Rust: Async statt RTOS?

Philipp Bormuth // awinia GmbH

Agenda

- Was ist „Async“?
- Async & Embedded
- Embassy.rs
- Zahlen

Was ist „Async“

- Kooperatives Multitasking im „async/await“ Stil
- In vielen Sprachen (z.B. Python, C#) schon lange verfügbar
- Async Konzepte seit C++11 im C++ Standard
- Rust mit flexiblem Ansatz der verschiedene Runtimes/Eventloops erlaubt
  - Tokio
  - Embassy.rs

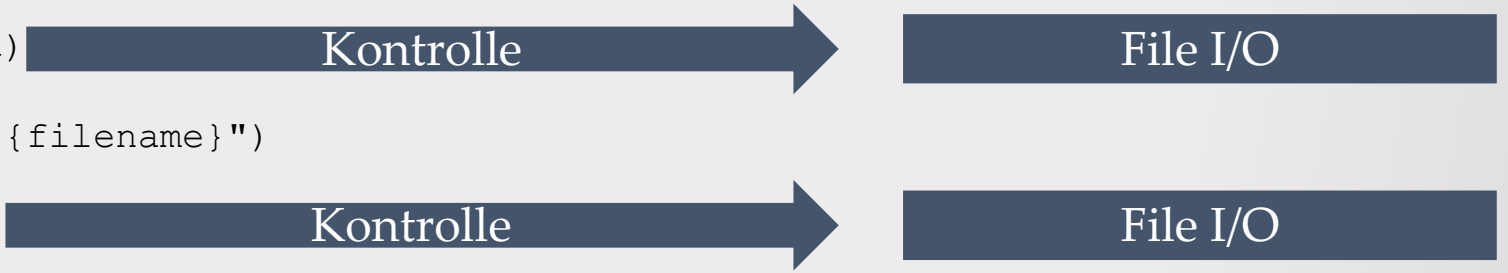
Scheduling Modell

- N „Tasks“
- 1 Eventloop
- Jeder Task läuft immer bis zum nächsten „await“
  - „await“: Task wird inaktiv
  - Aktivierung durch externes Ereignis (z.B. I/O, Socket, o.Ä.)
- Code wird i.d.R. in einen Zustandsautomaten übersetzt.
- Eventloop bekommt bei jedem „await“ die Kontrolle und führt Taskwechsel durch.

Task

Eventloop

```
async def write_to_file(filename, content):
    async with aiofiles.open(filename, 'w') as file:
        await file.write(content)
        print(f"Data written to {filename}")
        await file.write(„test“)
```



Plakatives Beispiel

```
TaskState MyTaskFunc(TaskState lastState)
{
  switch(lastState)
  {
     case TaskState::Begin:
       CodeForBeginState();
       return TaskState::Middle;
     case TaskState::Middle:
       CodeForMiddleState();
       return TaskState::End;
     case TaskState::End
       CodeForEndState();
       return TaskState::Finished
  }
}
```

```
async MyTaskFunc()
{
    await (CodeForBeginState());
    await (CodeForMiddleState());
    await (CodeForEndState());
}
```

Async & Embedded

- Kleine Systeme: Wenige aktive Tasks
- Pattern „stoße etwas an und warte auf Ergebnis“ ist gängig
  - Interrupts
  - Messagepassing
- In vielen Vendor SDKs sogar Teil der Schnittstelle mit completion Callbacks (z.B. Microchip, ST, TI)

Vorteile

- RAM Verbrauch: Nur ein einzelner Stack nötig
- ROM: Async Runtime deutlich kleiner als gängige RTOS
- Keine Races zwischen Tasks möglich (Interrupts sind eine andere Geschichte!)
- Stromverbrauch im Vergleich mit RTOS Systemen

Probleme

- Keine Preemption!
  - Busywaits nicht mehr möglich
  - Ein einzelner Task kann das ganze System anhalten
- Code ist sehr gewöhnungsbedürftig

Embassy.rs

- Async Runtime für Microcontrollersysteme
- Ports für
  - STM32
  - ESP
  - nRF
  - RP2040
- Neben Async auch noch jede Menge andere Features (USB, Eth…)

Embassy Beispiel

Zahlen*

Technologie	C/FreeRTOS	Rust/embassy	Unterschied in %	Unterschied absolut
Programmgröße (ROM)	12432	5280	-57%	-7152
Programmgröße (RAM)	4632	656	-85%	-3976

*nicht triviales Programm mit mehreren Tasks, das auch mit Hardwareperipherie interagiert

Fazit

- Asynchrone Programmierung als 3. Weg zwischen RTOS und Superloop
- Niedriger Ressourcenfootprint in RAM und ROM
- Bessere Wartbarkeit durch weniger „Noise“
- Bei harten Echtzeitanforderungen u.U. wenig passend
- Aktuelle Verfügbarkeit passender Libraries als limitierender Faktor

Newsflash: Certified toolchain

It's official: Ferrocene is ISO 26262 and IEC 61508 qualified!

Vielen Dank!

Philipp Bormuth
www.awinia.de