Safer, Simpler Embedded Programs with Rust on RIOT

Lup Yuen LEE

github.com/lupyuen
The year is 2020. Our story is about a learner (“Padawan”) and a teacher (“Sensei”).

Padawan: I want to make a watch face that looks like this. Can you guide me Sensei?

Sensei: That's easy. Here's a watch face.... Take this program and change it.

Padawan knows Arduino and tries to change the program...

```c
// Create a buffer on the stack
char buffer[6];

// Format the time
sprintf(buffer, "%02d:%02d", hour, minute);

// Set the LVGL label
lv_label_set_text(label, buffer);
```
Padawan programmed all night... But failed. The next day...

*Padawan*: Sorry Sensei, the program is acting really strange. I only added two newlines “\n\n” like this...

```c
// Create a buffer on the stack
char buffer[6];

// Format the date and time
sprintf(buffer, "%02d\n\n%02d", hours, minutes);
```

Sensei sighs.
It's 2020... There must be a better way to learn Embedded Programming....
Sensei: What we have here is a Buffer Overflow problem. Do you know what that is?

Padawan: Not really... May I ask some questions?

Padawan started asking many, many questions...

// Create a buffer on the stack
char buffer[6];

// Format the date and time
sprintf(buffer, "%02d\n%02d", hours, minutes);

Sensei wondered... How did a simple watch face... Become so complicated?
Sensei: Tell you what... Let's code this the Safer way with "snprintf"

```c
// Create a buffer on the stack
char buffer[64];

// Format the time
snprintf(buffer, sizeof(buffer), "%02d\n\n%02d", hour, minute);
```

Padawan: What's "snprintf"?

Sensei: Well to format something for printing we call "printf"...
   To format something into a string buffer we call "sprintf"...
   To format something into a string buffer limited by size we call "snprintf"...
   And to get the size of the string buffer we call "sizeof"...
(Silence)

Sensei: Are you still there, Padawan?

Padawan has slipped away to play Fortnite...
Never returning to Embedded Programming... Ever again!

Whose fault is it? Sensei’s fault of course!

Sensei failed to provide a safe and sensible environment for learners to experiment with Embedded Programming...

It’s A Trap!
DILEMMA FACING KIDS TODAY

1. SHALL I INVEST MY TIME IN LEARNING FORTNITE...

- BACK BANG
- UGH I'M DEAD!
- YAY GETTING BETTER
- BACK BANG
- UGH DEAD AGAIN
- BUT I'LL KEEP TRYING!

2. OR LEARN EMBEDDED PROGRAMMING?

- GOT CODE
- FLASH TO DEVICE
- OOPS CRASH WHY WHY
- I THINK I KNOW WHY
- FLASH TO DEVICE
- CRASH AGAIN!

BAD POINTER?

I GIVE UP!!!
We Need A Scaffold

... A Scaffold that prevents Padawans from falling into traps and never recovering

... Guide the learner towards difficult topics

... But feed them the skills one small chunk at a time

**Instructional scaffolding**

From Wikipedia, the free encyclopedia

*Instructional scaffolding* is the support given to a student by an instructor throughout the learning process. This support is specifically tailored to each student; this instructional approach allows students to experience student-centered learning, which tends to facilitate more efficient learning than teacher-centered learning. This learning process promotes a deeper level of learning than many other common teaching strategies.
Consider This Rust Scaffold

Mutable variables must be declared “mut”

And must be passed as “mut”

Rust works with LVGL and other C libraries

/// Create a buffer on the stack
let mut buffer = new_string();

/// Format the time
write!(
    &mut buffer,
    "{:02}:{:02}\0", // Terminate with null
    hour,
    minute
).expect("time fail");

/// Set the LVGL label
label::set_text(
    time_label,
    &to_strn( &buffer )
) ? ; // In case of error, return the error

Rust infers the types of our variables

"write!" is a macro that checks the type of each parameter

In case of overflow, program halts with an error “time fail”

Mandatory error checking with “?”

github.com/AppKaki/lvgl-wasm/blob/rust/rust/app/src/watch_face.rs
Watch Face: C vs Rust

// Create a buffer on the stack
char buffer[6];

// Format the time
sprintf(
    buffer,
    "%02d:%02d",
    hour,
    minute
);

// Set the LVGL label
lv_label_set_text(
    label,
    buffer
);

// Create a buffer on the stack
let mut buffer = new_string();

// Format the time
write!(
    &mut buffer,
    "{:02}:{:02}\0", // Terminate with null
    hour,
    minute
).expect("time fail");

// Set the LVGL label
label::set_text(
    label,
    &to_strn( &buffer )
) ? ; // In case of error, return the error
Safer Rust

Rust can detect subtle code safety issues... That most C coders won't notice

Uh-oh... Rust senses that the external C function “set_text” may have safety issues...

