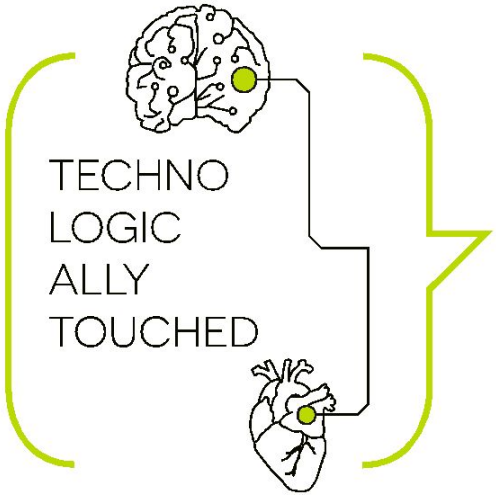


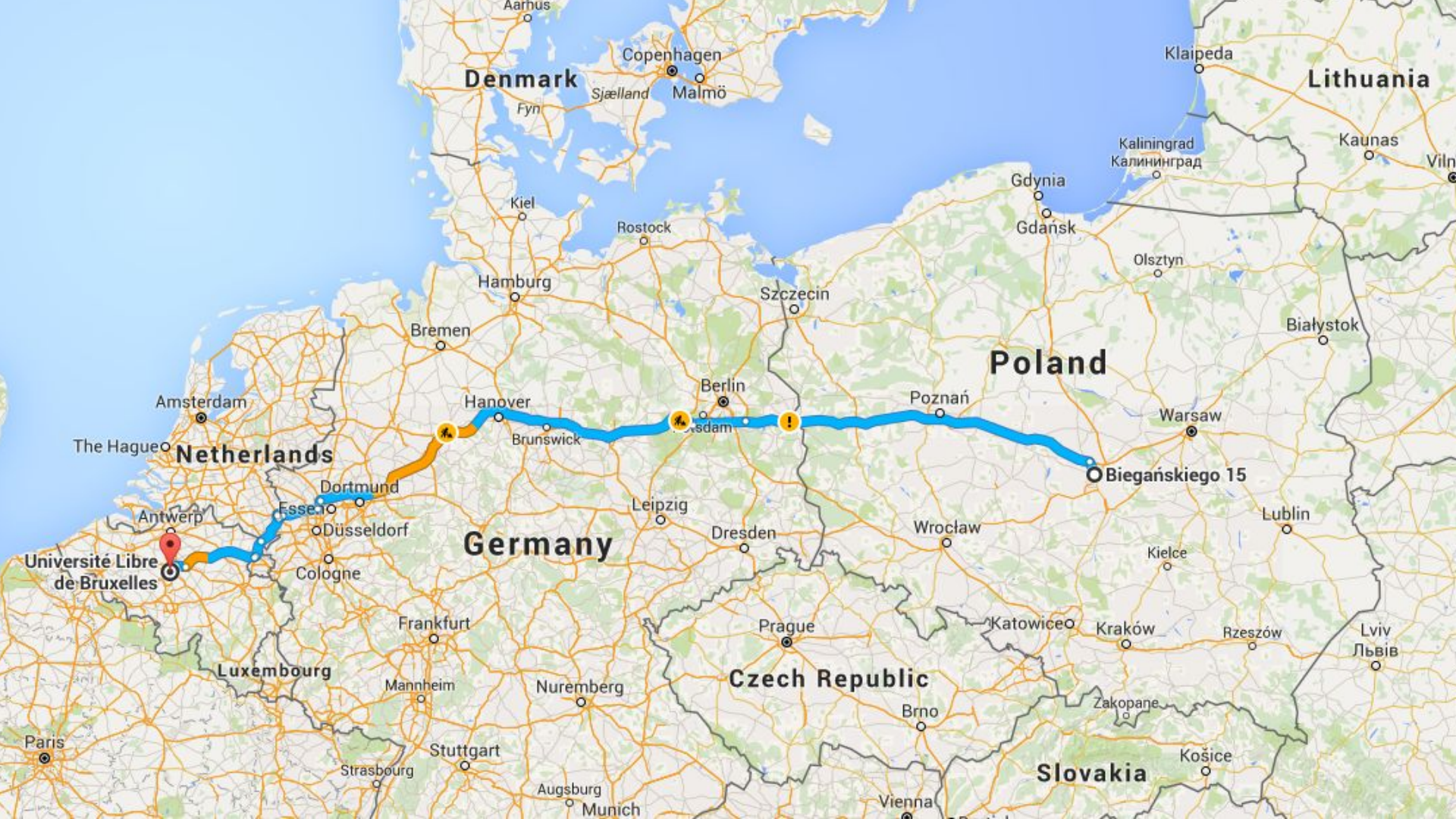
Yocto and IoT

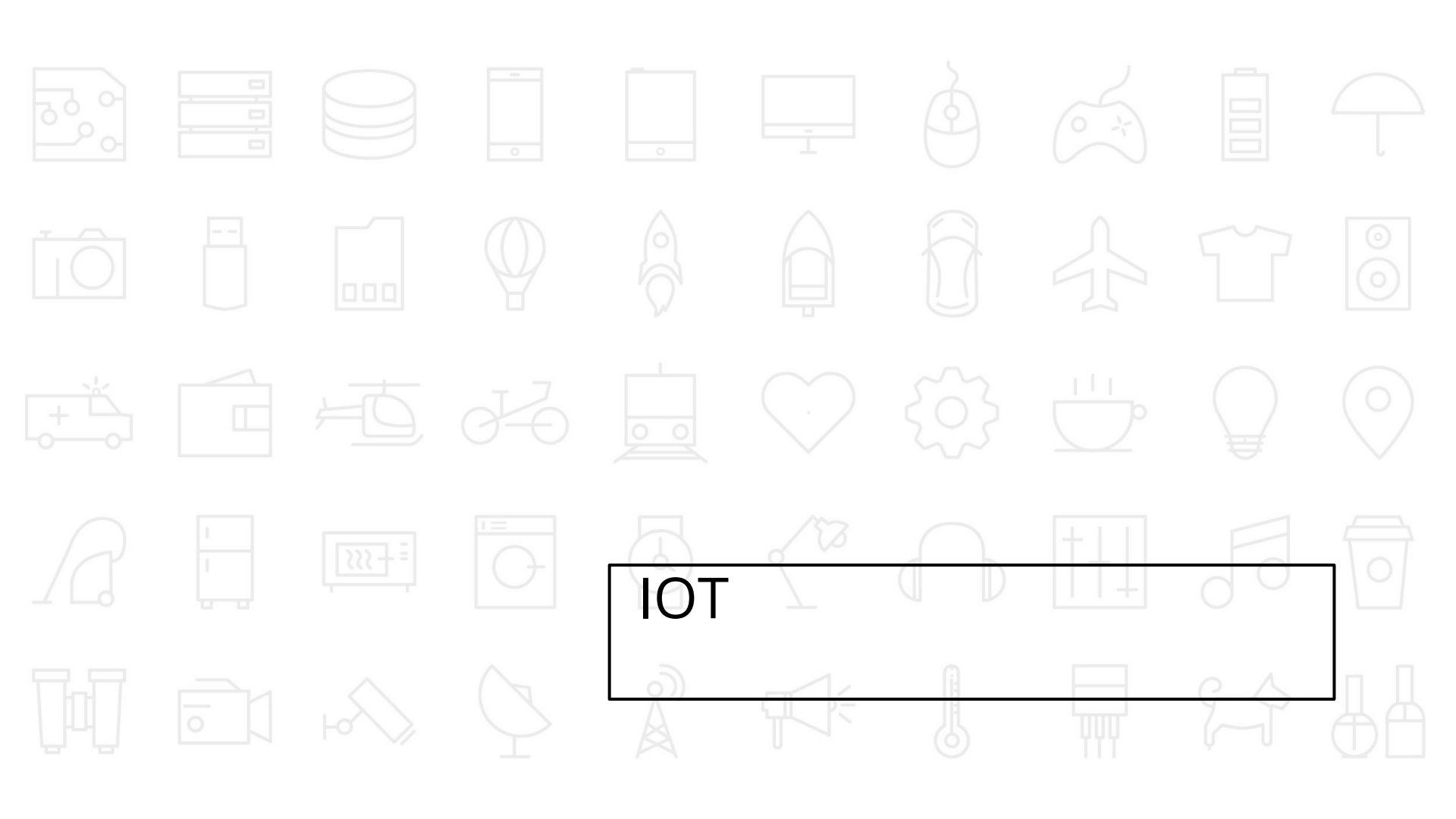
A retrospective

Maciej Borzęcki

FOSDEM 2016

- Working with embedded systems since 2006
- Since 2013 at Open-RnD Sp. z o.o.
- Software Architect


Aarhus
Copenhagen
Denmark
Sjælland
Malmö
Fyn
Klaipeda
Lithuania
Kaliningrad
Калининград
Kaunas
Gdynia
Gdańsk
Kiel
Rostock
Olsztyn
Hamburg
Szczecin
Białystok
Bremen
Poland
Berlin
Amsterdam
Hanover
Poznań
Warsaw
Brunswick
The Hague
Netherlands
Biegańskiego 15
Dortmund
Leipzig
Antwerp
Dresden
Wrocław
Lublin
Düsseldorf
Germany
Université Libre de Bruxelles
Cologne
Kielce
Frankfurt
Prague
Katowice
Kraków
Rzeszów
Lviv
Львів
Luxembourg
Mannheim
Nuremberg
Czech Republic
Zakopane
Brno
Paris
Stuttgart
Košice
Strasbourg
Slovakia
Augsburg
Vienna
Munich


