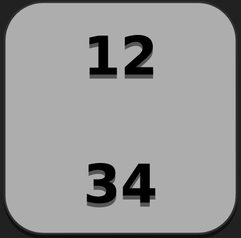
# Safer, Simpler Embedded Programs with Rust on RIOT

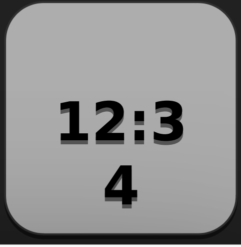
Lup Yuen LEE
github.com/lupyuen

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?

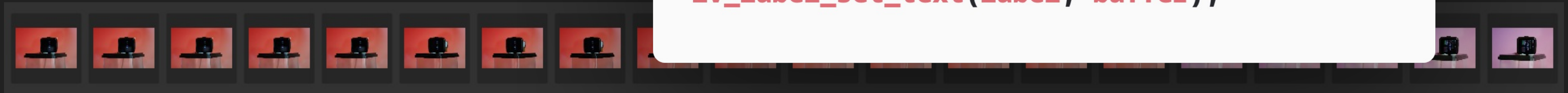


*Sensei*: That's easy. Here's a watch face.... Take this program and change it.



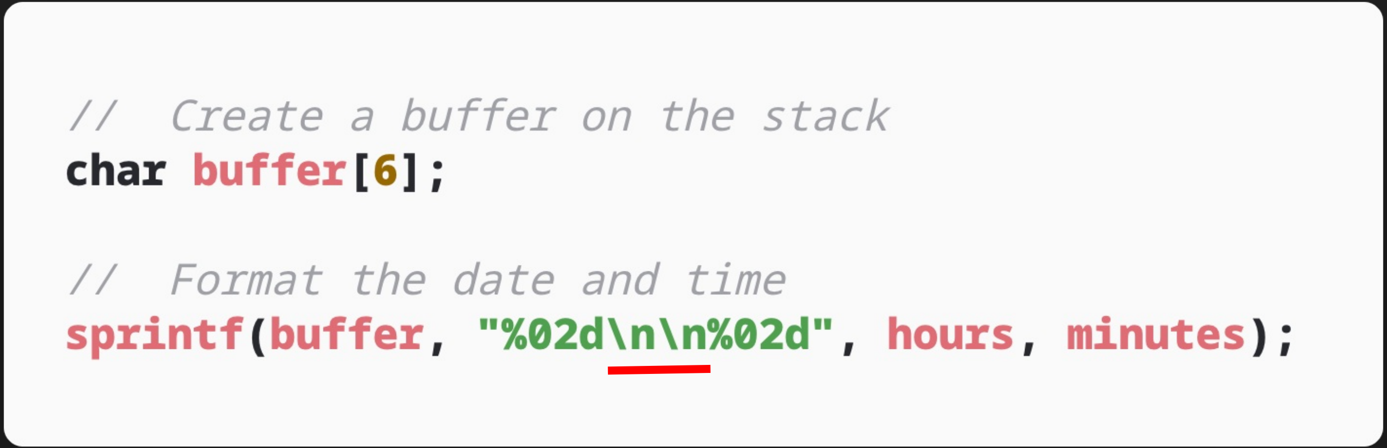
```
//  Create a buffer on the stack
char buffer[6];

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

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

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



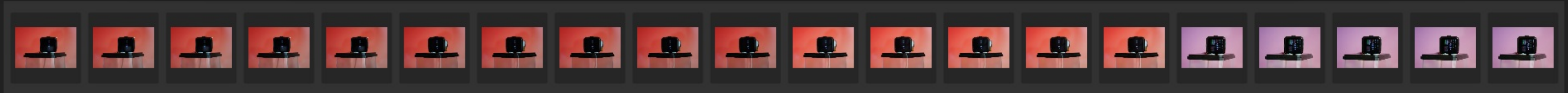
Padawan programmed all night... But failed. The next day...

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



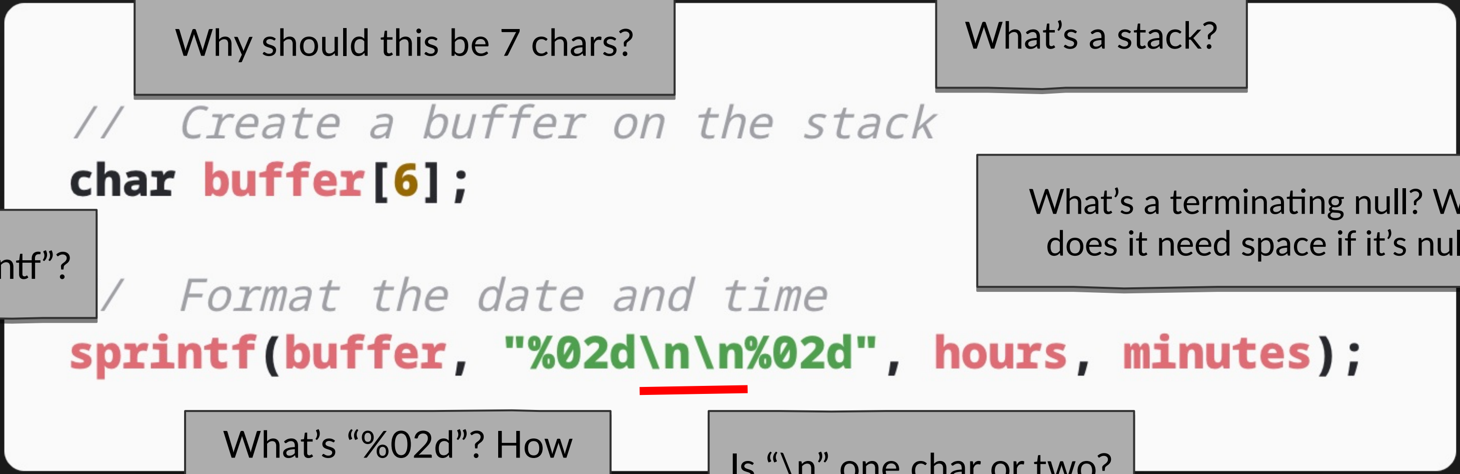
```
//  Create a buffer on the stack
char buffer[6];

//  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...



Why should this be 7 chars?

What's a stack?

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

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

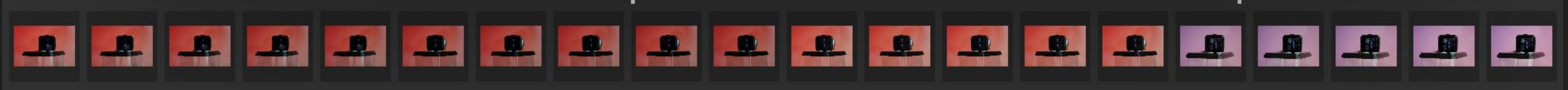
What's a terminating null? Why does it need space if it's null?

What's "sprintf"?

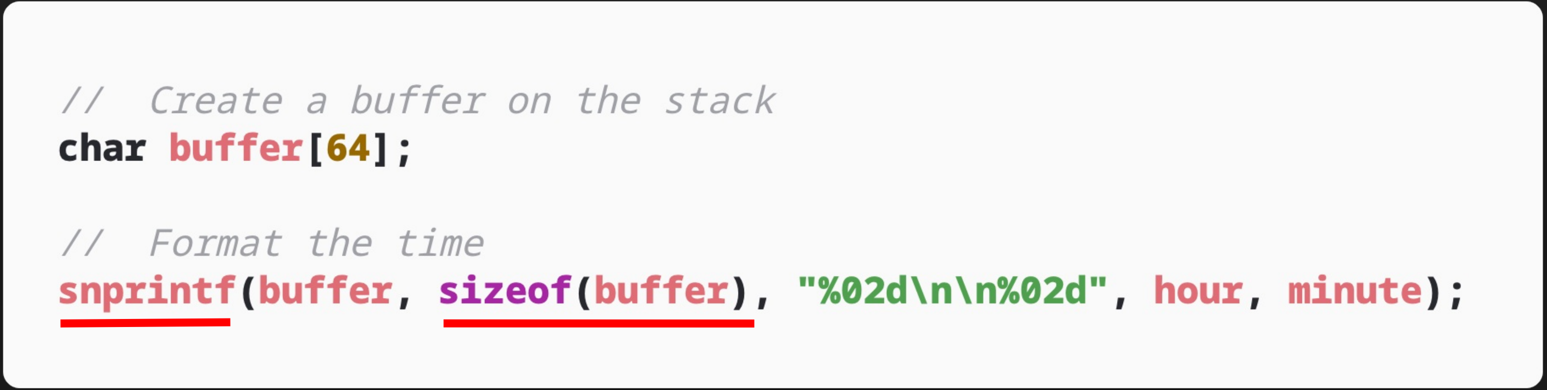
What's "%02d"? How many chars is that?

