Teaching Embedded Systems with Tock

Alexandru Radovici
Politehnica University of Bucharest

Embedded Systems in Rust

we taught an embedded systems course fully in Rust

Students learned

- how hardware works
- how to actually build their own hardware device
- the Rust programming Language

We used

- the `embassy` framework
- `async` Rust
- Rust Embedded `async` HAL

90 second year **students** built **70 projects** using the Raspberry Pi Pico and Rust

Theory

- How a microprocessor works
- How the ARM Cortex-M processor works
- Using digital signals to control devices
- Using analog signals to read data from sensors
- How interrupts work
- How asynchronous programming works (async/await)
- How embedded operating systems work

Practical

- How to use the Raspberry Pi Pico
  - Affordable
  - Powerful processor
  - Good documentation
- How to program in Rust
  - Memory Safe
  - *Java-like features, without Java’s penalties*
  - Defines an embedded standard interface *embedded-hal*

The Good

what worked well

The Good

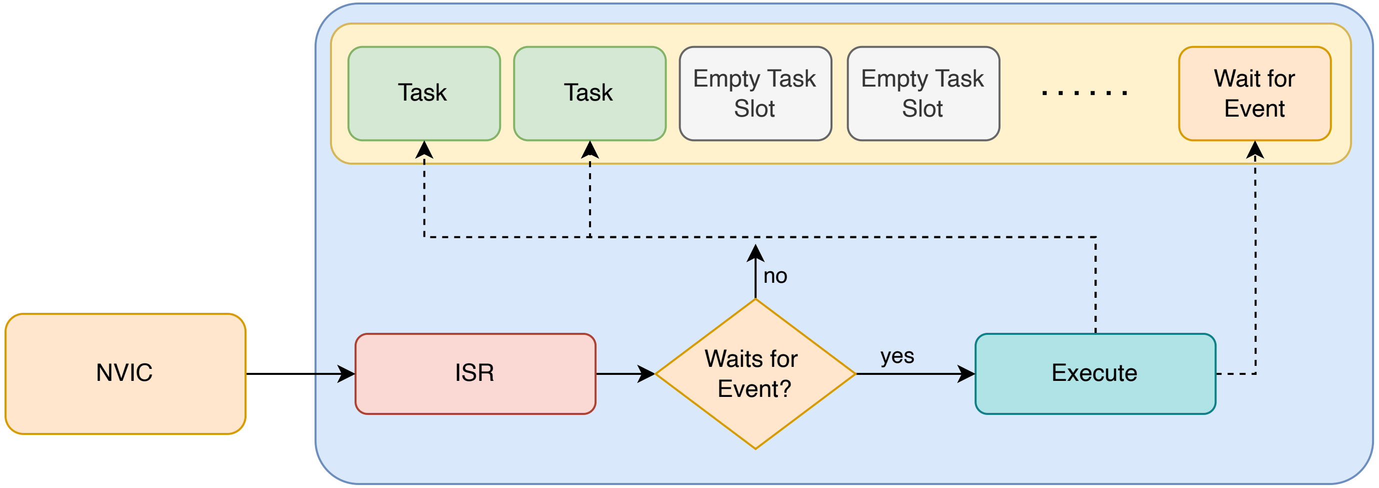
why did we use embassy

- `embassy` looks pretty simple to use
- the Raspberry Pi Pico is very well supported
  - has WiFi
- the *Rust Embedded HAL* is emplemented, in theory, students could you any crates
- allows the writing of *multi-threaded* applications
  - easier to do than writing state machines