The buffer lives in the stack. If “set_text” saves the buffer pointer for future access, this program may crash!

We solve this by creating a static mutable buffer... Which extends its Lifetime

But static mutable buffers are inherently unsafe... What if two threads try to update the same buffer?

Thus we need to flag the code as “unsafe”... And ensure that the buffer is used by only one RIOT thread

github.com/AppKaki/lvgl-wasm/blob/rust/rust/app/src/watch_face.rs
Rust on RIOT for PineTime Smart Watch

github.com/lupyuen/pinetime-rust-riot
Forked from Koen Zandberg: github.com/bosmoment/PineTime-apps

Why RIOT?

- Modern Embedded OS
- Strong Friendly Community
- Freedom to Innovate

We Need Your Help To Grow Rust On RIOT!

... Because many Padawans are waiting
WebAssembly Simulator for Rust on RIOT

github.com/AppKaki/lvgl-wasm/tree/rust

Star Trek has a Holodeck...
We have a WebAssembly Simulator to keep Padawans engaged

- Watch Face code (Rust) runs in a Web Browser
- Build in the cloud with GitHub Actions
- Great for learning and iterative development
WebAssembly Simulator for Rust on RIOT

Online Demo
appkaki.github.io/lvgl-wasm/rust.html

Source Code
github.com/AppKaki/lvgl-wasm/tree/rust
Simplify Embedded Programs with Rust on RIOT

github.com/AppKaki/druid-lvgl

Can we create watch faces with a Rust Declarative UI like Druid?

```rust
// Create a label for time (00:00)
let label_time = label::create(scr, ptr::null()) ? ;
label::set_long_mode(label_time, label::LV_LABEL_LONG_BREAK) ? ;
obj::set_width(label_time, 240) ? ;
obj::set_height(label_time, 200) ? ;
label::set_align(label_time, label::LV_LABEL_ALIGN_CENTER) ? ;
obj::align(label_time, scr, obj::LV_ALIGN_CENTER, 0, -30) ? ;

// Create a label for Date
let label_date = label::create(scr, ptr::null()) ? ;
label::set_long_mode(label_date, label::LV_LABEL_LONG_BREAK) ? ;
obj::set_width(label_date, 200) ? ;
obj::set_height(label_date, 200) ? ;
label::set_align(label_date, label::LV_LABEL_ALIGN_CENTER) ? ;
obj::align(label_date, scr, obj::LV_ALIGN_CENTER, 0, 40) ? ;
```

Flex::row()

.with_flex_child(Flex::column()

.with_flex_child(label_time, 1.0)

1.0

)

1.0

.with_flex_child(Flex::column()

.label_date, 1.0

)

1.0
Visual Embedded Rust

github.com/lupyuen/visual-embedded-rust

Drag and drop to create watch apps with Declarative UI

VSCode Extension with Druid + Blockly (Scratch)
Simplify Embedded Programs with Rust on RIOT

Let's create a watch app...

To make sure my family members (and my pet) don’t wander off too far away...

Perfect for Bluetooth Mesh with RIOT and NimBLE!
Unfortunately it takes 2,700 lines of C code…

To create a simple Bluetooth Mesh app

Can Rust on RIOT simplify this?

(Maybe with a Domain-Specific Language?)
CREATE YOUR OWN CUSTOM PINETIME Firmware

Hello to new Pinetime owners.

Select the operating system:
- ATMwatch
- PebbleOS
- Mynewlt
- RIOT
- WatchOS
- Zephyr

Upload custom boot logo

Select language

Select watch face

Choose apps

GitHub Actions workflow... in the cloud

Build operating system

Integrate boot logo

Add fonts

Link watch face

Link apps

Optimise

Security scan

Pinetime firmware gets rebuilt & reflashed when there are security updates.

Flashed to blue tooth

Custom Pinetime firmware
What’s “The New Normal” for IoT Development?

- Harder to get hardware in many parts of the world outside Asia
- We may need to build and test on Simulators... And verify on real hardware remotely
- Great time to rethink and reconstruct the way we teach IoT to a new generation of distracted learners

Will Rust on RIOT save our Padawan? Perhaps!

Shipping these Pogo Pins from Singapore to US now costs $100
Extra Slides
Rust on RIOT & Rust Embedded Complete Each Other

Two Complementary Approaches to Rustification: Top Down vs Bottom Up

- Start with Apps vs Start with Bare Metal Drivers
- One day the two shall meet... And we shall have a complete Rust Stack yay!
- If the two don’t meet... Then we shall have TWO complete Rust Stacks yay!