

Embedded TDD

For Cambridge Software Crafters

13 March 2024

Brice Fernandes

brice@fractallambda.com



Logistics and Wifi

- 👉 Behind the main screen to the right of the corridor.
- 🌐 Wifi is **The Bradfield Centre** password is **Ca3Br1d5e**
- ⚠️ We do not expect alarms. Assume a fire alarm is real and make your way to the car park.



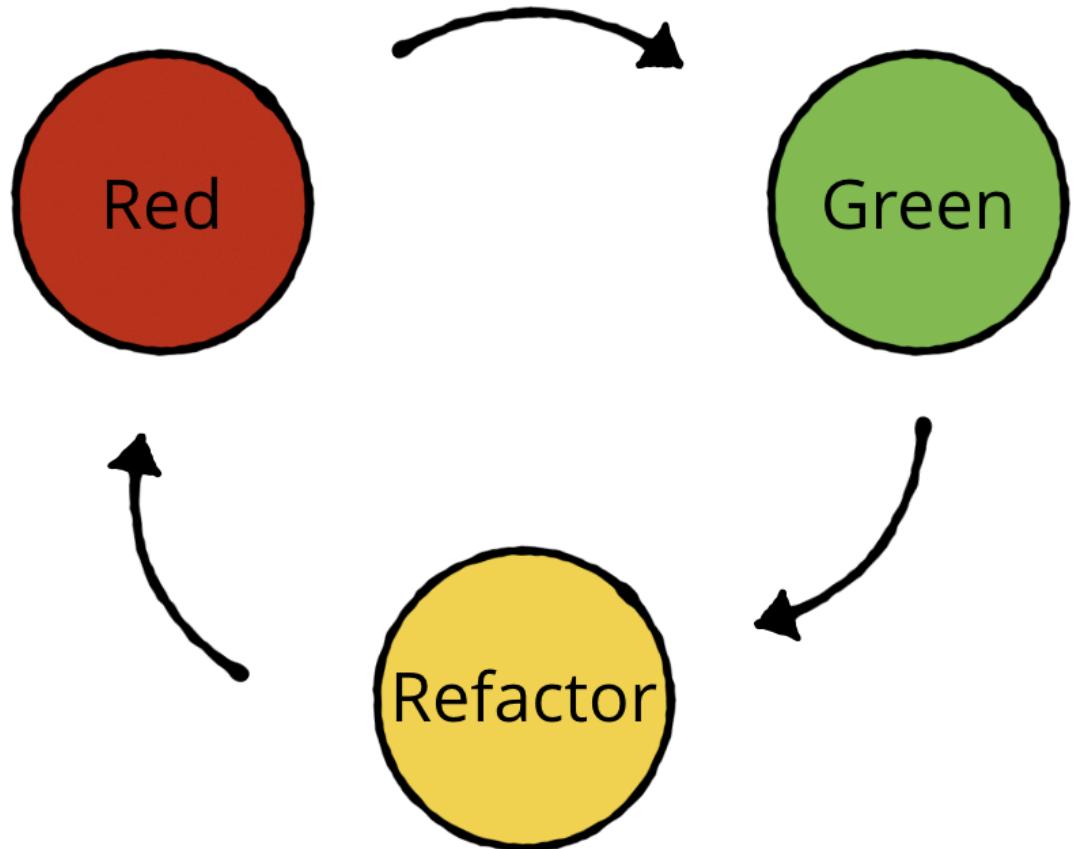
Slides and code available on Github
github.com/bricef/embedded-tdd-katas

Plan for this evening

1. Intro
2. What is "Embedded"
3. Embedded craftsmanship
4. Using Replit
5. LED Driver Kata
6. Recap

Why this talk?