`async`/`.await` worked out great

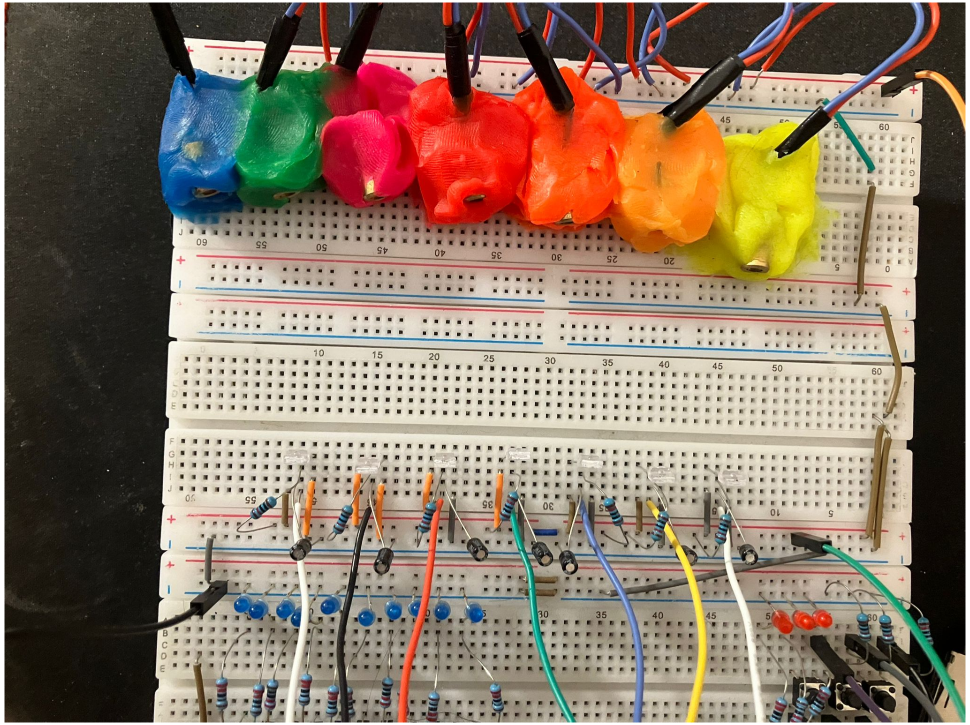
some say *do not use `async`/`.await` for beginners

- initially told students to just write `.await` at the end
- explained how `async` Rust works



