

# Práce s chybami v Rustu



## Marek Pšenka

- Technický vedoucí v Edhouse
- 7 let zkušeností
- Většinu kariéry jsem pracoval s C++ a C#
- Rust používám již dva roky

# Generátor vodíku H2Gem

- Zařízení pro výrobu zeleného vodíku
- V Edhouse jsme vyvinuli kompletní firmware
- Rust nám významně pomohl se spolehlivostí



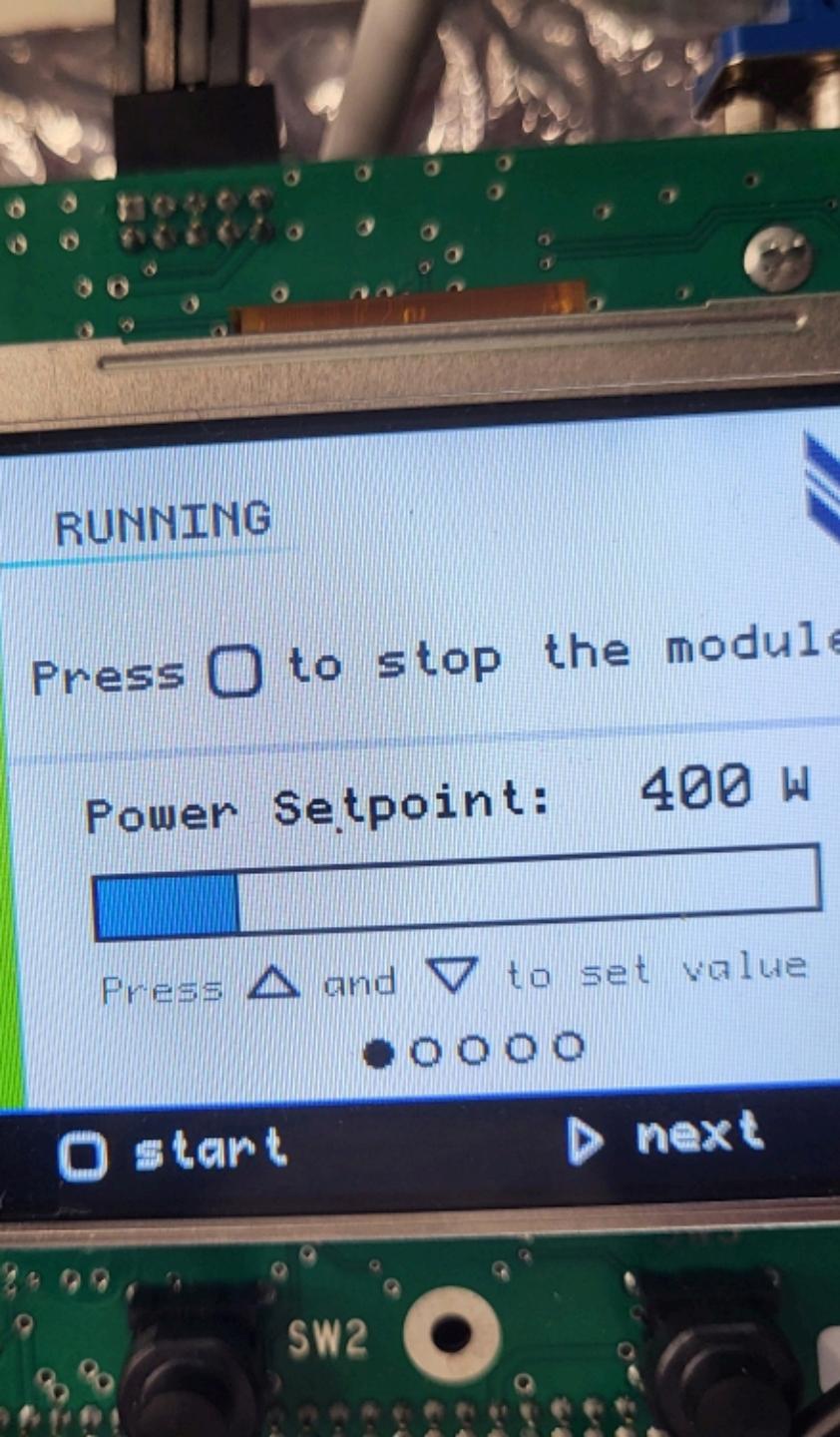
## H2Gem technicky

Řešené úlohy:

- komunikace a řízení zdroje elektrické energie
- komunikace se senzory a nadřazeným systémem
- zobrazení a vstupy na/z grafického displeje
- vše na platformě STM32.

