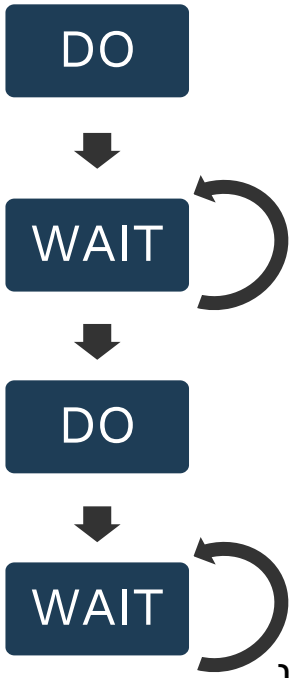
6th Workshop on Software Engineering for Cyber-Physical Production Systems (SECPPS) @ SE24 @ JKU

Machine Sequence Control with Lua Coroutines

Albrecht Wöß | Software Architect | Signum+ Software
Markus Löberbauer | Partner | Signum+ Software
Georg Koll | R&D Manager Plattform Software | TRUMPF Maschinen Austria

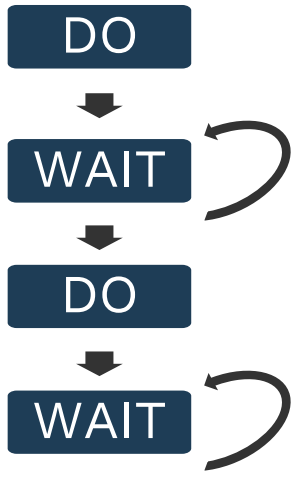
TRUMPF

Example: Step Chain vs. Sequence



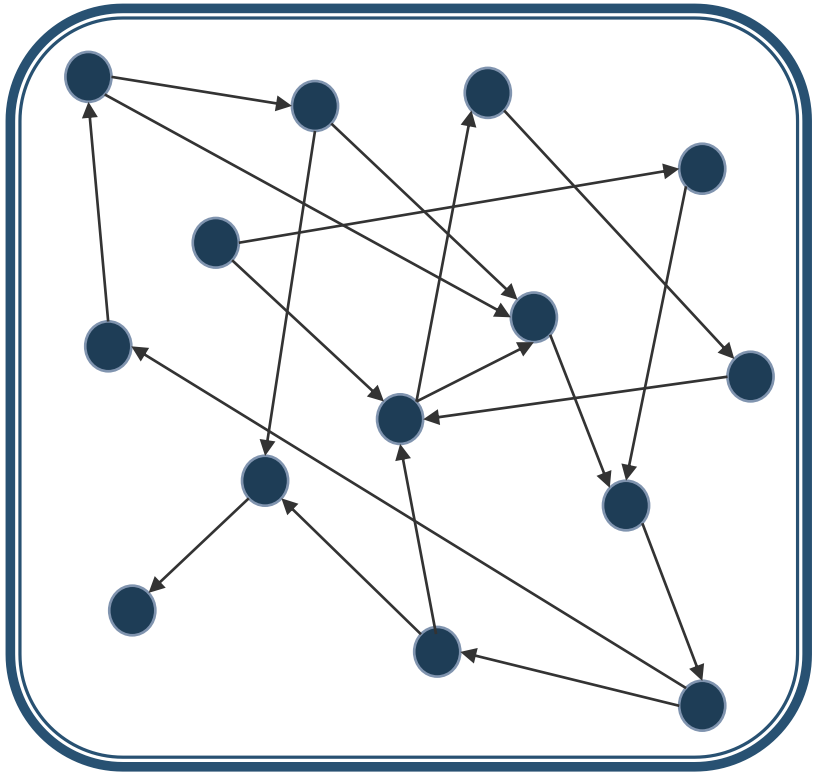

```
enum LED_BLINKER_STATE { ENABLE, ON, DISABLE, OFF };
LED_BLINKER_STATE current_state = ENABLE;
time_t led_state_change_time;

void led_blinker() {
  switch (current_state) {
    case ENABLE:
      enable_led();
      current_state = ON;
      led_state_change_time = get_current_time();
      break;
    case ON:
      if (get_current_time() > led_state_change_time + 50) { current_state = DISABLE; }
      break;
    case DISABLE:
      disable_led();
      current_state = OFF;
      led_state_change_time = get_current_time();
      break;
    case OFF:
      if (get_current_time() > led_state_change_time + 50) { current_state = ENABLE; }
    break;
  }
}
```

