Wireless Site Surveillance System

Thomas Eriksson

Master of Science in Engineering Technology
Electrical Engineering

Luleå University of Technology
Department of Computer science, Electrical and Space Engineering
Wireless site surveillance system

Thomas Eriksson

Luleå University of Technology
Dept. of Computer Science, Space and Electrical Engineering

March 13, 2012
In order to monitor telecommunications base stations more closely more and new sensors are needed. Installing new sensors can however prove difficult and costly since it will most probably include installing wiring to and from these sensors.

This master thesis examines the possibility to use existing wireless technology to connect such sensors within a base station and to a broader surveillance system. A small sensor node is designed, built and tested.

The Mulle wireless platform is used to collect sensor data and communicate over Bluetooth. The receiving end is an Internet connected, PAN-GW enabled unit running python and acting site controller.

Communication is kept secure with pre-shared secrets and AES encryption, three-way handshakes, sequence numbering and heartbeats.

Temperature and humidity is measured at very low power for proof-of-concept. The site controller communicates with OSS when preset thresholds are crossed.

Heartbeats and watchdog timers makes the system robust and reliable.

It is shown that the Mulle platform is well suited for implementing wireless battery powered sensors using Bluetooth communication.
This Masters Thesis is a part of the Master of Science Programme in Electrical Engineering at Luleå University of Technology. The Thesis has been done for Ericsson AB in Luleå during the years of 2009 and 2010. The work includes designing and building a low power wireless sensor device for environmental measurements using the Mulle platform and connecting it via a site-controller and SNMP to an operations support system.

I would like to thank Elizabeth Johansson and Torbjörn Burström who has been my supervisors and supported my work at Ericsson.

Thomas Eriksson
Contents

CHAPTER 1: INTRODUCTION .................................................. 1
  1.1 Purpose .................................................................. 1
  1.2 Delimitations ......................................................... 1
  1.3 Thesis outline ......................................................... 2
  1.4 Abbreviations ......................................................... 3

CHAPTER 2: PRESTUDY ...................................................... 4
  2.1 Ericsson ............................................................... 4
  2.2 The Mulle platform ................................................. 4
  2.3 OSS and SNMP .................................................... 5

CHAPTER 3: TECHNICAL BACKGROUND ................................ 6
  3.1 Reliable wireless sensors ......................................... 6
  3.2 Interesting measurements ........................................ 6

CHAPTER 4: RELATED WORK ............................................. 8
  4.1 WirelessHART ..................................................... 8
  4.2 BTnode ............................................................. 8

CHAPTER 5: IMPLEMENTATION .......................................... 9
  5.1 Wireless sensor node network layout ........................ 9
  5.2 Hardware .......................................................... 9
  5.3 Software .......................................................... 10

CHAPTER 6: ANALYSIS AND RESULT ............................... 16
  6.1 Design ............................................................. 16
  6.2 Deployment ........................................................ 20
  6.3 Power consumption .............................................. 21
  6.4 Reliability ........................................................ 22
  6.5 Security .......................................................... 22

CHAPTER 7: CONCLUSION .................................................. 23
  7.1 Mulle, low power Bluetooth active sensor node .......... 23
  7.2 Encryption and security ........................................... 23
  7.3 Sensors ............................................................ 23
7.4 Power consumption .................................................. 24
7.5 Controllability ......................................................... 24
7.6 Deployability ......................................................... 25

Chapter 8: Future work .................................................. 26
8.1 Different sensors ....................................................... 26
8.2 Node configurability .................................................. 26
8.3 Site controller ......................................................... 26
8.4 OSS integration ....................................................... 26
The need to monitor environmental conditions in telecommunications base stations is growing while the networks and systems are becoming more and more complex. Ericsson has more than two million base stations installed worldwide and therefore find it necessary to be able to easily and readily install any kind of sensor at a low cost\[1\]. It would therefore be beneficial to use wireless sensors since these are much simpler to install than their wired counterparts. There exist no solution for this today.

The Mulle platform is an extreme low power embedded Internet system with integrated Bluetooth which is an ideal base for wireless battery powered sensors.

1.1 Purpose

The purpose of this Master thesis is to examine how the Mulle platform could be implemented as a reliable and secure wireless monitoring sensor node and to design and build the necessary hardware and investigate EMC regulation conformance.

