The μracoli Source Package

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Overview

This package contains the uracoli-source code.

src	Source code of radio and ioutil library.
inc	The library header file.
wuart	The wireless UART application.
sniffer	The source code of the sniffer firmware.
wibo	The wireless bootloader source code, host application and examples.
xmpl	Some example applications, that illustrate how to use the µracoli functions.

In order to build the libraries and applications you need an installed AVR toolchain.

avr-gcc	AVR GCC C-compiler which we need to compile the libraries and the test examples
avr-binutils	Linker, Object File Converter, ..
avr-libc	Standard C library which provides a good set of C standard functions
avrdude	Tool to transfer the machine code (.hex files) via ISP or JTAG AVR interface to the internal Flash memory and/or EEPROM.
avr-gdb	GNU debugger to debug AVR programs
AVaRICE	This tool interfaces the AVR GNU debugger with the AVR JTAG interface which allows real in-circuit debugging

A detailed installation description is available at http://uracoli.nongnu.org/avrtools.html.

2 Compiling the Libraries

The libraries can be build with make. In order to get an overview, if your board is supported, type

```
make -C src/ list
```

The libraries for e.g. the "radiofaro" board, are build with the command:

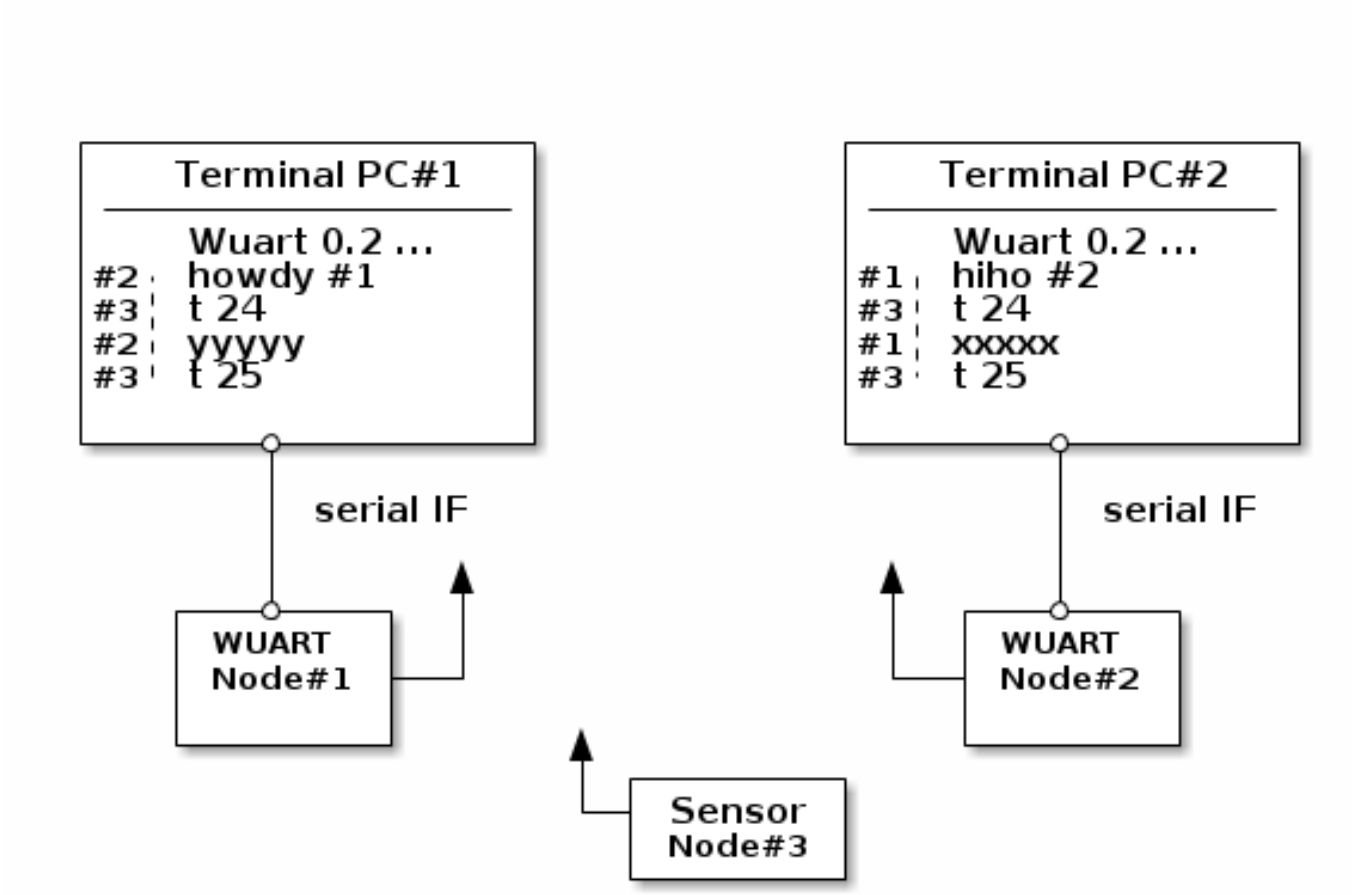
```
make -C src radiofaro
```

This will create the directory `lib` and the uracoli-library for radiofaro.

```
$ls lib/
liburacoli_radiofaro.a
```

