

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AIR)
SPECJALNOŚĆ: Embedded Robotics (AER)

**PRACA DYPLOMOWA
MAGISTERSKA**

Embedded Linux
build systems

Systemy implementacji
wbudowanego Linuxa

AUTOR:
Cezary Dynak

PROWADZĄCY PRACĘ:
dr Witold Paluszyński

OCENA PRACY:

To my wife and son

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 1.1 | Goals of the project | 3 |
| 2 | Literature review | 5 |
| 2.1 | elinux.org | 5 |
| 2.2 | wikipedia.org | 6 |
| 2.3 | Marcin Bis publications | 6 |
| 2.4 | Conferences and workshops | 6 |
| 2.5 | scholar.google.com | 7 |
| 2.6 | official documentation | 7 |
| 3 | Development boards | 9 |
| 3.1 | Raspberry Pi 1 | 11 |
| 3.2 | PandaBoard | 11 |
| 3.3 | BeagleBone Black | 13 |
| 3.4 | Wandboard Quad | 13 |
| 3.5 | Grinn liteboard | 13 |
| 3.6 | x86_64 (Asus Eee PC 1215n) | 13 |
| 4 | Build systems | 15 |
| 4.1 | Problem of definition | 15 |
| 4.2 | Common aspects | 15 |
| 4.3 | Buildroot | 17 |
| 4.4 | LEDE | 17 |
| 4.5 | LTIB | 17 |
| 4.6 | Pengutronix | 17 |
| 4.7 | Yocto | 17 |
| 4.8 | LFS | 17 |
| 5 | Use cases and testing | 19 |
| 5.1 | Building default configuration | 19 |
| 5.1.1 | Deploy image | 20 |
| 5.1.2 | Buildroot | 20 |
| 5.1.3 | OpenWrt / LEDE | 20 |
| 5.1.4 | LTIB | 20 |
| 5.1.5 | PTXdist / DistroKit | 21 |
| 5.1.6 | Yocto / OpenEmbedded | 22 |
| 5.1.7 | Linux From Scratch | 22 |
| 5.2 | POSIX Test Suites | 22 |

| | | |
|----------|-------------------------------|-----------|
| 5.3 | Node.js and IoT | 22 |
| 5.3.1 | Buildroot | 22 |
| 5.4 | MIPI CSI and OpenCV | 23 |
| 5.5 | Real time / Xenomai | 23 |
| 6 | Methodology | 25 |
| 6.1 | Build servers | 25 |
| 6.2 | Data storage | 25 |
| 7 | Conclusions | 27 |
| | Bibliography | 27 |

Chapter 1

Introduction

1.1 Goals of the project

The goal of this project is to explore, compare and extend embedded Linux build systems.
[10]

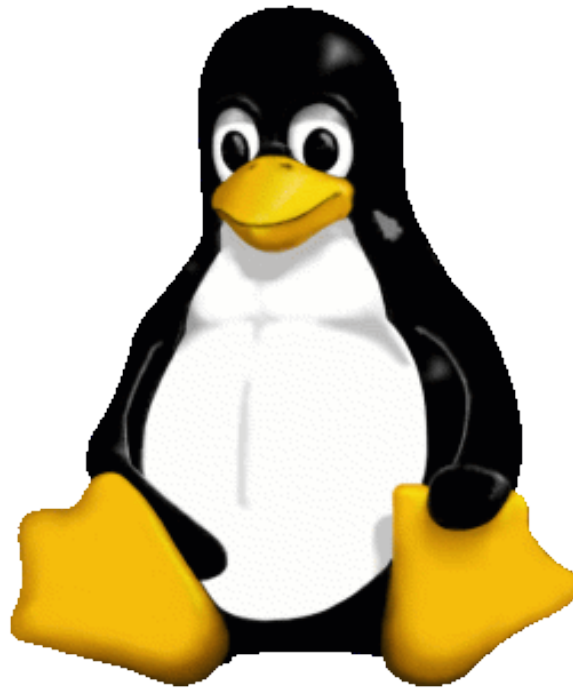


Figure 1.1: Penguin Tux - Linux mascot

Chapter 2

Literature review

There are two important issues, that I faced while making literature review: firstly what kind of materials should I analyze and secondly how to extend my searches, to get a full overview of current state of knowledge on this topic.

