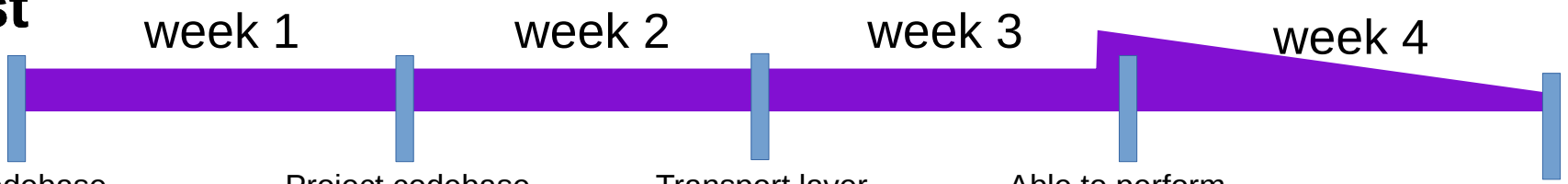
Internship @ Mindtribe:
Progress report 08-21-2015

RETROSPECTIVE: WEEK 3 – 08-17

- **Project progress:**
  - Last week, halting and memory read/write were possible (hard-coded into the debugger code)
  - Expanded this with possibility to continue execution and access registers
  - At this point, some interactive way to use the debug adapter was needed: just hard-coding these operations into the debugger sequence to test them became cumbersome
  - Therefore, instead of first finishing all necessary debug operations' implementation, decided to start implementing the gdbserver stub in order to be able to communicate with the debugger and instruct it what to do.
  - Result: basic gdbserver stub running over a UART link to the host pc. So far, it is already able to load a program (slowly) into the target, verify it and run it, and can halt/continue as well as inspect registers / memory – controlled by GDB running on the host PC.

ORIGINAL PROJECT PLANNING (JTAG → GDBserver → WiFi)

August



Project codebase set up, toolchain configured

Project codebase set up, toolchain configured

Transport layer done, able to read man. ID

Able to perform one debugging funtion (e.g. halt)

Able to perform most relevant debugging functions

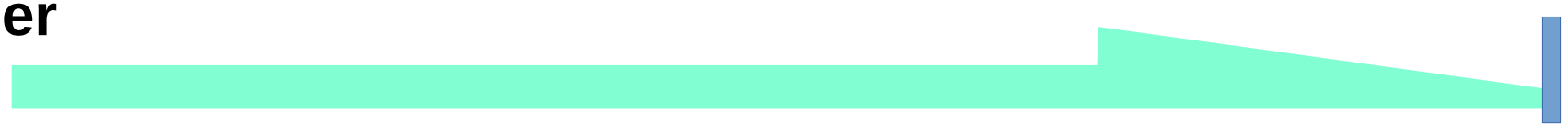
September

Sander on leave

Focus on getting a serial link to GDB host (UART or similar) and implementing the GDBserver stub for communicating to GDB. Meanwhile, finish implementing a minimal set of JTAG debugging functions (halt, step, continue, reset, memory read/write, register read/write).

A limited but working UART debugger

October



Focus on implementing a serial link over WiFi through TCP sockets, configuring the CC3200 as an access point. Once done, existing GDBserver stub should be easy to adapt to use this.
Other than that, polish and add any extra features that time may allow (breakpoints, watchpoints? Serial communication to debuggee? Etc)
Also: knowledge transfer, evaluation/code review, open-sourcing the project.

A (reasonably) complete WiFi debugger.

CURRENT STATUS

- Arguably, the goal of month two is already reached: “a limited but working UART debugger”.
- **However, there is much more to do, which may not be very well-represented in the original planning:**
  - A lot of debugging functionality should still be added before this becomes a debug adapter worth using (reset, faster transfers/loads, breakpoints/watchpoints, debug symbol reading, etc)
  - Performance improvements for loading require more features (pipelined memory accesses at the JTAG level, JTAG bitbanging speed, etc)
  - Not yet robust in GDB communication
  - Being able to write/read flash memory is high on the list of wanted features, and will require a new stub program
  - WiFi
  - For open-sourcing, a certain level of code quality, decent project structure etc is required