Projektautomatisierung

Am Beispiel von *Piracoli*

Axel Wachtler und Ralf Findeisen

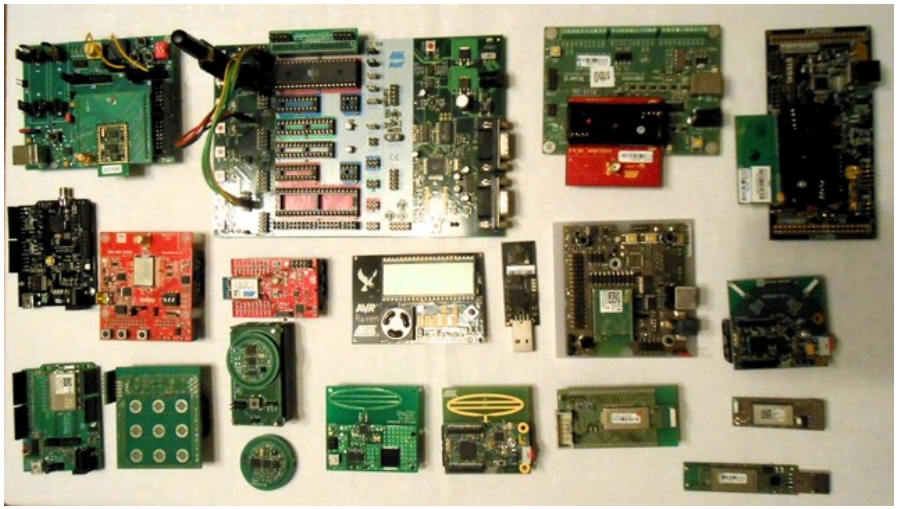
Einleitung

Open-Source Bibliothek für

- alle Atmel IEEE-802.15.4-Transceiver
- und AVR Mikrocontroller (ATtiny, ATmega, ATxmega)

Bereitstellung von Libraries, Beispielen und Anwendungen für über 80 verschiedene Board Varianten

Unser Problem ...



... funktionieren die ca. 15000 Codezeilen auf den Boards (noch)?

Motivation

- Funktionierende Software = Reputation des Projektes
  - Nutzer sollen keine Zeit mit Fehlersuche vergeuden.
- Manuelles Testen ist zeitaufwändig
- Automatisierung stets wiederkehrender Abläufe
- Fehlervermeidung
  - Wiederholbarkeit
  - Test verschiedener Setups/Platformen
- Zentrale Erfassung von Resultaten

Entwicklungsprozess

- Implementation neuer Features auf Branch
- Wenn Feature fertig, dann Merge mit Mainline
  - Konflikte beseitigen,
  - Einmal alles compilieren,
  - Pakete erzeugen
  - Fertig???
- Ein Release besteht aus mehreren Paketen (devel, src, doc, sniffer, arduino)
  - Sind die Pakete vollständig?
  - Funktioniert die Firmware auf den unterschiedlichen Boards?

Werkzeuge

- Konfigurierbares Buildsystem
  - scons bzw. GNU Make
- AVR Toolchain zum Bau der Images
  - avr-gcc, AVR Libc,
  - avrdude, GCF Flasher
- Tools
  - Python
  - Doxygen
- Continous-Intergation Server Jenkins

Jenkins

An extendable open source continuous integration server

Buildsystem

- scons
  - Leistungsfähig, plattformübergreifend
  - Automatische Abhängigkeiten.
- Metainformationen als INI-Files
  - `board.cfg`
  - `application.cfg`
  - `packages.cfg`
  - `setup.cfg`
- Abbildung von Abhängigkeiten in den Files
- Nutzung der INI-Files für:
  - Compilierung
  - Code- und Doc-Generierung
  - Paketerzeugung

Abhängigkeiten in den INI-Files

File `board.cfg`

```
[raspbee]
provides = trx hif led tmr
```

File `application.cfg`

```
[sniffer]
requires: trx hif led
```

File `packages.cfg`

```
[psniffer]
boards = psk230 psk230b stb230 stb230b ... raspbee
```