First issue was very clear in previous (20th) century - literature review was about analyzing two kind of printed materials: longer ones which were just specialist books and shorter ones which were peer reviewed articles in scientific or industry magazines. Right now they are also available and easy accessible in electronic form and still they are most reliable sources for literature review, but not every IT engineer is using it. Open Source Software and Open Source Hardware movement, beside creating lot of tools, also created a lot of written knowledge materials, but they are distributed in very different ways: project documentation, presentations, tutorials, source code comments, READMEs, How Tos, FAQs, wikis, finally blog and forum posts, but also in other forms. Hopefully, because they are all available via Internet, there exist one unified way to reference them: URL (Unified Resource Locator) also known as web address or link. Because of that, this and most of other bibliographies are filled with URLs, not ISBNs. I tried to divide all of those into groups as clearly as possible.

Second thing is how to be sure that I covered everything, at least by mentioning. Search results are personalized, both by searchers behavior who is choosing keywords and also Search Engine Optimization. Nobody could tell, that this is complete, but sources are cross referenced, so after some research, the loop closed. Most important factor for choosing materials is their universality and chance that they will be not so fast outdated.

2.1 elinux.org

The Embedded Linux Developer wiki elinux.org is undeniably the most extensive source of knowledge about all aspects of Embedded Linux. In this form it was possible to consolidate powerful community which extend its contests. From one side it is greatly filled with a lot of technical details and still extended, but from the other the content is not always universal and some pages are not maintained.

The elinux.org domain was registered on 1999-11-04 [1] and used by a Linux specialist Tim Riker just as a placeholder [2] [3]. About 2003 the Embedded Linux wiki was initiated here using MoinMoin framework [4]. In 2007 it was moved into Media Wiki engine, that is still in use. On the top level, its contents are divided into two main

parts: "Development Portals" about various aspects of embedded Linux and "Hardware Pages" for different development boards. The most relevant page for this thesis is http://elinux.org/Build_Systems, but build systems are only listed there without any comparison.

Currently it is maintained by the Core Embedded Linux Project which belongs to Linux foundation [5]. CELP is also coordinator of other important Embedded Linux activities.

2.2 wikipedia.org

Although Wikipedia allows anyone to edit articles it is the largest and most popular general reference work on the Internet. The most important thing from my point of view, is that it redirects and groups abstract entities on the high level. The table below summarizes pages that are most relevant for this work.

| | |
|---|--|
| https://en.wikipedia.org/wiki/Category:Embedded_Linux | |
| https://en.wikipedia.org/wiki/Category:Software_related_to_embedded_Linux | |
| https://en.wikipedia.org/wiki/Category:Embedded_Linux_distributions | |
| https://en.wikipedia.org/wiki/Linux_on_embedded_systems | |
| https://en.wikipedia.org/wiki/List_of_build_automation_software | |
| https://en.wikipedia.org/wiki/Cross_compiler | |
| https://en.wikipedia.org/wiki/Microprocessor_development_board | |
| https://en.wikipedia.org/wiki/Comparison_of_single-board_computers | |

During this project, I worked on extending enumerated categories and writing first version of article https://en.wikipedia.org/wiki/Embedded_Linux_build_systems.

2.3 Marcin Bis publications

In Poland, the most extensive source of written knowledge is provided by Mr Marcin Bis. He has published two books so far: "Linux in embedded systems" (pol. "Linux w systemach embedded") in 2011 and "Linux in i.MX 6 series systems" (pol. "Linux w systemach i.MX6 series") in 2015. The company BIS-LINUX.COM also offers various paid workshops and consultations.

2.4 Conferences and workshops

From one side there are public events, like Embedded Linux Conferences organized by CELP, from which slides and recordings are available on elinux.org:

- <http://elinux.org/Category:ELC> - Embedded Linux Conference (America)
- <http://elinux.org/Category:ELCE> - Embedded Linux Conference Europe

Embedded Linux issues connected to build systems are also present on Linux Sessions [6] coordinated by Academic IT Association from Wroclaw University of Technology.