Is "\n" one char or two?

Sensei wondered... How did a simple watch face... Become so complicated?



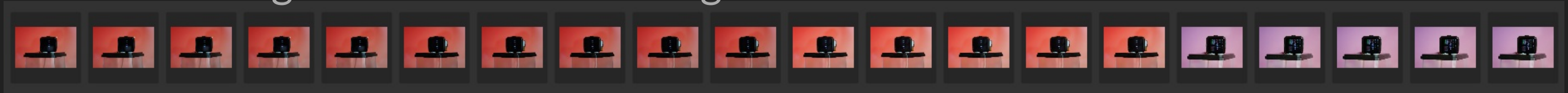
*Sensei*: Tell you what… Let’s code this the Safer way with “snprintf”



```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)*

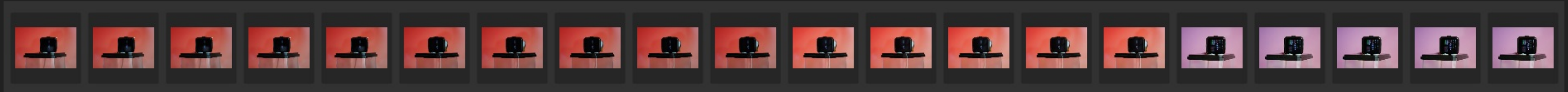
*Sensei*: Are you still there, Padawan?

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

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

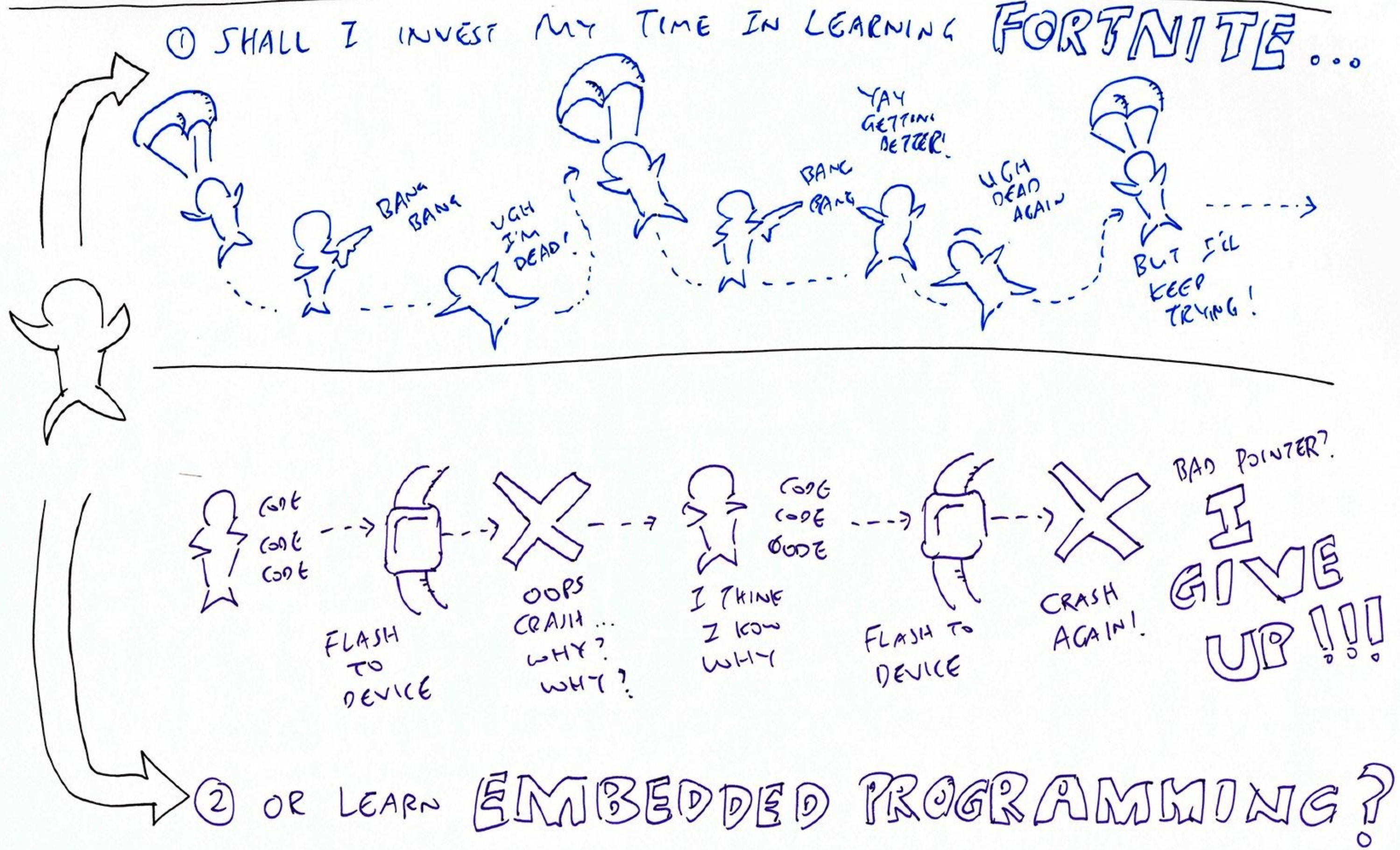
Sensei failed to provide a safe and sensible environment for learners to experiment with Embedded Programming…

*It's A Trap!*

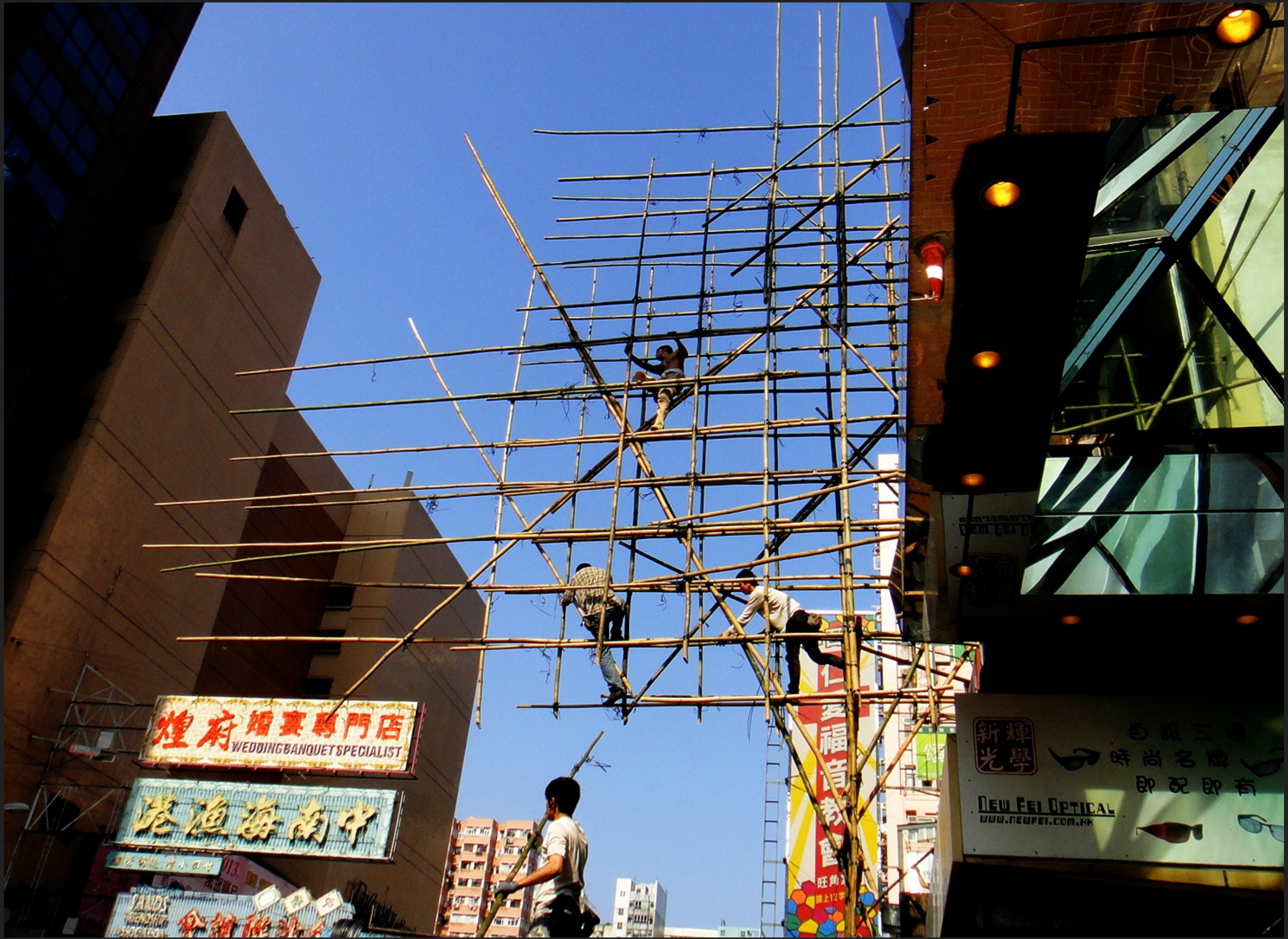


# DILEMMA FACING KIDS TODAY

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



② OR LEARN EMBEDDED PROGRAMMING?



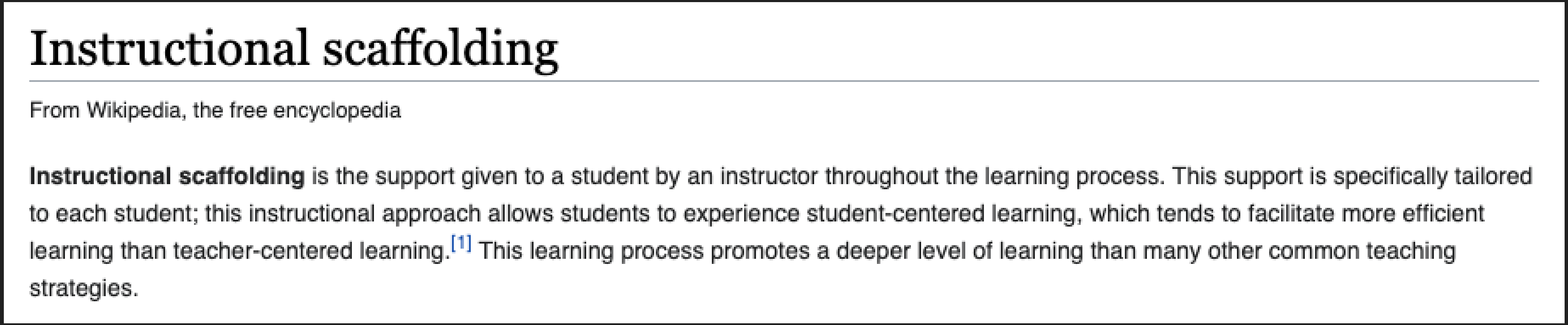
en.wikipedia.org/wiki/Scaffolding#/media/File:Usage_of_Bamboo.JPG

## We Need A Scaffold

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

… Guide the learner towards difficult topics

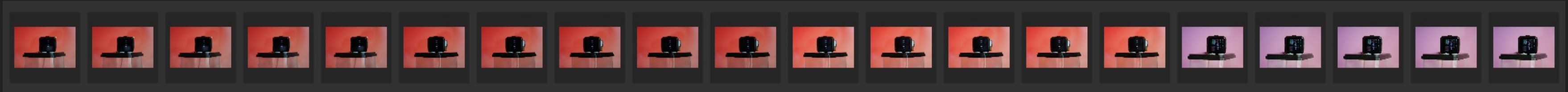
… But feed them the skills one small chunk at a time



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.[1] 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
```