Role Rustu:

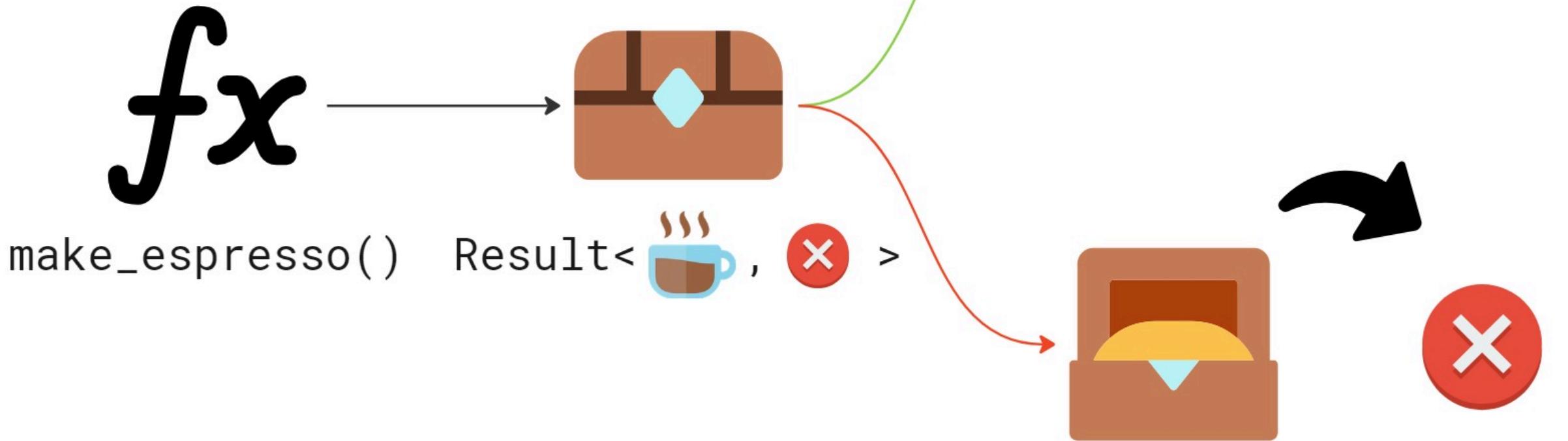
- Celé řešení, včetně ovladačů pomocí RTIC
- žádné runtime chyby v průběhu vývoje a testování
- rychlejší obrátky na HW ve srovnání s C++



```
pub struct CoffeeMachine {  
    water_tank_volume: f64,  
    available_coffee_beans: f64,  
}  
  
impl CoffeeMachine {  
    pub fn make_espresso(&self) -> Result<Espresso, String> {  
        if self.water_tank_volume < 25.0 {  
            Err("Not enough water in tank".to_string())  
        } else if self.available_coffee_beans < 7.0 {  
            Err("Not enough coffee beans".to_string())  
        } else {  
            Ok(Espresso {})  
        }  
    }  
}
```

```
#[test]
fn error_returned_when_making_espresso_without_beans() {
    let machine = CoffeeMachine {
        water_tank_volume: 300.0,
        available_coffee_beans: 2.0,
    };

    let result = machine.make_espresso();
    assert!(result.is_err());
    assert_eq!(result, Err("Not enough coffee beans".to_string()));
}
```



## Filozofie

Myšlenka vyhradit prostor pro chybové informace v návratové hodnotě není nová

```
int main(void)
{
    FILE *f = fopen("non_existent", "r");
    if (f == NULL) {
        perror("fopen() failed");
    } else {
        fclose(f);
    }
}
```

```
fopen() failed: No such file or directory
```