It is also part of this thesis to investigate which kinds of sensor would be feasible together with the Mulle.

1.2 Delimitations

To limit the scope of the thesis I have chosen to only implement temperature and humidity measuring, while keeping discussion of any other sensor type strictly theoretical.

Communication security has been limited not to include a full blown encryption secured communication channel since this is mostly computer science (out of scope of the programme of Electrical engineering) and the goal of the thesis is to implement a proof of concept rather than a deployment grade product.
1.3 Thesis outline

This thesis is divided into eight chapters, Introduction presents the thesis and explains what will come. Prestudy introduces important organizations and their roles for the thesis. Technical background looks into different possible approaches. Related work examines what others has already done in this area. Implementation describes the solution chosen in this work and Analysis and Result looks at how well that performed. Conclusion provides some thoughts about the results and Future work talks about what has been left out.
1.4 Abbreviations

BlueZ  The standard Bluetooth stack for the Linux kernel.
EMC    Electromagnetic compatibility, regulations considering unintentional electromagnetic interference.
ESD    Electrostatic discharge, a electrical phenomena which can damage electronics. Susceptibility regulated under EMC.
HART   Highway Addressable Remote Transducer Protocol.
IO      Input-Output.
IP      Internet protocol, both a specific protocol for routing traffic on the Internet and a suite of protocols for providing basic Internet functionality.
lwBT   Light-weight Bluetooth, a small open source Bluetooth stack for lwIP.
lwIP   Light-weight IP, a small open source TCP/IP stack for embedded systems.
OSS    Operations support system, a management tool for monitoring telecommunications networks.
PAN-GW Bluetooth profile for personal area network, gateway device.
RSA    Rivest, Shamir and Adleman, the most commonly used public-key cryptography scheme. Used in SSL to provide authentication and protection against eavesdropping.
SNMPD  Open source software implementing a service listening for SNMP commands.
SSL    Secure Sockets Layer, a standard protocol built on-top of TCP to provide cryptographic security for TCP connections.
TCP    Transmission control protocol, one of the core protocols of the Internet.
UDP    User datagram protocol, connections less communication and one of the core protocols of the Internet.
2.1 Ericsson

Ericsson is a global telecommunications company founded in 1879, Stockholm. It’s one of the largest telecommunications companies in the world with over 100 000 employees (as of 2011).

Ericsson delivers an immense amount of telecommunications equipment to operators in every part of the world. Included is not only the core network equipment but also among other things network monitoring solutions [2].

2.2 The Mulle platform

The Mulle platform is a small, extreme low power embedded Internet system developed and marketed by Eistec AB. With it’s many general purpose IO connections it is a very suitable wireless sensor node. The platform comes in different variations with either Bluetooth or ZigBee radio and supports true IP communications with the lwIP stack as well as sophisticated preprocessing of measurement data. The device is very small in size and multiple embedded operating systems has been proven to run on it [3].

2.2.1 Mulle software repository

Many of the components on-board the Mulle requires software drivers to interface. These drivers are open source with a BSD-style license and available from a software repository provided by Eistec. Eistec partners can get access to this repository upon request.
2.3. OSS and SNMP

2.2.2 The Mulle and EMC regulations

Since the Mulle runs on a rather low clock rate and very low power consumption radiation EMC considerations is not an issue. Earlier tests of immunity has proven the Mulle to be super resistant to both radiation and ESD [4].

2.3 OSS and SNMP

OSS, Operations Support Systems plays an important role in monitoring a telecommunication network from an operator point of view. It allows for every part of the network to report status upwards in the network using the Simple Network Management Protocol, which is an well established Internet standard for monitoring network functionality [5].
3.1 Reliable wireless sensors

For a wireless sensor solution to be reliable means that the system must somehow know that the sensors are working at any given moment and that the system is robust against anomalies such as unexpected reboots or hardware glitches.

The solution for this is usually to send periodical updates constantly convincing the system that this particularly sensor is still here and with a healthy battery.