IOT

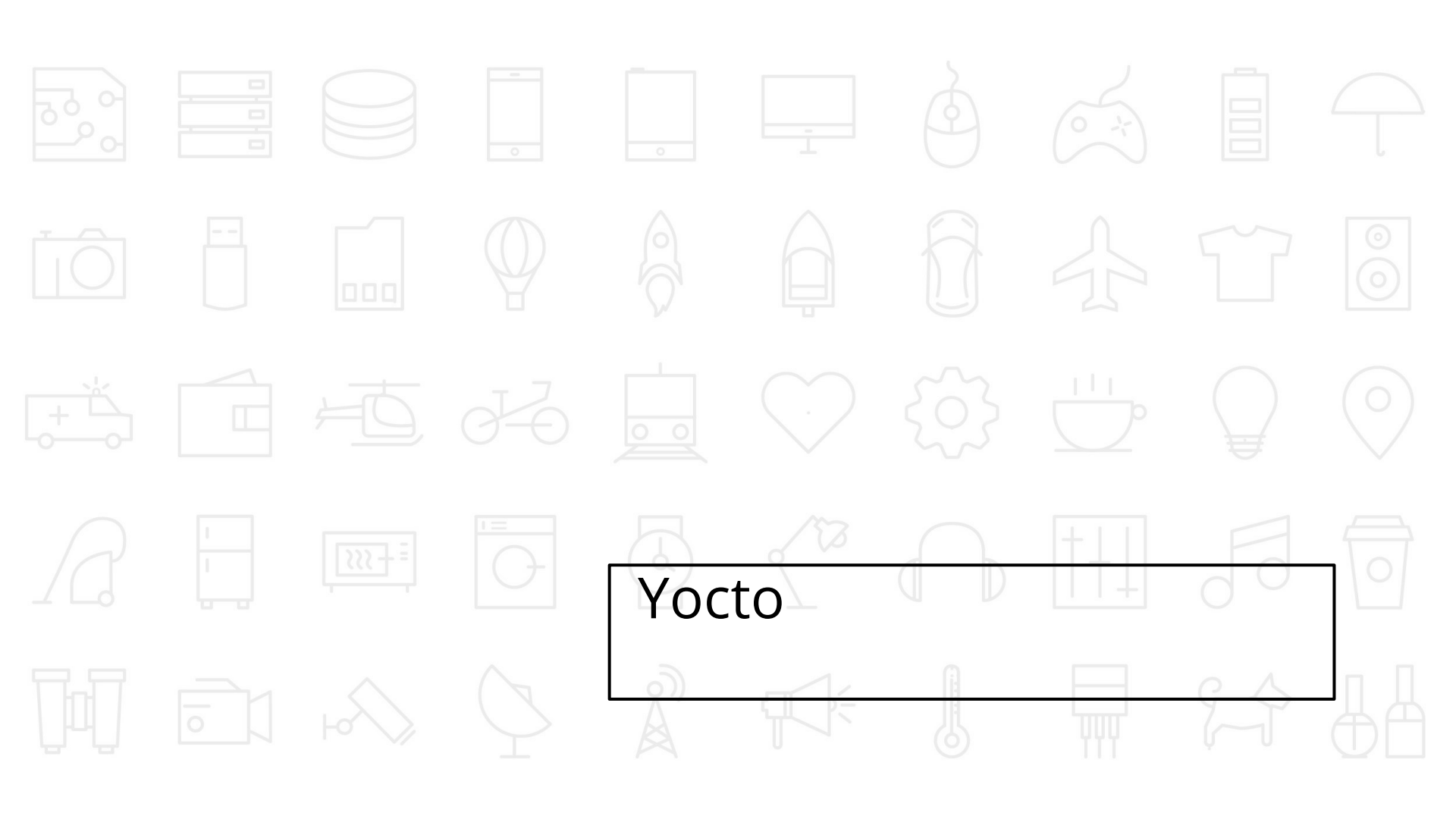
The **Internet of Things (IoT)** is the network of physical objects, devices, vehicles, buildings and other items which are embedded with electronics, software, sensors, and network connectivity, which enables these objects to collect and exchange data.

(Wikipedia)

The **Internet of Things (IoT)** is the network of physical objects, devices, vehicles, buildings and other items which are embedded with **electronics**, **software**, **sensors**, and **network connectivity**, which enables these objects to collect and **exchange data**.

(Wikipedia)

Gartner's 2015 Hype Cycle for Emerging Technologies Identifies the Computing Innovations That Organizations Should Monitor - https://www.gartner.com/newsroom/id/3114217


Yocto

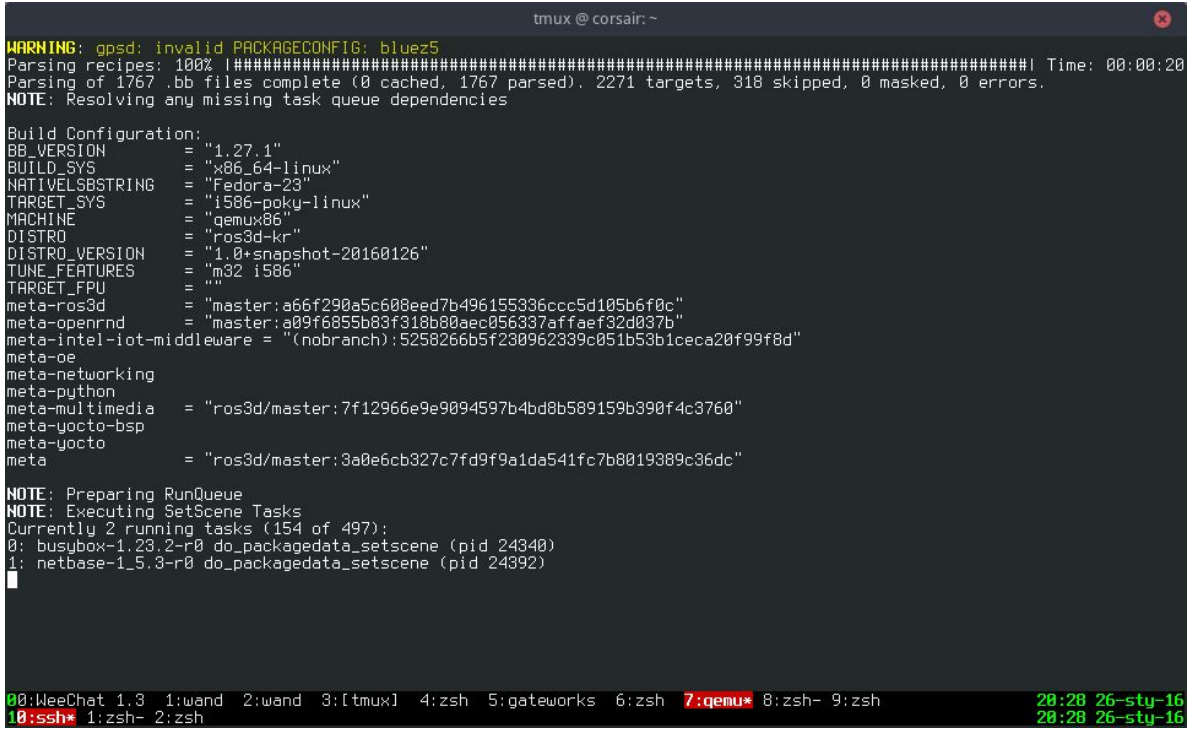
Taxonomy

- Yocto Project
  - Umbrella project
- Poky
  - Reference distribution
- Bitbake
  - Build tool
- OpenEmbedded
  - Build system
  - OE-core
  - meta-oe

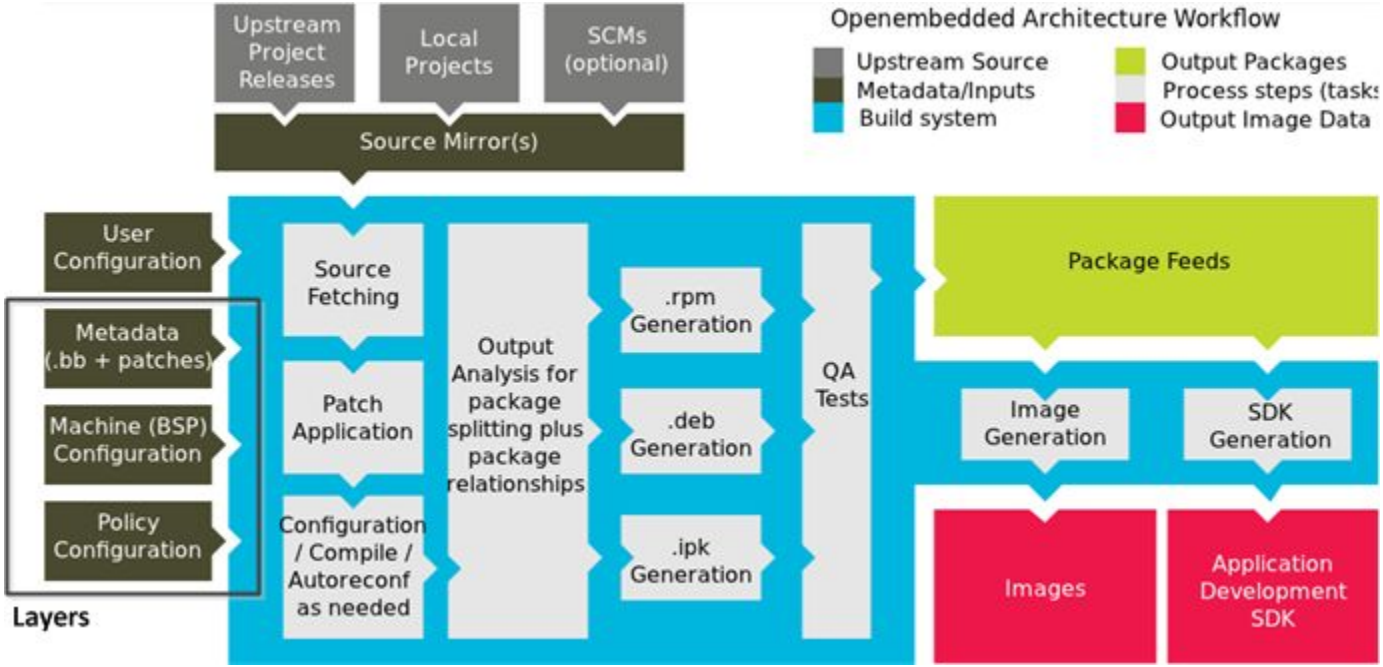
Yocto 101

1. Grab code (git://git.yoctoproject.org/poky)
2. Setup environment:
   ```
   $ source poky/oe-init-build-env qemu-x86
   ```
3. Build a reference image:
   ```
   $ bitbake core-image-minimal
   ```
4. Wait...
5. Start QEMU with your image:
   ```
   $ runqemu qemux86 core-image-minimal nographic
   ```

Yocto 101 condt.