TDD Refresh



TDD Loop

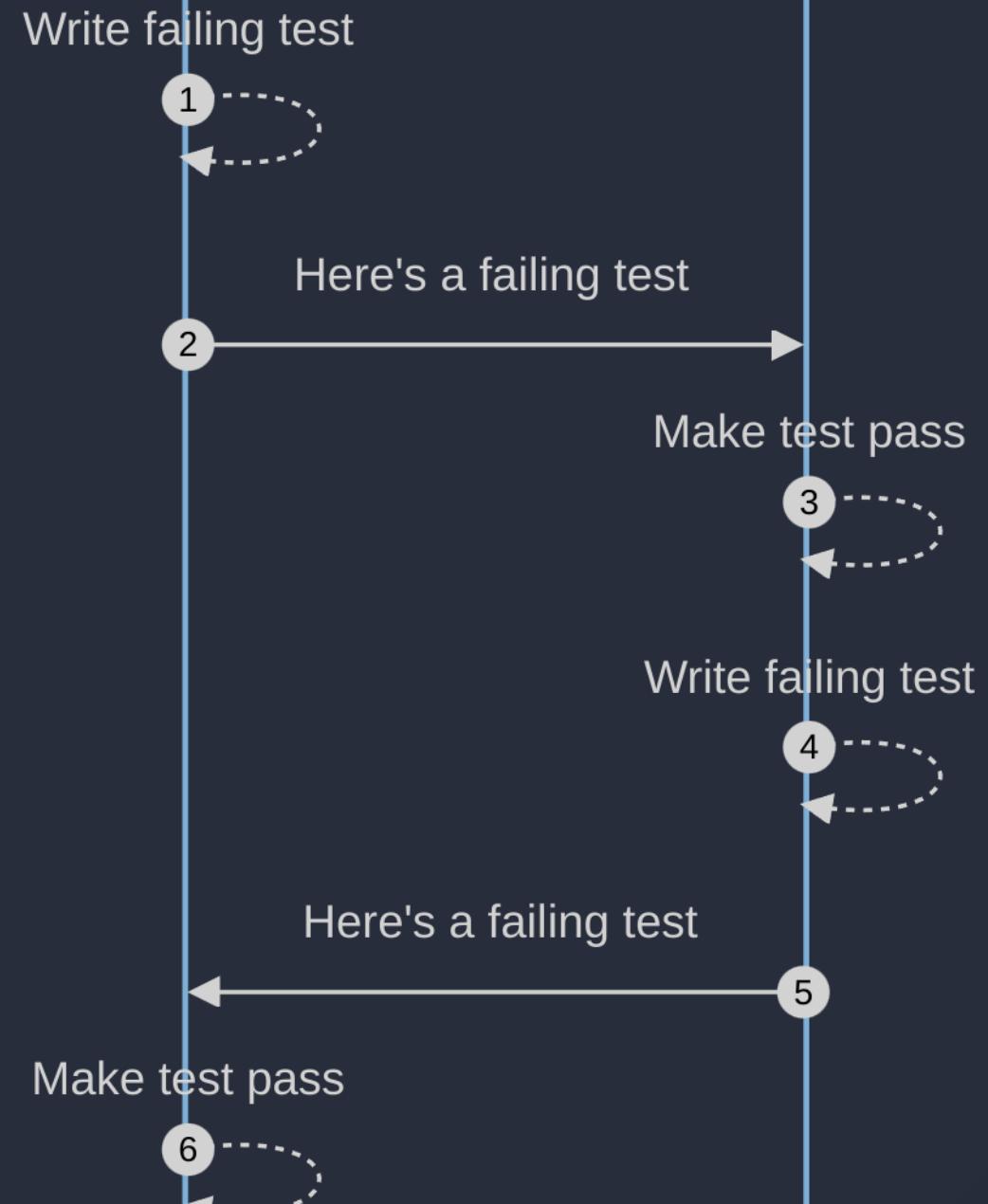
Write a failing test

Make the test pass

Refactor the code

```
1 TEST(LedDriver, ArrangeActAssertExample){  
2     // Arrange  
3     LedDriver_Create(&virtualLeds);  
4  
5     // Act  
6     LedDriver_TurnOn(4);  
7  
8     // Assert  
9     TEST_ASSERT_EQUAL_HEX16(0x08, virtualLeds);  
10  
11    // Teardown  
12    LedDriver_Destroy();  
13 }  
14 }
```

Ping Pong TDD



What is "Embedded"

Embedded constraints

- Resource constraints (RAM, CPU)
- Lack of standard libraries
- No or limited filesystem
- Limited Interface (serial? UART, SWI)
- No Operating System
- No standard library
- Direct hardware access
- Lack of MMU/PMMU

Special pains

- Late hardware delivery
- Hardware scarcity
- Hardware bugs
- Long target compile times
- Long target setup and upload time
- Compiler licenses

