µracoli Arduino Support Package Documentation

Release 0.6.0

Daniel Thiele, Axel Wachtler

Jun 03, 2017

CONTENTS

CHAPTER

ONE

INTRODUCTION

The µracoli project provides a support package for the Arduino-IDE .

The µracoli-Arduino-Support-Package (UASP) includes the HardwareRadio-Library, which makes the usage of the radio transceiver as easy as the *Serial* module.

UASP requires an Arduino-IDE with a versions above > 1.5.x.

The following boards are supported in the package:

- radiofaro - Radiofaro w/ ATmega128rfa1
- pinoccio - Pinoccio Scout
- atzb256rfr2xpro - Atmel ATmega256RFR2 Xplained Pro Evaluation Kit
- wdba1281 - Zigbit 2400MHz, w/ ATmega1281
- mnb900 - Zigbit 900MHz, w/ ATmega1281

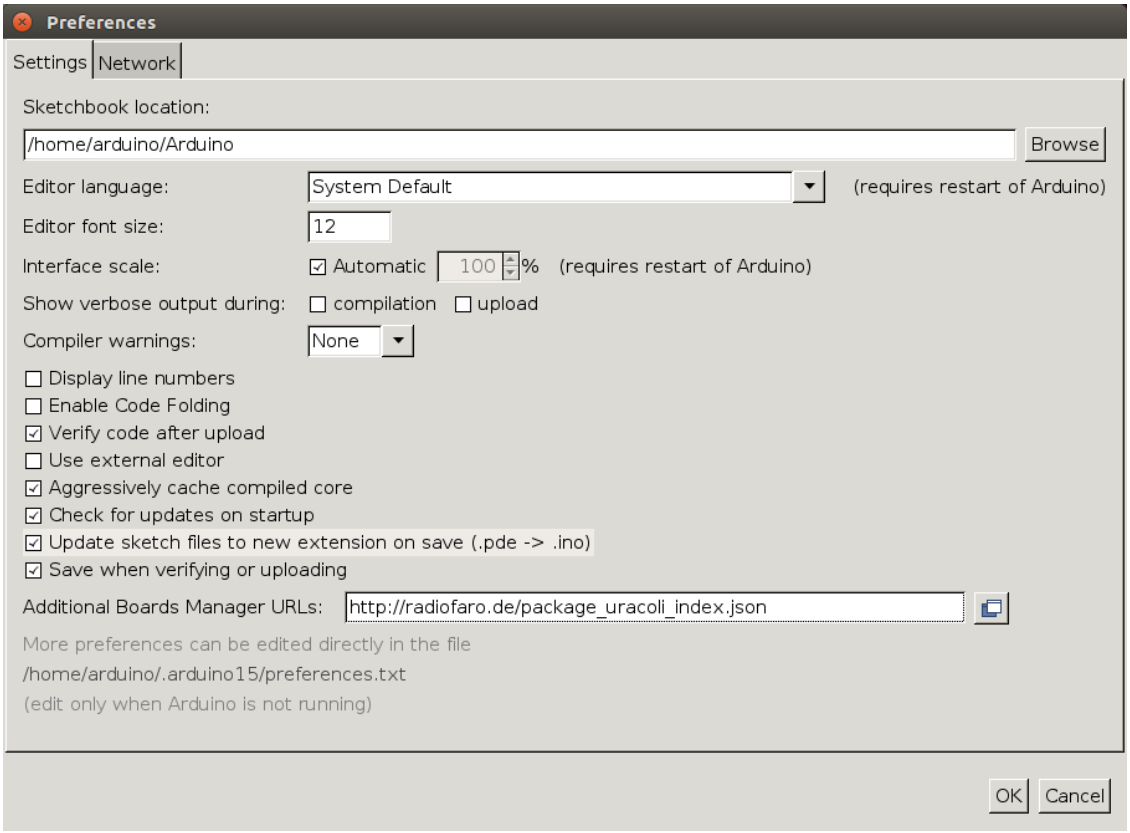
CHAPTER

TWO

INSTALLATION

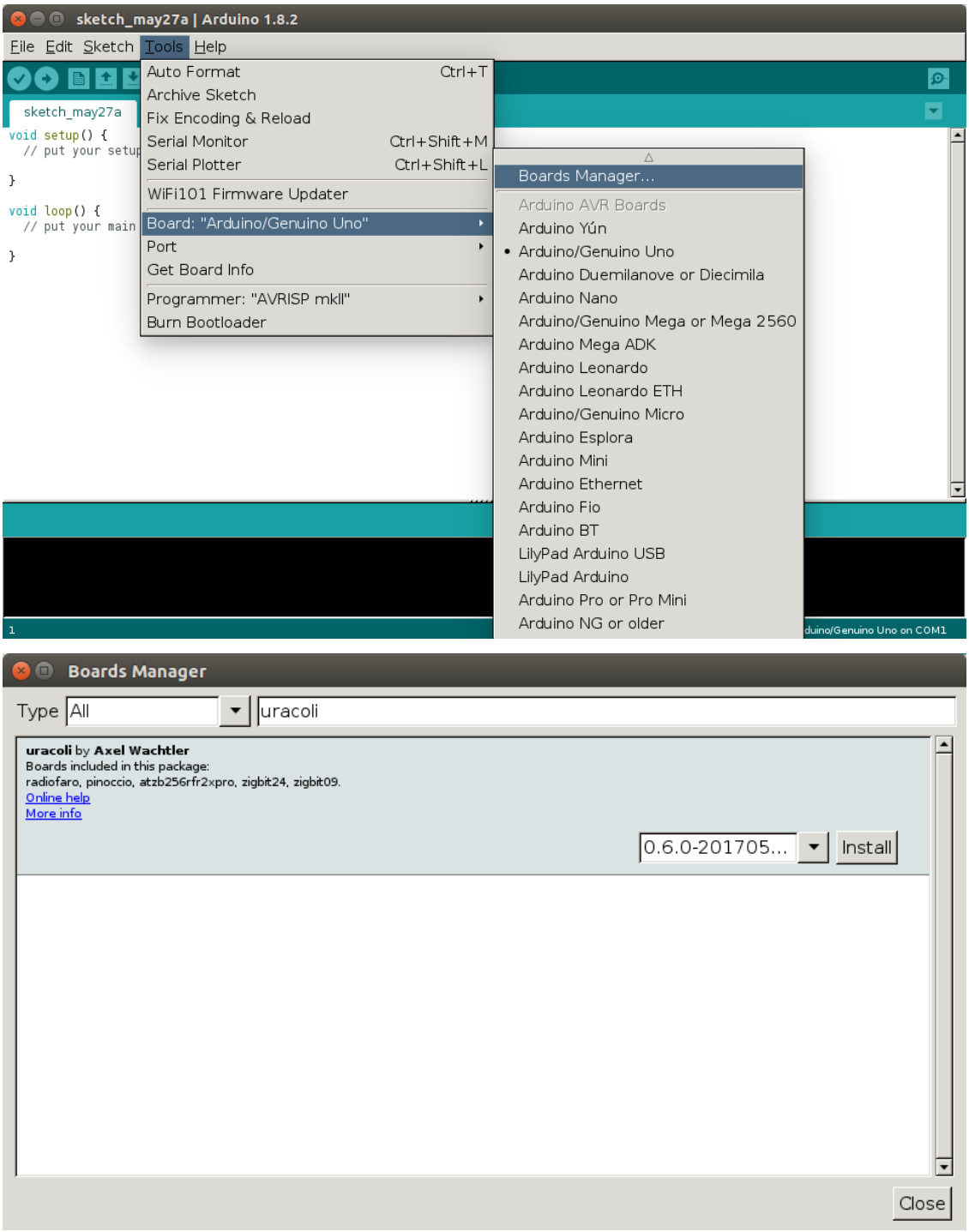
2.1 Add the Board Manager URL for the UASP

- From the Main Menu select *File / Preferences*
- In *Additional Boards Manager URLS* enter http://radiofaro.de/package_uracoli_index.json