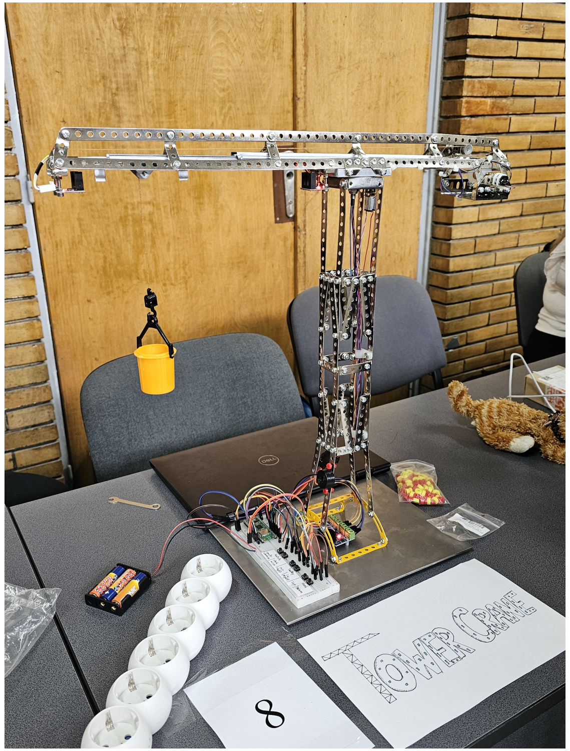
Laser Piano

a piano with laser keys - project page



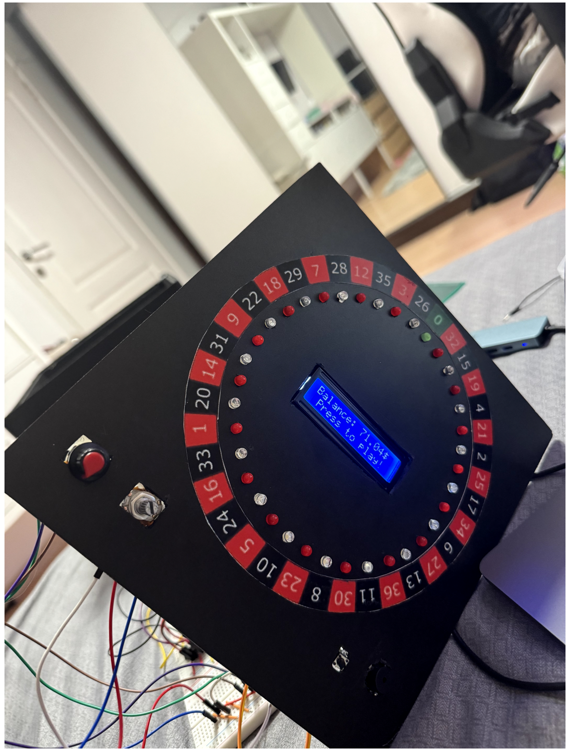
Tower Crane

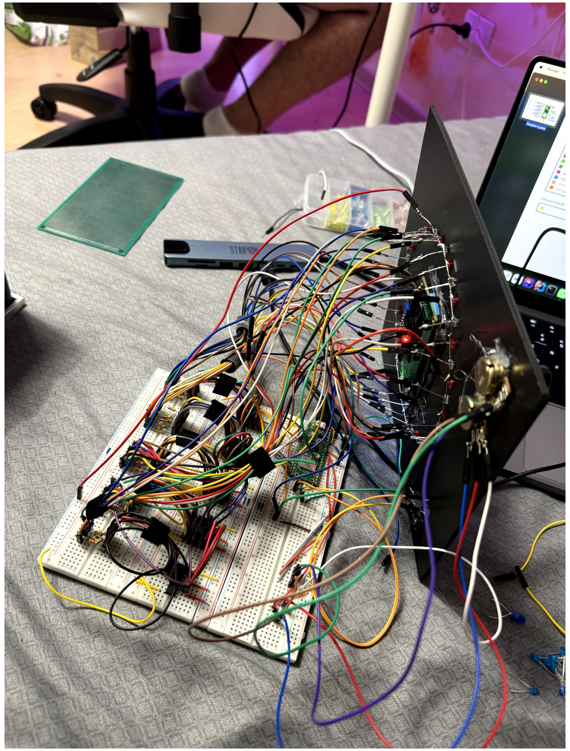
project page



Roulette

casino roulette - project page





Issues

these are some of the issues that we faced

`panic`

debugging was almost impossible

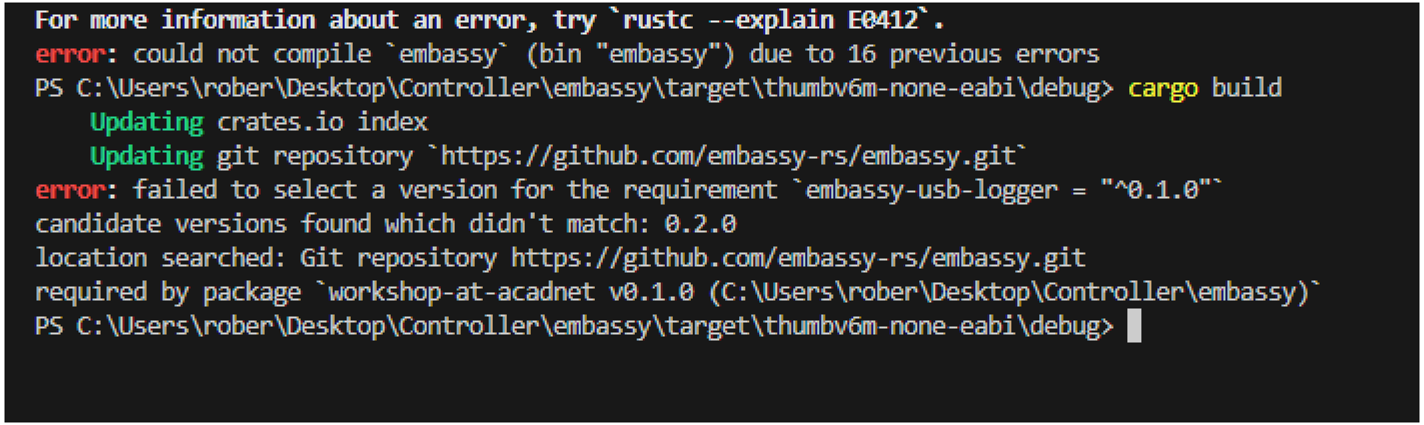
- RP2040 has no debugger (the debugger is more expensive than the actual chip)
- we used the USB logger for prints
- when `embassy` panics, everything stops, no output, maybe an LED blink
    - if RP2040 has WiFi, not even that, the LED is via SPI