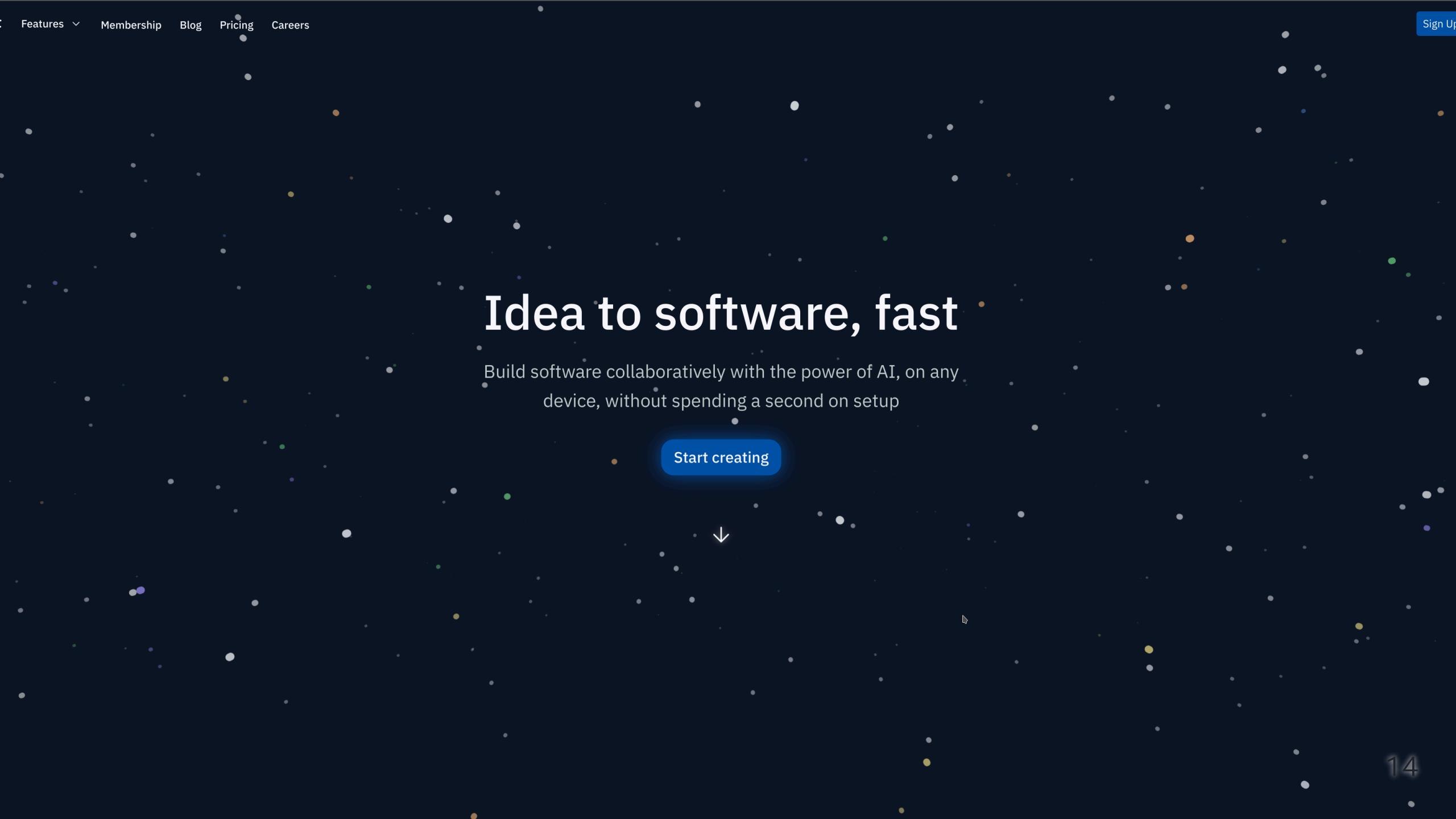


Let's get the party started!

(also, let's download dependencies whilst we talk!)

Using Replit

Create a replit.com account
(use a throwaway email if you'd like)





Search & run commands

Ctrl .

+ ?

+ Create Repl

me

Repls

Deployments

Page

Teams

Notes

Community

Jobs

Events

Presentation

Join Replit Core

Desktop App

Mobile App



bricefernandes

@bricefernandes

0 followers 0 following

• Online

This is where your Repls will be when you create them.

1

+ Create Repl

1. Create a new Repl

The screenshot shows the Replit interface. At the top, there's a search bar labeled "Search & run commands" and a "Ctrl . ." key binding. On the far right, there are icons for a plus sign, a bell, and a question mark. The left sidebar has a "Create Repl" button and several collapsed sections like "Home", "My Repls", "Deployments", "Code", and "Bans". The main area shows a user profile for "bricefernandes" with a picture, edit and camera icons, and a "Create Repl" button. Below the profile, it says "This is where your Repls will be when you create them." and "bricefernandes" with the handle "@bricefernandes". It shows "0 followers" and "0 following". A status indicator shows "Online". A central modal window titled "Create a Repl" is open. It has a "Template" section with a search bar and a "Name your Repl" input field. Below that is a "Public" checkbox with the note "Anyone can view and fork this Repl." and a "Upgrade to make private" button. There's also an "Import from GitHub" button with a red notification badge showing "2". A "Create Repl" button at the bottom of the modal matches the one in the header.

