Erlang-based Software Update Platform for remote devices

Authors:

Roman Janusz

Tomasz Kowal

Przemysław Dąbek

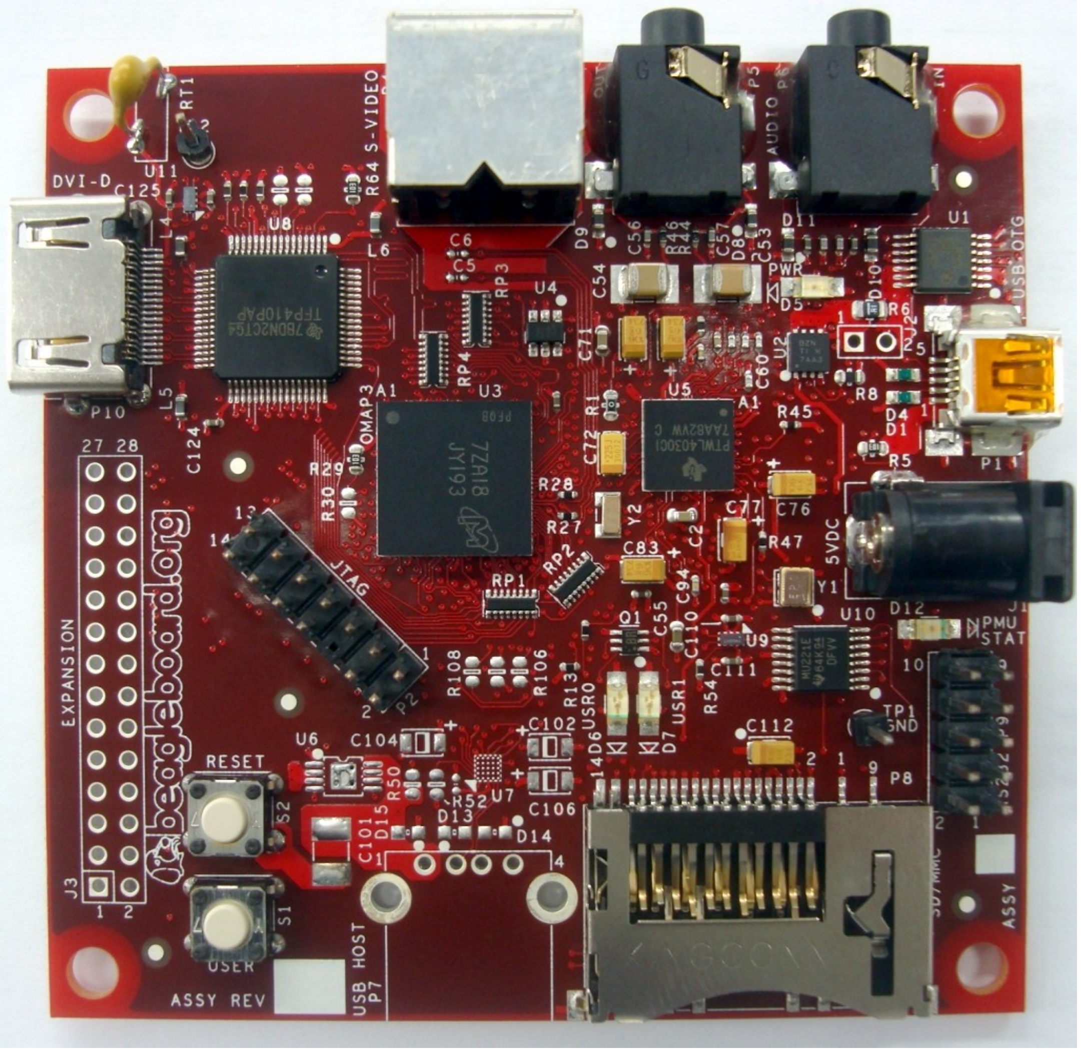
Małgorzata Wielgus