OpenEmbedded Architecture



Yocto Project Reference Manual - https://www.yoctoproject.org/docs/2.1/ref-manual/ref-manual.html

Most important pieces

- Machine
  - Usually the target platform, examples: raspberrypi, beaglebone, genericx86-64
  - Platform description - CPU, flash, boot devices, serial console
  - Influences how packages are built and which ones may get installed
- Distro
  - Policy and features at the distribution level, examples: poky, poky-tiny
  - Influences how packages are built and which ones get installed
  - X11 vs. wayland, init system, alsa & PulseAudio, OpenGL ..
- Image
  - Predefined images: core-image-minimal
  - System image features, influences which packages get installed
  - debug-tweaks, dropbear vs. openssh

Most important pieces

```
conf/local.conf:
  DISTRO = “poky”
  MACHINE = “beaglebone”
```

- Machine
  - Usually the target platform, examples: raspberrypi, beaglebone, genericx86-64
  - Platform description - CPU, flash, boot devices, serial console
  - Influences how packages are built and which ones may get installed
- Distro
  - Policy and features at the distribution level, examples: poky, poky-tiny
  - Influences how packages are built and which ones get installed
  - X11 vs. wayland, init system, alsa & PulseAudio, OpenGL ..
- Image
  - Predefined images: core-image-minimal
  - System image features, influences which packages
  - debug-tweaks, dropbear vs. openssh

```
$ bitbake core-image-minimal
```

Code organization

- layers
  - recipes
    - poky/meta/recipes-core/busybox/busybox_1.24.1.bb
    - poky/meta/recipes-devtools/python/python_2.7.9.bb
  - machines
    - poky/meta-yocto-bsp/conf/machine/beaglebone.conf
  - distributions
    - poky/meta-yocto/conf/distro/poky.conf
- upon layers (mix and match layers)
  - new recipes & extend existing ones
    - meta-virtualization/recipes-core/busybox/busybox_%.bbappend - extends busybox
  - even more machines
    - meta-ti/conf/machine/beaglebone.conf

Code organization

- layers
  - recipes
    - poky/meta/recipes-core/busybox/busybox_1.24.1.bb
    - poky/meta/recipes-devtools/python/python_2.7.9.bb
  - machines
    - poky/meta-yocto-bsp/conf/machine/beaglebone.conf
  - distributions
    - poky/meta-yocto/conf/distro/poky.conf
- upon layers (mix and match layers)
  - new recipes & extend existing ones
    - meta-virtualization/recipes-core/busybox/busybox_%.bbappend - extends busybox
  - even more machines
    - meta-ti/conf/machine/beaglebone.conf

```
$ bitbake-layers show-layers
$ bitbake-layers show-recipes
```

Code organization

- layers
  - recipes
    - **poky/meta/recipes-core/busybox/busybox_1.24.1.bb**
    - poky/meta/recipes-devtools/python/python_2.7.9.bb
  - machines
    - poky/meta-yocto-bsp/conf/machine/beaglebone.conf
  - distributions
    - poky/meta-yocto/conf/distro/poky.conf
- upon layers (mix and match layers)
  - new recipes & extend existing ones
    - **meta-virtualization/recipes-core/busybox/busybox_%.bbappend** - extends busybox
  - even more machines
    - meta-ti/conf/machine/beaglebone.conf

```
$ bitbake-layers show-appends
```

Code organization

- layers
  - recipes
    - poky/meta/recipes-core/busybox/busybox_1.24.1.bb
    - poky/meta/recipes-devtools/python/python_2.7.9.bb
  - machines
    - **poky/meta-yocto-bsp/conf/machine/beaglebone.conf**
  - distributions
    - poky/meta-yocto/conf/distro/poky.conf
- upon layers (mix and match layers)
  - new recipes & extend existing ones
    - meta-virtualization/recipes-core/busybox/busybox_%.bbappend - extends busybox
  - even more machines
    - **meta-ti/conf/machine/beaglebone.conf**

Directory structure

```
poky
├── bitbake
├── documentation
├── meta
├── meta-selftest
├── meta-skeleton
├── meta-yocto
├── meta-yocto-bsp
└── scripts
meta-ti
├── conf
├── licenses
├── recipes-bsp
├── recipes-connectivity
├── recipes-core
├── recipes-devtools
├── recipes-graphics
├── recipes-kernel
├── recipes-ti
└── scripts
```

```
meta-openembedded
├── contrib
├── meta-efl
├── meta-filesystems
├── meta-gnome
├── meta-gpe
├── meta-initramfs
├── meta-multimedia
├── meta-networking
├── meta-oe
├── meta-perl
├── meta-python
├── meta-ruby
├── meta-systemd
├── meta-webserver
└── meta-xfce
```

Recipe

```
SUMMARY = "Simple helloworld application"
SECTION = "examples"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"

SRC_URI = "file://helloworld.c"

S = "${WORKDIR}"

do_compile() {
     ${CC} helloworld.c -o helloworld
}

do_install() {
     install -d ${D}${bindir}
     install -m 0755 helloworld ${D}${bindir}
}
```

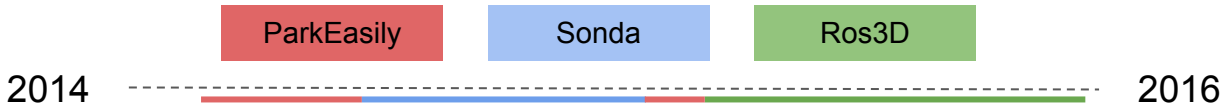
Yocto at Open-RnD

Background

- Previous experience with cross-compilation build systems
  - Buildroot ~2007
  - Custom solutions after that
- Maintenance headache
  - Ill-configured vendor supplied SDKs
- QA & tracking
- Reinventing the wheel
  - Is there any actual value in building from scratch?
- Gave Yocto a try
  - ~80 commits
  - 12 projects

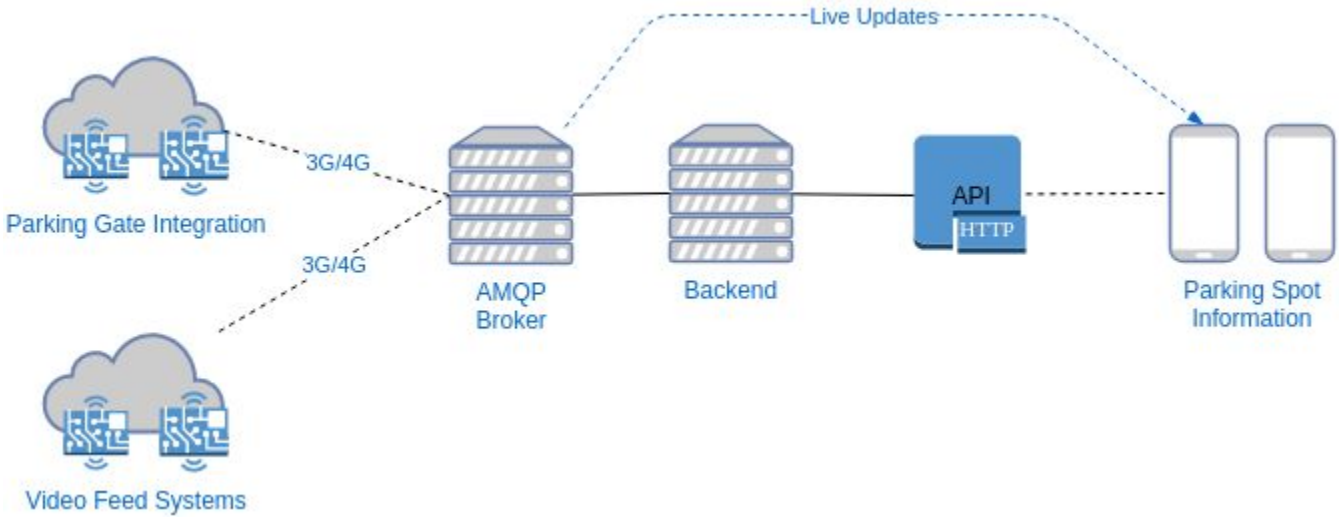
Projects

- ParkEasily
  - Distributed monitoring of parking spots
- Sonda
  - Athletic workout gadget
- Ros3D
  - Remote control of a 3D camera rig
- Timeline

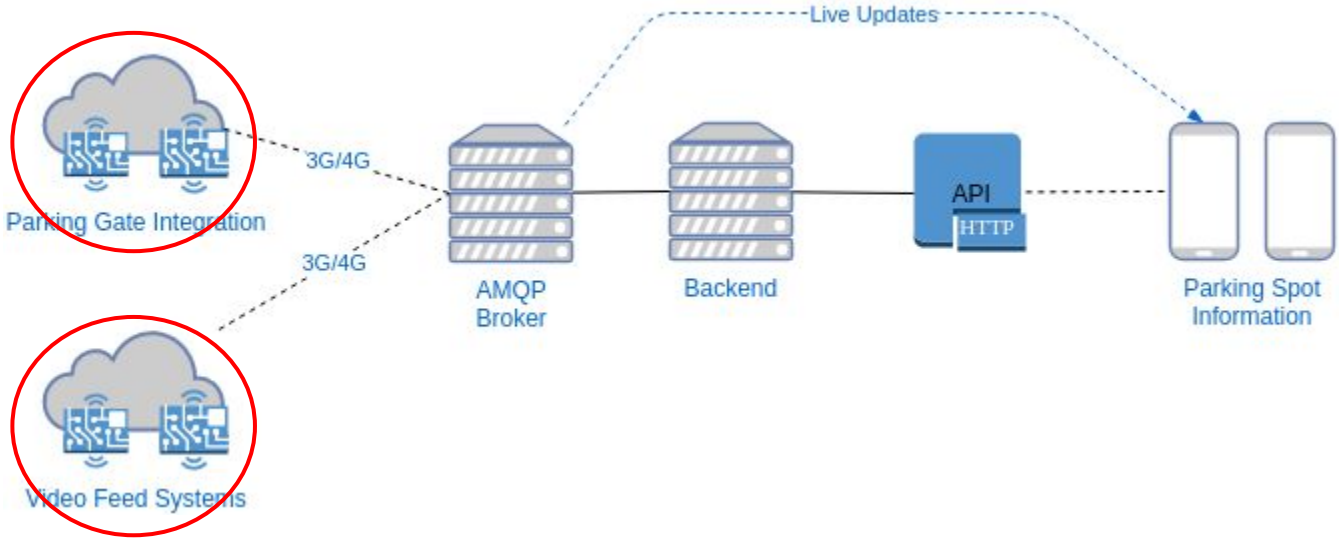


ParkEasily

System Architecture



System Architecture



Prototype devices

Baby steps

- Started with a point release
  - marked as stable release - must be good?
- Single layer with our packages - meta-openrnd
  - 4 core recipes, 7 packages, 2 package groups, 2 platform glue packages, 3 images
- Platforms - BeagleBone Black, Raspberry PI (early prototyping)
- Couple of bbappends
- First upstream contributions - wic fixes, rabbitmq-c
- Custom distro - openrnd-poky-systemd
- Used layers: meta-{ti, raspberrypi, oe, networking, python}

Baby steps - what went bad?

