⇓ Were porting to the RIoTboard. You probably know us by now...

1 Motivation

⇓Context also commonly known as: META-stuff-slide ↓
↓
because, why not? ↓
who doesn't? ↓
thank you (for the hardware) ↓
why wouldn't we? ↓
↓
because that's the only real reason to join a telematics-project ↓
just because these names go really well together.. ↓
seems like it's far more interesting than "deytabeysis" + plus we can keep the board for as long as we are working on it

2 Task-Division

⇓Task-Division aka "boring slide" ↓
we had help planning the steps before the project even started ↓
if only we had read them also... ↓
we asked for the hardware early and having the board and a linux-port really helps ↓
get something to run ↓
build "something" the RIoT-style ↓
interrupts are enabled ↓
stdio is somehow nice to have ↓
there are two.... ↓
have wiki pages in our github-fork ↓
these points really don't make sense in any other order ↓
the team urged us not to try to do too much and we were very conservative with the dates ↓
and we expected to be able to split them ad-hoc

⇓Milestones these are the milestones —let them read a bit— identify re-useable code: we tried to use the u-boot-port and ... panic mode!
learn about interfaces in RIoT: skipped this and ran straight to adjusting the code of the SDK

3 Outcome

⇓Outcome ↓
but.. ↓
we have working code that is in an "interesting" state

⇓Problems - documentation or: did everything work as expected? short answer - as with every project: no (or nööööö) while gathering documents we were expecting something like a straight-forward memory-map like we were used to from the micro-processor-lab ↓
warning: the ugliness of these slides depicts the ugliness of having to find out these things the hard way ↓
this is the same for any other component: there are eight DIP-switches on the board: go to the schematics. switches are numbered from 0 to 7 and have functions assigned (FUSEs). FUSEs are described in the reference manual. But the physical switches are numbered from 1 to 8. So =¿ back to the quick-start-guide where there are sample-configurations for flashing UART it says nowhere which of the three PINs is which.

⇓Problems - reference code ↓
which is kinda sad. They have targeted mainly android development and a bit of linux ↓
is spaghetti. we don't dare think about how our code would look like if be based it on spaghetti. ↓
which you can only use if you have valid configuration file which in turn can only be created and generated code from with a windows-only-tool

⇓Problems - different abstractions ↓
then there is a problem of different abstractions in the SDK and in RIoT ↓
CPU, interrupt-controller ↓
driver for UART and timers (which in RIoT belong to the CPU) or ethernet and USB ↓
iomux-configuration, register definitions (depending on the kind of i.MX6 - SL - SDL - D) ↓
We should we include the SDK ?

⇓Demo DEMO (of `printf`s and flashing LEDs)

⇓PRs At the time of writing we have one failed, a successful and another pending pull request: This would have allowed using LD for linking by changing the global Makefile and the Makefile of any board not using LD (so far: all except the RIoTboard). This basically does the same as #1355 but leaves other Makefiles untouched and assuming GCC to be used by default. Boards using LD have to explicitely supply a variable. It's purpose is to bring the software-project to a conclusion and add support for the riotboard and the i.MX6-SDK. This contains hundreds of file from the SDK that have been altered to just work. (obviously not fit for rolling out)

4 Future

⇓Perspectives restart in two weeks of time

⇓Questions?