From another side, private companies make mostly interactive workshops. The paid ones are organized i.e. by Marcin Bis [7]. There are also free of charge workshops i.e. ones organized by EBV in 2016, where I participated.

2.5 scholar.google.com

The phrase "embedded linux build systems" gives only 2 results [8]. First of them is presentation "Embedded Linux system development" by Thomas Petazzoni from year 2004. It describes only Open Embedded which is currently included as a part of Yocto Project. Second one is book "Mastering embedded Linux programming" by Chris Simmonds from year 2015. It differentiates only Buildroot and Yocto Project.

The phrase "embedded linux build system" gives 24 results [9]. Most of recent works enumerates Buildroot, Yocto Project, OpenWrt without comparison between them. Materials prior to 2010 are mostly using term "embedded linux distribution", rather than "embedded linux build system".

2.6 official documentation

The most valuable source of technical information is and always should be official documentation. URLs are situated in the bibliography and direct to the websites where more information about each project can be found.

Chapter 3

Development boards

From the beginning of embedded systems history, microprocessor development boards were just a training resource for engineers. Usually they were designed and produced by companies that have created certain chip and costs a lot of money, that it was not affordable for hobbyists or students. In recent years this state has been changed by a few factors:

- the spread of ARM architecture micro-controllers with efficiency comparable to personal computers (1GHz),
- creating development boards by community as open hardware, that resulted in cost reduction,
- adding peripherals specific not for embedded systems but for personal computers.

This led to discovering totally new use cases for development boards which explode due to widespread success of Raspberry Pi platform.

I have selected some of most popular devices with ARM to compare them, but also included my personal laptop computer as a reference for x86 architecture. Because of very low popularity, I haven't investigated other chips with SPARC or Power Architecture.

At minimum, each platform has:

- at least one UART, for serial console
- at least one Ethernet port, 100 MB/s or higher
- at least one USB port, 2.0 or higher
- SD card interface
- low voltage (5V-24V) power supply

| | | | | | | |
|----------------------------|---------------------------------------|------------------------------------|--|----------------------|--------------------|------------------------|
| name | Raspberry Pi 2 Model B | BeagleBone Black | PandaBoard | Wandboard Quad | Grinn liteboard | Asus Eee PC 1215n |
| release date | February 2015 | April 2013 | October 2010 | February 2013 | August 2016 | August 2010 |
| target price | \$35 | \$45 | \$174 | \$129 | \$120 | \$499 |
| word size | 32-bit | 32-bit | 32-bit | 32-bit | 32-bit | 32-bit/64-bit |
| SoC | Broadcom BCM2836 | Texas Instruments AM3358/9 | Texas Instruments OMAP4430 | Freescale i.MX6 Quad | i.MX 6Ultra Light | Intel Atom |
| CPU | ARM Cortex-A7 | ARM Cortex-A8 | ARM Cortex-A9 | ARM Cortex-A9 | ARM Cortex-A7 | x86 |
| frequency | 900 MHz | 1000 MHz | 1000 MHz | 1000 MHz | 528 MHz | 1800 MHz |
| RAM | 1 GB (GPU...) | 512 MiB DDR3 | 1 GB | 2GB DDR3 | 512MB DDR3 | 2GB DDR3 |
| Power source (consumption) | 5 V (Micro USB/GPIO) | Mini USB / 5 V jack | 5V | 5V | 5V/micro USB | 19V? tiny connector? |
| USB | 4 (via the on-board 5-port USB hub) | USB 2.0 | two USB host ports and one USB On-The-Go | USB 2.0 | USB 2.0 | USB 2.0 |
| Network | 10/100 Mbit/s Ethernet on the USB hub | Ethernet Fast Ethernet (MII based) | 10/100 Ethernet on USB hub | GbE | 10/100 half duplex | 10/100 Ethernet |
| storage | MicroSDHC slot | 4GB eMMC / MicroSDHC slot | SDHC slot | MicroSDHC | 2GB eMMC | SATA (i.e. 500 GB HDD) |