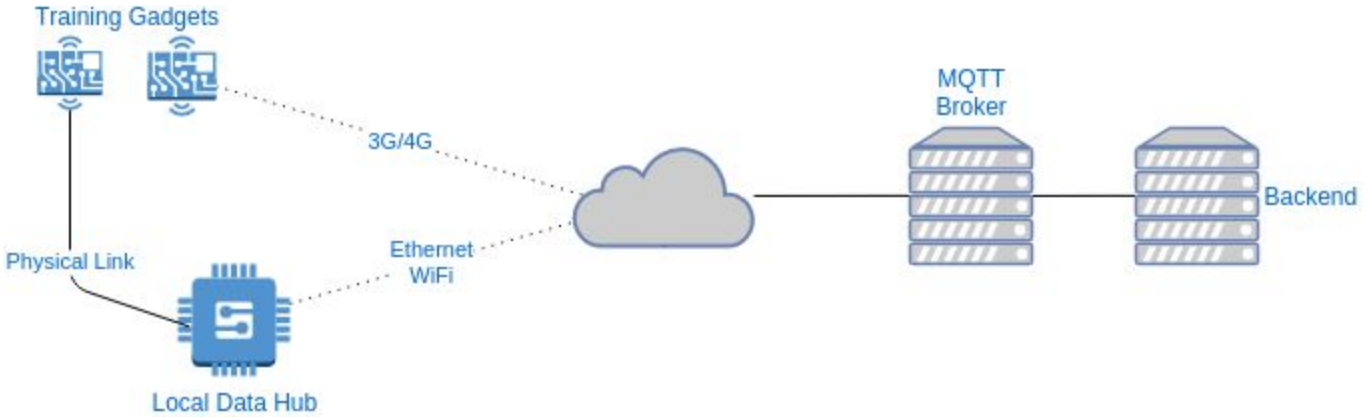
- Tracking a point release
  - Harder to contribute fixes
  - More work backporting desired patches
- Single layer with our packages
  - Reuse meta-openrnd for other projects
  - Recipes changed by bbappends must exist
- Inflexible

Baby steps -solutions

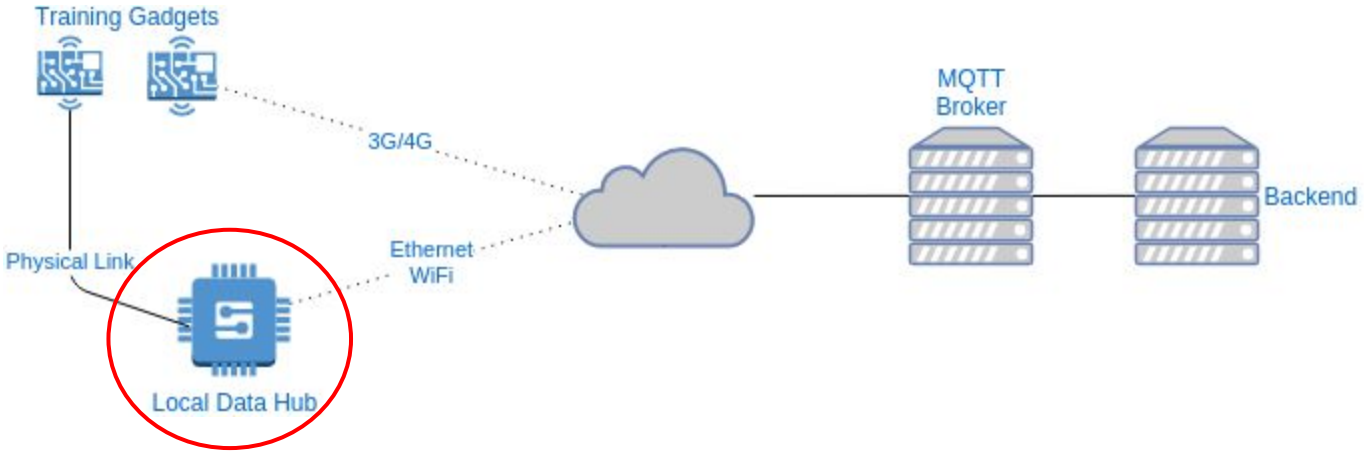
- Branch off master with a very simple policy
  - Preserve upstream branch names - easier tracking and syncing
  - Simple naming scheme **openrnd/master** based on master
  - Project branches **parkeasily/master**
  - Periodic merge
- Layer split
  - meta-openrnd & meta-parkeasily

Sonda

System Architecture

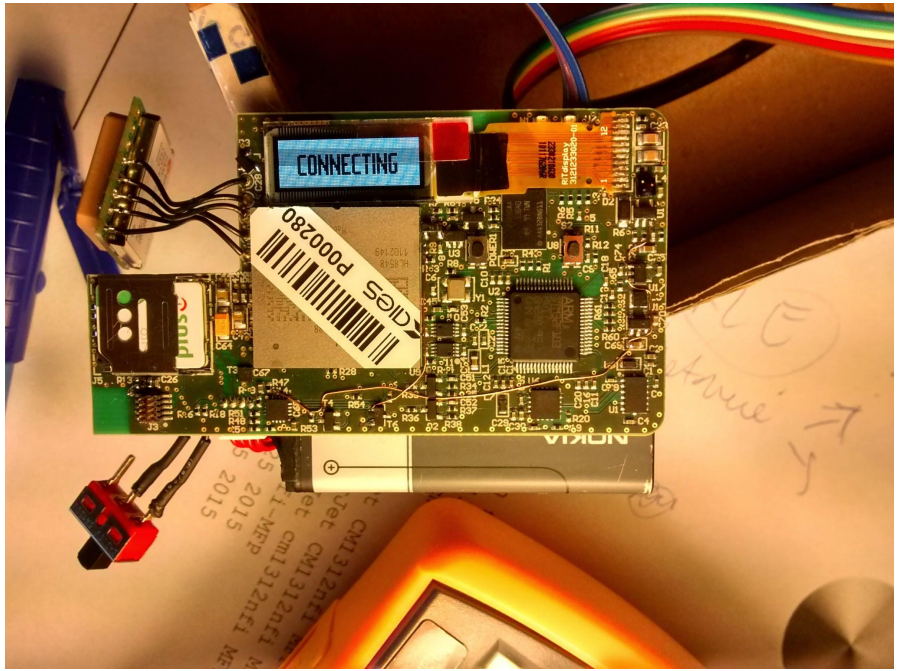


System Architecture



Workout gadget

- Accelerometers
- Gyroscopes
- Heartbeat
- Position (10Hz)
- On board storage
- 3G connectivity
- STM32F1x



Data hub

- Storage
- Recharge station
- Firmware update

Approach

- Project branching sonda/master
- Project layer
  - 3 core recipes, 3 packages, some platform glue
- Single platform only - BeagleBone Black
- Very few *.bbappends
- More wic fixes
  - MBR & multiple partitions
- Used layers: meta-{ti, oe}

Layer organization

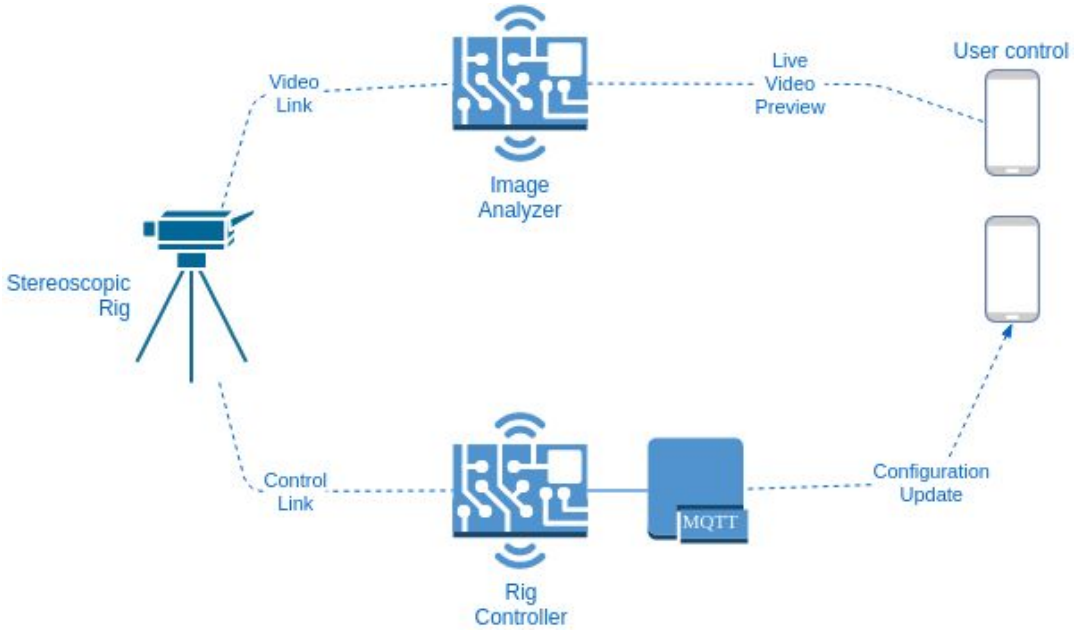
- meta-openrnd
  - Reuse some common recipes - ex. systemd-networkd default config
- meta-sonda
  - Custom distribution sonda
  - Override some default settings
    - Use systemd as init
    - Ship systemd-networkd

System firmware image - wic

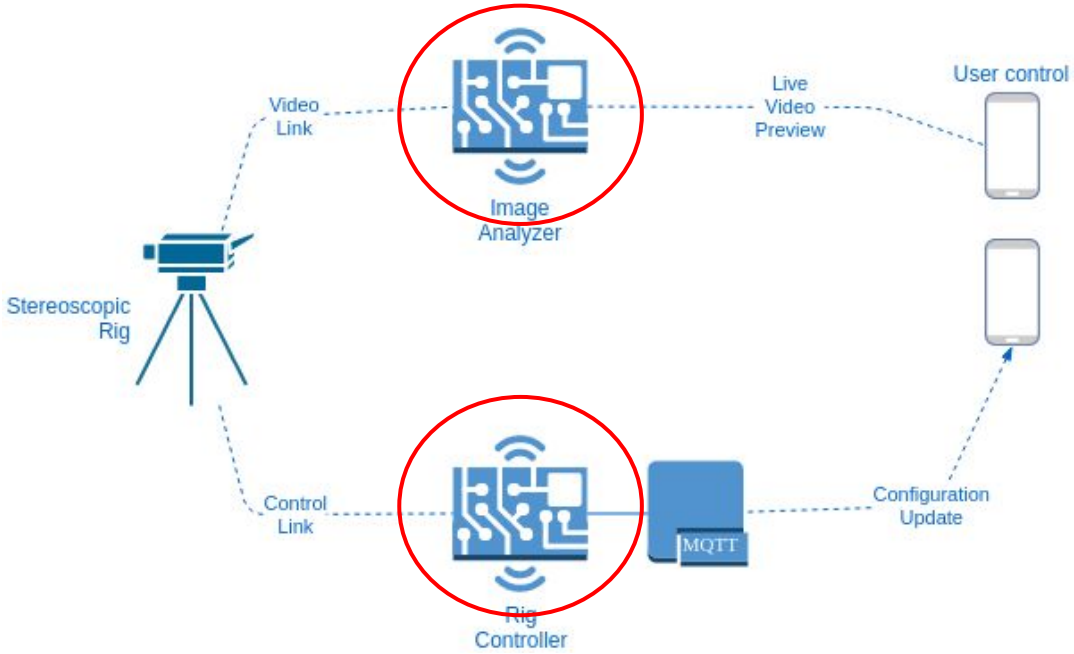
```
# short-description: Create SD card image for Sonda case appliance
# long-description: Creates a partitioned SD card image. Boot files
# are located in the first vfat partition. Usable in Sond case
# appliance.

part /boot --source bootimg-partition --ondisk mmcblk0 --fstype=vfat --label boot --active --
align 4 --size 16
part / --source rootfs --ondisk mmcblk0 --fstype=ext4 --label root --align 4
part /var/lib/sonda --ondisk mmcblk0 --fstype=ext4 --label sondadata --align 307200 --size 1024
```

