Embeded rust on microbit

Install

```
rustup target add thumbv7em-none-eabihf
rustup component add llvm-tools
cargo install cargo-binuitls
```

Install probe-rs/cargo embed

```
curl --proto '=https' --tlsv1.2 -LsSf
https://github.com/probe-rs/probe-rs/releases/latest/download/probe-rs-tools-
installer.sh | sh
```

Init project

```
cargo init --bin
```

Embed.toml

```
[default.general]
chip = "nRF52833_xxAA"
```

Add crate for CPU

```
cargo add cortex-m-rt
```

memory.x file

```
/* Linker script for the STM32F103C8T6 */
MEMORY
{
  FLASH : ORIGIN = 0x00000000, LENGTH = 512K
  RAM   : ORIGIN = 0x20000000, LENGTH = 128K
}
```

General embeded setup

```
cargo add panic-halt
```

main.rs

```
#![no_std]
#![no_main]

use cortex_m_rt::entry;
use panic_halt as _;

#[entry]
fn main() -> ! {
    loop {}
}
```

Add to Cargo.toml

```
[[bin]]
name = "mb_bare"
path = "src/main.rs"
test = false
doctest = false
bench = false
```

Set udev rules

Download rules file

```
sudo mv Downloads/69-probe-rs.rules /etc/udev/rules.d/
sudo udevadm control --reload
```

Unplug/plugin device

Add .cargo/config.toml

```
[build]
target = "thumbv7em-none-eabihf"

[target.thumbv7em-none-eabihf]
rustflags = [ "-C", "link-arg=-Tlink.x"]
```

Compile and run

```
cargo embed
```

Debugging

Setup RTT for debug

```
cargo add cortex-m –features critical-section-single-core
cargo add rtt-target
```

```
Add rtt to Embed.toml
```

```
[default.rtt]
enabled = true
```

Add rtt init to main.rs

```
...
use rtt_target::{rprintln, rtt_init_print};
use cortex_m as _;

#[entry]
fn main() -> ! {
    rtt_init_print!();
    rprintln!("Hello, world!");
    loop {}
}
```

GDB (on Debian)

```
sudo apt install gdb-multiarch
```

Modify Embed.toml

```
[default.rtt]
enabled = false

[default.gdb]
enabled = true

[default.reset]
halt_afterwards = true
```

Deactivate gdb and halt_afterwards for using rtt again

As rtt is disabled change main.rs

```
use cortex_m::asm::nop;
use cortex_m_rt::entry;
use panic_halt as _;

#[entry]
fn main() -> ! {
    loop {
        let mut x: usize = 0;
        x +=1;
        for _ in 0..x {
            nop();
        }
    }
}
```

Build and Run

```
cargo embed
gdb-multiarch target/…
```