Example: Step Chain vs. Sequence



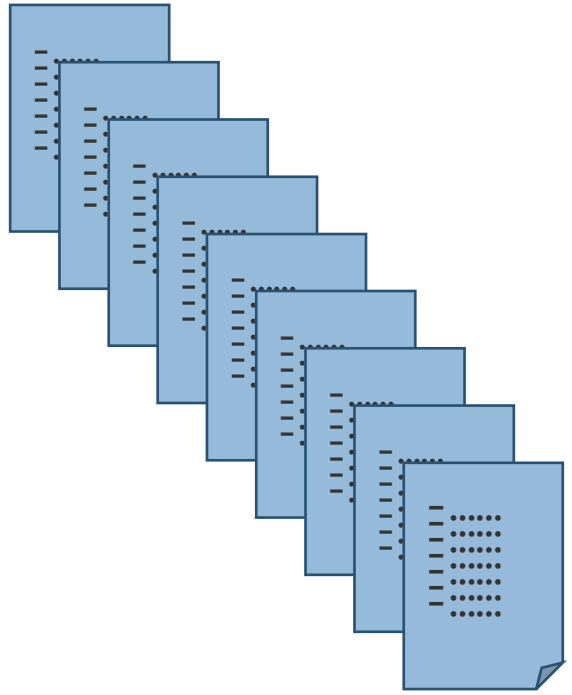

```
void led_blinker() {
  while (true) {
    enable_led();
    wait(50);
    disable_led();
    wait(50);
  }
}
```

Software Monolith vs. Loadable Scripts



All possible tasks connected in a complex *inseparable* **web of control scripts**

Complexity is many times as high as the sum of all tasks‘ complexities



One script per task (***un-/ loadable***)

Complexity is equal to each task‘s complexity

Thread vs. Coroutine

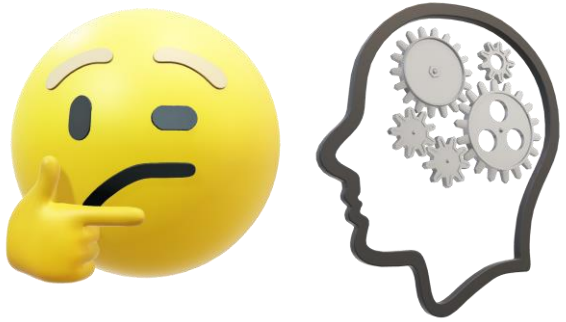
Who decides, when execution switches to the next script?

