Rust for embedded devices

Hello World

EchoKit

Star, clone and fork 👍

EchoKit devices: https://github.com/second-state/echokit_box

EchoKit server: https://github.com/second-state/echokit_server

Use Docker

Use the ESP32 IDF Rust container

Get a CLI inside the container

```
docker run --name=esp32build -it espressif/idf-rust:all_latest /bin/bash
```

Re-start the container after exit

```
docker start -ai esp32build
```

Build the EchoKit device firmware

```
git clone https://github.com/second-state/echokit_box

cd echokit_box

cargo build --release
```

Build the device image

```
espflash save-image --chip esp32s3 --merge --flash-size 16mb
target/xtensa-esp32s3-espidf/release/echokit echokit.bin
```

```
[2025-08-20T16:42:22Z INFO ] 🚀 A new version of espflash is available: v4.0.1

Chip type:         esp32s3

Merge:             true

Skip padding:      false

Partition table:   partitions.csv

App/part. size:    2,497,344/5,242,880 bytes, 47.63%

[2025-08-20T16:42:23Z INFO ] Image successfully saved!
```

Copy and flash

Copy the firmware out of the container

```
docker cp esp32build:/home/esp/echokit_box/target/xtensa-esp32s3-espidf/release/echokit .
```

Copy the device image out of the container

```
docker cp esp32build:/home/esp/echokit_box/echokit.bin .
```

Use espflash or ESP Launchpad tools to flash to your EchoKit device

Hello world

Generate a hello world project

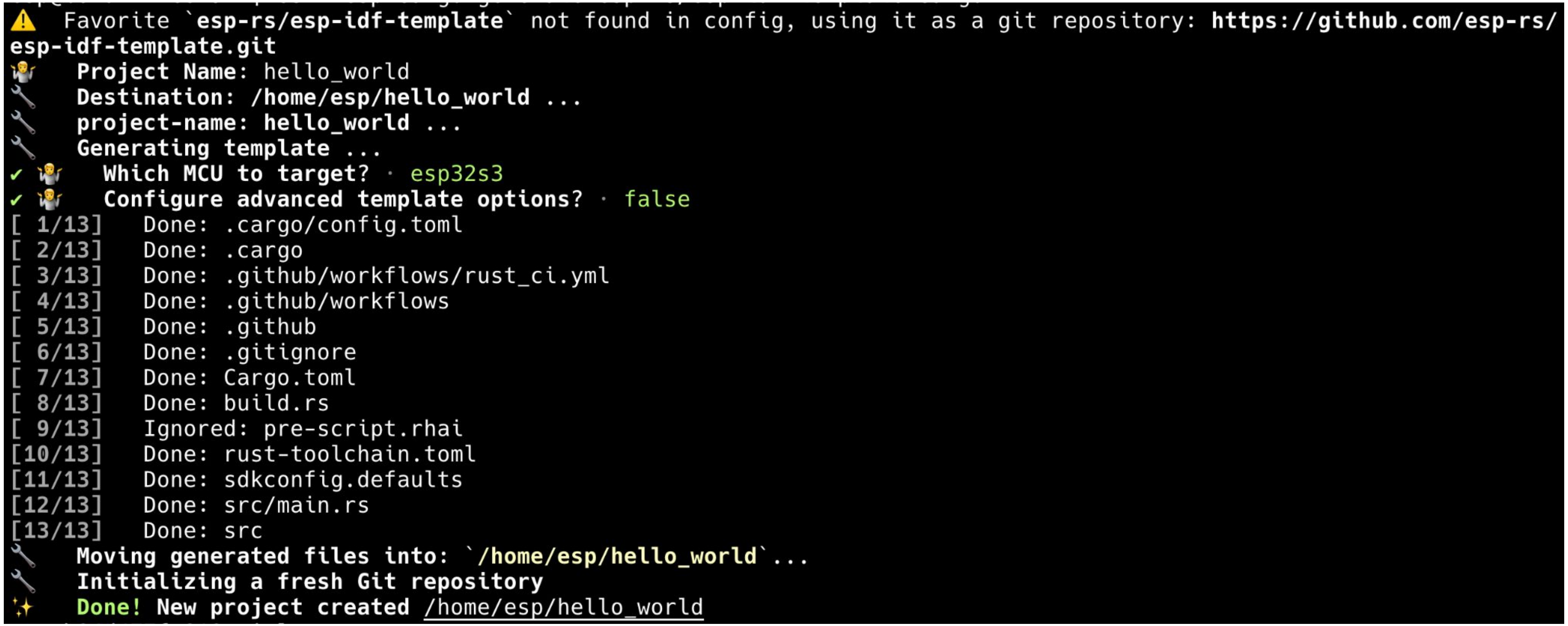
Install the template generator tool

```
cargo install --locked cargo-generate
```

Generate a hello world project

```
USER=esp cargo generate esp-rs/esp-idf-template cargo
```