Tabela. 3.1: Development boards comparison

3.1 Raspberry Pi 1

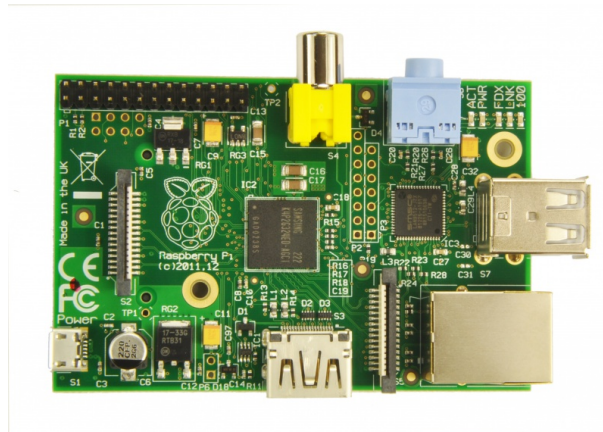


Figure 3.1: Raspberry Pi 1 Model B

3.2 PandaBoard

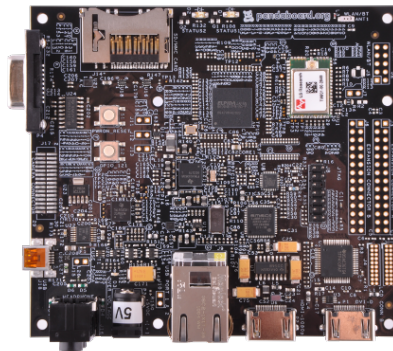


Figure 3.2: Pandaboard

drawbacks:

- poor support from community
- "big" SD card
- size: large surface + high audio and Ethernet/USB connectors
- Ethernet over USB
- serial db-9 connector
- strange USB type ab connector
- strange graphics solutions (two HDMI s)

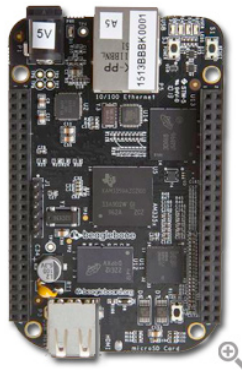


Figure 3.3: BeagleBone Black



Figure 3.4: Wandboard

3.3 BeagleBone Black

3.4 Wandboard Quad

3.5 Grinn liteboard

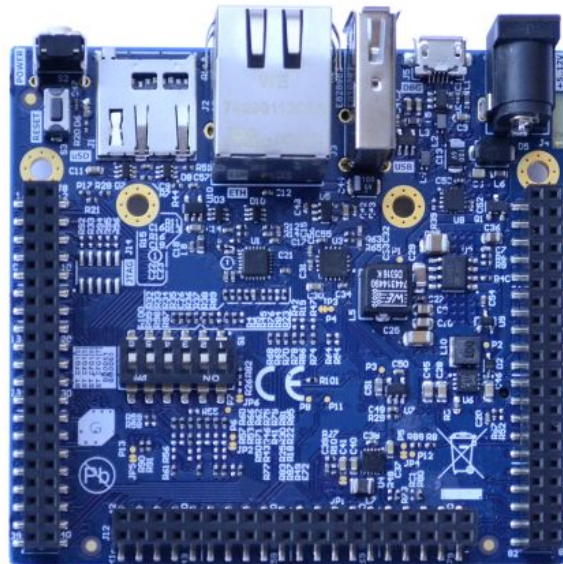


Figure 3.5: Liteboard

3.6 x86_64 (Asus Eee PC 1215n)



Figure 3.6: Asus Eee PC 1215n

Chapter 4

Build systems

4.1 Problem of definition

There is much confusion in semantic layer here. Linux is just a operating system kernel, it doesn't providing any user space. In the beginning, when it was created, there was no distributions, so if anybody want to utilize it, it had to be compiled and configured in a way later described in Linux From Scratch. After some conceptual projects, the two still most important distributions were created: Debian and Red Hat. They provides not only fully working solution, but also their own way to adding software - package management systems: deb and rpm.

