303)

- Å **Microcontroller drivers STM32F10x version 3.1.2**

Testbed

Å **Main function**

- ï Initialization of VSN peripherals
- ï OS Contiki initialization (processes, Rime protocol stack)

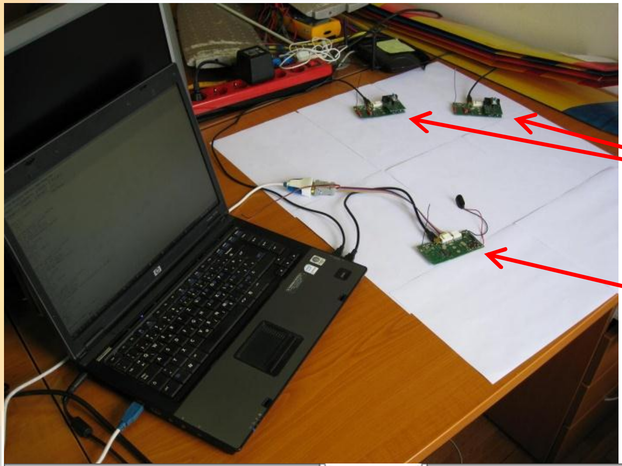
Å **Packets sending**

- ï on sensor nodes: code for sending packets
  - Å reading temperature and humidity
  - Å periodically broadcasts packets with measures for broadcasting periodically uses event timer

Å **Packets receiving**

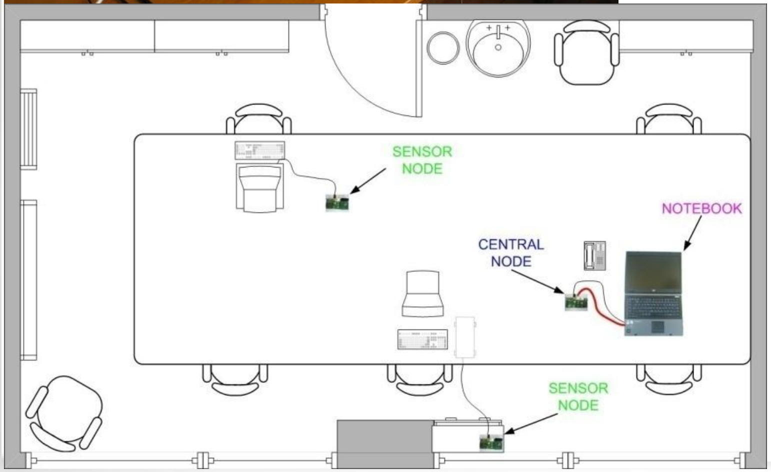
- ï on central node: code for receiving packets
  - Å processing of received packet
  - Å delivering ordered packet’s data to the user

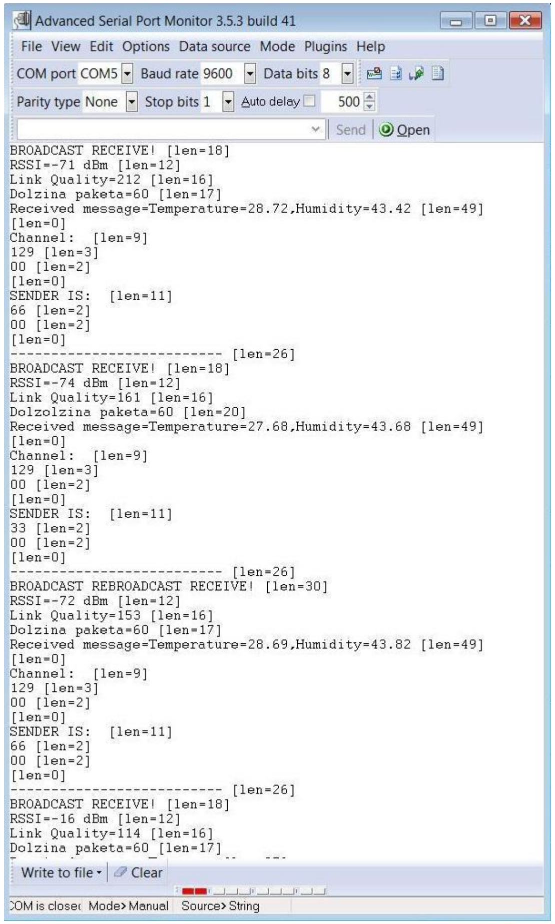
Testbed



Sensor nodes

Central node




Advanced Serial Port Monitor 3.5.3 build 41
File View Edit Options Data source Mode Plugins Help
Send
Open
Write to file
Clear

COM port COM5 Baud rate 9600 Data bits 8

Parity type None Stop bits 1 Auto delay □ 500

```
BROADCAST RECEIVE! [len=18]
RSSI=-71 dBm [len=12]
Link Quality=212 [len=16]
Dolzina paketa=60 [len=17]
Received message=Temperature=28.72,Humidity=43.42 [len=49]
[len=0]
Channel:  [len=9]
129 [len=3]
00 [len=2]
[len=0]
SENDER IS:  [len=11]
66 [len=2]
00 [len=2]
[len=0]
------------------------- [len=26]
BROADCAST RECEIVE! [len=18]
RSSI=-74 dBm [len=12]
Link Quality=161 [len=16]
Dolzolzina paketa=60 [len=20]
Received message=Temperature=27.68,Humidity=43.68 [len=49]
[len=0]
Channel:  [len=9]
129 [len=3]
00 [len=2]
[len=0]
SENDER IS:  [len=11]
33 [len=2]
00 [len=2]
[len=0]
------------------------- [len=26]
BROADCAST REBROADCAST RECEIVE! [len=30]
RSSI=-72 dBm [len=12]
Link Quality=153 [len=16]
Dolzina paketa=60 [len=17]
Received message=Temperature=28.69,Humidity=43.82 [len=49]
[len=0]
Channel:  [len=9]
129 [len=3]
00 [len=2]
[len=0]
SENDER IS:  [len=11]
66 [len=2]
00 [len=2]
[len=0]
------------------------- [len=26]
BROADCAST RECEIVE! [len=18]
RSSI=-16 dBm [len=12]
Link Quality=114 [len=16]
Dolzina paketa=60 [len=17]
```

COM is close(Mode> Manual Source> String

Thanks for your attention!

Miha Smolnikar

miha.smolnikar@ijs.si

AgroSense