When tampering and malicious activity is considered the pictures instantly gets a little more complicated. Someone might be blocking the sensor from sending data, or deliberately draining the sensor for power. Corrupted and/or injected data is also to be considered. To avoid tampering the sensor needs to be physically secured; not exposed to tampering at all. To avoid data corruption network transport layer security is needed (which was decided out of scope of this thesis.) And lastly to avoid the case of blocking the communications channel the channel itself must be robust. Bluetooth is a very robust wireless radio protocol and is very hard to block entirely. The periodical status updates will also help detecting the event of a successful attack of this kind.

ZigBee is also a widely used low power radio protocol for industrial purposes. The robustness of Bluetooth was preferred to IEEE 802.15.4 and ZigBee’s lower power consumption in this application since security is a bigger concern than power supply.

3.2 Interesting measurements

For this application to meet it’s purpose several different properties needs monitoring. Some properties deemed interesting are:
3.2. Interesting Measurements

- Temperature is a very important variable to monitor when dealing with electronics.
- Humidity is also an important variable affecting electronics.
- Acceleration might indicate building vibrations or earthquake which could affect the equipment.
- Water and fire detection for facility monitoring.
- Other properties such as fuel level or security breaches could be interesting from an operator point of view.
4.1 WirelessHART

WirelessHART is an open standard for wireless networking using IEEE 202.15.4 radio. It was developed specifically for the requirements of the process industry and utilizes a time synchronized and self-organizing mesh architecture. The standard was developed in 2004 by 37 HART Communications Foundation companies and approved by the IEC as a standard in 2010 [6].

WirelessHART builds upon the established HART protocol to provide a wireless solution backwards compatible with existing HART installations. It’s meant to be simple, reliable and robust.

4.2 BTnode

BTnode is a Bluetooth sensor node developed by Swiss ETH Zurich. It’s a competitor to the Mulle platform, while not as small and more power demanding it comes pre-assembled with a battery holder for standard AA batteries.

Based on Atmel’s AVR series of micro-controllers the BTnode might be more attractive to electronics hobbyists than the Mulle with it’s relatively unknown Renesas’ M16C micro-controller.
5.1 Wireless sensor node network layout

Several sensor nodes will connect to a single site controller within the base station via Bluetooth and UDP/IP. The site controller will then communicate with OSS over SNMP and IP and provide the latest measurements available. The Node to site controller communication is kept monitored with encryption and heartbeats and OSS will periodically poll the site controller status.

5.2 Hardware

The system includes two parts, sensor nodes and site controller.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{sensor_network_topology.png}
\caption{Sensor network topology.}
\end{figure}
5.2.1 Sensor node

The hardware, except for the Mulle itself, is mostly a connection board for power supply (battery), general purpose IO, and also for implementing the humidity sensor, which is not readily available on the Mulle.

The Mulle provides a temperature sensor, a real time clock, a battery monitor and a flash memory for data storage.

5.2.1.1 Humidity sensor

For humidity monitoring the Smartec capacitive humidity sensor, SMTHS07 [7], was chosen due to it’s small size and low cost. It’s essentially a humidity dependent capacitor, which means that to read the sensor one must measure it’s capacitance. A common way of measuring the capacitance is to build a oscillator and measure the obtained frequency. A typical and well known design using a 555-timer circuit was used here.

![Figure 5.2: Standard 555 Astable Circuit](image)

5.2.2 Site controller

For the proof of concept a Bluetooth enabled laptop running Debian GNU/Linux, with BlueZ Bluetooth stack version 3 and Python version 2, is used as the site controller of the system.

5.3 Software

5.3.1 Sensor node

The software on the Mulle is built on top the lwBT and lwIP stacks. It’s completely written in C and is all open sourced with a BSD-style license.
The light data security included in this project is provided by PolarSSL, more precisely the SHA256 and AES algorithms which will provide quite good security but not at the very tight level provided by full SSL, which also uses RSA encryption. RSA encryption on the Mulle has proven difficult due to the limited amount of RAM available and a rather low CPU clock rate.

5.3.2 Site controller

The software on the site controller is a short and simple python (2.x) script running a classic select-loop implementing a simple state machine which allows for multiple simultaneous node connections. For up-link communication the command line utility SNMPtrap from net-snmp is used.