supervised by dr inż. Wojciech Turek

Content

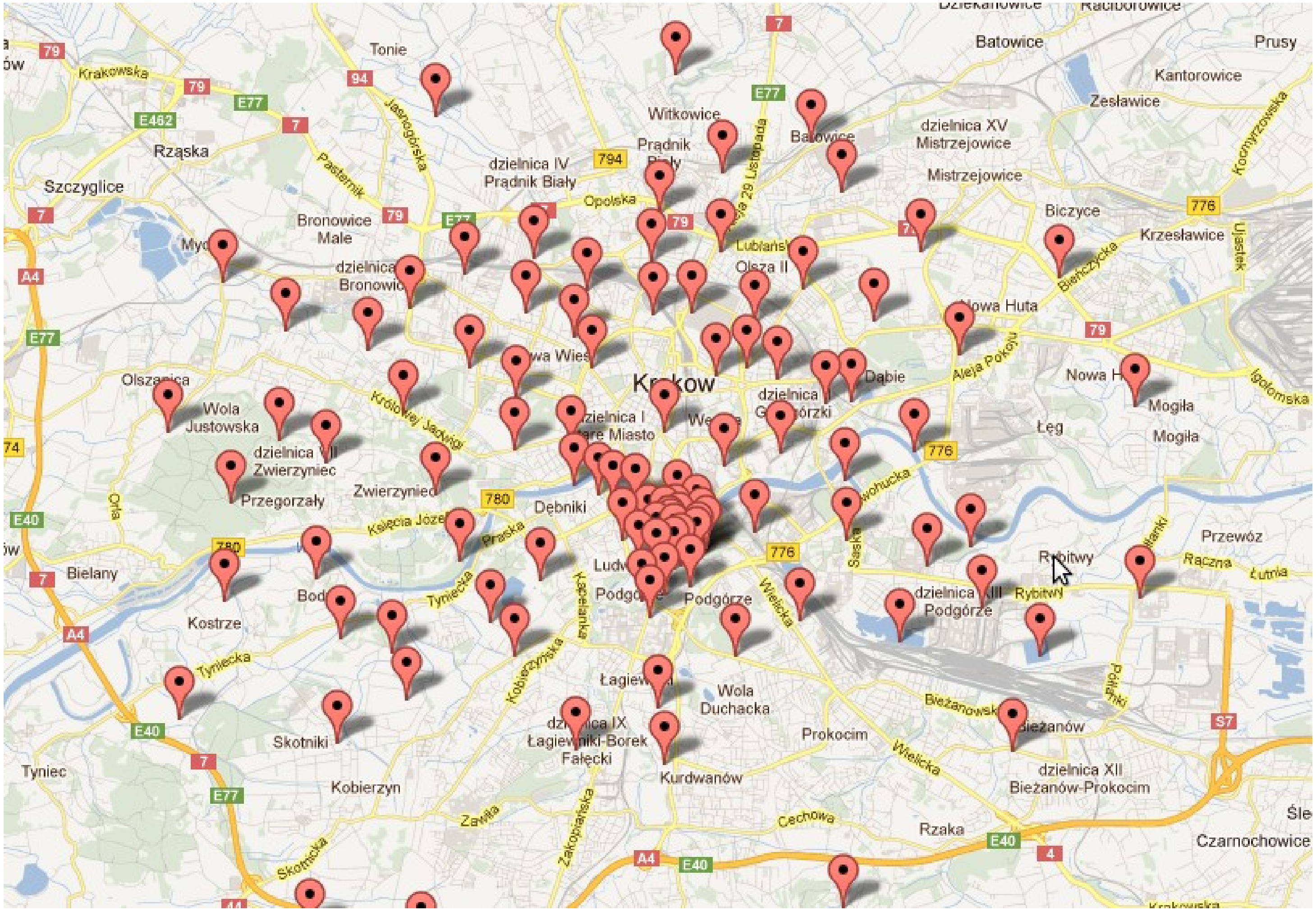
- Device description
- Use case example
- Requirements
- Technologies
- Architecture