ROS 3D

System Architecture



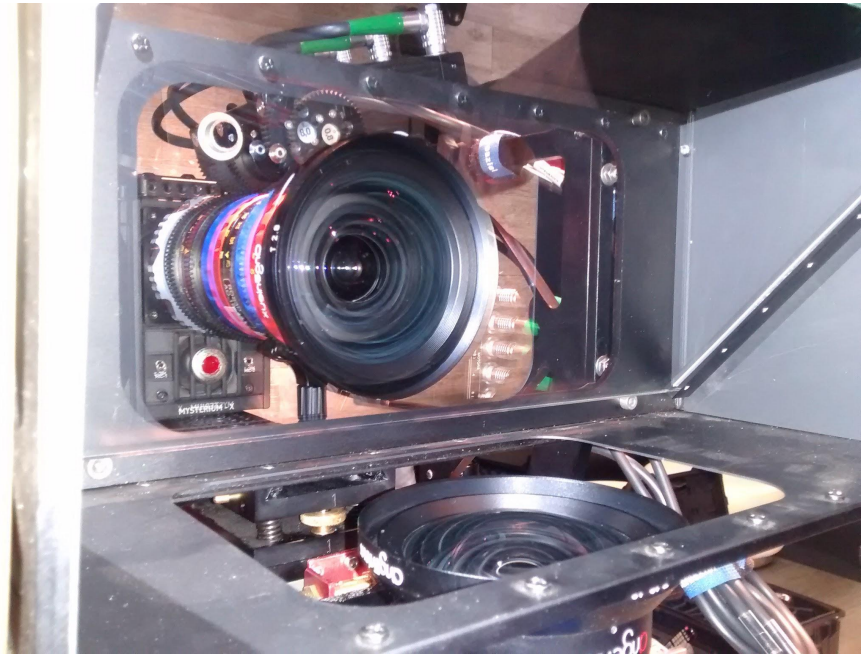
System Architecture



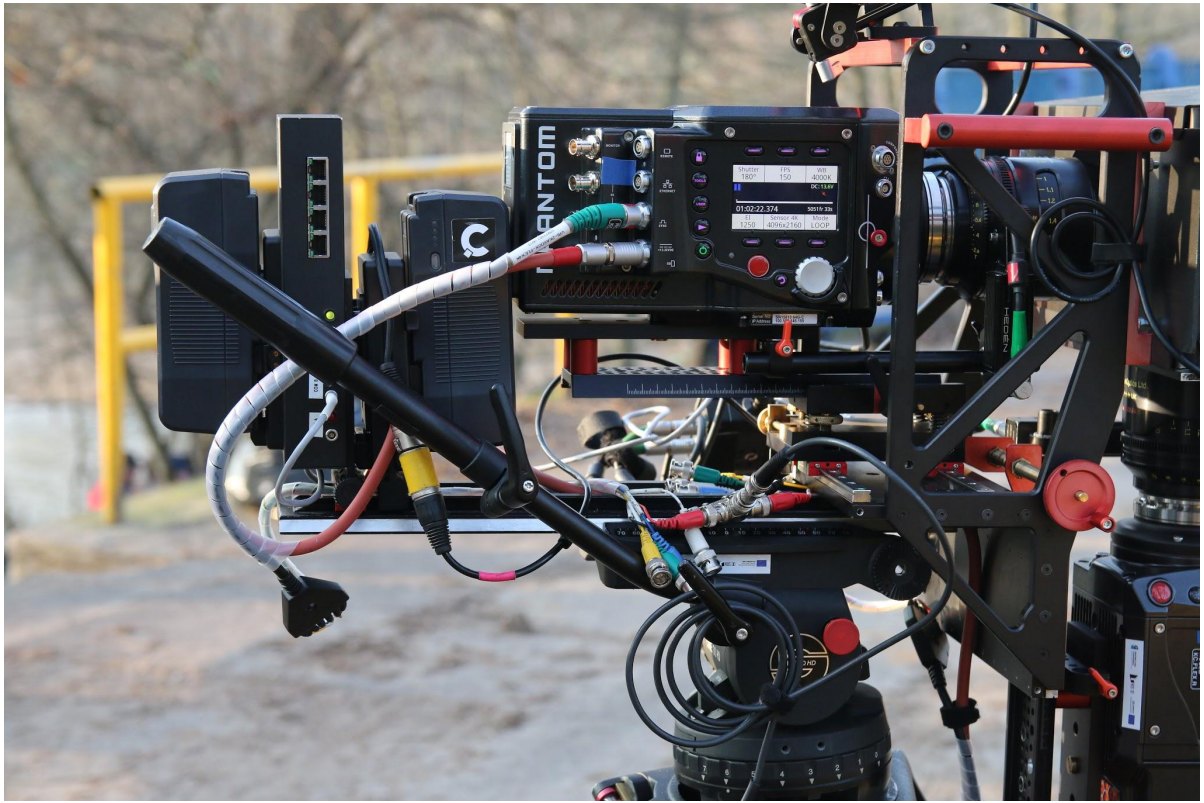
On the set


PRACA JEST MOŻLIWA DO WYKONANIA
ŚLI PODZIELIĆ JĄ NA MAŁE ODCINKI

Stereoscopic Camera Rig



Rig Controller



Rig Controller - Mobile App

Rig Controller

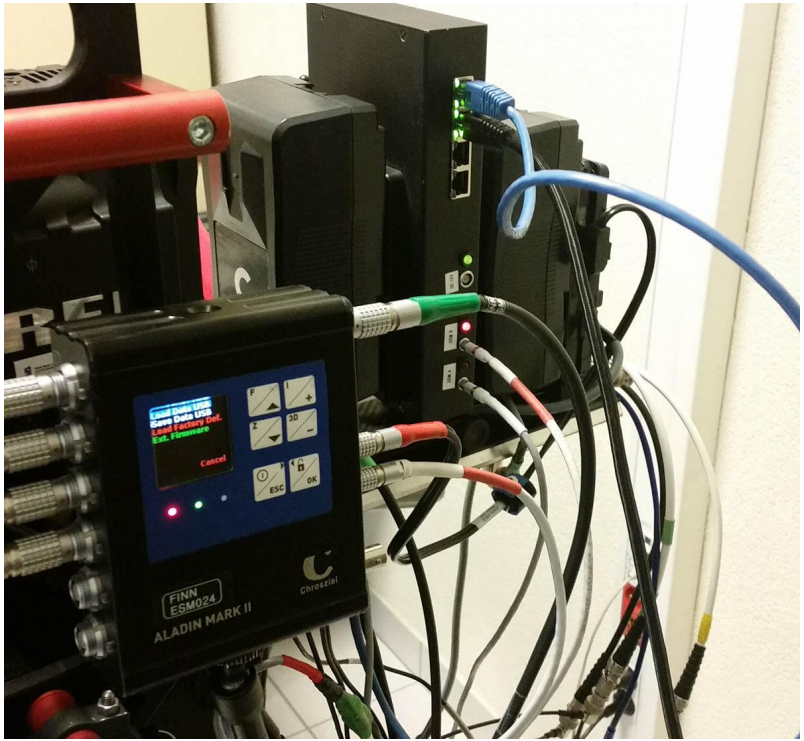


Image Analyzer

Source Code

- Most of source code is open source (MIT)
- Repositories
  - https://github.com/open-rnd/ros3d-stream
  - https://github.com/open-rnd/ros3d-dev-controller
  - https://github.com/open-rnd/ros3d-upnp
  - https://github.com/open-rnd/ros3d-platform-controller
  - https://github.com/open-rnd/ros3d-www

Approach

- Project branching **ros3d/master**
- Reused meta-openrnd
- Project layer
  - 11 core recipes, 4 images, 4 additional recipes
- Number of platforms
  - Wandboad Quad (meta-fsl-arm-extra) , GW5400 (meta-gateworks), QEMU
- Different target devices
  - KR - Rig Controller
  - AO - Image Analyzer
- SDK

Layers

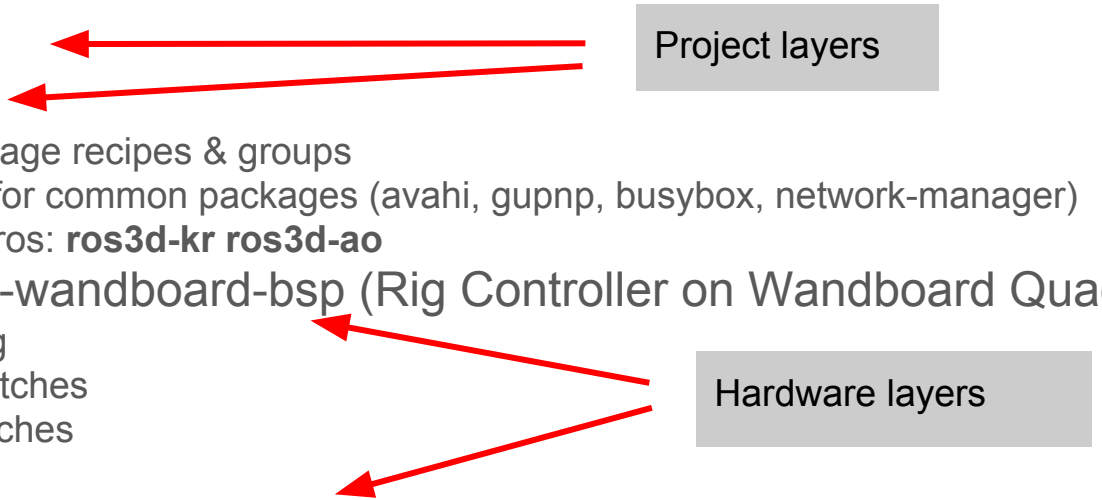
- meta-openrnd
- meta-ros3d
  - Ros3D package recipes & groups
  - bbappends for common packages (avahi, gupnp, busybox, network-manager)
  - Custom distros: **ros3d-kr ros3d-ao**
- meta-ros3d-kr-wandboard-bsp (Rig Controller on Wandboard Quad)
  - kernel config
  - LEDs dts patches
  - keys dts patches
  - udev rules
- meta-ros3d-ao-gateworks-bsp (Image Analyzer on GW5400)
  - video pipeline stream config for ros3d-stream
  - patches for gstreamer1.0-plugins-imx
  - udev rules

Layers

- meta-openrnd
- meta-ros3d
  - Ros3D package recipes & groups
  - bbappends for common packages (avahi, gupnp, busybox, network-manager)
  - Custom distros: **ros3d-kr ros3d-ao**
- meta-ros3d-kr-wandboard-bsp (Rig Controller on Wandboard Quad)
  - kernel config
  - LEDs dts patches
  - keys dts patches
  - udev rules
- meta-ros3d-ao-gateworks-bsp (Image Analyzer on GW5400)
  - video pipeline stream config for ros3d-stream
  - patches for gstreamer1.0-plugins-imx
  - udev rules

Layers

- meta-openrnd
- meta-ros3d
  - Ros3D package recipes & groups
  - bbappends for common packages (avahi, gupnp, busybox, network-manager)
  - Custom distros: **ros3d-kr ros3d-ao**
- meta-ros3d-kr-wandboard-bsp (Rig Controller on Wandboard Quad)
  - kernel config
  - LEDs dts patches
  - keys dts patches
  - udev rules
- meta-ros3d-ao-gateworks-bsp (Image Analyzer on GW5400)
  - video pipeline stream config for ros3d-stream
  - patches for gstreamer1.0-plugins-imx
  - udev rules