File `setup.cfg`

```
[b0]
hif=/dev/ttyAMA0
board = raspbee
```

board.cfg für RaspBee

```
[raspbee]
comment    = Dresden Elektronik Raspberry Pi Module
image      = raspbee.jpg
include    = boards/board_derfa.h
cpu        = atmega256rfr2
f_cpu      = 8000000UL
sensors    = mcu_vtg mcu_t
provides   = trx hif led tmr
...
```

application.cfg für den Sniffer

```
[sniffer]
requires: trx hif led
excludes: stkm8
sources: Sniffer/sniffer.c
         Sniffer/sniffer_ctrl.c
         Sniffer/sniffer_scan.c
```

packages.cfg für den Sniffer

```
[psniffer]
name = uracoli-sniffer
boards = psk230 psk230b stb230 stb230b ...
depends = build/sniffer
files = install/bin/sniffer_psk*.hex
        install/bin/sniffer_stb*.hex
        ...
relocate = install/bin:firmware
           Tools:script
        ...
```

setup.cfg für Radiofaro und RaspBee

```
[setup]
pairs = b1 : b0, b2: b0

[b0]
hif=/dev/ttyAMA0
board = raspbee
flvfy =
flwrt = GCFFlasher -f%(fw)s

[b1]
hif=/dev/ttyUSB0
board = radiofaro
flvfy = avrdude -P usb -c dragon_jtag -pm128rfa1 -U fl:v:%(fw)s:i
flwrt = avrdude -P usb -c dragon_jtag -pm128rfa1 -U fl:w:%(fw)s:i
fltmo = 5
```

Anforderungen an Tests

- Tests müssen:
  - Einfach ausführbar sein
  - Zuverlässig sein
  - Fehler zuverlässig aufdecken
- Klare Fehlermeldungen
  - "File not found!" oder 200 Zeilen Stacktrace
- Start der Tests von der Kommandozeile (interaktiver Mode)
- Benutzung der Tests in einem Build/CI-Server (automatischer Mode)
- Ergebnis-Logging

Testframework Py.Test

- http://pytest.org/latest/
- Wichtigste Eigenschaften
  - Tests können sehr einfach beschrieben werden (Python Sprachmittel sind ausreichend)
  - Assert Statement
  - Aussagekräftige Assert-Reports
  - Automatische Testdetektion (Durchsuchen von Python Skripts)
  - Fixtures: einfache Erweiterbarkeit durch flexibles Plugin-Konzept (Hooks)
- Print-Debugging während der Testentwicklung