It could be clearly said, that Linux distribution is a complete operating system, which includes certain set of software and distribution maintainers take care about preparing release version for certain hardware. Nevertheless, distributions are mostly used on personal computers, that have very standardized x86-based hardware, while embedded platforms differ very much between each other. The base idea behind distributions is that user is able to extend it and tune to own needs, while embedded systems mostly cope with non-human interfaces and need to be strongly pre-configured.

The lines are blurred in this area, but I will suggest using term "embedded Linux build systems" for every set of tools, that makes it possible to create complete operating system with Linux kernel on embedded device - starting with building from source for another CPU architecture and ending with complete disk image containing bootloader, kernel and root file system. Within this umbrella term some project identifies themselves differently as "distribution creators" or "executable documentation", but they share the same goal.

4.2 Common aspects

| name | Raspberry Pi 1 | BeagleBone Black | PandaBoard | Wandboard Quad | Grinn liteboard | Asus Eee PC 1215n |
|-------------|-----------------------|----------------------|----------------------|---------------------|---------------------------|-------------------------|
| Buildroot | raspberrypi_defconfig | beaglebone_defconfig | pandaboard_defconfig | wandboard_defconfig | grinn_liteboard_defconfig | pc_x86_64_efi_defconfig |
| LEDE | | | | | | |
| LTIB | | | | | | |
| Pengutronix | | | | | | |
| Yocto | | | | | | |

4.3 Buildroot

Buildroot is a simple and fast tool that enables to build a complete Linux embedded system by means of cross-compilation. During the building process a cross-compilation toolchain, a root filesystem, a Linux kernel and a boot loader can be generated. Furthermore, each of these steps can be performed separately.

4.4 LEDE

OpenWrt is a Linux distribution designed for embedded devices, commonly used on home routers. It gives users a possibility to make a fully customized operating system that is not dependent on firmware supplied by vendor. However, in May 2016 a group of developers decided to fork the project and create LEDE - Linux Embedded Development Environment.

4.5 LTIB

Linux Target Image Builder is a tool which allows to develop and deploy BSPs on target platforms. It was launched and financed by Freescale Semiconductor as and then transferred to Savannah. According to that, a Linux system can be built with a use of packages from the common pool.

4.6 Pengutronix

PTXdist is a build system that creates embedded Linux distributions directly from the source code. It is led by German company Pengutronix and supposed to maximize control and long-term maintainability. It starts with a cross toolchain and then next steps lead to packed binary image.

4.7 Yocto

Yocto is a set of tools and templates that enables to generate an operating system images for embedded Linux systems. The process is independent of the hardware architecture. Yocto includes core system components such as BitBake, OpenEmbedded-Core and Poky.

4.8 LFS

<http://www.linuxfromscratch.org>

<http://trac.clfs.org>

<http://intestate.com/pilfs/>

<https://raspberrypi.stackexchange.com/questions/25/is-there-a-linux-from-scratch-lfs->

Chapter 5

Use cases and testing

5.1 Building default configuration

OpenStack

1. Login to Horizon panel (<https://horizon.servery.pl>)
2. Select "instances" and "launch instance"
3. Choose instance name (mgr-builder-X), type (m1.2xlarge), image (Debian 8.4)
4. Choose or create key pair
5. Launch instance
6. Select "volumes" and "create volume" or... just select "Boot from image (creates a new volume)"
7. Choose volume name (mgr-volume-X), type (SAS), size (100 GB)
8. Create volume
9. Attach volume to instance

ssh

1. identify (`$HOST_KEYS $HOST_USER $HOST_IP $VOLUME_ID`)
2. Login with (`ssh -i $HOST_KEYS $HOST_USER@$HOST_IP`)
3. Update system (`sudo apt update && sudo apt upgrade`)
4. rest is possibly not needed... using home directory
5. Create partition on volume (`sudo mkfs.ext4 /dev/sdb`)
6. Mount volume (`sudo mount /dev/sdb /mnt/`)
7. (`sudo chown debian:debian /mnt/`)
8. Navigate to mounted directory (`cd /mnt/`)