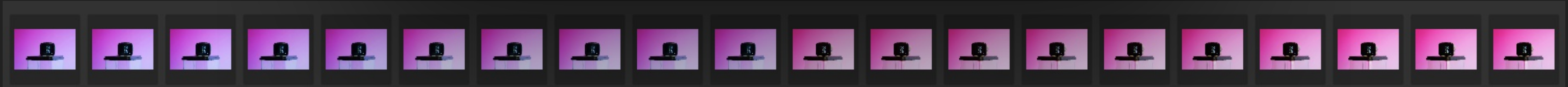
github.com/AppKaki/lvgl-wasm/blob/rust/rust/app/src/watch_face.rs

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”

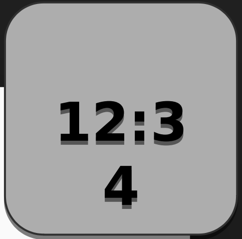
Mandatory error checking with “?”

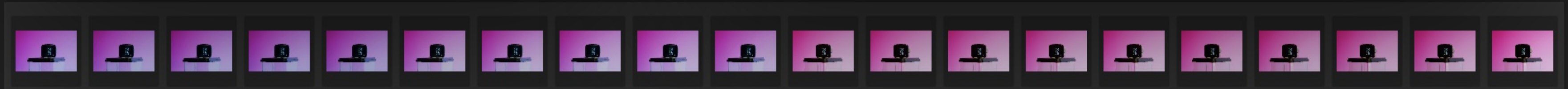


## Watch Face: C vs Rust

```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
);
```

```rust
//  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!

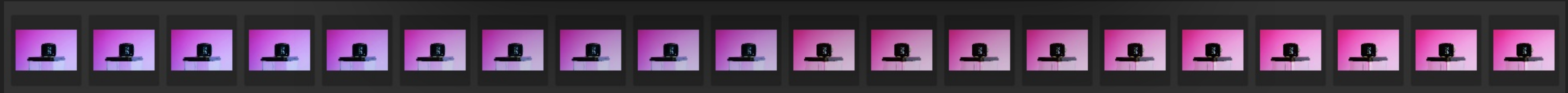
We solve this by creating a static mutable buffer… Which extends its Lifetime

```
//  Create a static mutable buffer
static mut BUFFER: String = new_string();

//  Unsafe because BUFFER is a mutable static
unsafe {
    //  Erase the buffer
    BUFFER.clear();