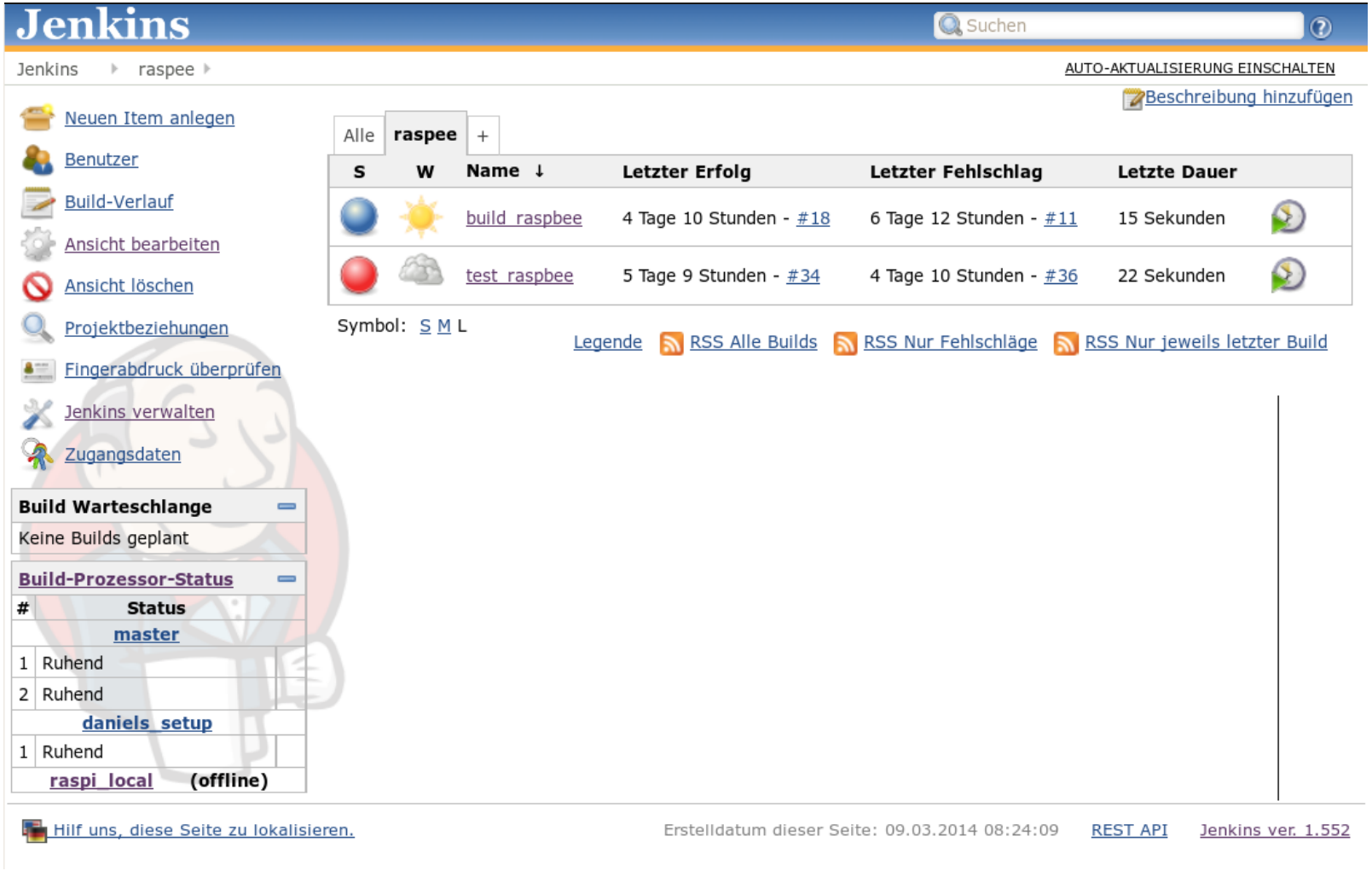
Py.Test Testcase

```
import pytest


@pytest.fixture
def answer_ultimate_question():
    return 24


class TestFoo:
    def test_answer(self, answer_ultimate_question):
        assert answer_ultimate_question == 42
```