2. Import from Github

Search & run commands

Ctrl .

+ Create Repl

me

Repls

Deployments

Page

Teams

+

Community

Join Replit Core

Desktop App

Mobile App

?

3. Choose "From URL"

Import from GitHub

Create from Template X

My Repositories From URL 3

Connect your GitHub account

Language ⓘ

Select a repository

⊕ Public

Anyone can view and fork this Repl.

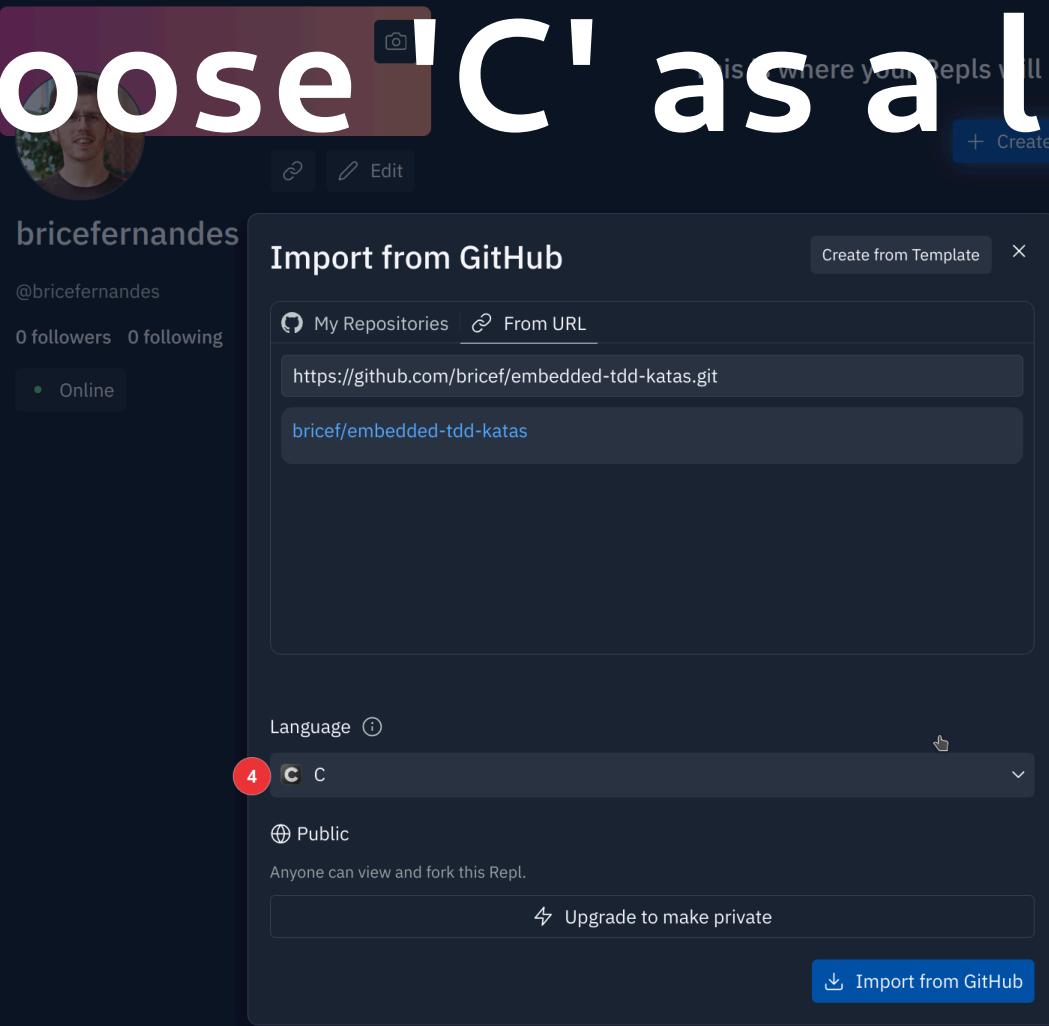
⚡ Upgrade to make private

Import from GitHub

<https://github.com/bricef/embedded-tdd-katas.git>

17

4. Choose 'C' as a language



<https://github.com/bricef/embedded-tdd-katas.git>

embeded-tdd-katas (1) Run Invite Deploy ?

README.md Open in Editor

Embedded TDD Katas

Embedded TDD

For Cambridge Software Crafters
13 March 2024

Brice Fernandes
brice@fractallambda.com

About

This talk was created for Cambridge Software Crafters and delivered there in Mach 2024.