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

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

github.com/AppKaki/lvgl-wasm/blob/rust/rust/app/src/watch_face.rs

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



# Rust on RIOT for PineTime Smart Watch

github.com/lupyuen/pinetime-rust-riot

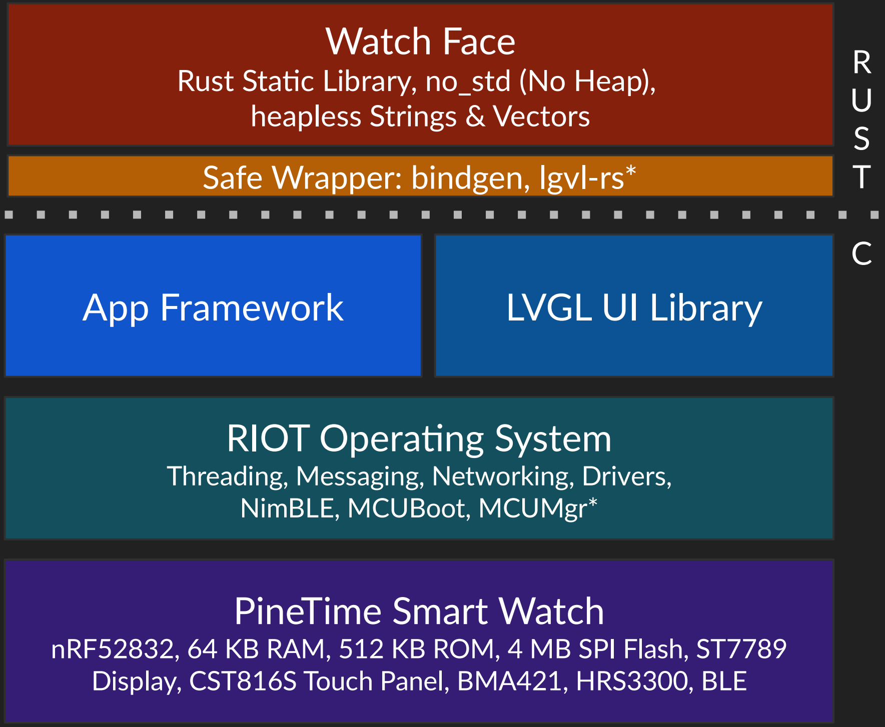
Forked from Koen Zandberg: github.com/bosmoment/PineTime-apps

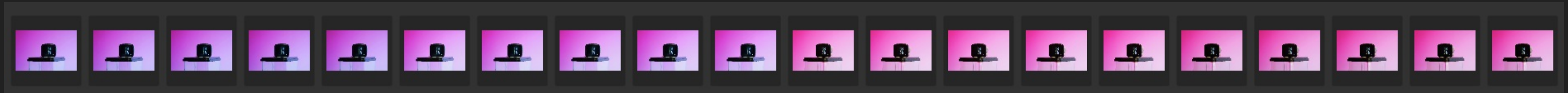
Why RIOT?

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

*We Need Your Help To Grow Rust On RIOT!*

*… Because many Padawans are waiting*


Watch Face
Rust Static Library, no_std (No Heap), heapless Strings & Vectors
Safe Wrapper: bindgen, lgvl-rs*
RUST
C
App Framework
LVGL UI Library
RIOT Operating System
Threading, Messaging, Networking, Drivers, NimBLE, MCUBoot, MCUMgr*
PineTime Smart Watch
nRF52832, 64 KB RAM, 512 KB ROM, 4 MB SPI Flash, ST7789 Display, CST816S Touch Panel, BMA421, HRS3300, BLE

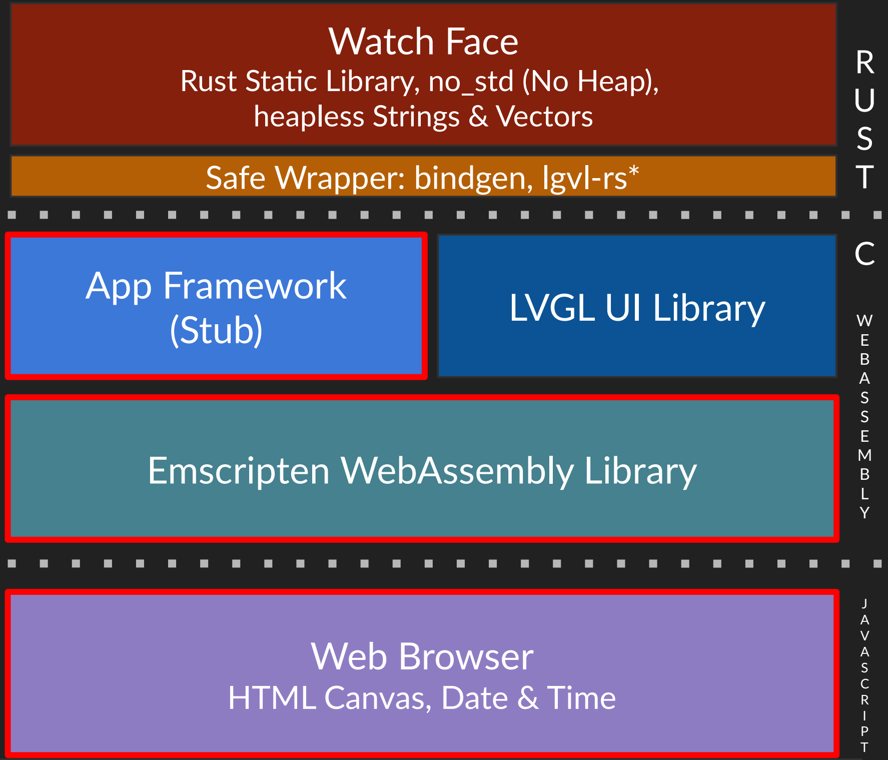


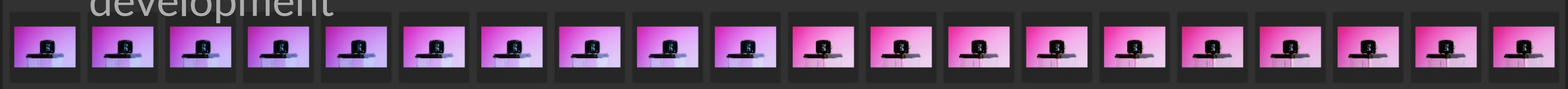
# WebAssembly Simulator for Rust on RIOT

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


Watch Face
Rust Static Library, no_std (No Heap), heapless Strings & Vectors
Safe Wrapper: bindgen, lgvl-rs*
RUST
App Framework (Stub)
LVGL UI Library
C
WEBASSEMBLY
Emscripten WebAssembly Library
Web Browser
HTML Canvas, Date & Time
JAVASCRIPT



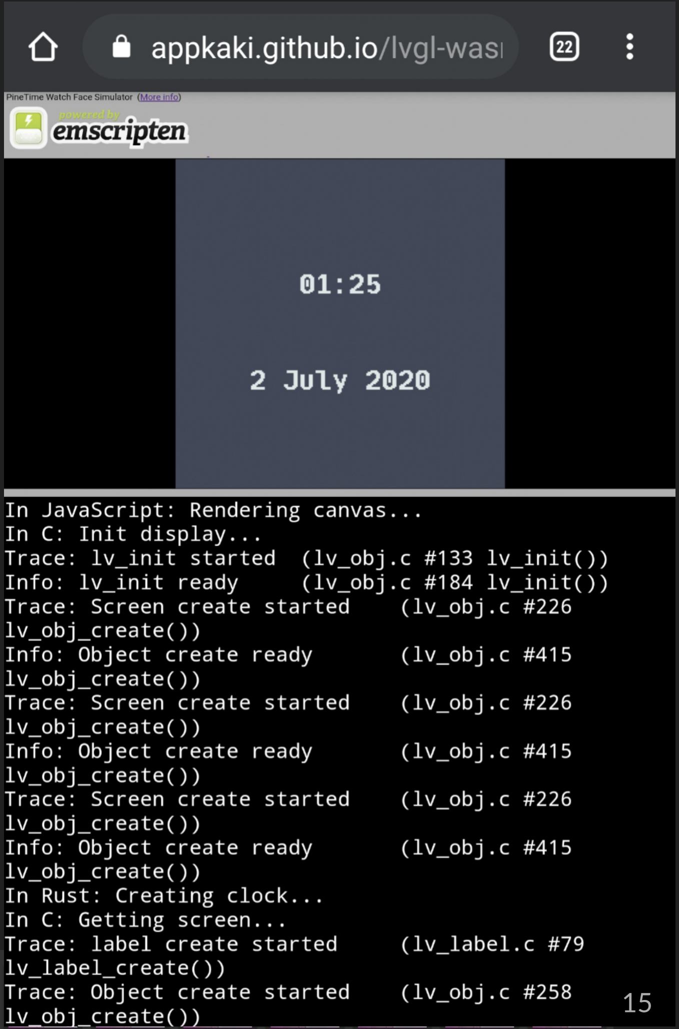
# WebAssembly Simulator for Rust on RIOT