Distribution

meta-ros3d/conf/distro/include/ros3d-common.inc:

```
TARGET_VENDOR = "-poky"
MAINTAINER = "Ros3D Project <ros3d@open-rnd.pl>"

LAYER_CONF_VERSION ?= "6"

# set preferred versions
PREFERRED_PROVIDER_udev ?= "systemd"
PREFERRED_PROVIDER_udev-utils ?= "systemd"
VIRTUAL-RUNTIME_init_manager = "systemd"
VIRTUAL-RUNTIME_initscripts = ""

DISTRO_FEATURES_append = " systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
# default to systemd-networkd & systemd-resolved
for network
PACKAGECONFIG_append_pn-systemd = " networkd
resolved"

# use ipk
PACKAGE_CLASSES = "package_ipk"
```

meta-ros3d/conf/distro/ros3d-kr.conf:

```
DISTRO = "ros3d-kr"
DISTRO_NAME = "Ros3D KR Baseline"
DISTRO_CODENAME ="ros3d-kr"
DISTRO_VERSION = "1.0+snapshot-${DATE}"

require conf/distro/include/ros3d-common.inc

DISTRO_FEATURES_append = " ros3d-kr"

DISTRO_FEATURES_remove = "x11 "
```

Distribution

meta-ros3d/conf/distro/include/ros3d-common.inc:

```
TARGET_VENDOR = "-poky"
MAINTAINER = "Ros3D Project <ros3d@open-rnd.pl>"

LAYER_CONF_VERSION ?= "6"

# set preferred versions
PREFERRED_PROVIDER_udev ?= "systemd"
PREFERRED_PROVIDER_udev-utils ?= "systemd"
VIRTUAL-RUNTIME_init_manager = "systemd"
VIRTUAL-RUNTIME_initscripts = ""

DISTRO_FEATURES_append = " systemd"
DISTRO_FEATURES_BACKFILL_CONSIDERED = "sysvinit"
# default to systemd-networkd & systemd-resolved
for network
PACKAGECONFIG_append_pn-systemd = " networkd
resolved"

# use ipk
PACKAGE_CLASSES = "package_ipk"
```

meta-ros3d/conf/distro/ros3d-kr.conf:

```
DISTRO = "ros3d-kr"
DISTRO_NAME = "Ros3D KR Baseline"
DISTRO_CODENAME ="ros3d-kr"
DISTRO_VERSION = "1.0+snapshot-${DATE}"

require conf/distro/include/ros3d-common.inc

DISTRO_FEATURES_append = " ros3d-kr"

DISTRO_FEATURES_remove = "x11"
```

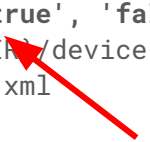
If needed test for distro features when building common package

Distribution - in recipe

```
do_install_append() {
    ...

    # replace device.xml with target specific one
    rm -f ${D}${datadir}/ros3d-upnp/device.xml

    if ${@bb.utils.contains('DISTRO_FEATURES', 'ros3d-kr', 'true', 'false', d)}; then
        install -m 0644 -t ${D}${datadir}/ros3d-upnp ${WORKDIR}/device-kr.xml
        ln -s device-kr.xml ${D}${datadir}/ros3d-upnp/device.xml
    fi

    ...
}
```

Distribution - in recipe

```
do_install_append() {
    ...

    # replace device.xml with target specific one
    rm -f ${D}${datadir}/ros3d-upnp/device.xml

    if ${@bb.utils.contains('DISTRO_FEATURES', 'ros3d-kr', 'true', 'false', d)}; then
        install -m 0644 -t ${D}${datadir}/ros3d-upnp ${WORKDIR}/device-kr.xml
        ln -s device-kr.xml ${D}${datadir}/ros3d-upnp/device.xml
    fi

    ...
}
```



Inline Python

Layers - Rig Controller

```
mborzecki@comp_016_pc_buildenv:ros3d/ros3d-head/wandboard bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta-ros3d-kr-wandboard-bsp  /home/mborzecki/yocto/ros3d/ros3d-head/meta-ros3d-kr-wandboard-bsp  6
meta-ros3d            /home/mborzecki/yocto/ros3d/ros3d-head/meta-ros3d  7
meta-openrnd          /home/mborzecki/yocto/ros3d/ros3d-head/meta-openrnd  6
meta-intel-iot-middleware  /home/mborzecki/yocto/ros3d/ros3d-head/meta-intel-iot-middleware  8
meta-oe               /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-oe  6
meta-gnome            /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-gnome  7
meta-networking       /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-networking  5
meta-python           /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-python  7
meta-multimedia       /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-multimedia  6
meta-fsl-arm-extra    /home/mborzecki/yocto/ros3d/ros3d-head/meta-fsl-arm-extra  4
meta-fsl-arm          /home/mborzecki/yocto/ros3d/ros3d-head/meta-fsl-arm  5
meta-yocto-bsp        /home/mborzecki/yocto/ros3d/ros3d-head/poky/meta-yocto-bsp  5
meta-yocto            /home/mborzecki/yocto/ros3d/ros3d-head/poky/meta-yocto  5
meta                  /home/mborzecki/yocto/ros3d/ros3d-head/poky/meta  5
workspace             /home/mborzecki/yocto/ros3d/ros3d-head/wandboard/workspace  99
```

Custom machine - Rig Controller

```
#@TYPE: Machine
#@NAME: Ros3D Rig Controller based on Wandboard i.MX6 Wandboard Quad
#@SOC: i.MX6Q
#@DESCRIPTION: Machine configuration for Ros3D Rig Controller based on  i.MX6 Wandboard Quad
#@MAINTAINER: Maciej Borzecki <maciej.borzecki@open-rnd.pl>

require conf/machine/wandboard-quad.conf

MACHINE_EXTRA_RRECOMMENDS += " \
              bcm4329-nvram-config \
              kr-leds \
              kr-wandboard-keys \
              kernel-devicetree \
"
```

Custom machine - Rig Controller

```
#@TYPE: Machine
#@NAME: Ros3D Rig Controller based on Wandboard i.MX6 Wandboard Quad
#@SOC: i.MX6Q
#@DESCRIPTION: Machine configuration for Ros3D Rig Controller based on  i.MX6 Wandboard Quad
#@MAINTAINER: Maciej Borzecki <maciej.borzecki@open-rnd.pl>

require conf/machine/wandboard-quad.conf

MACHINE_EXTRA_RRECOMMENDS += " \
              bcm4329-nvram-config \
              kr-leds \
              kr-wandboard-keys \
              kernel-devicetree \
"
```

Base on Wandboard Quad config

Custom machine - Rig Controller

```
#@TYPE: Machine
#@NAME: Ros3D Rig Controller based on Wandboard i.MX6 Wandboard Quad
#@SOC: i.MX6Q
#@DESCRIPTION: Machine configuration for Ros3D Rig Controller based on  i.MX6 Wandboard Quad
#@MAINTAINER: Maciej Borzecki <maciej.borzecki@open-rnd.pl>

require conf/machine/wandboard-quad.conf

MACHINE_EXTRA_RRECOMMENDS += " \
              bcm4329-nvram-config \
              kr-leds \
              kr-wandboard-keys \
              kernel-devicetree \
"
```