Py.Test Testlauf

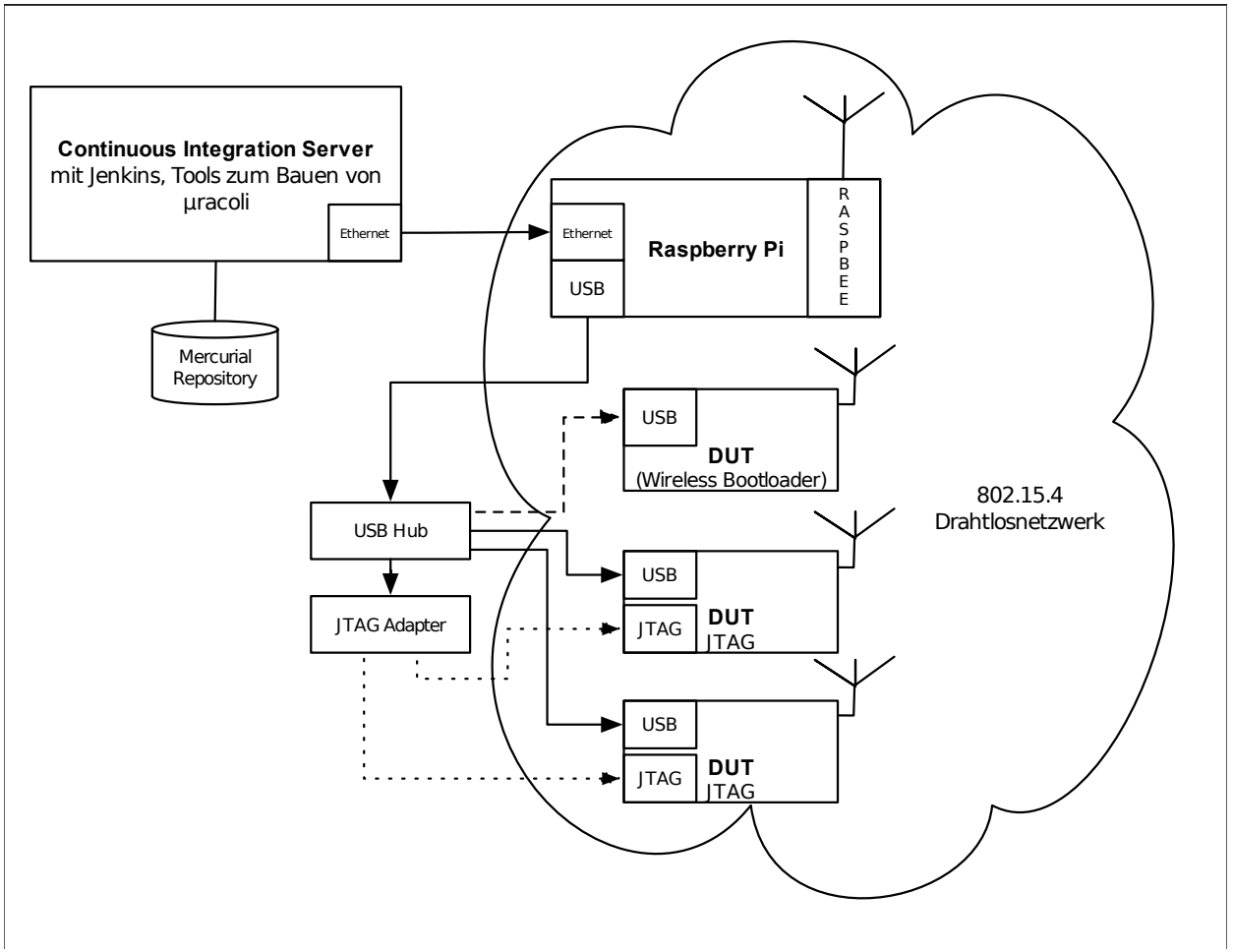
```
$ py.test
======================== test session starts ================
platform linux2 -- Python 2.7.3 -- py-1.4.20 -- pytest-2.5.2
collected 1 items

test_foo.py F

============================ FAILURES =======================
_____________________ TestFoo.test_answer ___________________

self = <test_foo.TestFoo instance at 0x2c014d0>
answer__ultimate_question = 24

def test_answer(self, answer__ultimate_question):
>       assert answer__ultimate_question == 42
E       assert 24 == 42

test_foo.py:10: AssertionError
==================== 1 failed in 0.01 seconds ===============
```

Jenkins Server



Testautomatisierung

- Jenkins als Build- und Continuous Integration Server
- Verteilte Slaves erlauben auf unterschiedlicher Hardware zu testen



Jenkins + Plugins

- Mercurial plugin
  - Zugriff auf das µracoli-Repository
- Credentials Plugin (ssh key)
  - Erlaubt den SSH-Zugang des Users "Jenkins" auf den Slave
- Artifact Deployer Plug-in
  - Verteilen von Build-Ergebnissen auf die Slaves
- Junit XML Reports
  - Aufbereitung und Anzeige der Testergebnisse