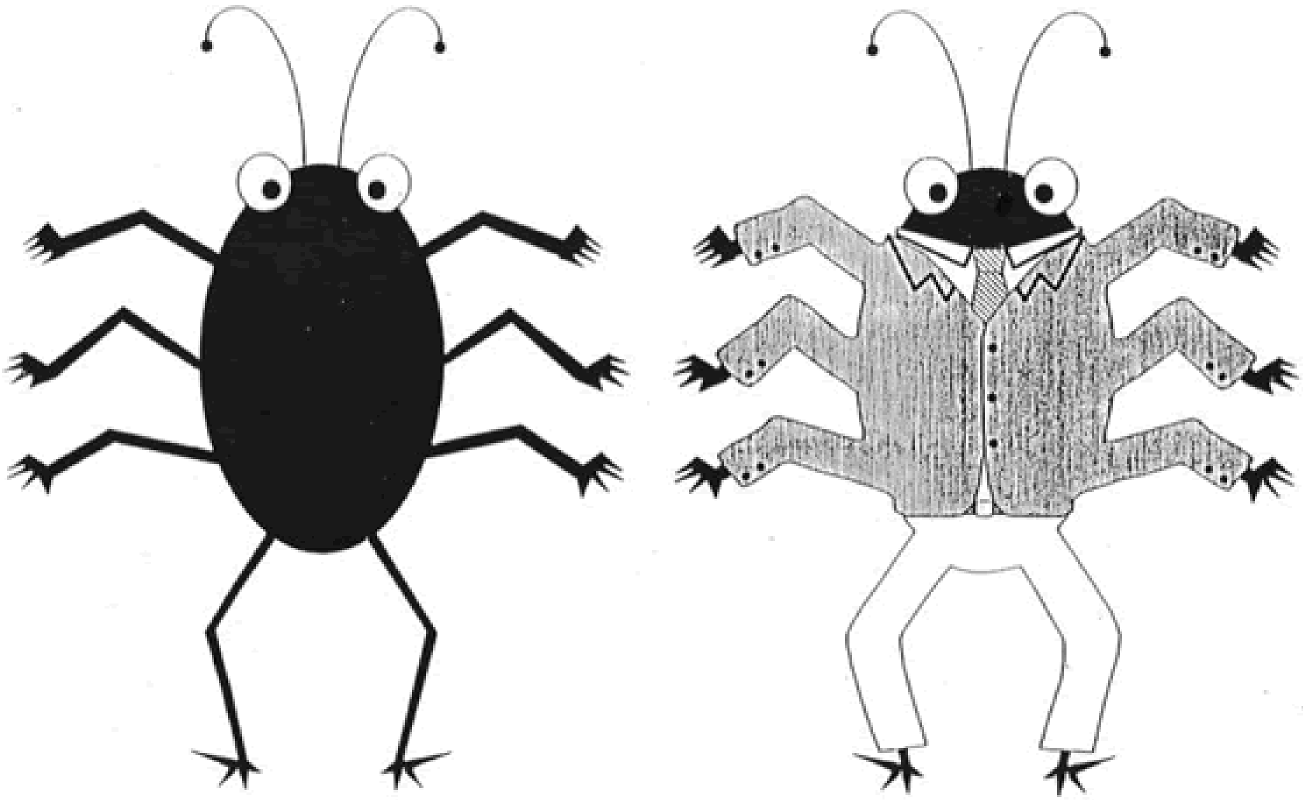
Beagleboard



Use case

Monitoring traffic in Cracow





BUG FEATURE

Requirements

- mass automatic upgrades
- reliability
- fine-grained control over upgrade process
- easy to install and use
- sending small amount of data via network

Erlang Features

- mass automatic upgrades
- reliability
- fine-grained control over upgrade process
- easy to install and use
- sending small amount of data via network