5.3.3 Sensor network protocol

The protocol with which the sensors contacts the site controller is fairly simple. There exists currently four kinds of packets, sent using UDP and each identified with a different command word of 16 bits. The choice stands between UDP and TCP since well established standards are preferred and UDP was deemed more appropriate since we don’t need handshaking and reliability is provided at application level. Every packet has an inbound sequence number and an outbound sequence number, each 16 bits long. The payload is 80 bits and all packets end with a 256 bits checksum using the SHA256 algorithm [8]. Packets are then encrypted with a 256 bits key using the AES algorithm [9], which together with the packet checksum ensures both packet integrity and authenticity. The encryption key used can be different for each sensor node and the nodes are identified by their IP-address and authenticated with the AES/SHA256 integrity. A three-way
handshake with randomized sequence numbers protects against replay-attacks [10]. All data is encoded in big endian.

### 5.3.3.1 Request connection (0x0000)

The sensor node sends a command requesting a new connection with a random inbound sequence number. The packet is checksummed and encrypted and therefore nearly impossible [9, 8] to forge. The Request connection packet has the following elements:

- **INSEQ** Inbound sequence number, unsigned 16 bit integer. Randomized.
- **OUTSEQ** Outbound sequence number, unsigned 16 bit integer. Set to zero.
- **CMD** command number, unsigned 16 bit integer. 0x0000
- **Padding**
- **Checksum** SHA256 checksum of all other fields.

### 5.3.3.2 Offer connection (0x0001)

In response to the request connection command the site controller answers with a connection offer packet with unmodified inbound sequence and a randomized outbound sequence. The packet is checksummed and encrypted so the sensor node knows that if the inbound sequence number is correct the answer is legitimate. The Offer connection packet has the following elements:

- **INSEQ** Inbound sequence number, unsigned 16 bit integer. Unmodified.
- **OUTSEQ** Outbound sequence number, unsigned 16 bit integer. Randomized.
- **CMD** command number, unsigned 16 bit integer. 0x0001
- **Padding**
- **Checksum** SHA256 checksum of all other fields.

*Figure 5.4: Request connection packet, 384 bits.*
5.3. Software

5.3.3.3 Status update (0x0002)

Upon receiving a connection offer the client confirms the connection with a status update. This command is also used to report any change in status when already connected. The site controller validates the message with both inbound and outbound sequence numbers. After receiving this packet the site controller flags the connection as authenticated and live. The Status update packet has the following elements:

- INSEQ Inbound sequence number, unsigned 16 bit integer. Incremented.
- OUTSEQ Outbound sequence number, unsigned 16 bit integer. Unmodified.
- CMD command number, unsigned 16 bit integer. 0x0002
- Temperature, signed 16 bit integer. Temperature in degrees Celsius multiplied by 100.
- Humidity, unsigned 16 bit integer. Relative humidity in percent multiplied by 100.
- Battery voltage, unsigned 16 bit integer. Battery voltage in millivolts.
- IO Current state of digital IO port.
- Padding
- Checksum SHA256 checksum of all other fields.

5.3.3.4 Status acknowledge (0x0003)

The site controller acknowledges every status update received with an acknowledge packet. The sensor node validates both inbound and outbound sequence numbers. After receiving the first acknowledge the sensor node flags the connection as authenticated and live. The Status acknowledge packet has the following elements:
### 5.3. Software

<table>
<thead>
<tr>
<th>INSEQ 16 bits</th>
<th>OUTSEQ 16 bits</th>
<th>CMD 16 bits</th>
<th>Padding 80 bits</th>
<th>Checksum 256 bits</th>
</tr>
</thead>
</table>

![Figure 5.7: Status acknowledge packet, 384 bits.](image)

- INSEQ Inbound sequence number, unsigned 16 bit integer. Unmodified.
- OUTSEQ Outbound sequence number, unsigned 16 bit integer. Incremented.
- CMD command number, unsigned 16 bit integer. 0x0003
- Temperature low threshold, signed 16 bit integer.
- Temperature high threshold, signed 16 bit integer.
- Humidity low threshold, unsigned 16 bit integer.
- Humidity high threshold, unsigned 16 bit integer.
- Battery voltage low threshold, unsigned 16 bit integer.
- Checksum SHA256 checksum of all other fields.

5.3.4 Temperature and battery voltage sensor

Temperature and battery voltage is measured by the battery monitor chip included on-board the Mulle platform, which is interfaced using the driver from the standard Mulle software repository.