Online Demo

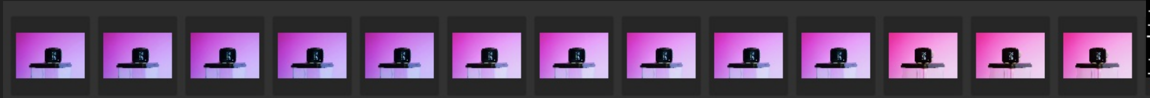
appkaki.github.io/lvgl-wasm/rust.html

Source Code

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


appkaki.github.io/lvgl-wasi
22
PineTime Watch Face Simulator (More info)
powered by emscripten
01:25
2 July 2020

```
In JavaScript: Rendering canvas...
In C: Init display...
Trace: lv_init started  (lv_obj.c #133 lv_init())
Info: lv_init ready     (lv_obj.c #184 lv_init())
Trace: Screen create started    (lv_obj.c #226
lv_obj_create())
Info: Object create ready       (lv_obj.c #415
lv_obj_create())
Trace: Screen create started    (lv_obj.c #226
lv_obj_create())
Info: Object create ready       (lv_obj.c #415
lv_obj_create())
Trace: Screen create started    (lv_obj.c #226
lv_obj_create())
Info: Object create ready       (lv_obj.c #415
lv_obj_create())
In Rust: Creating clock...
In C: Getting screen...
Trace: label create started     (lv_label.c #79
lv_label_create())
Trace: Object create started    (lv_obj.c #258
lv_obj_create())
```



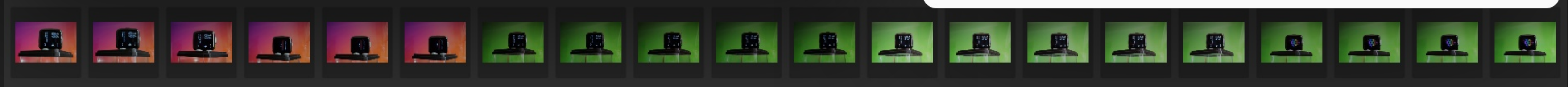
## Simplify Embedded Programs with Rust on RIOT

github.com/AppKaki/druid-lvgl

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

```
//  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
    )
    .with_flex_child(
        Flex::column()
            .with_flex_child(
                label_date, 1.0
            ),
        1.0
    )
```

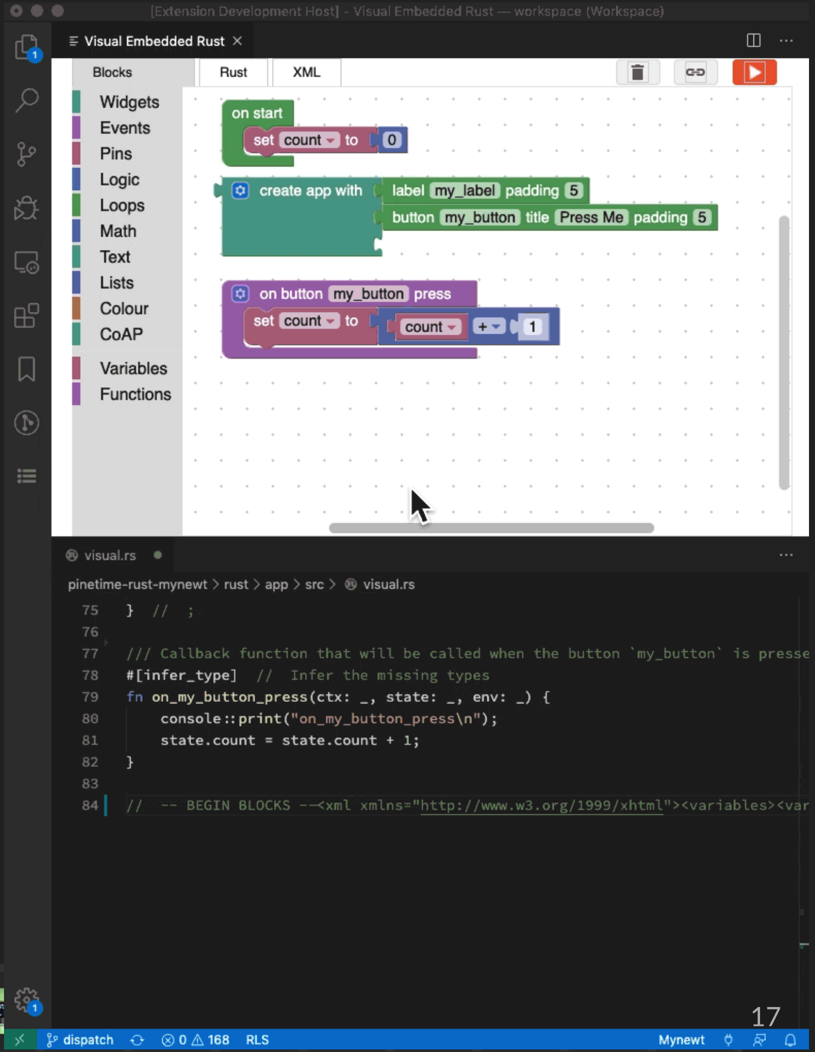


# Visual Embedded Rust

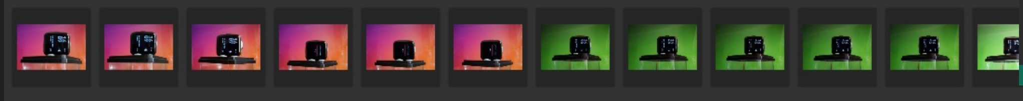
github.com/lupyuen/visual-embedded-rust

Drag and drop to create watch apps with Declarative UI

VSCode Extension with Druid + Blockly (Scratch)



```
} //  ;

/// Callback function that will be called when the button `my_button` is presse
#[infer_type]  //  Infer the missing types
fn on_my_button_press(ctx: _, state: _, env: _) {
    console::print("on_my_button_press\n");
    state.count = state.count + 1;
}

//  -- BEGIN BLOCKS --<xml xmlns="http://www.w3.org/1999/xhtml"><variables><var
```