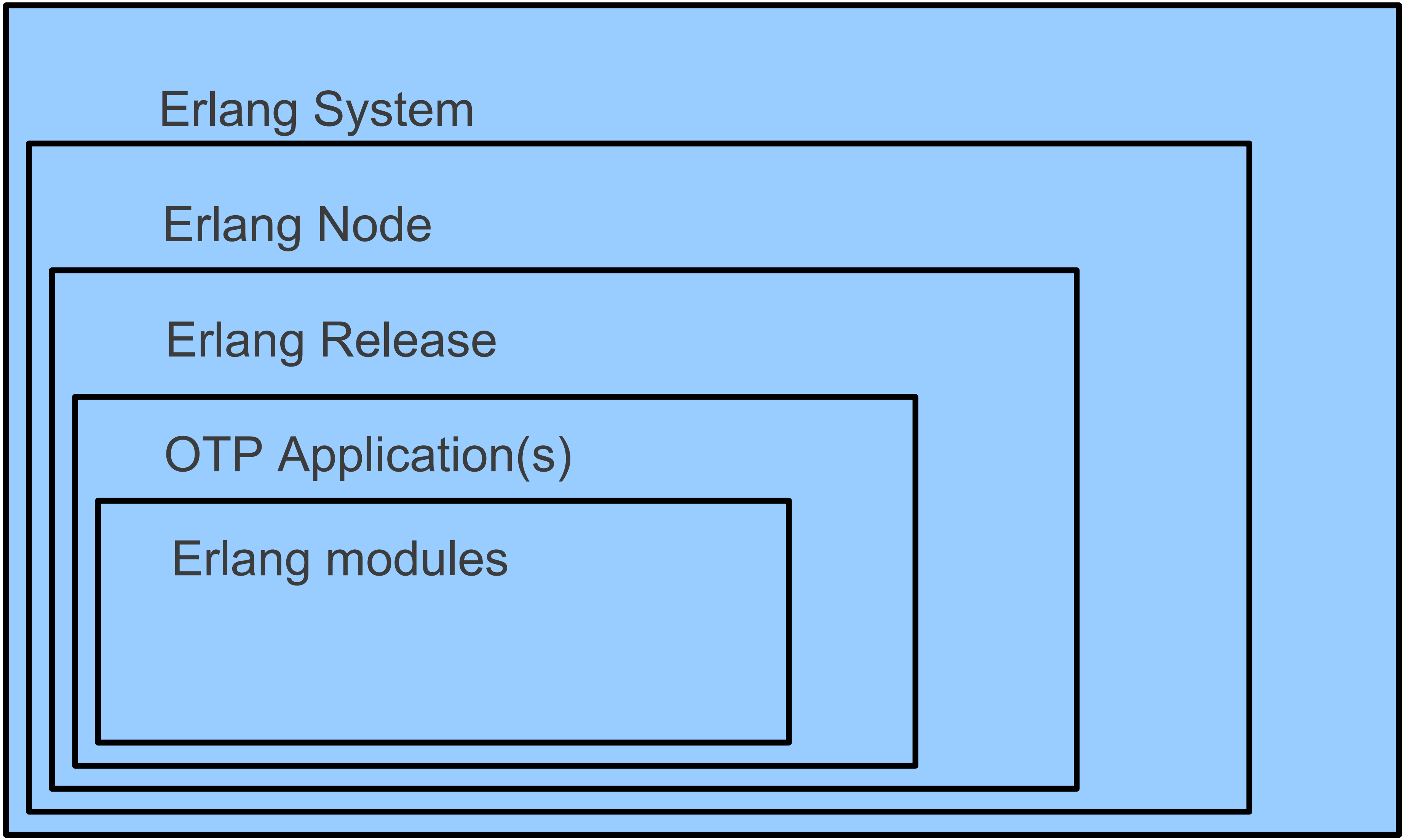
- massively parallel
- fault tolerant
- hot code swapping

DPKG advantages

- mass automatic upgrades
- reliability
- fine-grained control over upgrade process
- easy to install and use
- sending small amount of data via network

- massively parallel
- reliable
- hot code swapping
- easy to use
- saves bandwidth

Glossary



Erlang System

Erlang Node

Erlang Release

OTP Application(s)

Erlang modules

Design and architecture details

- Remote software development model
- Package manager integration
- General platform architecture
- Communication protocol

Development model

- Developer maintains an Erlang OTP release
- Main tool for building – **`rebar`**
- A few helpful scripts
  - **`genrelup`** for generation of **`relup`** files
  - **`makebasedeb, makeappdeb, makereldeb`** – for easy generation of .deb packages