## Rust nám to usnadňuje

```
pub enum Result<T, E> {  
    Ok(T),  
    Err(E),  
}
```

```
fn open_nonexistent_file() {  
    match std::fs::File::open("non_existent") {  
        Ok(file) => drop(file),  
        Err(err) => println!("open() failed: {}", err),  
    }  
}
```

```
open() failed: The system cannot find the file specified. (os error 2)
```

# Porovnej

```
int main(void) {
    FILE *f = fopen("non_existent", "r");
    if (f == NULL) {
        perror("fopen() failed");
    } else {
        fclose(f);
    }
}
```

```
fn open_nonexistent_file() {
    match std::fs::File::open("non_existent") {
        Ok(file) => drop(file),
        Err(err) => println!("open() failed: {}", err),
    }
}
```

## Jiná strategie - výjimky v C# - nejsou vidět

```
public static System.IO.FileStream Open (string path, System.IO.FileMode mode);
```

Kde se dozvím jak vypadá chyba? V dokumentaci:

- ArgumentNullException
- PathTooLongException
- (...)

Rust je explicitní. Dozvím se to v kódu:

```
pub fn open<P: AsRef<Path>>(path: P) -> std::Result<T, std::io::Error>;
```

## Vyjímky střílí

```
void OpenNonexistentFile() {
    File.Open("non_existent", FileMode.Open);
}

OpenNonexistentFile();

DoSomethingElse();
```

```
C:\code\rust-error-handling\_examples_cs>dotnet run
Unhandled exception. System.IO.FileNotFoundException: Could not find file 'non_existent'.
(...)
```

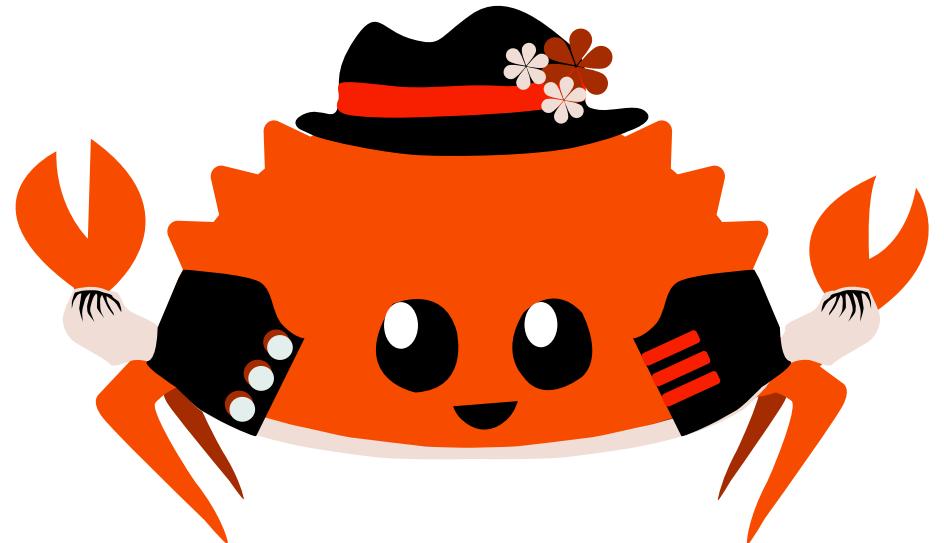
# Porovnej

```
void OpenNonexistentFile() {
    try
    {
        File.Open("non_existent", FileMode.Open);
    }
    catch (Exception e) {
        Console.WriteLine($"{e}");
    }
}
```

```
fn open_nonexistent_file() {
    match std::fs::File::open("non_existent") {
        Ok(file) => drop(file),
        Err(err) => println!("open() failed: {}", err),
    }
}
```

## Shrnutí

- Rust nám na zákaznických projektech pomáhá psát spolehlivý kód
- Myšlenka vyhradit prostor pro chybové informace v návratové hodnotě není nová
- Rust nám to usnadňuje standardním typem `Result<T, E>`
- Příklad alternativní strategie jsou výjimky.
- Nejsou ale vidět a střílí - nutná bdělost



**Rust Moravia**