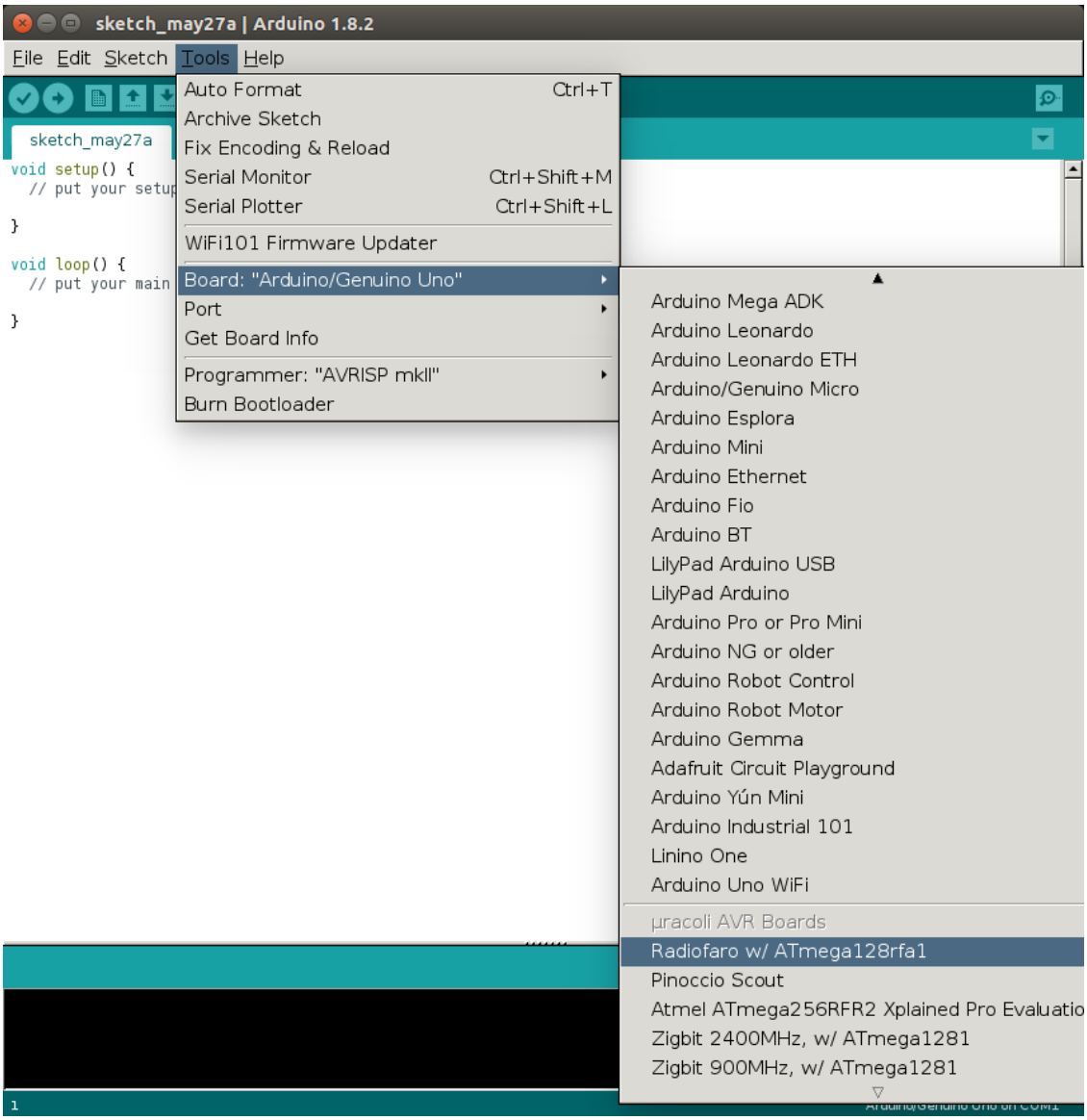
2.2 Install the Package

- From the Main Menu select *Tools / Board / Boards Manager*
- type “uracoli” in the search field and press *Install* to select the latest package version.
- Press close to exit the *Boards Manager*



2.3 Select the Radio Transceiver Board

- From the *Tools /Board* menu select one of the supported radio transceiver boards.


sketch_may27a | Arduino 1.8.2
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
sketch_may27a
void setup() {
// put your setup
}
void loop() {
// put your main
}
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino/Genuino Micro
Arduino Esplora
Arduino Mini
Arduino Ethernet
Arduino Fio
Arduino BT
LilyPad Arduino USB
LilyPad Arduino
Arduino Pro or Pro Mini
Arduino NG or older
Arduino Robot Control
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
Arduino Yún Mini
Arduino Industrial 101
Linino One
Arduino Uno WiFi
µracoli AVR Boards
Radiofaro w/ ATmega128rfa1
Pinoccio Scout
Atmel ATmega256RFR2 Xplained Pro Evaluatio
Zigbit 2400MHz, w/ ATmega1281
Zigbit 900MHz, w/ ATmega1281
1