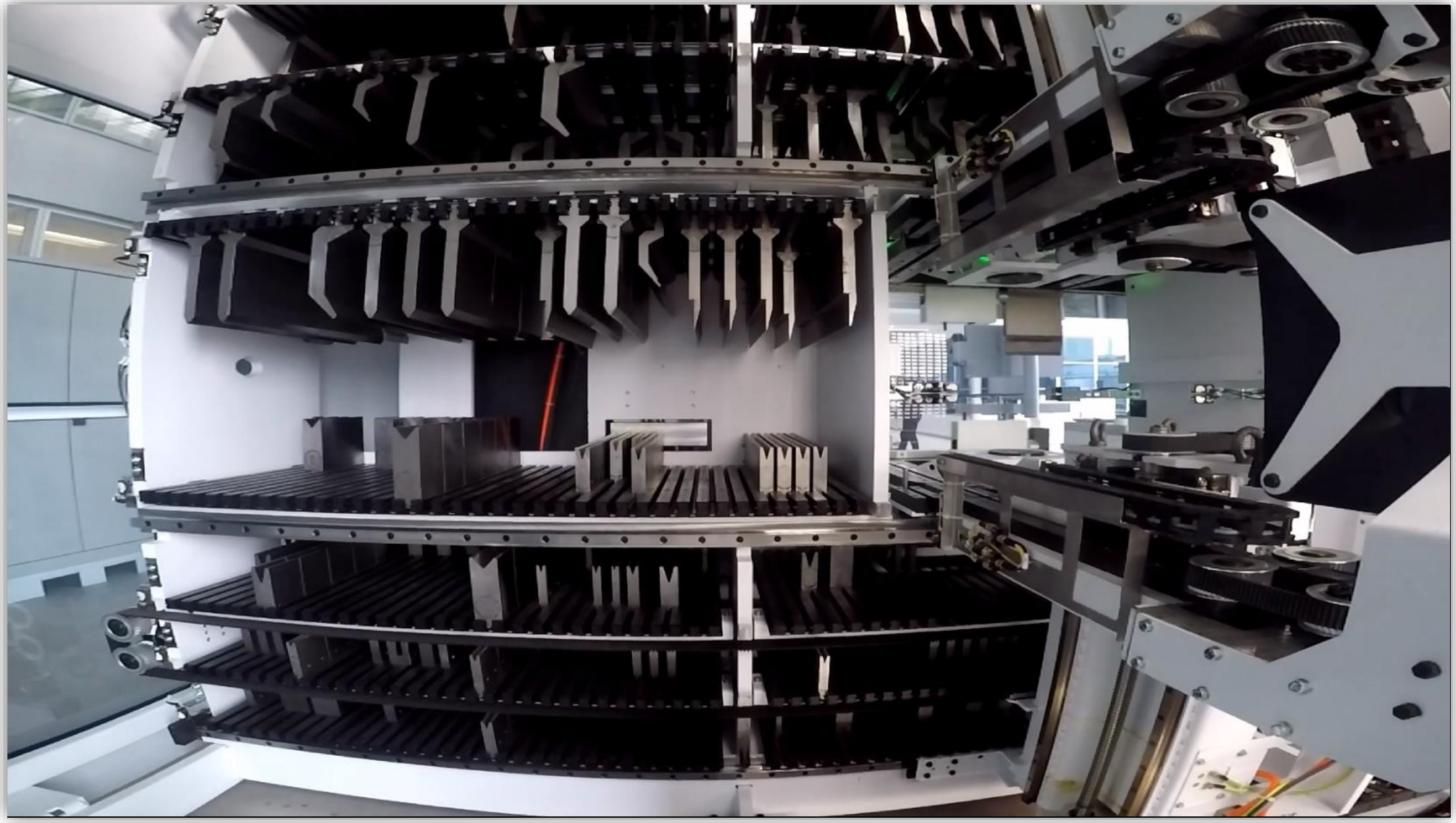
Thread = preemptive multi-threading

➔ Computer / Operating System / Machine

Coroutine = cooperative multi-threading

➔ **Program / Programmer / Human Person**

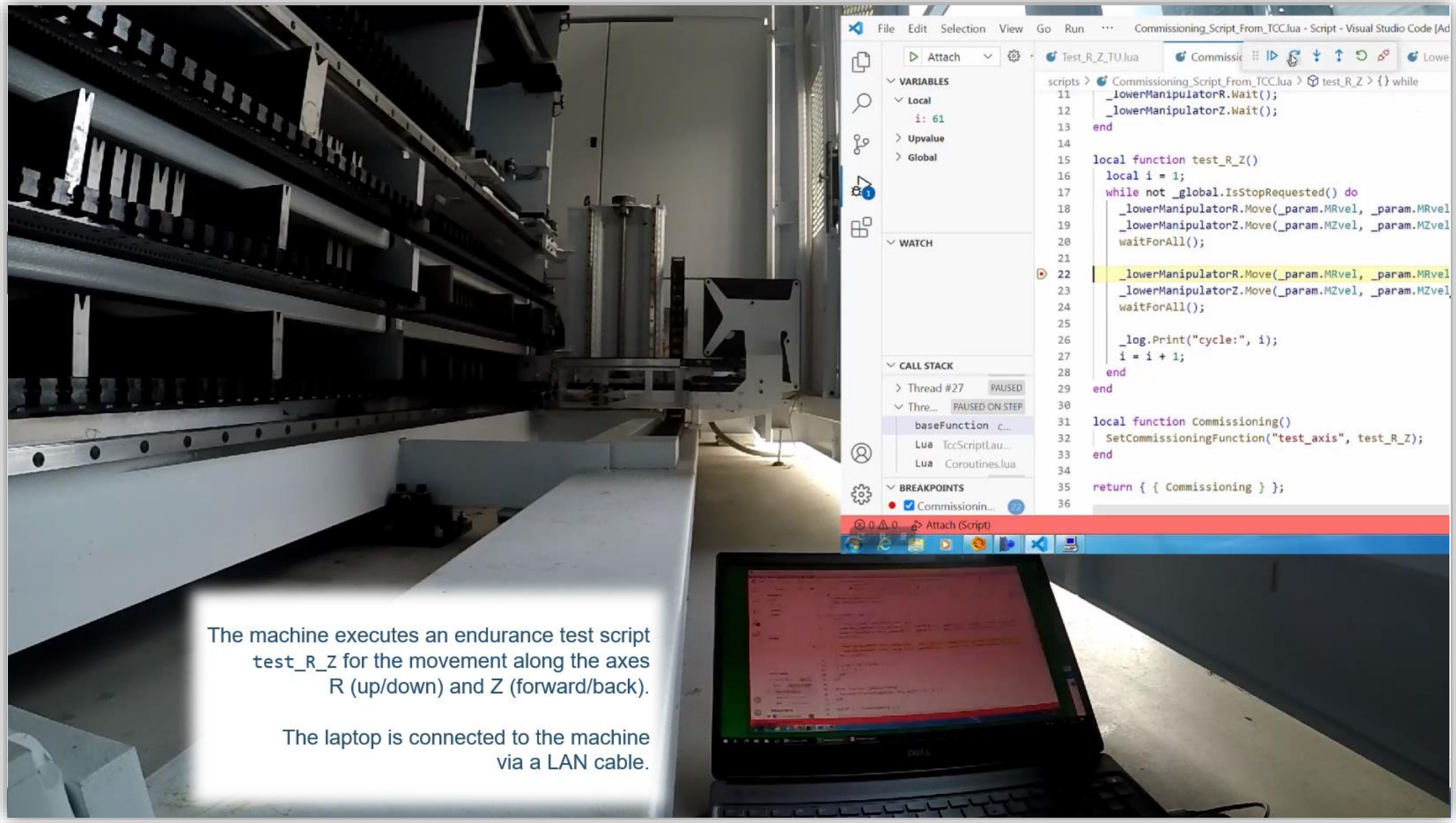




Lua Controls the Machine

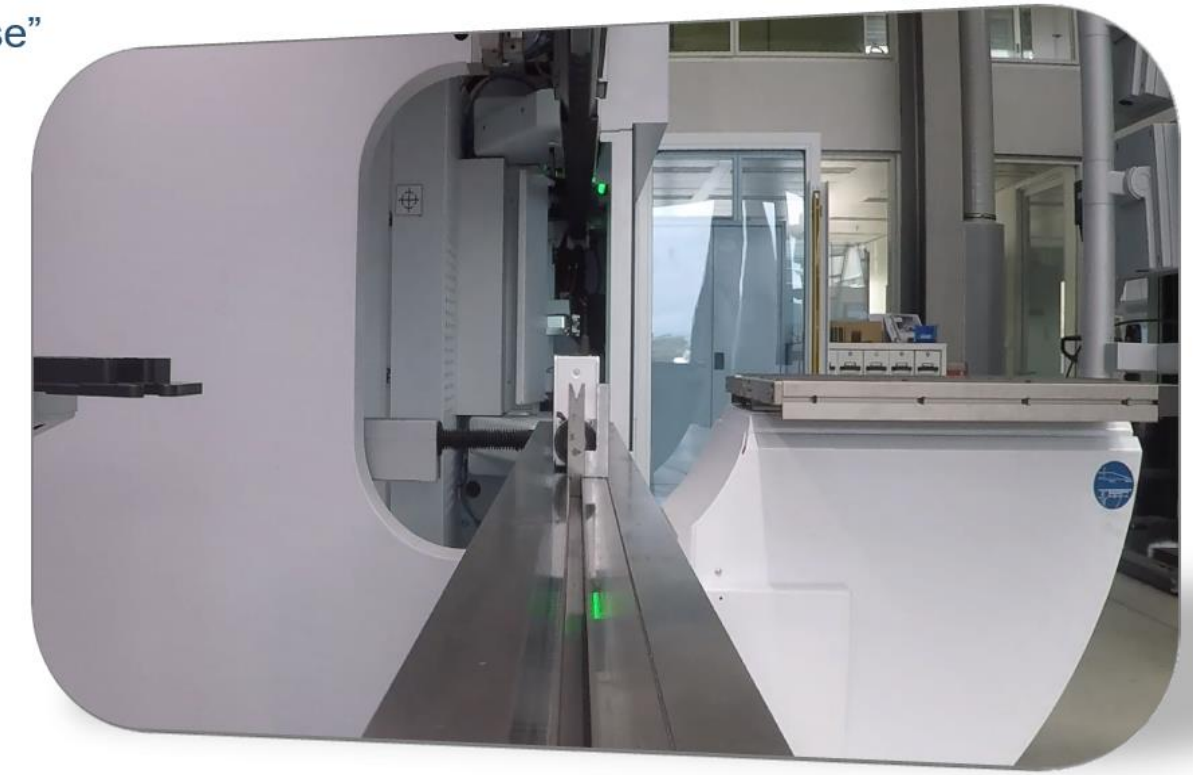
- **We provide all elementary functions for each actor of the machine as coroutines.**
  - Such coroutines quickly check, if they have completed their current task or should start a new one.
  - Then they immediately yield control to the next coroutine (`coroutine.yield()`).
  - **Cooperative!**
    The programmer has to pay attention to only take control for a minimal amount of time (**microseconds**!).
- **The main loop executes 1000x per second**
  - and resumes each coroutine (`coroutine.resume()`).
- **Development Environment: Visual Studio Code with Lua Language Server** (by sumneko) **extension**
- **Our Lua Interpreter supports yield even from C subroutines** (under Windows)**.**