Base on Wandboard Quad config

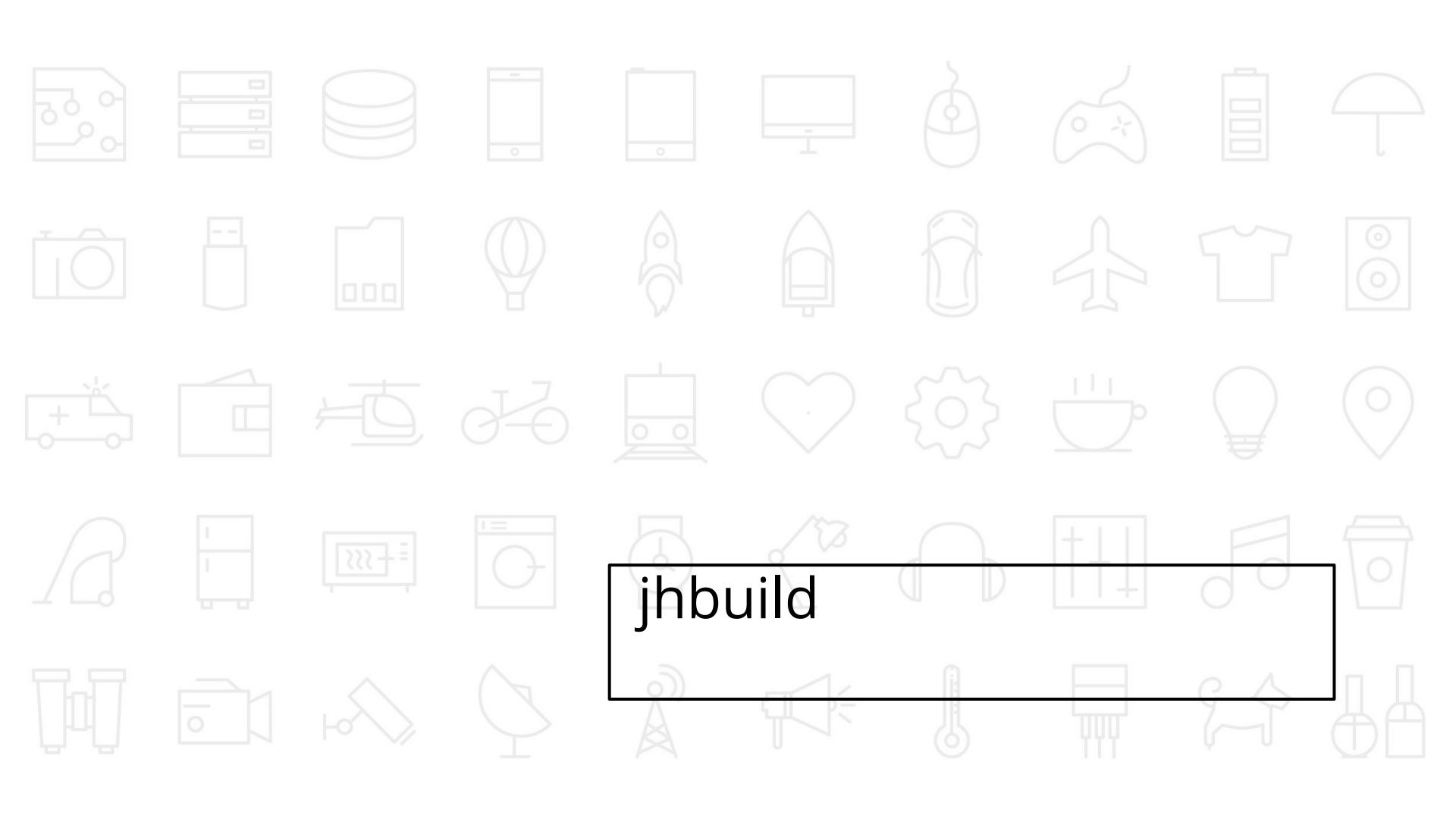
udev glue

Layers - Image Analyzer

```
mborzecki@comp_016_pc_buildenv:ros3d/ros3d-head/gateworks bitbake-layers show-layers
layer                 path                                      priority
==========================================================================
meta-ros3d-ao-gateworks-bsp  /home/mborzecki/yocto/ros3d/ros3d-head/meta-ros3d-ao-gateworks-bsp  6
meta-ros3d            /home/mborzecki/yocto/ros3d/ros3d-head/meta-ros3d  7
meta-openrnd          /home/mborzecki/yocto/ros3d/ros3d-head/meta-openrnd  6
meta-intel-iot-middleware  /home/mborzecki/yocto/ros3d/ros3d-head/meta-intel-iot-middleware  8
meta-oe               /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-oe  6
meta-networking       /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-networking  5
meta-python           /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-python  7
meta-multimedia       /home/mborzecki/yocto/ros3d/ros3d-head/meta-openembedded/meta-multimedia  6
meta-gateworks        /home/mborzecki/yocto/ros3d/ros3d-head/meta-gateworks  6
meta-fsl-arm-extra    /home/mborzecki/yocto/ros3d/ros3d-head/meta-fsl-arm-extra  4
meta-fsl-arm          /home/mborzecki/yocto/ros3d/ros3d-head/meta-fsl-arm  5
meta-yocto-bsp        /home/mborzecki/yocto/ros3d/ros3d-head/poky/meta-yocto-bsp  5
meta-yocto            /home/mborzecki/yocto/ros3d/ros3d-head/poky/meta-yocto  5
meta                  /home/mborzecki/yocto/ros3d/ros3d-head/poky/meta  5
workspace             /home/mborzecki/yocto/ros3d/ros3d-head/gateworks/workspace  99
```

How did it work out?

- Flexible
- Easily exchange platforms
  - Hassle free builds for QEMU and even for genericx86-64
- Clear separation between project and hardware
- Software configuration via files extremely convenient
  - ros3d-stream pipeline different between platforms
- Packager friendly software


jhbuild

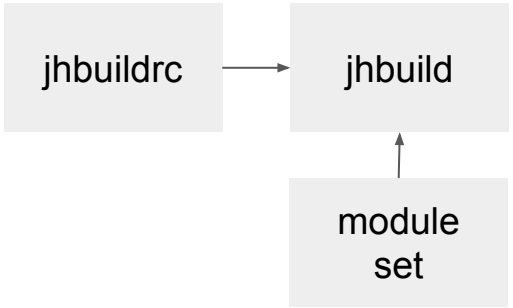
jhbuild

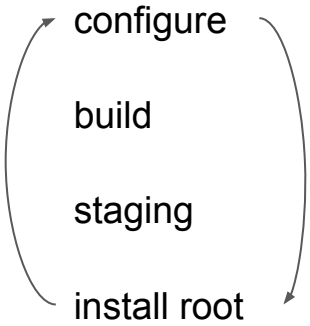
- Looking for a decent chroot-like development tools, alternatives?
  - Bunch of custom scripts
    - PKG_CONFIG_PATH, PATH, ACLOCAL_PATH, GI_TYPELIB_PATH, LD_PRELOAD, LD_LIBRARY_PATH, PYTHONPATH, PERL5LIB, GST_PLUGIN_PATH, LDFLAGS, CFLAGS, CXXFLAGS
  - autotools? cmake? setuptools?
  - Out of the source builds
  - Confined chroot
  - Code checkout

jhbuild

- Looking for a decent chroot-like development tools, alternatives?
  - Bunch of custom scripts
    - PKG_CONFIG_PATH, PATH, ACLOCAL_PATH, GI_TYPELIB_PATH, LD_PRELOAD, LD_LIBRARY_PATH, PYTHONPATH, PERL5LIB, GST_PLUGIN_PATH, LDFLAGS, CFLAGS, CXXFLAGS
  - autotools? cmake? setuptools?
  - Out of the source builds
  - Confined chroot
  - Code checkout
- Builds GNOME, Xorg, Mesa 3D
- https://github.com/bboozzoo/jhbuild-helper

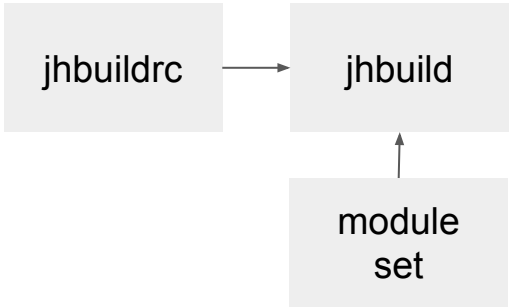
jhbuild

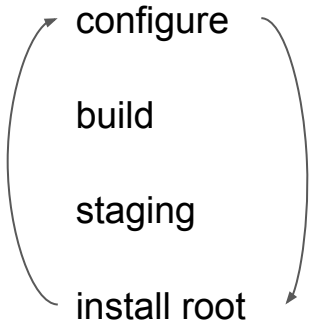





```
$ tree install-root build-root/stream
checkout/stream
install-root
├── bin
│   └── ros3d-stream
├── etc
│   └── ros3d-stream
│       └── ros3d-stream.conf
└── _jhbuild
    ├── info
    │   └── ros3d-stream
    └── manifests
        └── ros3d-stream
build-root/stream
├── config.log
├── config.status
├── libtool
├── Makefile
├── ros3d-stream
└── src
    ├── ros3d_stream-config.o
    ├── ros3d_stream-httpapi.o
    ├── ros3d_stream-main.o
    ├── ros3d_stream-stream-client.o
    ├── ros3d_stream-stream-manager.o
    ├── ros3d_stream-stream.o
    └── ros3d_stream-zeroconf.o
```

jhbuild






```
$ tree install-root build-root/stream
checkout/stream
install-root
├── bin
│   └── ros3d-stream
├── etc
│   └── ros3d-stream
│       └── ros3d-stream.conf
└── _jhbuild
    ├── info
    │   └── ros3d-stream
    └── manifests
        └── ros3d-stream
build-root/stream
├── config.log
├── config.status
├── libtool
├── Makefile
├── ros3d-stream
└── src
    ├── ros3d_stream-config.o
    ├── ros3d_stream-httpapi.o
    ├── ros3d_stream-main.o
    ├── ros3d_stream-stream-client.o
    ├── ros3d_stream-stream-manager.o
    ├── ros3d_stream-stream.o
    └── ros3d_stream-zeroconf.o
```

All under a custom prefix

jhbuild - module set

```
<?xml version="1.0"?><!--*- mode: nxml -*-->
<!DOCTYPE moduleset SYSTEM "moduleset.dtd">
<?xml-stylesheet type="text/xsl" href="moduleset.xsl"?>
<moduleset>
     <!-- repositories -->
     <!-- modules -->
</moduleset>
```

jhbuild - repositories

```
<?xml version="1.0"?><!--*- mode: nxml -*-->
<!DOCTYPE moduleset SYSTEM "moduleset.dtd">
<?xml-stylesheet type="text/xsl" href="moduleset.xsl"?>
<moduleset>
     <!-- repositories -->
     <repository type="system" name="system" />
     <repository type="git" href="ssh://git.open-rnd.net:29418/"
               name="git-rnd" default="yes"/>
     <!-- modules -->
</moduleset>
```

jhbuild - modules

```
<?xml version="1.0"?><!--*- mode: nxml -*-->
<!DOCTYPE moduleset SYSTEM "moduleset.dtd">
<?xml-stylesheet type="text/xsl" href="moduleset.xsl"?>
<moduleset>
     <!-- repositories -->
     <repository type="system" name="system" />
     <repository type="git" href="ssh://git.open-rnd.net:29418/"
               name="git-rnd" default="yes"/>
     <!-- modules -->
     <systemmodule id="glib-2.0">
          <pkg-config>glib-2.0.pc</pkg-config>
          <branch repo="system" version="2.32"/>
     </systemmodule>

     <systemmodule id="gstreamer-1.0">
          <pkg-config>gstreamer-1.0.pc</pkg-config>
          <branch repo="system"/>
     </systemmodule>
</moduleset>
```

jhbuild - modules

```
...
     <autotools id="ros3d-stream"
                autogen-sh="autoreconf">
          <branch module="open-rnd.ros3d.stream"
                  checkoutdir="stream"/>
          <dependencies>
               <dep package="gstreamer-1.0" />
               <dep package="libsoup-2.4" />
               <dep package="glib-2.0" />
               <dep package="gio-2.0" />
          </dependencies>
     </autotools>
...
```

jhbuild - modules

```
...
     <autotools id="ros3d-stream"
                autogen-sh="autoreconf">
          <branch module="open-rnd.ros3d.stream"
                  checkoutdir="stream"/>
          <dependencies>
               <dep package="gstreamer-1.0" />
               <dep package="libsoup-2.4" />
               <dep package="glib-2.0" />
               <dep package="gio-2.0" />
          </dependencies>
     </autotools>
...
```

```
$ jh sysdeps --install ros3d-stream
$ jh buildone ros3d-stream
$ jh build ros3d-stream
```

