



DesignNews

Embedded Software using RUST

DAY 1: Introduction to Rust for Embedded Systems

Sponsored by



Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Group Chat’ by maximizing the chat widget in your dock.

THE SPEAKER



Jacob Beningo

Visit 'Lecturer Profile'

Beningo Embedded Group - President

Focus: Embedded Software Consulting

An independent consultant who specializes in the design of real-time, microcontroller based embedded software.

He has published two books:

- [Reusable Firmware Development](#)
- [MicroPython Projects](#)
- [Embedded Software Design](#)

Writes a weekly blog for DesignNews.com focused on embedded system design techniques and challenges.

Visit www.beningo.com to learn more ...

Visit 'Lecturer Profile' in your console for more details.

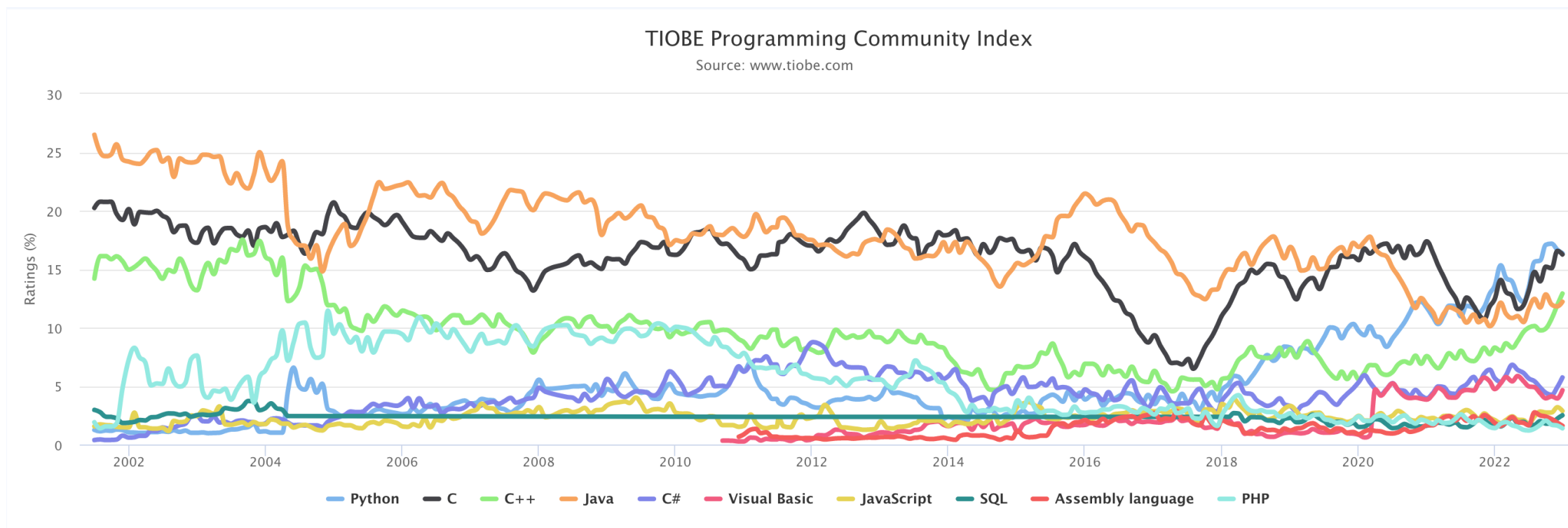
Course Sessions

- Introduction to Rust for Embedded Systems
- "Hello Rust!", using QEMU
- "Hello Rust!", using the STM32F3
- Interfacing to Peripherals in Rust
- Becoming a Rust Expert

1

Embedded Software Languages












Embedded Software Languages



Embedded Software Languages

Most Popular Embedded

- C (80%+)
- C++ (10%+)
- Python (<5%)
- Assembly

Jan 2023	Jan 2022	Change	Programming Language	Ratings	Change
1	1		 Python	16.36%	+2.78%
2	2		 C	16.26%	+3.82%
3	4	^	 C++	12.91%	+4.62%
4	3	v	 Java	12.21%	+1.55%
5	5		 C#	5.73%	+0.05%
6	6		 Visual Basic	4.64%	-0.10%
7	7		 JavaScript	2.87%	+0.78%
8	9	^	 SQL	2.50%	+0.70%
9	8	v	 Assembly language	1.60%	-0.25%
10	11	^	 PHP	1.39%	-0.00%
18	26	^	 Rust	0.61%	+0.11%

Embedded Software Languages

Rust is a systems programming language that is designed to be fast, efficient, and safe from memory-related bugs.