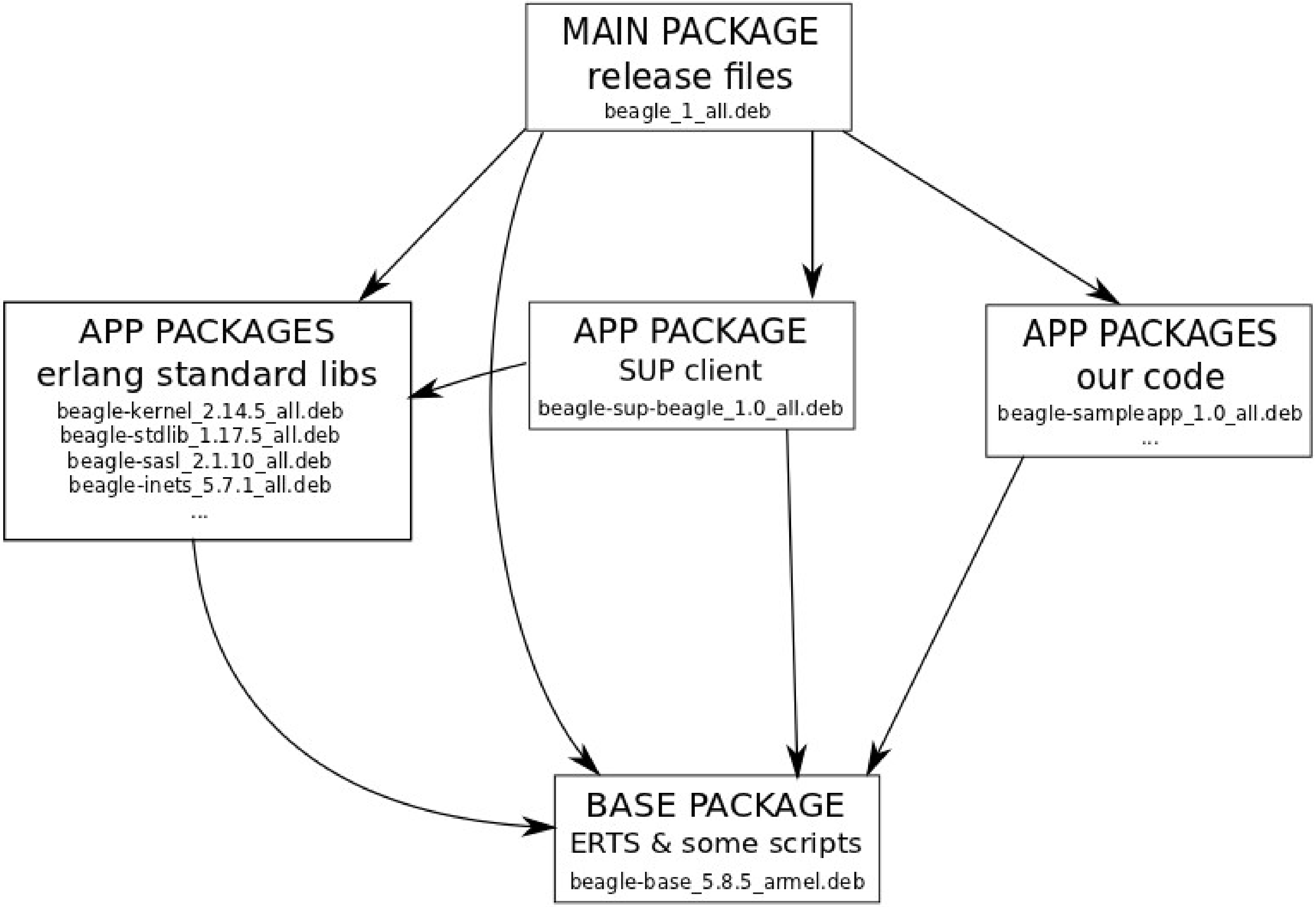
Package manager integration

- a layer over native erlang release handling
  - native erlang api is called by maintainer scripts
  - preserves ability to gain from hot code swapping

- benefits
  - easy manual device administration
  - saves bandwidth during upgrade - thanks to automatic dependency resolution