3 Wireless UART

The wireless UART can be used to communicate between two PCs via a RF link, or to communicate between a PC and an autnomous sensor/actor node. The WUART application starts on a fixed channel and is automatically in the transparent data mode.



The wireless UART for the radiofaro board is build with the following commands:

```
make -C src radiofaro
make -C wuart radiofaro
```

The firmware file `wuart_radiofaro.hex` will be now available in the directoy `bin/`. It can be flashed in the microcontroller e.g. using an AVR Dragon programmer with the command:

```
avrdude -Pusb -cdragon_jtag -pm128rfa1 -U bin/wuart
```

Now open a serial terminal programm, adjust the baudrate, set the hardware handshake to *none* and after a reset of WUART node you will see a boot message, similiar to this:

```
Wuart 0.2 chan=17 radio 02.01
```

Do the same steps for a second board an try to chat between the terminal windows. Alternatively you can compile the example programm `xmpl_linbuf_tx.hex` and watch the text that appears in the terminal window of the PC.

```
make -C src anotherboard
make -C xmpl -f xmpl_linbuf_tx.mk anotherboard
```

4 Multiport Serial Terminal Program

This python script `wuart/sterm.py` is a usefull addon for debugging wireless applications. How to use the script is described in the document `doc/uracoli-sterm.pdf` or `doc/uracoli-sterm.txt`.

5 Sniffer Firmware

With the sniffer firmware the frames exchanged in an IEEE 802.15.4 network can be captured and transmitted over the serial interface to the PC. This firmware is intended for use with a the Python script `sniffer.py` and wireshark. The script together with the documentation, which explains the sniffer setup, can be found in the package uracoli-sniffer-<version>.zip on http://uracoli.nongnu.org/download.html.

The sniffer firmware can be compiled with the commands

```
make -C src mysnifferboard
make -C sniffer mysnifferboard
```

For which boards the sniffer application is available, you can find out with the command:

```
make -C sniffer list
```

6 Wireless Bootloader

The wireless bootloader (WiBo) is an application that resides in the bootloader section of the AVR and therefore it can erase and rewrite the programm flash memory. In Difference to a regular bootloader, WiBo receives its data over the RF link. WiBo modifies the flash directly, therefore no special backup flash memory, like in other over the air upgrade (OTA) solutions, is required on the transceiver board.

A detailed description how WiBo is used can be found in `doc/uracoli-wibo.pdf` or `doc/uracoli-wibo.txt`.

7 Examples

The example source code can be found in the directory `xmpl/`. This simple example programms are thought as starters for your application.

xmpl_leds.c	Example use of the LED macros
xmpl_key_events.c	Example for key event processing with a single key
xmpl_keys.c	Example use of the KEY macros
xmpl_hif.c	Example for use of the HIF functions
xmpl_hif_echo.c	Example that implements HIF echo, usefull to test the HIF troughput
xmpl_timer.c	Example for using the timer macros
xmpl_dbg.c	Example for use of the DBG_XXX macros
xmpl_linbuf_tx.c	Example use of the buffer functions
xmpl_trx_base.c	Example for accessing the transceiver

xmpl_trx_echo.c	Example for echoing received frames
xmpl_trx_rxaack.c	Example for receiving frames in rx_aack mode
xmpl_trx_rx.c	Example for receiving frames
xmpl_trx_txaret.c	Example for transmitting frames in tx_aret mode
xmpl_trx_tx.c	Example for transmitting frames
xmpl_radio_range.c	Example use of the radio and ioutil functions for a simple range test
xmpl_radio_stream.c	Example use of the radio stream functions

The example firmware can be build with the following commands:

```
make -C src myboard
make -C xmpl -f xmpl_<myexample>.mk myboard
```

Note: Some of the examples are not available on all boards, due to the lack of some hardware features, e.g. if the LEDs, the KEYs or/and the HIF is absent. A list of supported boards of the example can be obtained by:

```
make -C xmpl -f xmpl_<myexample>.mk list
```