µracoli Support Package for Arduino 1.x

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Abstract

This custom hardware package enables the use of the following radio boards with Arduino 1.x:

- Radiofaro
- 2.4GHz ZigBit Modules
- 900MHz ZigBit Modules

It provides a custom Arduino core, that includes the radio functionality of µracoli, that can be used from sketches. The core is derived from Arduino package version 1.0. For updates check http://uracoli.nongnu.org/download.html.

Beside the radio board core and example sketches, this package comes with a copy of Arscons, a tool that enables to build and upload sketches from the command line with Scons.

Since this package contains a mix of source code from different packages check out the license section at the of the file.

1 Installation and First Run

At first install the Arduino-IDE with a version 1.0 or later according to the instructions given here http://arduino.cc/en/Guide/-HomePage and check if it runs. You will get prompted for a **sketchbook location** when you start the Arduino-IDE for the first time.

Download the latest µracoli Arduino support package `uracoli-arduino-<version>.zip` from http://uracoli.nongnu.org/-download.html and unzip it directly in the sketchbook directory. This will create three folders in the sketchbook directory, `hardware`, `examples` and `doc`. The folder `hardware` contains the Radiofaro and Zigbit core, `examples` contains sketches, that use the uracoli radio core functions and a copy of Arscons [?]. In `doc` you will find this file and a copy of the license files Section 5, that are involved in this package.

If you forgot the location of your sketchbook directory, you will see it in the dialogue, that opens by using the menu entry `File/Preferences`.

Restart the Arduino-IDE after unpacking `uracoli-arduino-<version>.zip` and check if you see the new boards at the end of the list that opens if you click the menu entry `Tools/Board` and select your current radio board, e.g. "Radiofaro", "Zigbit 2400MHz", etc. Now select the serial port to which your radio board is connected via the menu `Tools/Serial Port`. After adjusting this settings open the `HelloRadio` sketch by selecting `File/Sketchbook/Radio/HelloRadio`. Now select the menu entry "File/Upload". If the upload fails for some reason, mark the options "compile verbose" and "upload verbose" in the dialogue that opens when clicking `File/Preferences` and see in the lower window of the Arduino-IDE what goes wrong.

The sketch "HelloRadio", that you have currently uploaded, sends a short frame every 500ms on channel 17 and reports the transmission also on its serial port. You can open a terminal with `Tools/SerialMonitor` and you should see in the terminal window:

```
HelloRadio
TX: 0
TX: 1
TX: 2
...
```

Each printed number shows, that a frame was successfully transmitted. If you see this output, that means that you are now "on air".

2 API documentation

The class `HardwareRadio` provides the following methods. TODO: describe more

```
    /** constructor 1*/
    void begin(void);

    /** constructor 2*/
    void begin(uint8_t channel, uint8_t idlestate);

    /** Method to write a byte */
    void write(uint8_t byte);
    /** Method to write a string */
    void write(char * str);

    /** Insert a 16 bit integer value in the stream */
    void put(int16_t value);

    /** Flush the TX buffer. */
    void flush(void);

    /** Check if data in the RX buffer is availbale. */
    int available(void);
```

```
/** Read a byte from the RX bufffer. */
int read(void);

/** Retireve a 16 bit integer value from the stream */
void get_int(int16_t& value);

/** Allocate a radio buffer */
radio_buffer_t * alloc_buffer(void);
/** Free a radio buffer */
void free_buffer(radio_buffer_t * pbuf);
```

3 Bootloaders

TODO: decribe and add zigbit in the package

4 Building Sketches with Arsons

Basics Arscons is a scons script by Homin Lee, that builds and uploads Arduino sketches on the command line. It uses no Java and since SCons is used as build engine, it tracks the dependencies among the source files automatically, rather then the Arduino-IDE. Thus Arscons reduces the development cycle time drastically. Arscons can be simply used by:

- copying or symlinking the file `examples/Arscons/SCostruct` in a sketch directory,
- change to the sketch directory and run
  `scons ARDUIO_BOARD=<yourboard> ARDUINO_PORT=<yourport> upload`

Arscons Options Arscons is controlled by command line options and environment variables. The following commands are available:

- `scons [OPTIONS]`
  just compile the sketch
- `scons upload [OPTIONS]`
  compile sketch if needed and uploaded it with to the Arduino board.

The following variables can be set either with an enviroment variable or with a command line argument. If both are given, the command line option overrides the environment variable.

- `ARDUINO_BOARD`
  The name of the board, where Arduino is connected.
- `ARDUINO_HOME`
  Directory where the Arduino-IDE is installed.
- `ARDUINO_PORT`
  The name of the serial port , where Arduio is connected.
- `ARDUINO_VER`
  The IDE version can be forced with this variabled, normally the version is automatically determined.
- `EXTRA_LIB`
  List of extra directories with Arduino libraries.

- `SKETCHBOOK_HOME`
  The location of the sketchbook, this is needed to find libraries in it.

This options can be overwritten by command line only:

- `MCU`
  Name of the microcontroller.
- `F_CPU`
  Clock frequency of microcontroller in Hz.