Generate a hello world project



```
⚠️  Favorite `esp-rs/esp-idf-template` not found in config, using it as a git repository: https://github.com/esp-rs/
esp-idf-template.git
🤷   Project Name: hello_world
🔧   Destination: /home/esp/hello_world ...
🔧   project-name: hello_world ...
🔧   Generating template ...
✔ 🤷   Which MCU to target? · esp32s3
✔ 🤷   Configure advanced template options? · false
[ 1/13]   Done: .cargo/config.toml
[ 2/13]   Done: .cargo
[ 3/13]   Done: .github/workflows/rust_ci.yml
[ 4/13]   Done: .github/workflows
[ 5/13]   Done: .github
[ 6/13]   Done: .gitignore
[ 7/13]   Done: Cargo.toml
[ 8/13]   Done: build.rs
[ 9/13]   Ignored: pre-script.rhai
[10/13]   Done: rust-toolchain.toml
[11/13]   Done: sdkconfig.defaults
[12/13]   Done: src/main.rs
[13/13]   Done: src
🔧   Moving generated files into: `/home/esp/hello_world`...
🔧   Initializing a fresh Git repository
✨   Done! New project created /home/esp/hello_world
```

What's generated?

```
fn main() {
    // It is necessary to call this function once. Otherwise some patches to the runtime
    // implemented by esp-idf-sys might not link properly.
     esp_idf_svc::sys::link_patches();

    // Bind the log crate to the ESP Logging facilities
    esp_idf_svc::log::EspLogger::initialize_default();

    log::info!("Hello, world!");
}
```

Build and run the hello world

Build the project

```
cargo build --release
```

Copy the firmware out of the container

```
docker cp esp32build:/home/esp/hello_world/target/xtensa-esp32s3-espidf/release/hello_world .
```

Use espflash to flash to your EchoKit device https://github.com/second-state/echokit_box

Display messages on the LCD screen

Display text in main.rs

```
ui::lcd_init().unwrap();

let mut gui = ui::UI::new(None).unwrap();

... ...

gui.state = "Failed to connect to wifi".to_string();

gui.text = "Press K0 to restart".to_string();

gui.display_flush().unwrap();
```

Peek at the UI struct

```
pub struct UI {
    pub state: String,
    state_area: Rectangle,
    state_background: Vec<Pixel<ColorFormat>>,
    pub text: String,
    text_area: Rectangle,
    text_background: Vec<Pixel<ColorFormat>>,

    display: Box<Framebuffer< ... ...  >,>,
}
```

Create the UI

```
pub fn new(backgroud_gif: Option<&[u8]>) -> anyhow::Result<Self> {

    let mut display = Box::new(Framebuffer::<ColorFormat, _, LittleEndian, DISPLAY_WIDTH,
DISPLAY_HEIGHT,  { buffer_size::<ColorFormat>(DISPLAY_WIDTH, DISPLAY_HEIGHT) }, >::new());

    display.clear(ColorFormat::WHITE).unwrap();

    let state_area = Rectangle::new(

        display.bounding_box().top_left + Point::new(0, 0),

        Size::new(DISPLAY_WIDTH as u32, 32),

    );

    let text_area = Rectangle::new(

        display.bounding_box().top_left + Point::new(0, 32),

        Size::new(DISPLAY_WIDTH as u32, DISPLAY_HEIGHT as u32 - 32),

    );
```

Draw text on the UI

```
pub fn display_flush(&mut self) -> anyhow::Result<()> {
    ... ...
    let text_box = TextBox::with_textbox_style(
        &self.text, self.text_area,
        MyTextStyle(... ...),
        textbox_style,
    );
    text_box.draw(self.display.as_mut())?;
```

Display image in main.rs

```
let mut new_gif = Vec::new();
std::mem::swap(&mut setting.0.background_gif.0, &mut new_gif);

let _ = ui::backgroud(&new_gif);
gui.text = "Background GIF set OK".to_string();
gui.display_flush().unwrap();
```

https://github.com/second-state/echokit_box

Play a sound

Inter-IC Sound (I2S)

```
let sck = peripherals.pins.gpio5;

let din = peripherals.pins.gpio6;

let dout = peripherals.pins.gpio7;

let ws = peripherals.pins.gpio4;

let bclk = peripherals.pins.gpio15;

let lrclk = peripherals.pins.gpio16;
```

https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/i2s.html

Play audio in main.rs

```
audio::player_welcome(
    peripherals.i2s0,
    bclk.into(),
    dout.into(),
    lrclk.into(),
    None,
    None,
);
```

Play audio in audio.rs

```
pub fn player_welcome() {

    ... ...

    let i2s_config =
config::StdConfig::new(config::Config::default().auto_clear(true),
config::StdClkConfig::from_sample_rate_hz(SAMPLE_RATE),
config::StdSlotConfig::philips_slot_default(config::DataBitWidth::Bits16,
config::SlotMode::Mono,), config::StdGpioConfig::default(),

    );

    let mut tx_driver = I2sDriver::new_std_tx(i2s, &i2s_config, bclk, dout, mclk,
lrclk).unwrap();

    tx_driver.tx_enable().unwrap();

    tx_driver.write_all(WELCOME_WAV, 1000).unwrap();
```

Until next time!