No release plan

embassy is a *one person show*

there is no release plan

throughout the course, at some point, running `cargo build` *on the repo would fail due to version incompatibilities between embassy's own crates*



```
For more information about an error, try `rustc --explain E0412`.
error: could not compile `embassy` (bin "embassy") due to 16 previous errors
PS C:\Users\rober\Desktop\Controller\embassy\target\thumbv6m-none-eabi\debug> cargo build
    Updating crates.io index
    Updating git repository `https://github.com/embassy-rs/embassy.git`
error: failed to select a version for the requirement `embassy-usb-logger = "^0.1.0"`
candidate versions found which didn't match: 0.2.0
location searched: Git repository https://github.com/embassy-rs/embassy.git
required by package `workshop-at-acadnet v0.1.0 (C:\Users\rober\Desktop\Controller\embassy)`
PS C:\Users\rober\Desktop\Controller\embassy\target\thumbv6m-none-eabi\debug>
```

- talked to Dario, wrote `embassy` for himself, not sure he wants to fully support it

Breaking changes

with no major version increase

1. we sumitted a PR and renamed the `PWM_CHANNEL` to `PWM_SLICE`
   - got accepted immediatly
   - public doc changed immediatly
   - no major version increase
2. the `Pin` type changed throughout the semester
   - depeding on when students downloade embassy, they had to use it differently
   - libraries would fail
   - no git tag for the most recent working release
3. WiFi only worked with the git creates

Tock

use as main the tool embedded systems course teaching

Why Tock?

- *Applications* - it runs full applications that fault and print a debug message
  - over USB, a debugger is not needed
  - apps are simpler to write
- *OS Intrenals* - students cand easily understand the OS internals
  - it is easy to write a driver
- *Security* - easy way to introduce security in an embedded systems course
  - App IDs
  - System call filter
- *Development* - students can you several languages to write projects
- *No Dependencies* - there are not dependencies that break

TODOs

we have things to do to actually use Tock

Connectivity

support for WiFi/Ethernet mostly

Work in progress

- Arduino Nano RP2040 #2625
- Ethernet for STM32 #3695
- PacketBuffers (Amalia)

TODOs

- Port the RP2040 WiFi Driver to Tock
- TCP/IP stack implementation
  - smoltcp in userspace
  - smoltcp in the kernel

Thread is not great is you do need gateways that students do not have at home

Fix the USB stack

The USB stack is broken, at least serial port jams frequently

Issues

- the issue #4011
- refactor the USB stack

TODOs

- add the USB IAD for MS Windows
- document how the stack works

Configurator

The `main.rs` file is way to complicated, a `menuconfig` like system would be great

Work in progress

- Write a configurator (OxidOS / Irina)
- Tweedegolf is happy to help

TODOs

- a lot of feedback is needed

Configurator Demo

The `main.rs` file is way to complicated, a `menuconfig` like system would be great

`async` / `.await` support for libtock-rs

it is easier to write asynchronous apps

Work in progress

- add Tock as a backend to `embassy-executor`
- define `async` APIs in `libtock-rs` #494

TODOs

- might be tricky to add async, due to the way in which `scope` works

Support the Rust Embedded HAL

so that users can add libraries to their applications

Done

- Embedded HAL #540

TODOs

- implement the full embedded HAL

Userspace drivers

safely expose devices to userspace

Work in progress

- Device Passthrough #4020
- Stub out device pass through support #4044

TODOs

- define some special API?

Windows support

- using VMs for Tock is difficult due to bad support fom VM providers
  - VMWare Workstation might not be available
  - VirtualBox gets stuck
  - WSL2 has an issue with mapping USB ports

TODOs

- add support for building Tock in Windows
- use probe-rs to replace *openocd* and *JLink*
- linker scripts might be problematic

Dev board Kit

everyone has different hardware platforms

Requirements

- be able to build it with off-the-shelf components
- cost under $50
- debugger!

Work in progress

- lab board
    - RP2040 as a debuger
    - Pico W SMD mounted (cheaper than bying the components)
    - buttons, LEDs, screen, buzzer and extension sockets

Conclusion

Tock could be the standard for embedded systems courses

- There is a lof of work to do
- We have 5 interns for the summer that will work on this
- Try to teach common courses or at least parts of them

x86 port?

we want to use it in a OS design course

- 4th year
- mostly a driver design course

@microsoft:

- How fast can be this upstreamed?
- Can we help to speed this up?