CHAPTER

THREE

EXAMPLE SKETCHES

3.1 Wireless UART

This example shows, how with a few lines of code, data from the serial interface of one board can be transfered wirelessly to a remote board.

```
/* RadioFaro wirless/serial bridge. */
#include "HardwareRadio.h"

/* data captured from UART */
char serial_inbyte = 0;

/* data received from Radio */
char serial_outbyte = 0;

/* next time the radio buffer will flushed */
unsigned long tx_time;

void setup() {
    Radio.begin();
    Serial.begin(57600);
    tx_time = 0;
    Serial.println("RadioUart.ino");
}

void loop() {

    if ((Serial.available() > 0))
    {
        serial_inbyte = Serial.read();
        Radio.print(serial_inbyte);
    }

    if (millis() > tx_time){
        tx_time = millis() + 25;
        Radio.flush();
    }

    if (Radio.available() > 0)
    {
        serial_outbyte = Radio.read();
        Serial.print(serial_outbyte);
    }
}
```

3.2 Radio LED

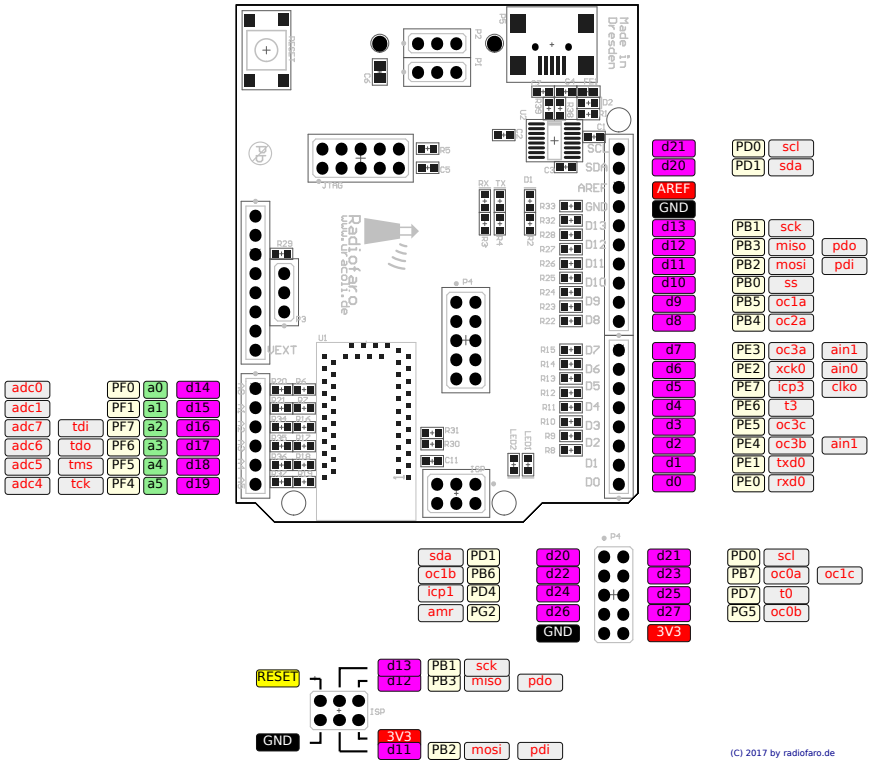
```
/*
 * Example sketch, chatting to Radiofaros by keypress + LED toggling
 *
 * Radiofaro LED1 used as keypress indicator, toggling each press
 * Button connected between d8 and d10, plugged into female header
 */

#include "HardwareRadio.h"

static const uint8_t pinLed = LED1_BUILTIN;
static const uint8_t pinButton = 8;
static const uint8_t pinButtonGnd = 10;

void setup() {
  pinMode(pinLed, OUTPUT);
  pinMode(pinButtonGnd, OUTPUT);
  digitalWrite(pinButtonGnd, 0);
  pinMode(pinButton, INPUT_PULLUP);
  Radio.begin(17, STATE_RX);
}

void loop() {
  static uint8_t ledtgl=0;
  static uint8_t buttonstate=0;
  static uint8_t lastbuttonstate=0;
  static uint16_t debounce_tmr=0;

  buttonstate = digitalRead(pinButton);
  if((millis() > debounce_tmr) && (buttonstate != lastbuttonstate)){
    switch(buttonstate){
      case LOW:
        ledtgl=1-ledtgl;
        Radio.write('A');
        Radio.flush();
        break;
      case HIGH:
        // do nothing
        break;
      default:
        break;
    }
    lastbuttonstate = buttonstate;
    debounce_tmr = millis()+30; // some button debouncing
  }

  if (Radio.available() > 0)
  {
      Radio.read();
      ledtgl=1-ledtgl;
      debounce_tmr = millis()+30; // some button debouncing
  }

  digitalWrite(pinLed, ledtgl);
}
```

CHAPTER

FOUR

REFERENCES

4.1 Radiofaro Pin Mappings



4.2 Function Reference

class `HardwareRadio`

Hardware Radio class

Inherits from Stream

Public Functions

radio_buffer_t ***`alloc_buffer`** (void)

Allocate a radio buffer

void **free_buffer**(radio_buffer_t **pbuf*)
Free a radio buffer

HardwareRadio(void)
constructor

void **begin**(void)
Starting the hardware radio class with default parameters

Note The following default parameters are used implicitely.