jhbuild - modules

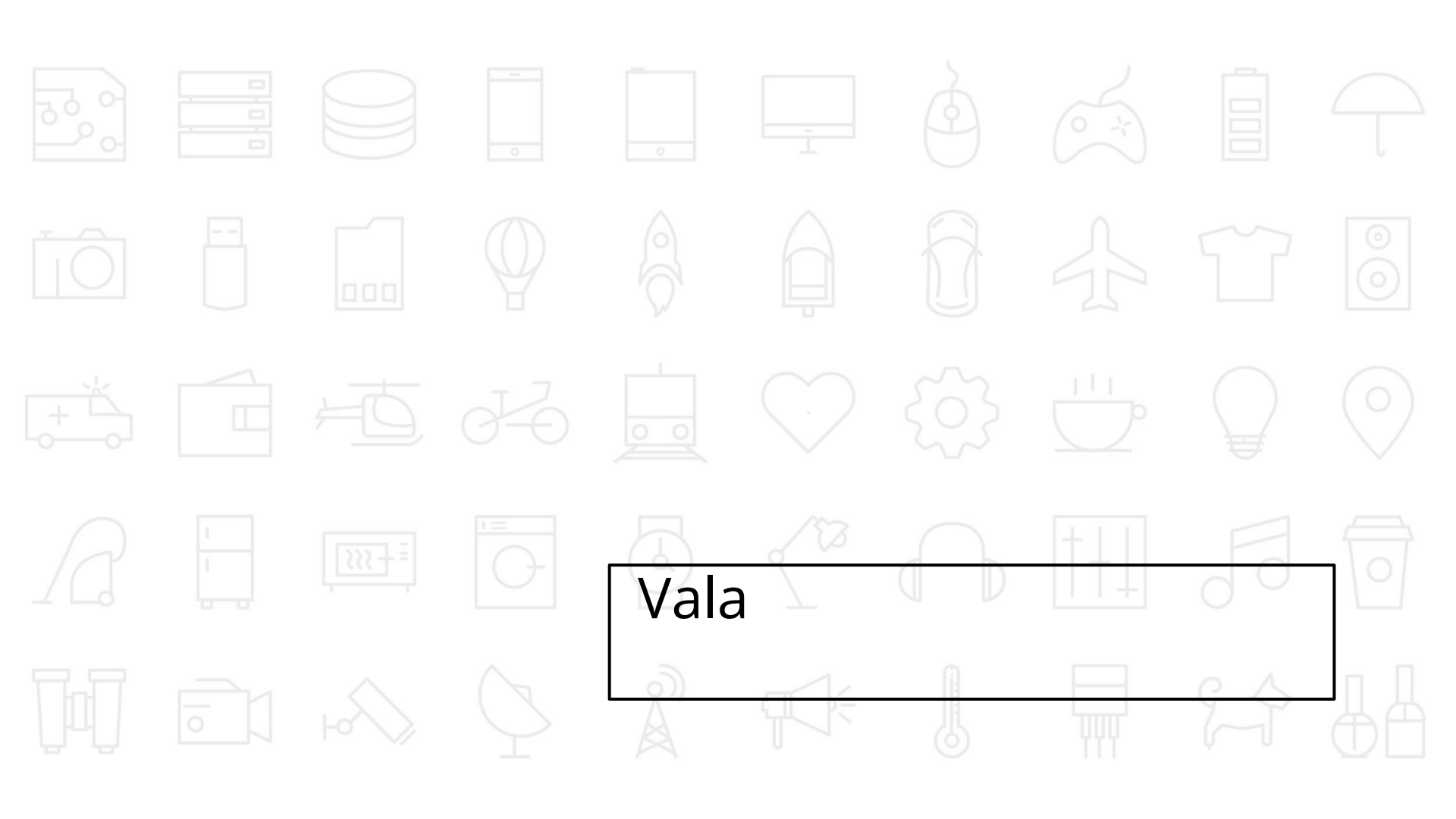
```
...
     <cmake id="red-rcp-api">
          <branch module="open-rnd.ros3d.camera"
                  checkoutdir="camera"
                  revision="ros3d/master"
                  />
          <dependencies>
          </dependencies>
     </cmake>
...
```

jhbuild - modules

```
...
     <distutils id="ros3d-dev-controller">
          <branch module="open-rnd.ros3d.controler"
                  checkoutdir="dev-controller"/>
          <dependencies>
               <dep package="ros3d-servo" />
          </dependencies>
     </distutils>
...
```

jhbuild

- Force proper build system tooling
  - automake, cmake, setuptools
- Less packaging effort
  - Early testing!!!
- Confined development root
  - Libraries not present in your repositories
  - Particular library versions


Vala

Vala

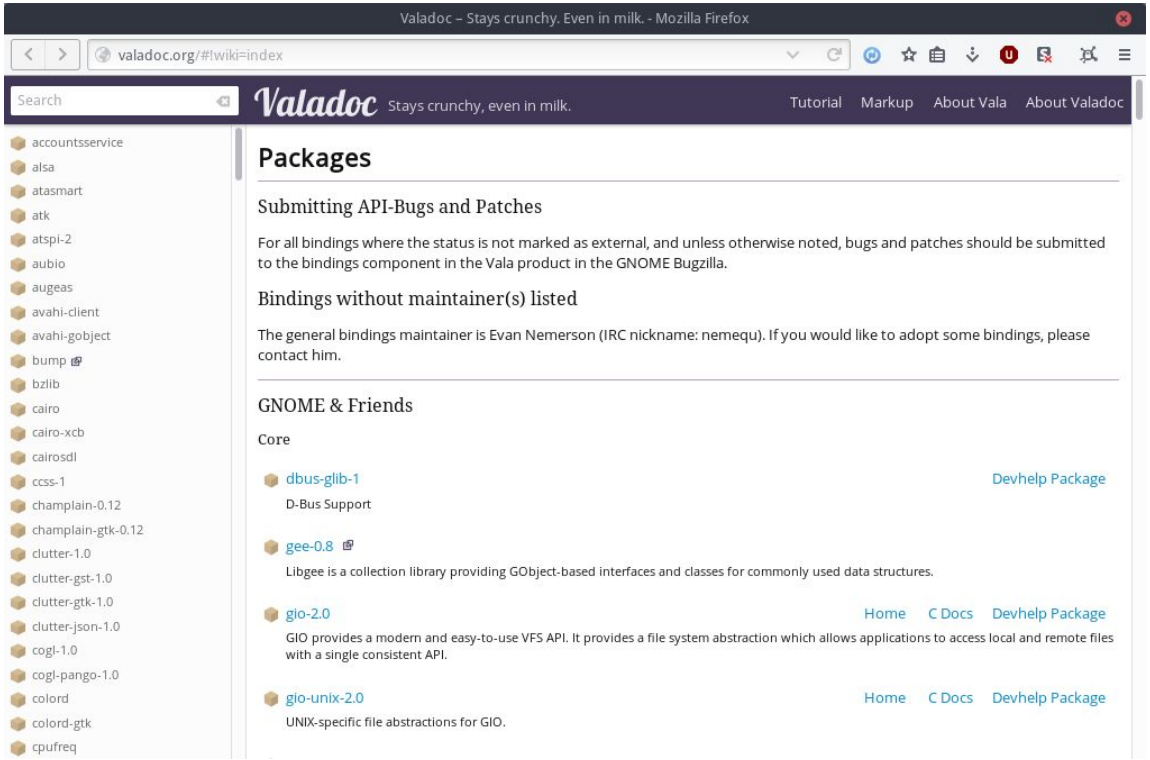
- High level, C# like syntax
- `valac`
  - C# compiles to C with GObject, Glib, Gio
  - C compiles with your favourite C compiler
- Zero overhead interfacing with C
- VAPI == C header
  - Lets Vala know how to call C code
- `autotools` and Yocto support:
  `inherit autotools pkgconfig vala`
  - no extra configuration needed
- Broken VAPI autogeneration
  - Why not ship VAPI files with source code?

Vala

- High level, C# like syntax
- `valac`
  - C# compiles to C with GObject, Glib, Gio
  - C compiles with your favourite C compiler
- Zero overhead interfacing with C
- VAPI == C header
  - Lets Vala know how to call C code
- `autotools` and Yocto support:
  `inherit autotools pkgconfig vala`
  - no extra configuration needed
- Broken VAPI autogeneration
  - Why not ship VAPI files with source code?

https://github.com/open-rnd/ros3d-stream

Valadoc


Valadoc – Stays crunchy. Even in milk. - Mozilla Firefox
valadoc.org/#!wiki=index
Search
Valadoc
Stays crunchy, even in milk.
Tutorial
Markup
About Vala
About Valadoc
accountsservice
alsa
atasmart
atk
atspi-2
aubio
augeas
avahi-client
avahi-gobject
bump
bzlib
cairo
cairo-xcb
cairosdl
ccss-1
champlain-0.12
champlain-gtk-0.12
clutter-1.0
clutter-gst-1.0
clutter-gtk-1.0
clutter-json-1.0
cogl-1.0
cogl-pango-1.0
colord
colord-gtk
cpufreq
Packages
Submitting API-Bugs and Patches
For all bindings where the status is not marked as external, and unless otherwise noted, bugs and patches should be submitted to the bindings component in the Vala product in the GNOME Bugzilla.
Bindings without maintainer(s) listed
The general bindings maintainer is Evan Nemerson (IRC nickname: nemequ). If you would like to adopt some bindings, please contact him.
GNOME & Friends
Core
dbus-glib-1
Devhelp Package
D-Bus Support
gee-0.8
Libgee is a collection library providing GObject-based interfaces and classes for commonly used data structures.
gio-2.0
Home
C Docs
Devhelp Package
GIO provides a modern and easy-to-use VFS API. It provides a file system abstraction which allows applications to access local and remote files with a single consistent API.
gio-unix-2.0
Home
C Docs
Devhelp Package
UNIX-specific file abstractions for GIO.

Conclusions

Conclusions

- Yocto worked out great
- Split layers
  - Project specific functionality
  - Platform dependent functionality
- Software configuration through files
- Packager friendly build systems
- Work close to the upstream
  - Contribute your patches
- Do not be afraid of following development branches

Thank you

Maciej Borzęcki
maciej.borzecki@open-rnd.pl

QUESTIONS? THOUGHTS? COMMENTS?

Feel free to contact us!

INFO@OPEN-RND.PL
WWW.OPEN-RND.pl

Tips - poor man’s kernel config fragments

```
FILESEXTRAPATHS_prepend := "${THISDIR}/${PN}-${PV}:"
SRC_URI += "\
        file://0001-ARM-dts-wandboard-leds-for-Ros3D-KR-platform.patch \
        file://0001-ARM-dts-wandboard-setup-GPIO-keys-for-Ros3D-KR-platf.patch \
        file://0001-tty-serial-imx-register-LED-triggers.patch \
        file://0002-usb-serial-register-LED-triggers-for-each-USB-serial.patch \
        file://usbserial.cfg \
        "

do_configure_prepend() {

    cfgs="${@ ' '.join([n for n in src_patches(d, True) if n.endswith('.cfg')])}"

    bbnote "configs: ${cfgs}"
    if [ -n "${cfgs}" ]; then
        for cfg in "${cfgs}"; do
            bbnote "Applying config ${cfg}"
            cat ${cfg} >> ${WORKDIR}/defconfig
        done
    fi
}
```

Yocto 101 condt.

- Cross-compilation toolchain
- Set of packages for host
  - *-native
- Set of packages for the target
  - one of RPM, DEB, IPK, tar.gz
- Root filesystem tree (core-image-minimal)
- Disk image
- SDK

Vala and Yocto

```
inherit autotools pkgconfig vala gitpkgv systemd

DESCRIPTION = "Ros3D Streaming Service"
LICENSE = "MIT"
SRC_URI = "\
git://git.open-rnd.net:29418/open-rnd.ros3d.stream;
protocol=ssh \
    file://ros3d-stream.service \
"

SRCREV = "${AUTOREV}"
PVBASE := "1.0"
PV = "${PVBASE}+gitr${SRCPV}"
PKGV = "${PVBASE}+gitr${GITPKGV}"

DEPENDS = "\
        glib-2.0 \
        libsoup-2.4 \
        gstreamer1.0 \
        avahi \
"
...
```

```
RDEPENDS_${PN} += " \
    glib-2.0 \
    libsoup-2.4 \
    gstreamer1.0 \
    libavahi-gobject \
    libavahi-client \
"
CONFFILES_${PN} += "\
    ${sysconfdir}/ros3d-stream/ros3d-stream.conf \
"

S = "${WORKDIR}/git"

SYSTEMD_SERVICE_${PN} = "ros3d-stream.service"

do_install_append() {
    # install systemd service files
    install -d ${D}${systemd_unitdir}/system
    install -m 0644 -t ${D}${systemd_unitdir}/system/ \
       ${WORKDIR}/ros3d-stream.service
}
```