5.1.1 Deploy image

- `scp`
- `pv sdcard.img | dd of=/dev/sdb bs=4M oflag=dsync`

5.1.2 Buildroot

Build procedure

Buildroot build procedure is the simplest from every point of view. The only difference between different boards is only `defconfig` name.

```
sudo apt install make gcc g++ libncurses-dev unzip git
wget https://buildroot.org/downloads/buildroot-2017.02.4.tar.gz
tar -zxf buildroot-2017.02.4.tar.gz
cd buildroot-2017.02.4/
make raspberrypi_defconfig
make
cp output/images/sdcard.img ..
```

5.1.3 OpenWrt / LEDE

```
sudo apt install make gcc g++ libncurses-dev unzip git
sudo apt install gawk zlib1g-dev
git clone https://git.lede-project.org/source.git lede
cd lede
git checkout v17.01.2
./scripts/feeds update -a
./scripts/feeds install -a
make defconfig
make menuconfig
Target System (Broadcom BCM27xx)
Target Profile (Raspberry Pi B/B+/CM/Zero/ZeroW)
make -j 12
cp build_dir/target-arm_arm1176jzf-s+vfp_musl-1.1.16_eabi/\
linux-brcm2708_bcm2708/tmp/\
lede-brcm2708-bcm2708-rpi-ext4-sdcard.img ..
```

5.1.4 LTIB

```
sudo apt install make gcc g++ libncurses-dev unzip
sudo apt install rpm bison tcl zlib1g-dev

wget https://github.com/downloads/midnightyell/RPi-LTIB/raspberrypi-tools-9
sudo mkdir -p /opt/ltib/pkgs/
sudo cp raspberrypi-tools-9c3d7b6-1.i386.rpm /opt/ltib/pkgs/
```



```

sudo dpkg --add-architecture i386
sudo apt update
sudo apt upgrade
sudo apt install zlib1g:i386
sudo apt install libstdc++6-4.8-dbg:i386

wget http://download.savannah.nongnu.org/releases/ltib/ltib-13-2-1-sv.tar.gz
tar -zxf ltib-13-2-1-sv.tar.gz
cd ltib-13-2-1-sv/
./ltib
Platform choice (Raspberry Pi with BCM2835 SoC)

https://github.com/raspberrypi/linux/commit/e6ef7f6bbc05fdcd7b1f115bb55a32c
rpm/BUILD/raspberrypi-linux-118e2d3/arch/arm/mach-bcm2708/bcm2708.c
rpm/BUILD/raspberrypi-linux-118e2d3/drivers/usb/host/dwc_otg/dwc_otg_driver

```

5.1.5 PTXdist / DistroKit

```

sudo apt install make gcc g++ libncurses-dev unzip git
sudo apt install gawk flex bison gettext python-dev lzop

wget http://public.pengutronix.de/software/ptxdist/ptxdist-2017.06.0.tar.bz2
tar -xjf ptxdist-2017.06.0.tar.bz2
cd ptxdist-2017.06.0
./configure
make
sudo make install
cd ..

# wget http://public.pengutronix.de/oselas/toolchain/SELAS.Toolchain-2016.06.1.tar.bz2
# tar -xjf SELAS.Toolchain-2016.06.1.tar.bz2
# cd SELAS.Toolchain-2016.06.1/
# ptxdist select ptxconfigs/arm-1136jfs-linux-gnueabihf_gcc-5.4.0_glibc-2.24
# sudo mkdir -p /opt/SELAS.Toolchain-2016.06.1/
# sudo chown debian:debian /opt/SELAS.Toolchain-2016.06.1/
# ptxdist go

wget http://debian.pengutronix.de/debian/pool/main/o/oselas.toolchain/oselas-toolchain-2016.06.1-arm-1136jfs-linux-gnueabihf-gcc-5.4.0-glibc-2.24.tar.gz
sudo dpkg -i oselas-toolchain-2016.06.1-arm-1136jfs-linux-gnueabihf-gcc-5.4.0-glibc-2.24.tar.gz

git clone https://git.pengutronix.de/cgit/DistroKit/
cd DistroKit/
ptxdist platform configs/platform-rpi/platformconfig
ptxdist images

