Bringing RIoT-OS to the RIoTboard

Lennart Dührsen and Leon Martin George
Freie Universität Berlin

Softwareproject - Telematics, 2014

Context

- what do we want?
  - RIoT-OS running on the RIoTboard
  - have fun coding
  - fancy hardware
  - credit points
- what did we expect to achieve?
  - get the hardware for free
  - basic support of the RIoTboard for RIoT-OS
  - be motivated enough to continue working on the port after the software-project

Task-Division

- sub-goals:
  - gather relevant documents
  - find out how on the board ”works”
  - build a basic application that runs on the board (LED-blinking)
  - build this application from within RIoT-OS (run our program from the `board_init`)
  - enable interrupts
  - UART for STDIO
  - implement timer-interface
  - wiki pages
- milestone arrangement
- milestones have dates assigned
- milestones are coarse

Milestones

- get familiar with the board
  - boot it, read manuals and documentation
  - try features with existing OS that supports it
  - understand target architecture
  - flash it
  - cross-compile
  - be able to actually run bare-metal code
- find out what needs to be done for a port
  - identify re-useable code
  - learn about interfaces in RIoT
- port it
  - patch SDK for use in RIoT
  - successfully build
  - debug

Outcome

- all goals reached
- spaghetti

Demo

DEMO (of `printfs` and flashing LEDs)
It works! (But is far from being complete or a good codebase)

Problems - documentation

- expectation: to turn on the LED write a bit to $beef1337:3
- reality:
  - i.MX6-reference-manual
    - the status of a GPIO-pin is determined by a bit in a register that can be anywhere - based on the configuration of the muxer
    - names of channels in the muxer are from the same namespace as the functions mapped onto them
  - RIoTboard-schematics
    - one LED on the RIoTboard is connected to a function "EIM_DATAwx" which you can then map GPIOyz on

Problems - reference code

- embest-tech doesn't supply a muxer-configuration-file that can be used with the SDK
- maybe looking at how they did it for their u-boot- and linux-ports helps?
- reconsideration: the i.MX6-platform-SDK has macros to abstract to and from the muxer config

Problems - different abstractions

- The platform-SDK differentiates between code concerning
  - the i.MX6-architecture
  - peripherals
  - board-specific headers and iomux-configuration
- RIoT has its own abstraction for each of those (dividing the sub-topics differently)

PRs

- #1355 was closed in favour of #1359 Leon had trouble keeping the git-log tidy
- #1359 was merged - with the help of staff members and RIoT-maintainers.
- #1411 is still pending and we do not know whether it will be merged.

Perspectives

- still motivated
- perhaps scrap the existing code and restart
- or: clean the existing code

Questions?