- channel: PHY_DEFAULT_CHANNEL
- idlestate: STATE_RXAUTO
- pan id: DEFAULT_PAN_ID
- destination address: DEFAULT_SHORT_ADDRESS
- source address: DEFAULT_SHORT_ADDRESS

void **begin**(uint8_t *channel*, uint8_t *idlestate*)
Starting the hardware radio class with explicit parameters

Note The following default parameters are used implicitely.

- pan id: DEFAULT_PAN_ID
- destination address: DEFAULT_SHORT_ADDRESS
- source address: DEFAULT_SHORT_ADDRESS

Parameters

- `channel`: radio channel (11 - 26 for 2.4GHz radios, 0 - 10 for SubGHz radios)
- `idlestate`: default state of the radio, supported values are listed in radio_state_t.

void **begin**(uint8_t *channel*, uint8_t *idlestate*, uint16_t *pan*, uint16_t *dst*, uint16_t *src*)
Starting the hardware radio class with explicit parameters

Parameters

- `channel`: radio channel (11 - 26 for 2.4GHz radios, 0 - 10 for SubGHz radios)
- `idlestate`: default state of the radio, supported values are listed in radio_state_t.
- `pan`: _p_ersonal _a_rea _n_etwor ID
- `dst`: 16 bit destination address
- `src`: 16 bit node address

virtual int **available**(void)
return number of available bytes in current RX buffer

virtual int **peek**(void)
Returns the next byte (character) of incoming data (RX) without removing it from the internal serial buffer.

Return EOF (-1) if no data available, otherwise a value from 0 . . . 255

virtual int **read**(void)
Returns the next byte (character) of incoming data (RX)

Return EOF (-1) if no data available, otherwise a value from 0 . . . 255

virtual void **flush**(void)
flush TX and RX queues. RX queue data are discarded, TX data is sent.

virtual size_t **write**(uint8_t)
write a byte to the TX stream

void **write**(char *_str_)
write a string to the TX stream

Parameters

- `str`: \0 terminated string

void **write**(uint8_t *_buf_, uint8_t _size_)
write a binary buffer (buf, size) to the TX stream

Parameters

- `buf`: pointer to the buffer
- `size`: number of bytes in the buffer.

uint8_t **sendto**(uint16_t _dst_, uint8_t *_pbuf_, uint8_t _size_)
send binary data direct to a node, adressed by dst.

Return number of bytes transmitted

Note At the end of this routine *flush* is called, and therefore the data collected in *write* and and print are sent out to.

Parameters

- `dst`: destination address
- `pbuf`: pointer to data array
- `size`: number of bytes to transmit