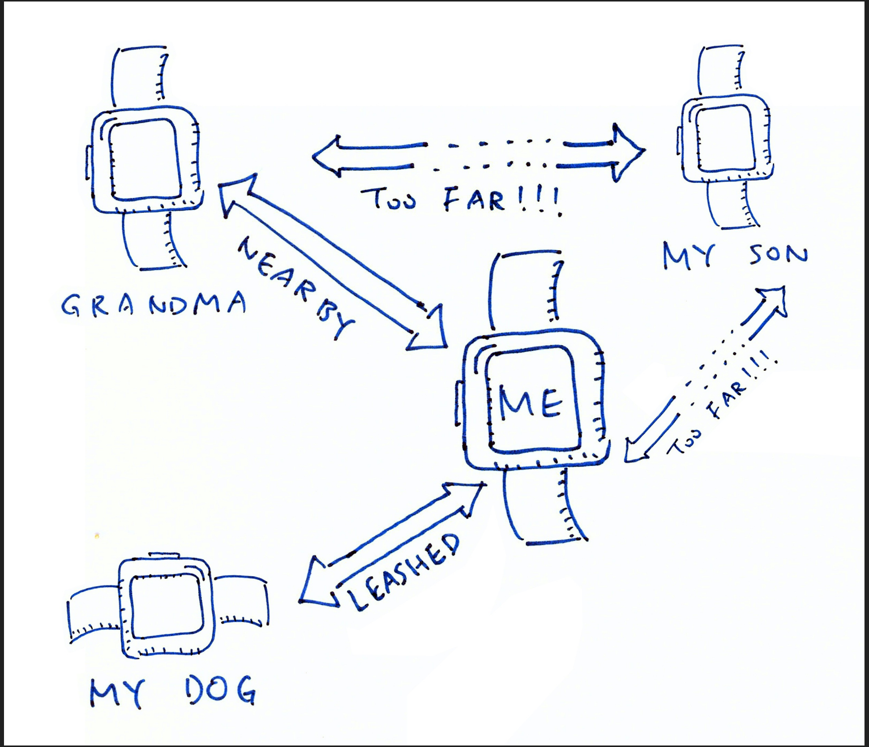


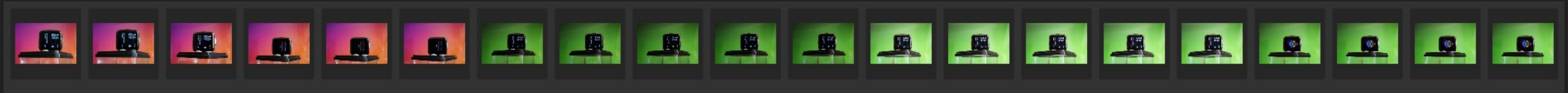
# Simplify Embedded Programs with Rust on RIOT