- Created by Mozilla in 2010 by Graydon Hoare
- Writes high-performance, low-level code for systems like operating systems, game engines, and **embedded systems**.
- Memory safety is one of the key driving factors of interest and adoption

What language do you primarily use for embedded programming?

- C
- C++
- Python
- Assembly
- Rust
- Other

3

Rust Memory Safety

Rust Memory Safety

C Example

```
#include <stdio.h>
#include <stdint.h>

int main (void){
    printf ("Hello, world!\n");
    printf ("Let's overflow the buffer!\r\n");

    uint32_t array[5] = {0, 0, 0, 0, 0};

    for(int index = 0; index < 6; index++) {
        printf("Index %d: %d\r\n", index, array[index]);
    }

    return 0;
}
```

Buffer Overflows

A buffer overflow is a type of software vulnerability that occurs when more data is written to a buffer in memory than the buffer can store. This can cause the data stored in adjacent memory locations to be overwritten, leading to unintended and potentially harmful consequences.

Buffer overflows can cause crashes, corrupt data, or even allow attackers to execute arbitrary code on the system.

Rust Memory Safety

Rust Example

```
fn main() {  
    println!("Hello World!");  
    println!("Let's overflow the buffer!");  
  
    let array: [u32; 5] = [0;5];  
  
    for index in 0..array.len() + 1 {  
        println!("Index {}: {} ", index, array[index])  
    }  
}
```

Buffer Overflow Detection

- **Bounds Checking:** Data can never be written outside a buffer's boundaries
- **Ownership and Borrowing:** Prevents use-after-free and double-free bugs.
- **Safe Rust:** Language subset that makes it easier to write memory safe code.
- **Mutability:** by default, variables are not mutable and must be declared so.

Rust Memory Safety

C Example

```
Hello, world!  
Let's overflow the buffer!  
Index 0: 0  
Index 1: 0  
Index 2: 0  
Index 3: 0  
Index 4: 0  
Index 5: 1
```

Rust Example

```
[beningo@Jacobs-MacBook-Pro buffer_ouerrun % cargo run  
  Compiling buffer_ouerrun v0.1.0 (/Users/beningo/rust/buffer_ouerrun)  
  Finished dev [unoptimized + debuginfo] target(s) in 0.39s  
  Running `target/debug/buffer_ouerrun`  
Hello World!  
Let's overflow the buffer!  
Index 0: 0  
Index 1: 0  
Index 2: 0  
Index 3: 0  
Index 4: 0  
thread 'main' panicked at 'index out of bounds: the len is 5 but the index is 5', src/main.rs:8:42  
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace  
beningo@Jacobs-MacBook-Pro buffer_ouerrun %
```

How important are memory safety related bugs to you?

- Critically important
- Important
- Could go either way
- Not important

4

Language Concept Mapping

Language Concept Mapping

Ownership and borrowing: Critical to safety. Similar to C++ concepts like smart pointers, references, move semantics etc.

Generics: C++ templates

No inheritance: Only uses composition.

Traits: behave like abstract classes and interfaces. Provides polymorphism.

Language Concept Mapping

Cargo: Built-in package manager and build system tool(s)

Attributes: think preprocessor directives

Unsafe: allows unsafe libraries and other language code to be executed.

Build Types: standard builds included in the language: release, debug, and test.

You're here! When do you think you'll use Rust in a production system?

- Within the next 3 months
- Within the next 3 – 6 months
- Within the next 6 – 12 months
- I'm not sure

5

Going Further

Rust Resources

- [Rust Website](#)
- [Rust Book](#)
- [Rust for Embedded Book](#)
- [Learning Rust for Embedded Systems](#)
- [Rust By Example](#)
- [RTIC: Real-Time Interrupt Driven Concurrency](#)



Thank you for attending

Please consider the resources below:

- www.beningo.com
 - Blog, White Papers, Courses
 - Embedded Bytes Newsletter
 - <http://bit.ly/1BAHYXm>
 - Embedded Software Design
 - <https://www.beningo.com/embedded-software-design/>

From www.beningo.com under

- Blog > CEC – Embedded Software using Rust





DesignNews

Thank You

Sponsored by



© 2022 Beningo Embedded Group, LLC. All Rights Reserved.