5.3.5 Humidity sensor

The relative humidity can be calculated as follows from capacitance, which in turn can be calculated from the frequency of the 555 circuit.

\[
f = \frac{1}{\ln(2) \times C \times (R_1 + 2R_2)} \quad \iff \quad C = \frac{1}{\ln(2) \times f \times (R_1 + 2R_2)} \quad (5.1)
\]

\[
C = C_s + S_X X_{rh} + S_T T \quad \iff \quad X_{rh} = \frac{(C - C_s - S_{CT} T)}{S_{CX}} \quad (5.2)
\]

Put (5.1), \((R1 = R2 = R + S_{RT} T)\) into (5.2).

\[
X_{rh} = \left( \frac{1}{\ln(2) \times f \times 3 \times (R + S_{RT} \times T)} - C_s - S_{CT} \times T \right) / S_{CX} \quad (5.3)
\]
Where

\[ R = \text{Resistor in 555 circuit (10k\Omega)} \]
\[ S_{RT} = \text{Resistor temperature coefficient (1\Omega/K)} \]
\[ C_S = \text{Nominal capacitor value at 0\% humidity. (Calibration needed.)} \]
\[ S_{CT} = \text{Capacitor temperature coefficient (0.16pF/K)} \]
\[ S_{CX} = \text{Capacitor humidity coefficient (0.6pF/\%)} \]
\[ T = \text{Temperature} \]
\[ X_{rh} = \text{Relative humidity} \]

The frequency is measured by counting cycles over a set period of time.

\[ f = c/T \] (5.4)
6.1 Design

The over-all design and information flow is illustrated in figure 6.1 while the design of some parts are discussed in subsections of this section.

6.1.1 Schematics

The board schematics is where electrical function was designed. The goals here was to provide means to reprogram the Mulle in circuit, connect the battery, provide general purpose digital and analog IO and design a capacitance meter with the Smartec humidity sensor connected.

The reprogramming interface is at pins 53 to 56 of the 60 pin Mulle-connector, J1. Pins 6, 8-21 and 38-45 are general purpose IO with analog to digital conversion. Pins 6, 8, 9, 10 and 11 are used for analog inputs. Pin 4 is analog power output. Digital IO from pins 24, 26, 27, 28, 29 and 30 with one counter capable input and three interrupts [11].

The humidity sensor circuit is a standard 555 astable circuit powered from an IO pin of the Mulle and connected to a hardware counter input (TimerB4 input.) It’s designed with 10kΩ resistors and the Smartec SMTHS07 with a capacitance of 330pF for a nominal frequency of 150kHz.

6.1.2 Board layout

The board layout is a simple two layer design with one layer on each side of the substrate.

There are 6 connectors on the board, J1 is the Mulle connector [11], J2 - J6 are documented in tables 1 to 5. The layout is shown in figure 6.2, bottom layer not mirrored.
6.1. Design

Figure 6.1: Information flow within the system.

(a) Top
(b) Top placing
(c) Bottom
(d) Bottom placing

Figure 6.2: Board layout.
Table 1: Connector J2, SMTHS07

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Humidity sensor input</td>
</tr>
<tr>
<td>2</td>
<td>Ground</td>
</tr>
</tbody>
</table>

Table 2: Connector J3, digital GPIO

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VCC, regulated 3.3V</td>
</tr>
<tr>
<td>2</td>
<td>GPIO P17, INT5</td>
</tr>
<tr>
<td>3</td>
<td>GPIO P16, INT4</td>
</tr>
<tr>
<td>4</td>
<td>GPIO P15, INT3</td>
</tr>
<tr>
<td>5</td>
<td>GPIO P14</td>
</tr>
<tr>
<td>6</td>
<td>GPIO P13</td>
</tr>
<tr>
<td>7</td>
<td>Ground</td>
</tr>
<tr>
<td>8</td>
<td>GPIO P93, TimerB3 input</td>
</tr>
</tbody>
</table>

Table 3: Connector J4, battery input

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>Battery positive 2.7V – 5V</td>
</tr>
</tbody>
</table>