CHAPTER

FIVE

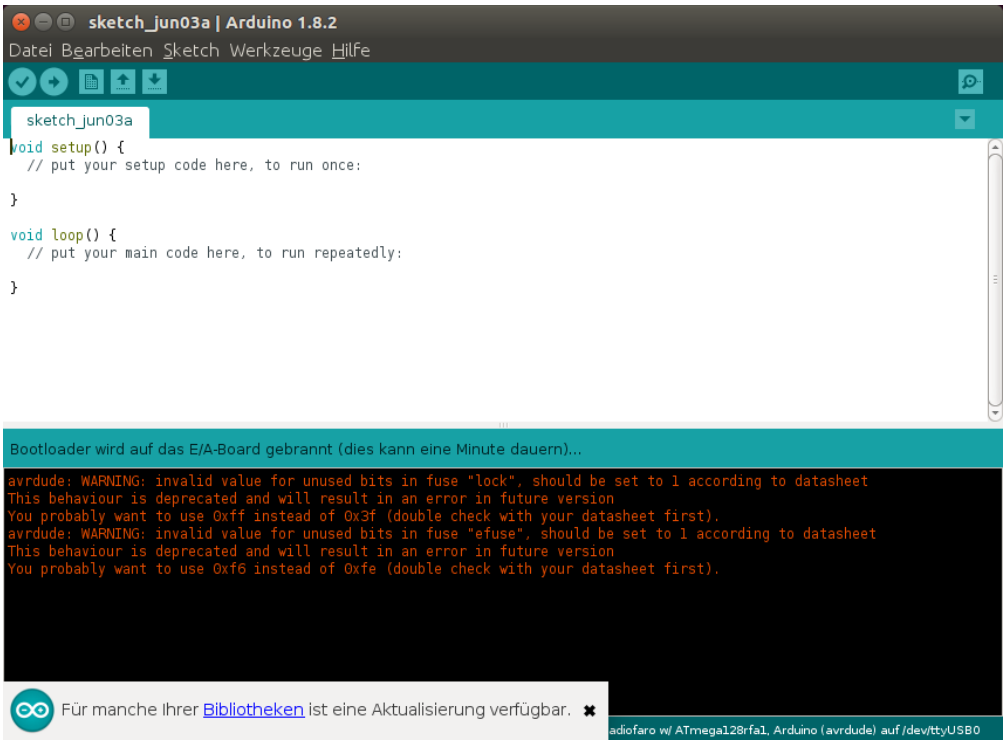
BOOTLOADER INSTALLATION

In the case that sketches can not be uploaded, one reason maybe, that there is no a working Bootloader on the board. So before flashing the bootloader, check your setup, e.g. serial port and board type are selected correctly and probably remove all external components.

5.1 Installation via Arduino IDE

The Bootloader can be flashed over the Arduino-IDE.

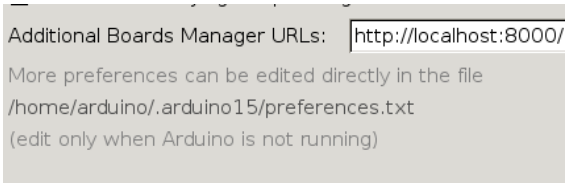
- connect an external programmer (AVRISP, UBStinyISP, . . .) to the board.
- from the Main Menu select in *Tools / Programmer:* the programmer device you have connected.
- now select *Tools / Burn Bootloader*, ignore the avrdude-warnings and after a while the board has a new bootloader.
- to verify that the new bootloader is working, upload a sketch to the board
- disconnect the external programmer.



5.2 Installation with arvdude

The Bootloader can be found in the package install path, e.g. under Linux it might look so *~/.arduino15/packages/uracoli*.

You can find the local Arduino installation directory via *File / Preferences*, in the dialogue window, check out the location of the config file



The bootloader can then flashed with this commands.

```
cd ~/.arduino15/packages/uracoli/hardware/avr/0.6.0-20170509/bootloaders/radiofaro
avrdude -P usb -c jtag2 -p m128rfa1 -U ATmegaBOOT_radiofaro.hex
avrdude -P usb -c jtag2 -p m128rfa1 \
      -U lf:w:0xe2:m -U hf:w:0x1a:m -U ef:w:0xfe:m
```

CHAPTER

SIX

WHAT IS ARDUINO?

6.1 Overview

A few words about Arduino. Who knows what Arduino is, may skip this section.

Arduino is a Microcontroller development plattform that consists of a Java-IDE which supports various microcontroller boards. Most of the supported boards are equipped mit 8-bit-AVR microcontrollers.

Basically the Arduino-IDE provides a simple code editor and the firmware (denoted as "sketches") can be compiled and flashed with a button click. A serial terminal completes the IDE. This high level of abstration makes it easy for none technicians and first-time users to start with embedded programming.

6.2 What is a Sketch?

A sketch basically implements two functions setup() and loop() that are called from the main function of the core library. The available API-functions are described at http://arduino.cc/en/Reference/HomePage.

Beside the standard functions to operate digital and analogue IO-pins, **Arduino_** also provides various libraries for different hardware peripherals and shields.

6.3 Arduino.org vs. Arduino.cc

The Arduino project did fork in 2014, see http://hackaday.com/2015/04/06/arduino-ide-forked/ for details. Since this time there are two versions of the IDE avaiable.

IDE from arduino.cc:: https://www.arduino.cc/en/Main/Software

IDE from arduio.org:: http://www.arduino.org/downloads

Meanwhile the projects has rejoined and reference to the same IDE.

An interesting reading about the Arduino-History can be found here: http://arduinohistory.github.io/

INDEX

H