- **The debugger can attach directly to the running machine, even in production!**
  - It stops the whole system and then allows single stepping.
  - There is almost no runtime overhead.

Debugging the Machine


File Edit Selection View Go Run ···
Commissioning_Script_From_TCC.lua - Script - Visual Studio Code [Ad
Attach
Test_R_Z_TU.lua
VARIABLES
Local
i: 61
Upvalue
Global
WATCH
CALL STACK
Thread #27 PAUSED
Thre... PAUSED ON STEP
baseFunction c...
Lua TccScriptLau...
Lua Coroutines.lua
BREAKPOINTS
Commissionin... 22
Attach (Script)
scripts > Commissioning_Script_From_TCC.lua > test_R_Z > {} while
11 _lowerManipulatorR.Wait();
12 _lowerManipulatorZ.Wait();
13 end
14
15 local function test_R_Z()
16 local i = 1;
17 while not _global.IsStopRequested() do
18 _lowerManipulatorR.Move(_param.MRvel, _param.MRvel
19 _lowerManipulatorZ.Move(_param.MZvel, _param.MZvel
20 waitForAll();
21
22 _lowerManipulatorR.Move(_param.MRvel, _param.MRvel
23 _lowerManipulatorZ.Move(_param.MZvel, _param.MZvel
24 waitForAll();
25
26 _log.Print("cycle:", i);
27 i = i + 1;
28 end
29 end
30
31 local function Commissioning()
32 SetCommissioningFunction("test_axis", test_R_Z);
33 end
34
35 return { { Commissioning } };
36
The machine executes an endurance test script test_R_Z for the movement along the axes R (up/down) and Z (forward/back).
The laptop is connected to the machine via a LAN cable.

Lua Community Participation

- **We want to return our extensions of the Lua interpreter to the community.**
- **Our Lua Language Server adaptations are already integrated into the official release.**
- **We started and will continue publishing our approach in papers and talks:**
  Paper at VST2024 at SANER24, Finland:
  “Introducing a Linter in an Industrial Lua Code Base”



Albrecht Wöß | Signum+ Software
albrecht.woess@signum.plus

Markus Löberbauer | Signum+ Software
markus.loeberbauer@signum.plus
Georg Koll | TRUMPF Maschinen Austria
georg.koll@trumpf.com

2/28/2024