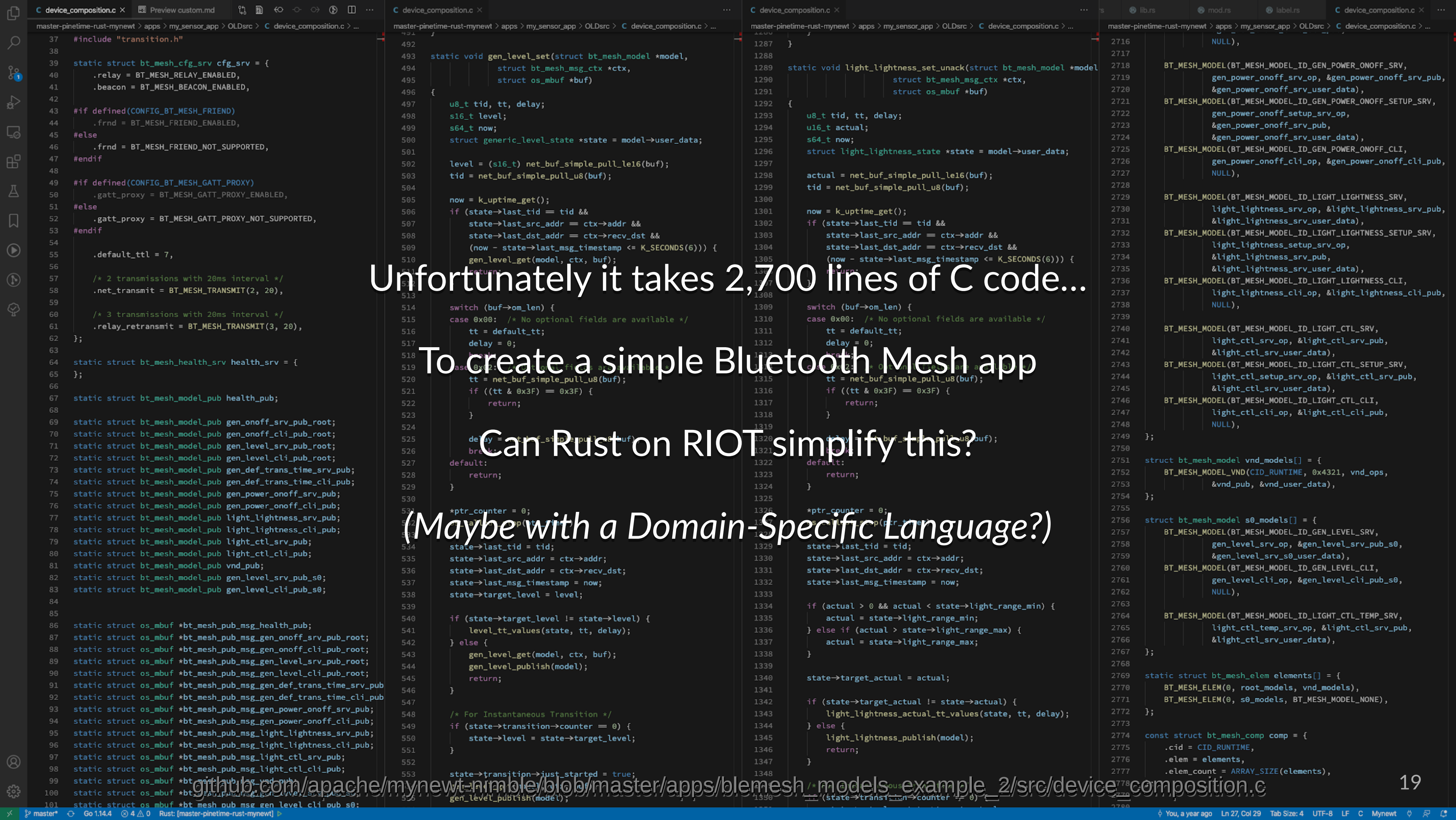
Let's create a watch app…

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

Perfect for Bluetooth Mesh
with RIOT and NimBLE!







```c
#include "transition.h"

static struct bt_mesh_cfg_srv cfg_srv = {
    .relay = BT_MESH_RELAY_ENABLED,
    .beacon = BT_MESH_BEACON_ENABLED,

#if defined(CONFIG_BT_MESH_FRIEND)
    .frnd = BT_MESH_FRIEND_ENABLED,
#else
    .frnd = BT_MESH_FRIEND_NOT_SUPPORTED,
#endif

#if defined(CONFIG_BT_MESH_GATT_PROXY)
    .gatt_proxy = BT_MESH_GATT_PROXY_ENABLED,
#else
    .gatt_proxy = BT_MESH_GATT_PROXY_NOT_SUPPORTED,
#endif

    .default_ttl = 7,

    /* 2 transmissions with 20ms interval */
    .net_transmit = BT_MESH_TRANSMIT(2, 20),

    /* 3 transmissions with 20ms interval */
    .relay_retransmit = BT_MESH_TRANSMIT(3, 20),
};

static struct bt_mesh_health_srv health_srv = {
};

static struct bt_mesh_model_pub health_pub;

static struct bt_mesh_model_pub gen_onoff_srv_pub_root;
static struct bt_mesh_model_pub gen_onoff_cli_pub_root;
static struct bt_mesh_model_pub gen_level_srv_pub_root;
static struct bt_mesh_model_pub gen_level_cli_pub_root;
static struct bt_mesh_model_pub gen_def_trans_time_srv_pub;
static struct bt_mesh_model_pub gen_def_trans_time_cli_pub;
static struct bt_mesh_model_pub gen_power_onoff_srv_pub;
static struct bt_mesh_model_pub gen_power_onoff_cli_pub;
static struct bt_mesh_model_pub light_lightness_srv_pub;
static struct bt_mesh_model_pub light_lightness_cli_pub;
static struct bt_mesh_model_pub light_ctl_srv_pub;
static struct bt_mesh_model_pub light_ctl_cli_pub;
static struct bt_mesh_model_pub vnd_pub;
static struct bt_mesh_model_pub gen_level_srv_pub_s0;
static struct bt_mesh_model_pub gen_level_cli_pub_s0;


static struct os_mbuf *bt_mesh_pub_msg_health_pub;
static struct os_mbuf *bt_mesh_pub_msg_gen_onoff_srv_pub_root;
static struct os_mbuf *bt_mesh_pub_msg_gen_onoff_cli_pub_root;
static struct os_mbuf *bt_mesh_pub_msg_gen_level_srv_pub_root;
static struct os_mbuf *bt_mesh_pub_msg_gen_level_cli_pub_root;
static struct os_mbuf *bt_mesh_pub_msg_gen_def_trans_time_srv_pub
static struct os_mbuf *bt_mesh_pub_msg_gen_def_trans_time_cli_pub
static struct os_mbuf *bt_mesh_pub_msg_gen_power_onoff_srv_pub;
static struct os_mbuf *bt_mesh_pub_msg_gen_power_onoff_cli_pub;
static struct os_mbuf *bt_mesh_pub_msg_light_lightness_srv_pub;
static struct os_mbuf *bt_mesh_pub_msg_light_lightness_cli_pub;
static struct os_mbuf *bt_mesh_pub_msg_light_ctl_srv_pub;
static struct os_mbuf *bt_mesh_pub_msg_light_ctl_cli_pub;
static struct os_mbuf *bt_mesh_pub_msg_vnd_pub;
static struct os_mbuf *bt[illegible]
static struct os_mbuf *bt_mesh_pub_msg_gen_level_cli_pub_s0;
```

```c

static void gen_level_set(struct bt_mesh_model *model,
                          struct bt_mesh_msg_ctx *ctx,
                          struct os_mbuf *buf)
{
    u8_t tid, tt, delay;
    s16_t level;
    s64_t now;
    struct generic_level_state *state = model->user_data;

    level = (s16_t) net_buf_simple_pull_le16(buf);
    tid = net_buf_simple_pull_u8(buf);

    now = k_uptime_get();
    if (state->last_tid == tid &&
        state->last_src_addr == ctx->addr &&
        state->last_dst_addr == ctx->recv_dst &&
        (now - state->last_msg_timestamp <= K_SECONDS(6))) {
        gen_level_get(model, ctx, buf);
        return;
    [illegible]

    switch (buf->om_len) {
    case 0x00:  /* No optional fields are available */
        tt = default_tt;
        delay = 0;
        [illegible]
    [illegible]
        tt = net_buf_simple_pull_u8(buf);
        if ((tt & 0x3F) == 0x3F) {
            return;
        }

        delay = [illegible]
        bre[illegible]
    default:
        return;
    }

    *ptr_counter = 0;
    [illegible]
    state->last_tid = tid;
    state->last_src_addr = ctx->addr;
    state->last_dst_addr = ctx->recv_dst;
    state->last_msg_timestamp = now;
    state->target_level = level;

    if (state->target_level != state->level) {
        level_tt_values(state, tt, delay);
    } else {
        gen_level_get(model, ctx, buf);
        gen_level_publish(model);
        return;
    }

    /* For Instantaneous Transition */
    if (state->transition->counter == 0) {
        state->level = state->target_level;
    }

    state->transition->just_started = true;
    [illegible]
    gen_level_publish(model);
```

```c
}

static void light_lightness_set_unack(struct bt_mesh_model *model
                                      struct bt_mesh_msg_ctx *ctx,
                                      struct os_mbuf *buf)
{
    u8_t tid, tt, delay;
    u16_t actual;
    s64_t now;
    struct light_lightness_state *state = model->user_data;

    actual = net_buf_simple_pull_le16(buf);
    tid = net_buf_simple_pull_u8(buf);

    now = k_uptime_get();
    if (state->last_tid == tid &&
        state->last_src_addr == ctx->addr &&
        state->last_dst_addr == ctx->recv_dst &&
        (now - state->last_msg_timestamp <= K_SECONDS(6))) {
        return;
    [illegible]

    switch (buf->om_len) {
    case 0x00:  /* No optional fields are available */
        tt = default_tt;
        delay = 0;
        [illegible]
    [illegible]
        tt = net_buf_simple_pull_u8(buf);
        if ((tt & 0x3F) == 0x3F) {
            return;
        }

        [illegible] buf);
        [illegible]
    default:
        return;
    }

    *ptr_counter = 0;
    [illegible]
    state->last_tid = tid;
    state->last_src_addr = ctx->addr;
    state->last_dst_addr = ctx->recv_dst;
    state->last_msg_timestamp = now;

    if (actual > 0 && actual < state->light_range_min) {
        actual = state->light_range_min;
    } else if (actual > state->light_range_max) {
        actual = state->light_range_max;
    }

    state->target_actual = actual;

    if (state->target_actual != state->actual) {
        light_lightness_actual_tt_values(state, tt, delay);
    } else {
        light_lightness_publish(model);
        return;
    }

    [illegible]
    [illegible](state->transi[illegible]counter [illegible]
```

```c
                NULL),

        BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_POWER_ONOFF_SRV,
                      gen_power_onoff_srv_op, &gen_power_onoff_srv_pub,
                      &gen_power_onoff_srv_user_data),
        BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_POWER_ONOFF_SETUP_SRV,
                      gen_power_onoff_setup_srv_op,
                      &gen_power_onoff_srv_pub,
                      &gen_power_onoff_srv_user_data),
        BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_POWER_ONOFF_CLI,
                      gen_power_onoff_cli_op, &gen_power_onoff_cli_pub,
                      NULL),

        BT_MESH_MODEL(BT_MESH_MODEL_ID_LIGHT_LIGHTNESS_SRV,
                      light_lightness_srv_op, &light_lightness_srv_pub,
                      &light_lightness_srv_user_data),
        BT_MESH_MODEL(BT_MESH_MODEL_ID_LIGHT_LIGHTNESS_SETUP_SRV,
                      light_lightness_setup_srv_op,
                      &light_lightness_srv_pub,
                      &light_lightness_srv_user_data),
        BT_MESH_MODEL(BT_MESH_MODEL_ID_LIGHT_LIGHTNESS_CLI,
                      light_lightness_cli_op, &light_lightness_cli_pub,
                      NULL),

        BT_MESH_MODEL(BT_MESH_MODEL_ID_LIGHT_CTL_SRV,
                      light_ctl_srv_op, &light_ctl_srv_pub,
                      &light_ctl_srv_user_data),
        BT_MESH_MODEL(BT_MESH_MODEL_ID_LIGHT_CTL_SETUP_SRV,
                      light_ctl_setup_srv_op, &light_ctl_srv_pub,
                      &light_ctl_srv_user_data),
        BT_MESH_MODEL(BT_MESH_MODEL_ID_LIGHT_CTL_CLI,
                      light_ctl_cli_op, &light_ctl_cli_pub,
                      NULL),
};

struct bt_mesh_model vnd_models[] = {
        BT_MESH_MODEL_VND(CID_RUNTIME, 0x4321, vnd_ops,
                          &vnd_pub, &vnd_user_data),
};

struct bt_mesh_model s0_models[] = {
        BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_LEVEL_SRV,
                      gen_level_srv_op, &gen_level_srv_pub_s0,
                      &gen_level_srv_s0_user_data),
        BT_MESH_MODEL(BT_MESH_MODEL_ID_GEN_LEVEL_CLI,
                      gen_level_cli_op, &gen_level_cli_pub_s0,
                      NULL),

        BT_MESH_MODEL(BT_MESH_MODEL_ID_LIGHT_CTL_TEMP_SRV,
                      light_ctl_temp_srv_op, &light_ctl_srv_pub,
                      &light_ctl_srv_user_data),
};

static struct bt_mesh_elem elements[] = {
        BT_MESH_ELEM(0, root_models, vnd_models),
        BT_MESH_ELEM(0, s0_models, BT_MESH_MODEL_NONE),
};

const struct bt_mesh_comp comp = {
        .cid = CID_RUNTIME,
        .elem = elements,
        .elem_count = ARRAY_SIZE(elements),
```