Table 4: Connector J5, analog/digital GPIO

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GPIO P104, AN04</td>
</tr>
<tr>
<td>2</td>
<td>GPIO P103, AN03</td>
</tr>
<tr>
<td>3</td>
<td>GPIO P102, AN02</td>
</tr>
<tr>
<td>3</td>
<td>GPIO P101, AN01</td>
</tr>
<tr>
<td>5</td>
<td>GPIO P100, AN00</td>
</tr>
<tr>
<td>6</td>
<td>Ground</td>
</tr>
<tr>
<td>7</td>
<td>Analog VREF</td>
</tr>
<tr>
<td>8</td>
<td>Analog VCC</td>
</tr>
</tbody>
</table>

6.1.3 Encapsulation

The prototypes has been encapsulated in a reused component-box from Maxim IC.
Table 5: Connector J6, Reprogramming

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VCC, regulated 3.3V</td>
</tr>
<tr>
<td>2</td>
<td>CNVSS</td>
</tr>
<tr>
<td>3</td>
<td>RX</td>
</tr>
<tr>
<td>4</td>
<td>TX</td>
</tr>
<tr>
<td>5</td>
<td>Ground</td>
</tr>
<tr>
<td>6</td>
<td>#Reset (active low)</td>
</tr>
</tbody>
</table>

6.1.4 Firmware

The firmware is a modified version of the Mulle PAN-U Demo application from the Mulle repository, designed as a state-machine with 6 different states.

- State 0, inactive, not connected, sleeping or connecting.
- State 1, inactive, connected, sending connection requests.
- State 2, inactive, connected, got connection offer, sending status packets.
- State 3, active, sleeping or measuring.
- State 4, active, got data, connecting.
- State 5, active, got data, connected, sending status packets.

Figure 6.4 shows a flow chart of the firmware logic.
6.1.5 Site controller

The site controller software is a simple Python (2.x) script implementing a classic select-loop with object-oriented node-connections. Figure 6.5 shows a flow chart of the site controller software. It’s of a state-machine design where each connection maintains it’s own state.

6.2 Deployment

Deploying the prototypes requires programming them with firmware where IP-addresses and encryption keys has been set and the humidity sensor has been calibrated.
6.3. Power consumption

6.2.1 Calibrating humidity sensor

Calibrating the humidity sensor can be done by measuring the humidity, temperature and frequency and then calculating the nominal capacitance:

\[
C_S = \frac{1}{\ln(2) \times f \times (3 \times (R + S_{RT} T))} - S_{CT} \times T + X_{rh} \times S_{CX} \tag{6.1}
\]

This value in picofarads is then inserted into the source code file `measure.c` line 9.

6.3 Power consumption

The device as essentially three different power states:

- Sleeping, 130µA
- Measuring, 10mA, 2 seconds every minute in active mode.
- Sending, 50mA, 0.3 seconds every half hour in active mode.

The total average power consumption in active mode is thus 

\[
50 \times 0.3/1800 + 10 \times 2.0/60 + 0.130 = 0.470mA.
\]

A 4.2 volts, 150mAh battery will be good for about 110 hours. Inactive mode average power consumption is 

\[
50 \times 0.3/15360 + 0.130 = 0.131mA.
\]

6.3.1 Sleeping

This state is the lowest power state, in this state the device consumes 130µA.

The device enter this state for one minute after a successful transmission. If the transmission fails the device will sleep for 2 minutes after the first failure, 4 minutes after the second, 8, 16 and so on up to 256 minutes which is the maximum. This ensures quick recovery and low power consumption during longer outages.

6.3.2 Measuring

This state is active for 2 seconds each time and entered once every minute when connection is flagged OK. This is when we count cycles for humidity measurement. The device consumes 10mA in this state.

6.3.3 Sending

When ever the device has something to report to the site controller, it enter this state. This is either when retrying connection after disconnection or reboot, or after a measurement is found outside the currently specified limits, a heartbeat is sent every 30 minutes regardless of measurement values. It’s active for about 0.3 seconds if the transmission is successful or about 2 seconds if it is not. The device consumes about 50mA in this state.
6.4 Reliability

The sensor node will consider a connection dead if transmission has failed for 5 minutes, therefore detecting a connection problem in maximum 35 minutes. The site controller will likewise consider a connection dead after not hearing from a sensor node for 35 minutes. When a node considers a link dead it will enter reconnect mode which will keep trying until the battery runs out (with a 256 minutes period a 150mAH battery will last for 17 days, 6 years with power consumption optimizations.)

