# Poster Abstract: MansOS: Easy to Use, Portable and Resource Efficient Operating System for Networked Embedded Devices

Girts Strazdins
gstazdins@acm.org

Atis Elsts
aelsts@acm.org

Leo Selavo
selavo@acm.org

Faculty of Computing, University of Latvia
19 Raina Blvd., Riga, LV 1586, Latvia
Institute of Electronics and Computer Science
14 Dzerbenes Str., Riga, LV 1006, Latvia

## Abstract

Often software for wireless sensor networks (WSNs) is developed using a specific event based operating system (OS) such as TinyOS. However, this requires steep learning curve for the new developers. Other operating systems for embedded devices have limited support for new hardware platforms. Our goal is to provide an operating system for resource constrained devices that is easy to use for the wide range of researchers and developers familiar with C programming language and Unix operating system concepts. In addition, we provide a framework for agile portability to new hardware platforms, due to the nature of WSN systems that require specific hardware or features for the sensing tasks at hand. We propose *Multiple agent netted sensor Operating System* (MansOS), that demonstrates ability to execute the same application on different platforms including x86 CPU based computers for easy simulation and debugging of a single node or a whole network. The new platforms are easily added to MansOS thanks to the well defined abstraction layers. MansOS is being successfully used in real world applications, such as wild animal tracking and environmental monitoring in a fruit care research garden. It has also been used in academic environment for teaching wireless sensor networks.

## Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design; D.1 [**Programming Techniques**]; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and Embedded Systems

## General Terms

Design

## Keywords

Portable operating system, C programming, wireless sensor networks

## 1 Related Work

Several operating systems (OS) are available to the WSN community. Perhaps, the most popular is TinyOS[3]. However, having a specific, event driven paradigm, TinyOS is hard to learn. We are designing MansOS as easy-to-learn operating system based on concepts known to C and UNIX programmer. MansOS started as a branch of LiteOS [1] - a Unix-like OS for WSNs, therefore we share several common Unix concepts. In contrast to LiteOS, which supports only AVR-MCU based motes (MicaZ and IRIS), MansOS is designed with different architectures in mind and allows easy adoption of new platforms. The development of MansOS has been influenced by Mantis [2] and Contiki [6] operating systems, though the code in MansOS is optimized both for resource efficiency and portability.

## 2 MansOS Architecture and Features

The OS architecture is divided in three tiers: HIL, HAL and HPL, see Figure 1. The flexible hardware abstraction approach is borrowed from previous work by Vlado et al [8]. The lowest level, HPL, contains all chip-specific code. HAL level represents platforms and extension modules, and contains chip selection, pin assignments and platform-specific constants. HIL level abstracts the lower layers and provides users with platform-independent services. Complexity of adding new platforms is reduced by separating chip specific code, platform specific wiring and platform-independent routines.

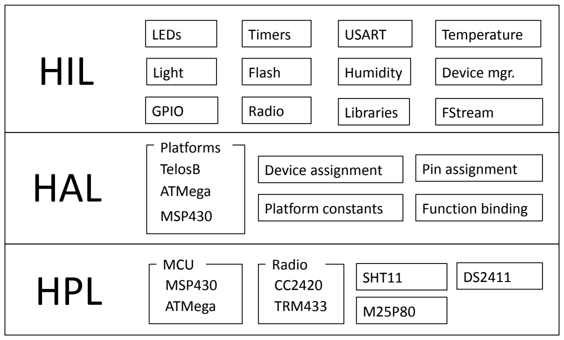

**Figure 1. MansOS architecture**

Key features of the MansOS:

- Easy, plain C programming. See App example 1. Paradigms from the Unix world are available as op-

*SenSys'10*, November 3–5, 2010, Zurich, Switzerland.
ACM 978-1-4503-0344-6/10/11

tional features: sockets, threads, device manager.

