- Remote-Debugging with Eclipse -

Goal:
This exercise helps you to gain first experiences with remote debugging using Eclipse.

Exercise 3.1: Cross-Compilation of Hello_World for the Target

In Eclipse select the platform-config „Debug with Cross GCC” and build the executable in ARM format. Check that Eclipse has used the correct cross-compiler by watching the console output. In the hello_word project directory you now find a new subfolder and the ARM executable within. Using the **`file`** comand you might doublecheck the code.

Now you copy the executable for ARM to the target using **`scp`** (Q: do you know an alternative?). Start the program on the target and watch the result.

Note: if you are not familiar with the gdb, you might want to create local debug configuration and execute it.

Exercise 3.2: Remote-Execution in Eclipse

As described in the lecture, now you specify your first target connection. Create a cross-run-config (“C/C++ remote application”) and set the target connection instead of “local”. Fill in additional information such as the target directory (absolute file path) and the pre-execution commands. Make sure, the target directory exists. Now execute the cross-run-config. Doublecheck and understand the output in the Eclipse console.

Exercise 3.3: Remote-Debugging with Eclipse

Prepare the target and the host environment for remote debugging, i.e. make sure, the package **`gdb_server`** is installed and ready on the target, and a cross-debugger is available on the host (which is part of the cross-toolchain provided). Now extend your cross-run-config to specify a remote-debug-configuration.

Note:
Both remote-run-configs and remote-debug-configs require a target-configuration to specify how to connect to the target (“connection”). However, while the target-config used for the remote-run-config might address the target using the name of the target (by means of the Avahi / Bonjour protocol), this is not possible for a remote-debug-config. Here we need a target-config with the explicit IP-address of our target.

Then start the debug session. Switch to the debugger perspective in Eclipse and exercise a single-step execution using the GUI of the debugger. Just as in the case of local debugging, set normal breakpoints, conditional breakpoints, read and write program variables during the live debugging session.