
Aktuelle Themen der IT-Sicherheit: RIOT Challenges

WS 22/23: Prof. Dr. Jan Seedorf

Thomas Jakkel (Mat. Nr), Severin Nonenmann (Mat. Nr), Lukas

Reinke (1001213) (Mat. Nr), Lars Weiß (Mat. Nr)

29. Januar 2023



Inhalt

1	Woche 1	2
1.1	Challenge 1: VM installation	2
1.2	Challenge 2: Erste schritte mit RIOT	2
1.2.1	First_test Application	2
1.2.2	Simple Network communication	4

1 Woche 1

1.1 Challenge 1: VM installation

Das VM setup wird hier nicht genauer beschrieben.

1.2 Challenge 2: Erste schritte mit RIOT

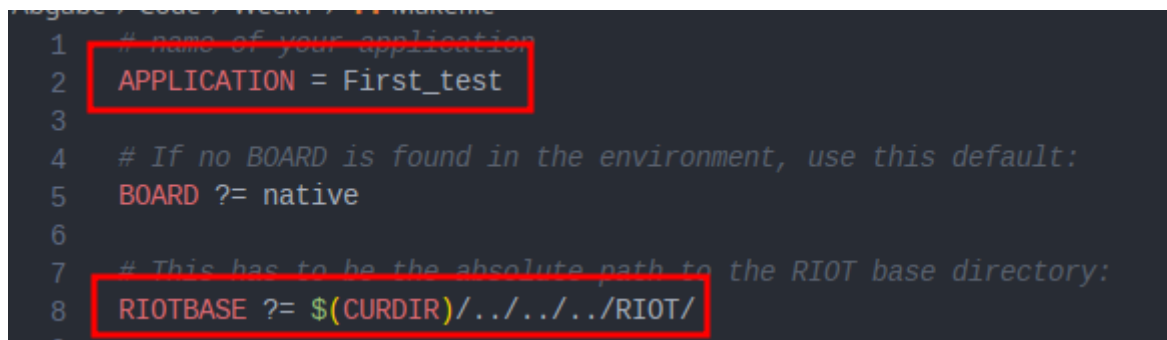
Die Anleitung zum Setup von RIOT OS aus den RIOT Tutorials wurde durchlaufen und ein funktionierender Workspace erstellt.

Wir haben das Setup insofern verändert, dass unser Code in einem separaten Ordner neben dem von GitHub geklonten RIOT Dateien liegt.

1.2.1 First_test Application

Das Ziel ist ein erstes RIOT-OS selber zu kompilieren, mit einer eigen Funktion zu versehen und zu starten.

Im default Makefile müssen zwei Änderungen vorgenommen werden: 1. In der Variable `APPLICATION` der Name der Ausführbaren binary zu setzen 2. Die `RIOTBASE`, dem Pfad zu den Hauptdateien des RIOT-OS, zu setzen.



```
1 # name of your application
2 APPLICATION = First_test
3
4 # If no BOARD is found in the environment, use this default:
5 BOARD ?= native
6
7 # This has to be the absolute path to the RIOT base directory:
8 RIOTBASE ?= $(CURDIR)/../../../../RIOT/
```

Abbildung 1: Einfaches Makefile

Es soll eine shell command geschrieben werden der bei Aufruf einen String aufgibt. Zugrunde liegt eine einfache C Funktion mit einem `printf()` statement:

Des weiteren muss die Funktion in einem Array eingetragen und dieses Array als Quelle für Shell-befehle in der `main` Funktion registriert werden.

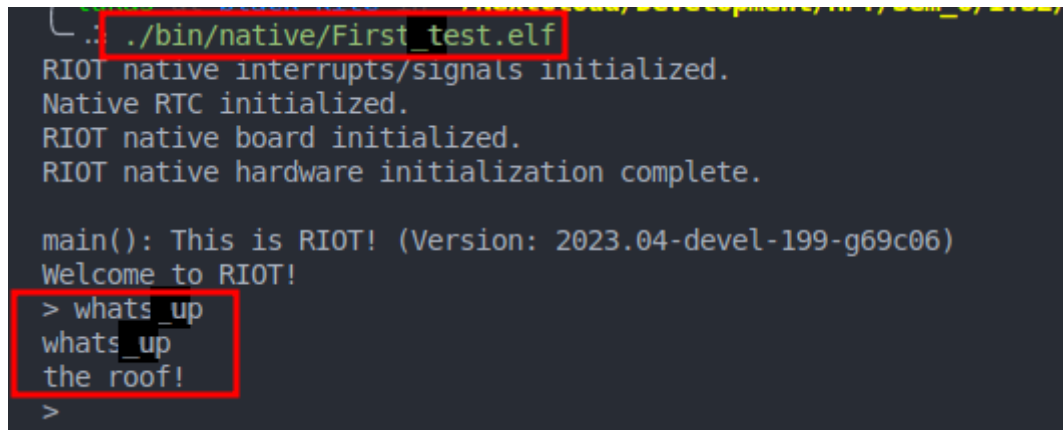
```
35
36 static int whats_up(int argc, char **argv) {
37     (void)argc;
38     (void)argv;
39
40     printf("The roof!\n");
41     return 0;
42 }
43
```

Abbildung 2: Funktion Whats_up

```
43
44 const shell_command_t shell_commands[] = {
45     {"whats_up", "prints the roof", whats_up},
46     { NULL, NULL, NULL}
47 };
48
49 int main(void)
50 {
51     #ifdef MODULE_NETIF
52     gnrc_netreg_entry_t dump = GNRC_NETREG_ENTRY_INIT_PID(GNRC_NETREG_DEMUX_CTX_ALL,
53     gnrc_pktdump_pid);
54     gnrc_netreg_register(GNRC_NETTYPE_UNDEF, &dump);
55     #endif
56
57     (void) puts("Welcome to RIOT!");
58
59     char line_buf[SHELL_DEFAULT_BUFSIZE];
60     shell_run(shell_commands, line_buf, SHELL_DEFAULT_BUFSIZE);
61
62     return 0;
63 }
64
```

Abbildung 3: Shell Kommando Registrieren

Nun kann mithilfe des `make`-Kommandos ein build gestartet werden und die resultierende Binary mit dem Namen **First_test.elf** ausgeführt werden. In der RIOT Shell kann nun der Befehl `whats_up` ausgeführt werden.



```
./bin/native/First_test.elf
RIOT native interrupts/signals initialized.
Native RTC initialized.
RIOT native board initialized.
RIOT native hardware initialization complete.

main(): This is RIOT! (Version: 2023.04-devel-199-g69c06)
Welcome to RIOT!
> whats_up
whats_up
the roof!
>
```

Abbildung 4: `whats_up` Befehl in RIOT shell

1.2.2 Simple Network communication