Bluetooth communication uses several different frequencies and not depending on any of them, thus making it very robust to interference.

6.5 Security

AES and SHA256 with randomized sequence numbers protects against replay attacks. Radio jamming is possible but difficult given that Bluetooth uses 79 different frequencies. Denial of service attacks against the site controller is probably possible in several fashions, targeting either the BlueZ Bluetooth stack, the IP stack or the site controller application, a maliciously modified Bluetooth hardware could probably completely corrupt and disable the communication. Tampering is not protected against. Each node can have its own encryption key which means that if a single node is compromised the rest of the network can still function with full security. The OSS uses unencrypted, unauthenticated SNMP to communicate with the site controller which defeats all security measurements provided here.
7.1 Mulle, low power Bluetooth active sensor node

The Mulle sensor platform with Bluetooth communication is a good choice for implementing reliable robust battery powered wireless sensors with measurement data preprocessing. It allows for optimizations such as not sending uninteresting data, which was used to provide "long" battery life time in this work.

7.2 Encryption and security

In order to bring this solution to production grade a more robust data encryption solution would be needed, such as SSL. Also the SNMP connection to the OSS would need proper encryption and authentication.

One could employ some filtering to protect against denial of service attacks at HCI (Bluetooth) level and IP level.
In cases where tampering proofness is required extensive work remains.

7.3 Sensors

7.3.1 Temperature sensor

The temperature sensor used here is not any kind of high precision device, but for the purpose of monitoring suitable operating temperature it will suffice.
7.3.2 Humidity sensor

The Smartec SMTHS07 is an analog device and therefore capable of providing very high resolution, however the accuracy is not very good. In applications where high precision is desired a calibrated mapping table could provide very good accuracy.

7.3.3 Accelerometer

An analog accelerometer could easily be connected via a suitable filter/amplifier to the analog GPIO pins provided on the J5 connector.

7.3.4 Tampering and intrusion detection

Most existing wired intrusion detection systems use active sensors with opening or closing signaling, these could be connected to one of the Mulle’s general purpose IO pins using a few resistors. One of the pins available on the board presented here also has interrupting capabilities so that a breach could be detected and signaled immediately.

7.3.5 Other sensors

The board presented here has a few general purpose digital IO pins as well as some analog inputs on the J3 and J5 connectors. There is also a serial bus available on connector J6. Almost any type of sensor should be interface-able with this board, provided that suitable modifications is made to the node firmware.

7.4 Power consumption

The current power consumption is too high for battery powering, a software upgrade is needed. The stop mode consumption could easily be lowered to 10\(\mu\)A and the active measuring time to less than 1 ms, thus lowering the total average consumption by 98%, increasing battery life-time from 110 hours to more than 230 days. To further increase battery life-time one would need to increase the heartbeat period, measure period and/or battery capacity (this would scale battery life time linearly).

7.5 Controllability

The sensor node firmware allows for controlling threshold values for temperature, humidity and battery voltage level, but not the measurement period (currently set to 1 minute) and heartbeat period (30 minutes.) The site controller software however is not controllable at all, one would expect all settings to be adjustable via SNMP and the OSS.
7.6 Deployability

The source code is written mostly for proof-of-concept purposes where certain properties are hard-coded at compile time, such as IP addresses, encryption keys and humidity sensor calibration. In a production grade product these settings would need to be configurable.

The configuration at the site controller is also hard coded in the proof-of-concept Python program running on the site controller.
8.1 Different sensors

Implementation and integration of different sensor types is expected. This will include designing new sensor node hardware, writing new sensor drivers and extending the node communications protocol.

8.2 Node configurability

The sensor nodes needs more and easier configuration regarding identity (network settings), delays and calibration. In order to simplify deployment such settings should not require recompiling and programming the micro controller but rather be performed over Bluetooth and/or wired serial communications.

8.3 Site controller

Site controller design remains mostly undone, specialized hardware is needed and the software is nowhere close to deployment grade.

8.4 OSS integration

The current OSS integration is one-way. Reconfiguration of the system should be possible via OSS. This includes integrating the site controller software with SNMPD or replacing SNMPD with extended functionality of the site controller software itself.
References