```

5.1.6 Yocto / OpenEmbedded

```

sudo apt install make gcc g++ unzip git
sudo apt install gawk diffstat texinfo build-essential chrpath

mkdir raspberrypi-yocto
repo init -u ...
repo synch
source poky/oe-init-build-env
vi conf/bblayers.conf
export MACHINE=raspberrypi
bitbake core-image-minimal

```

5.1.7 Linux From Scratch

5.2 POSIX Test Suites

https://en.wikipedia.org/wiki/Linux_Standard_Base https://en.wikipedia.org/wiki/Single_UNIX_Specification <https://en.wikipedia.org/wiki/POSIX>

<https://www.opengroup.org/testing/downloads.html> <https://www.opengroup.org/testing/linux-test/> <https://wiki.linuxfoundation.org/en/Testing>

[posixtestsuite-1.5.2.tar.gz https://sourceforge.net/projects/posixtest/](https://sourceforge.net/projects/posixtest/) <http://posixtest.sourceforge.net> <https://sourceforge.net/p/posixtest/mailman/posixtest-discuss/>

<https://github.com/search?utf8=%E2%9C%93&q=posixtest> <https://github.com/juj/posixtestsuite>

5.3 Node.js and IoT

5.3.1 Buildroot

```

cd buildroot-2017.02.4/
make clean # because of wchar...
make menuconfig
Toolchain -> Enable WCHAR support
Toolchain -> Enable C++ support
Target packages
Interpreter languages and scripting -> nodejs -> NPM for the target
Networking applications -> openssh
Networking applications -> ntp -> ntpd
Libraries -> Database
mysql support -> mysql variant (mariadb) -> mariadb server

```

5.4 MIPI CSI and OpenCV

5.5 Real time / Xenomai

meta-eldk/linux-xenomai <https://layers.openembedded.org/layerindex/recipe/8333/>

Chapter 6

Methodology

6.1 Build servers

minimal requirements vs maximum usable resources

1. local machine (laptop)
 2. cloud server (payed m1.2 x large - count time on various sizes + cost + servery.pl vs Amazon AWS)
 3. dedicated server (own... Intel Xeon)
 4. dedicated server (payed)
- count costs in comparison to... big mac / salary?

6.2 Data storage

WARNING: DEGRADATION OF SD CARDS! (ram file system... and other mechanisms)

Chapter 7

Conclusions

Bibliography

- [1] <https://who.is/whois/elixir.org>.
- [2] <http://rikers.org>.
- [3] <https://web.archive.org/web/20010721180347/elixir.org>.
- [4] <https://web.archive.org/web/20030220110526/http://www.elixir.org:80/wiki/>.
- [5] <https://www.linuxfoundation.org/projects/core-embedded-linux>.
- [6] sesja.linuxsowa.pl.
- [7] <http://www.bis-linux.com/szkolenia>.
- [8] <https://scholar.google.pl/scholar?q=embedded+linux+build+systems>. [Online; access 2017-06-01].
- [9] <https://scholar.google.pl/scholar?q=embedded+linux+build+system>. [Online; access 2017-06-01].
- [10] M. Bis. Linux w systemach embeded. <http://bis.org.pl/en:start>, 2013. [Online; access 2017-01-01].

List of Figures

| | | |
|-----|--------------------------------------|----|
| 1.1 | Penguin Tux - Linux mascot | 3 |
| 3.1 | Raspberry Pi 1 Model B | 11 |
| 3.2 | Pandaboard | 11 |
| 3.3 | BeagleBone Black | 12 |
| 3.4 | Wandboard | 12 |
| 3.5 | Liteboard | 13 |
| 3.6 | Asus Eee PC 1215n | 13 |

List of Tables

| | | |
|-----|---|----|
| 3.1 | Development boards comparison | 10 |
|-----|---|----|