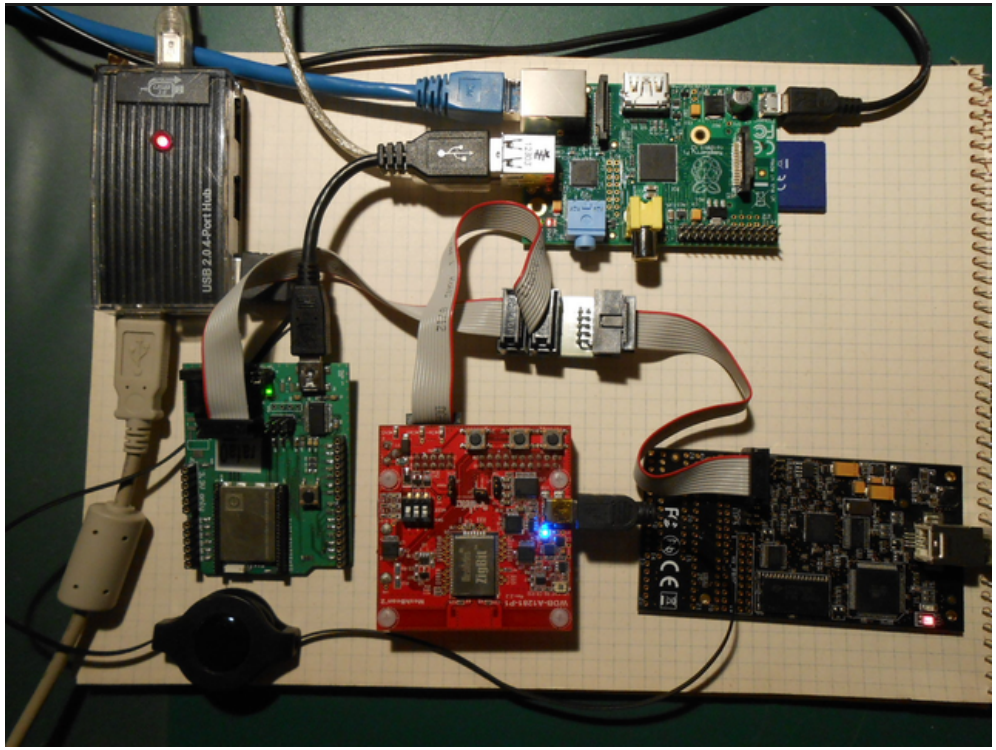
Jenkins Slaves

- Warum Slaves?
  - Verteiltes Setup an mehreren Orten
  - Entwickler stellen unterschiedliche Hardware zur Verfügung
- Anforderungen
  - Stromsparender Dauerbetrieb
  - Per SSH erreichbar, ggf. Portforwarding und DynDNS
- Man kann Slaves natürlich noch an vielen anderen Stellen im Build- und Test Prozess, z.b. beim *verteilten Softwarebau* einsetzen.

RaspberryPi + RaspBee

- Raspberry-Pi
  - Hat alles was man braucht: USB, Netzwerk, Funkadapter, Linux
  - 6,5W Standby = 56.64 kWh / Jahr = 14,80€/Jahr.
- RaspBee
  - Zusatzmodul mit Atmega256RFR2
  - Verbindung: serielle Schnittstelle + GPIO für Reset
  - Modul kann per Bootloader geflasht werden

Live Demo



Lokales Setup per USB

Remote Setup mit SSH Slave

Danksagung

Dietzsch und Thiele Partnerschaftsgesellschaft
Für die Bereitstellung des Testservers.

Jörg Wunsch (DL8DTL):
Für die Unterstützung von µracoli in den vergangenen 6 Jahren und für das Hosten von www.uracoli.de.

dresden elektronik ingenieurtechnik gmbh
Für die Bereitstellung von Test-Hardware.

Fragen?