Arscons Example Session under Linux Be sure to have the Arduino-IDE closed, before you run `scons [...] upload` The preparation of a Sketch directory for Arscons looks under Linux so:

```
cd examples/Radio/HelloRadio
ln -sfv ../../Arscons/SConstruct
```

and an example session so:

```
scons ARDUINO_BOARD=radiofaro ARDUINO_PORT=/dev/ttyUSB0 \
      SKETCHBOOK_HOME=/mnt/netshare/arduino/sketchbook upload
```

or so:

```
export ARDUINO_BOARD=radiofaro
export ARDUINO_PORT=/dev/ttyUSB0
export SKETCHBOOK_HOME=/mnt/netshare/arduino/sketchbook
scons upload
```

5 Licenses

This package incorporates source code from different license models, which has an influence of the use of the code in proprieatary projects and environments.

GPL version 3.0 All files in the Arscons directory are licensed with GPL version 3. Arscons is a build script, which does not generate code itself. So it can be regarded like a compiler or a linker. See also http://www.gnu.org/licenses/gpl.txt or file license_gpl_3v0.txt.

GPL version 2.0 According to the file header, the bootloader is licensed under GNU General Public License version 2.0. See http://www.gnu.org/licenses/gpl-2.0.txt or file license_gpl_2v0.txt.

```
hardware/uracoli/bootloaders/radiofaro/ATmegaBOOT.c
```

LGPL version 2.1 The files copied from the original Arduino core are licensed under the GNU Lesser General Public License version 2.1. This code is linked to each sketch. See http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt or file license_lgpl_2v1.txt.

```
hardware/uracoli/variants/zigbit900/pins_arduino.h
hardware/uracoli/variants/zigbit2400/pins_arduino.h
hardware/uracoli/variants/radiofaro/pins_arduino.h
hardware/uracoli/cores/uracoli/Print.cpp
hardware/uracoli/cores/uracoli/wiring_pulse.c
hardware/uracoli/cores/uracoli/Printable.h
hardware/uracoli/cores/uracoli/Stream.cpp
hardware/uracoli/cores/uracoli/HardwareSerial.cpp
hardware/uracoli/cores/uracoli/WCharacter.h
hardware/uracoli/cores/uracoli/wiring_digital.c
hardware/uracoli/cores/uracoli/Tone.cpp
hardware/uracoli/cores/uracoli/Stream.h
```

```
hardware/uracoli/cores/uracoli/WInterrupts.c
hardware/uracoli/cores/uracoli/WString.cpp
hardware/uracoli/cores/uracoli/WMath.cpp
hardware/uracoli/cores/uracoli/wiring.c
hardware/uracoli/cores/uracoli/wiring_analog.c
hardware/uracoli/cores/uracoli/wiring_shift.c
hardware/uracoli/cores/uracoli/Print.h
hardware/uracoli/cores/uracoli/HardwareSerial.h
hardware/uracoli/cores/uracoli/WString.h
hardware/uracoli/cores/uracoli/wiring_private.h
```

Modified BSD license The sources of the µracoli radio functions are licensed under the modified 3 clause BSD license. See file license_uracoli.txt.

```
hardware/uracoli/cores/uracoli/trx_rf230_param.c
hardware/uracoli/cores/uracoli/const.h
hardware/uracoli/cores/uracoli/boards/base_zdma1281.h
hardware/uracoli/cores/uracoli/boards/board_derfa.h
hardware/uracoli/cores/uracoli/boards/board_wdba1281.h
hardware/uracoli/cores/uracoli/trx_rf230_sram.c
hardware/uracoli/cores/uracoli/trx_datarate_str.c
hardware/uracoli/cores/uracoli/board.h
hardware/uracoli/cores/uracoli/at86rf230b.h
hardware/uracoli/cores/uracoli/trx_rf230.c
hardware/uracoli/cores/uracoli/at86rf212.h
hardware/uracoli/cores/uracoli/atmega_rfa1.h
hardware/uracoli/cores/uracoli/trx_datarate.c
hardware/uracoli/cores/uracoli/trx_rf230_frame.c
hardware/uracoli/cores/uracoli/trx_rfa.c
hardware/uracoli/cores/uracoli/trx_rf230_irq.c
hardware/uracoli/cores/uracoli/trx_rf230_bitwr.c
hardware/uracoli/cores/uracoli/trx_rf230_bitrd.c
hardware/uracoli/cores/uracoli/radio_rf230.c
hardware/uracoli/cores/uracoli/at86rf230a.h
hardware/uracoli/cores/uracoli/radio_rfa.c
hardware/uracoli/cores/uracoli/radio.h
hardware/uracoli/cores/uracoli/trx_rf230_crc.c
hardware/uracoli/cores/uracoli/transceiver.h
hardware/uracoli/cores/uracoli/usr_radio_irq.c
hardware/uracoli/cores/uracoli/trx_rf230_misc.c
examples/Radio/RadioUart/RadioUart.ino
examples/Radio/IoCheck/IoCheck.ino
examples/Radio/HelloRadio/HelloRadio.ino
examples/Radio/IoRadio/IoRadio.ino
```

Conclusions IANAL, but I would presume that LGPL version 2.0 has the strongest influence on a compiled sketch, because code from LGPL version 2 and 3 clause BSD is linked. The bootloader is a seperate programm and is not linked with the sketch. The Arscons script is a compilation helper tool, it does not create any code that is linked. However all this conclusions maybe obsolete, since I am as a technician have not the in depth knowledge of all the license aspects. However, if you are fine to publish your sketches according LGPL version 2.0, you will be fine with this non trivial licensing constellation.