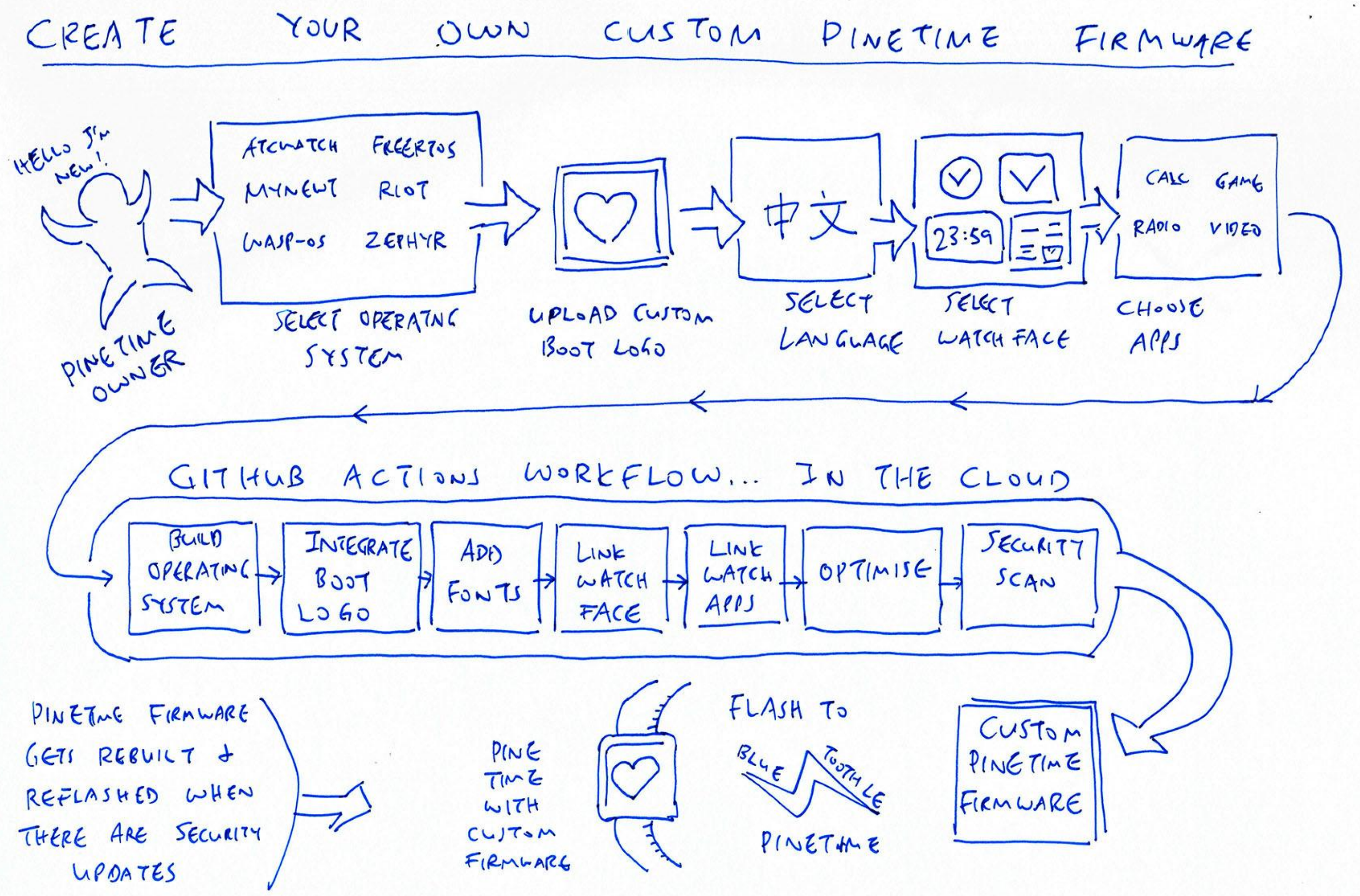
Unfortunately it takes 2,700 lines of C code…

To create a simple Bluetooth Mesh app

Can Rust on RIOT simplify this?

*(Maybe with a Domain-Specific Language?)*

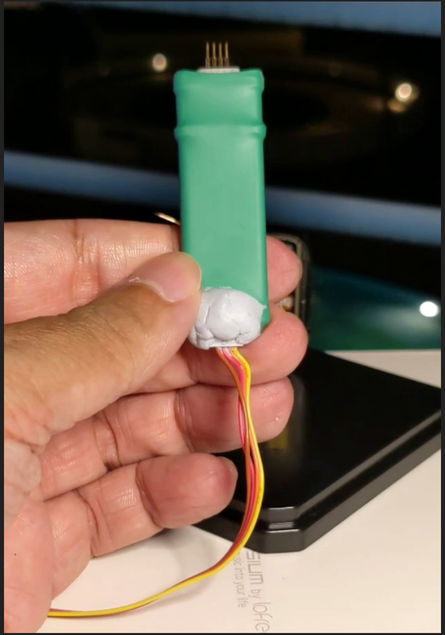
github.com/apache/mynewt-nimble/blob/master/apps/blemesh_models_example_2/src/device_composition.c

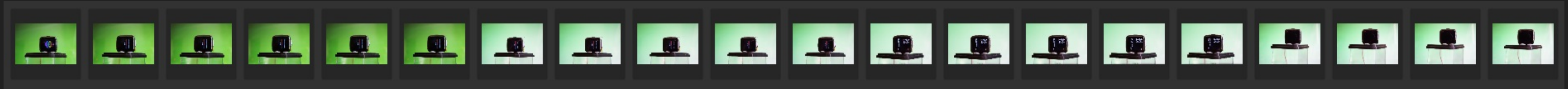

CREATE YOUR OWN CUSTOM PINETIME FIRMWARE
HELLO I'M NEW!
PINETIME OWNER
ATCWATCH
FREERTOS
MYNEWT
RIOT
WASP-OS
ZEPHYR
SELECT OPERATING SYSTEM
UPLOAD CUSTOM BOOT LOGO
中文
SELECT LANGUAGE
23:59
SELECT WATCH FACE
CALC
GAME
RADIO
VIDEO
CHOOSE APPS
GITHUB ACTIONS WORKFLOW... IN THE CLOUD
BUILD OPERATING SYSTEM
INTEGRATE BOOT LOGO
ADD FONTS
LINK WATCH FACE
LINK WATCH APPS
OPTIMISE
SECURITY SCAN
CUSTOM PINETIME FIRMWARE
FLASH TO
BLUETOOTH LE
PINETIME
PINE TIME WITH CUSTOM FIRMWARE
PINETIME FIRMWARE GETS REBUILT & REFLASHED WHEN THERE ARE SECURITY UPDATES

## 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

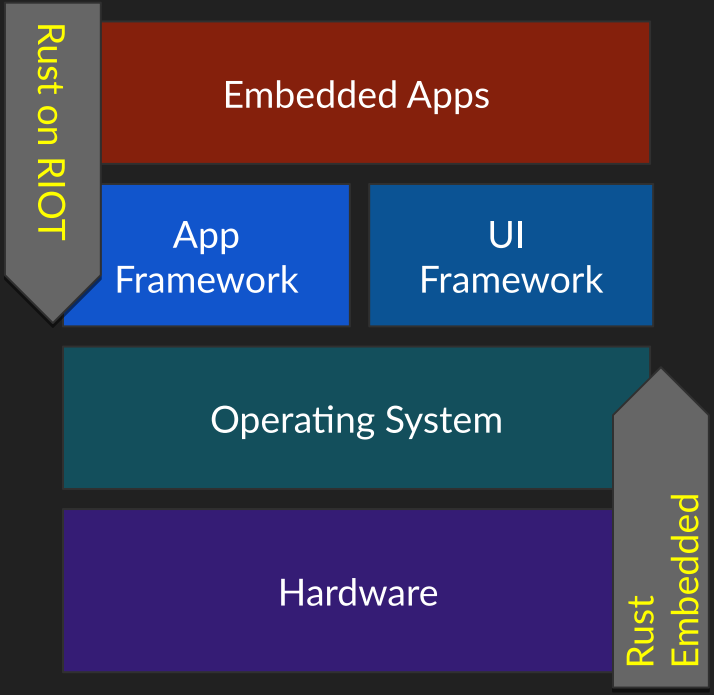
# Rust on RIOT & Rust Embedded Complete Each Other

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!


Rust on RIOT
Embedded Apps
App Framework
UI Framework
Operating System
Hardware
Rust Embedded