**App example 1** The SensorToSerial application in MansOS
```
#include "mansos.h"
#include "dprint.h"
void appMain() {
    PRINT_INIT(128);
    while(1) {
        uint16_t light = readVisibleLight();
        PRINTF("light = %u\n", light);
        msleep(1000); // sleep one second
    }
}
```

- Resource efficiency: macros and inlining are used where possible to minimize and reuse code. In addition, Python linker script analyzes exported source file symbols and assures, that only the required modules are linked together in the final application. Each Unix-like feature requires system resources, therefore they are made optional - features not used can be turned off. Figure 2 illustrates flash memory savings comparing non-optimized code to linker optimizations and turning off thread support. With thread support, MansOS code size is similar to TinyOS. With threads turned off, MansOS code size is significantly smaller than that of TinyOS.

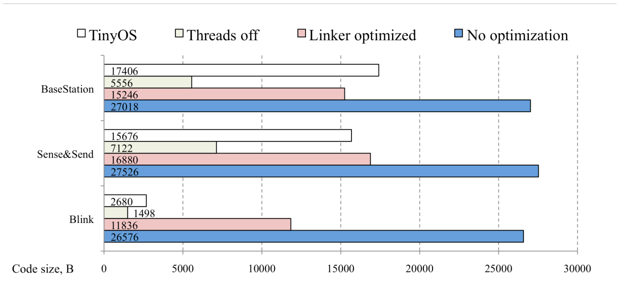

**Figure 2. MansOS code size optimizations**

- Easy portability: the source code is written as platform-independent as possible. As a case study the Arduino (ATMega) platform implementation took 29 hours in total: 11 hours for AVR datasheet studies and 18 hours for code implementation in MansOS.
- TelosB is the first implemented platform. Generic MSP430 platform is also supported - users can build their own motes based on MSP430 MCU (Such as Epic mote), specifying only pin assignments of the used peripherals. The second generic platform supported is Arduino Duemilanove [4], which is based on ATMega328P MCU. Arduino is widely used among embedded system enthusiasts with limited programming knowledge. Epic mote [7] and Nordic Semiconductor nRF24LE1 System-on-Chip platform [5] support is under development.
- Sensor network simulation on the PC platform is supported - each mote is simulated as a separate process on the PC, connecting using sockets to the cloud application simulating the virtual communication medium. Currently one hop network can be simulated.

## 3 Practical Applications

MansOS is used in two projects: *Sensors in the Fruit Garden* (Figure 3), where spread of light inside tree crown is investigated, and *LynxNet* (Figure 4): wildlife monitoring using sensor nodes carried by animals and base stations in the forest.

**Figure 3. Sensor nodes in the tree**

**Figure 4. Mote attached to a "lynx" emulator**

## 4 Conclusions and Future Work

MansOS is a fully functional OS, used in real applications. It supports easy interface to sensors, radio, UART and other peripherals, components similar to Unix environment such as sockets and device abstraction thus saving sensor network application development time for users with C language and Unix skills. Improvements and additional functionality are planned, including remote reprogramming and debugging, IPv6 (6lowpan) support and configurable network for PC and heterogenous simulations.

## 5 Acknowledgements

This work has been partially supported by ESF under grants Nr.2009/0219/1DP/1.1.1.2.0/APIA/VIAA/020 and Nr.2009/0138/1DP/1.1.2.1.2/09/IPIA/VIAA/004.

## 6 References

[1] http://liteos.net.

[2] http://mantisos.org.

[3] http://tinyos.net.

[4] http://www.arduino.cc.

[5] http://www.nordicsemi.com.

[6] http://www.sics.se/contiki/.

[7] P. Dutta, J. Taneja, J. Jeong, X. Jiang, and D. Culler. A Building Block Approach to Sensornet Systems. In *SenSys '08: Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 267–280, New York, NY, USA, 2008. ACM.

[8] H. Vlado, J. Polastre, J. Hauer, C. Sharp, A. Wolisz, and D. Culler. Flexible Hardware Abstraction for Wireless Sensor Networks. In *Proceedings of the 2nd European Workshop on WirelessSensor Networks (EWSN 2005)*, 2005.