- `dpkg` (`apt`, `aptitude`, ...) - integrated for now

Decomposition into .deb files



MAIN PACKAGE
release files
beagle_1_all.deb

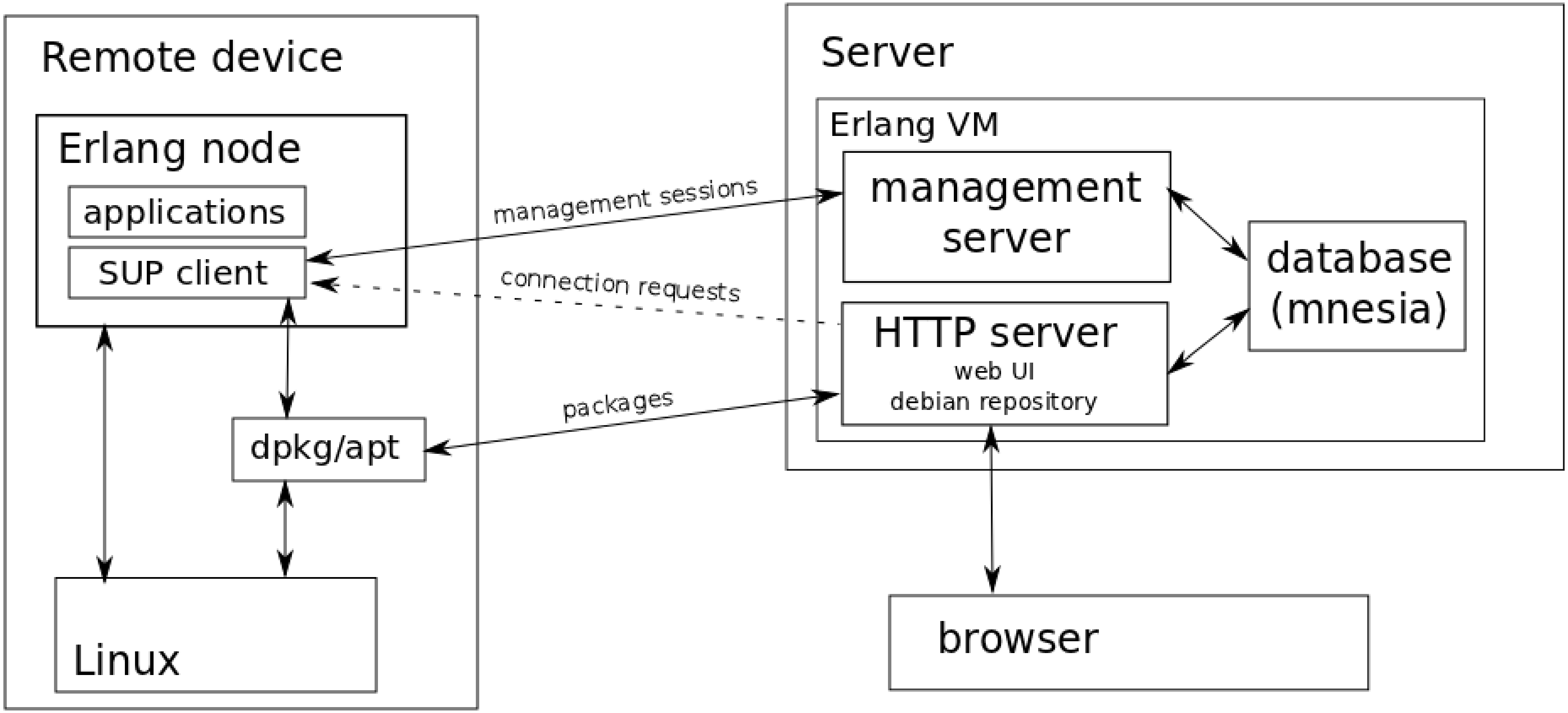
APP PACKAGES
erlang standard libs
beagle-kernel_2.14.5_all.deb
beagle-stdlib_1.17.5_all.deb
beagle-sasl_2.1.10_all.deb
beagle-inets_5.7.1_all.deb
...

APP PACKAGE
SUP client
beagle-sup-beagle_1.0_all.deb

APP PACKAGES
our code
beagle-sampleapp_1.0_all.deb
...

BASE PACKAGE
ERTS & some scripts
beagle-base_5.8.5_armel.deb

Platform architecture



Communication protocol

- Management sessions
- Always initiated by the device
- Possibility of connection requests
- Protocol – **`gen_tcp`**, **`term_to_binary`**, **`binary_to_term`**
- Generic, simple and extensible

Management session

- Initiated by device with an inform message
- Server looks into the message and database and decides what to do
- Server sends job to device
- Device performs job and sends back result
- Server sends another job or closes session

Summary

- Development model for erlang software on embedded devices with usage of rebar
- Mass management of remote devices
- Easy maintenance thanks to package manager
- Optimized size of downloads during upgrade
- Usage of hot code swapping – little downtime and fine-grained control over upgrade

Future development

- More general management platform
- Another package managers (`rpm`, `pacman`, …)
- Erlang distributed application management

Repository – open for clones!

https://github.com/tomekowal/SUP