Colophon

This talk is built in Marp and the PDF can be built from source by running `make` from the `Talk` folder.

License

Unless otherwise specified the slides and code in this repository by Brice Fernandes are licensed under CC BY-SA 4.0

 [Unsupported image](#)  [Unsupported image](#)  [Unsupported image](#)

Configure Repl .replit

Configure the Compile Command
This is optional, and is executed before your run command.
`make -s`

Configure the Run Command
`./main`

Not sure what to do?
[Done](#)

Console Shell Run

Results of your code will appear here when you Run the project.

Join Replit Core X

19

Local alternative

If you're confident in your local toolchain

Clone the repository locally:

```
$ git clone https://github.com/bricef/embedded-tdd-katas.git
```

Run the setup script

Open a shell in Replit, then run

```
$ cd Code  
$ ./setup.sh
```

Now we wait for dependencies to get downloaded...



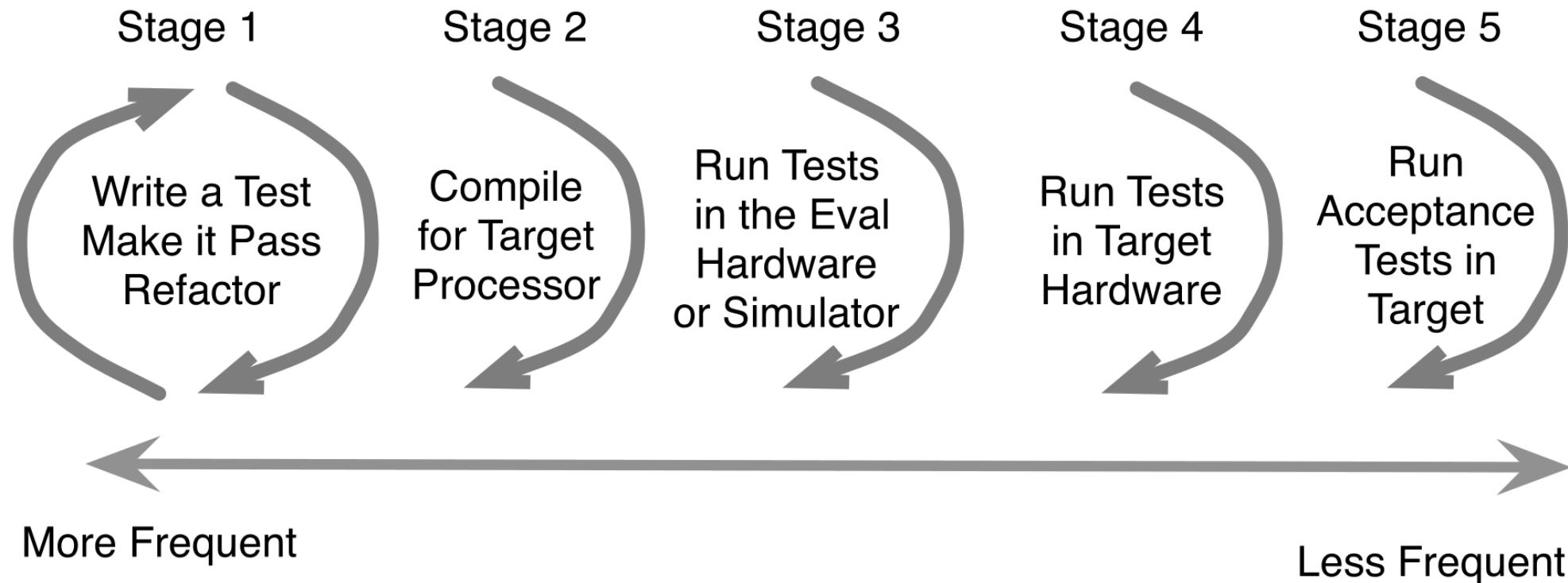
Whilst we wait...

Embedded Strategy

Dual targeting

- Dual targeting
 - Simulate hard-to-duplicate conditions
 - Get around target bottleneck
 - Running the test suite locally
 - Automated CI

Embedded TDD Cycles



Automated HW tests

- There's no reason why you can't create an automated harness that runs the unit test on test devices.
- There's no reason why your CI builds couldn't use HW tests. Including cloud runners!
- You might want to ship tests in production devices as part of a HW self-test suite.

We won't go into depth in this topic tonight.

Test Doubles

- Critical for embedded
- Mock the HAL
- Mock the clock



How to Mock?

In order of preference

1. Link time substitution
(Requires appropriate code structure)
2. Function pointer substitution
3. Syntactic substitution (preprocessor)

Combine at will...

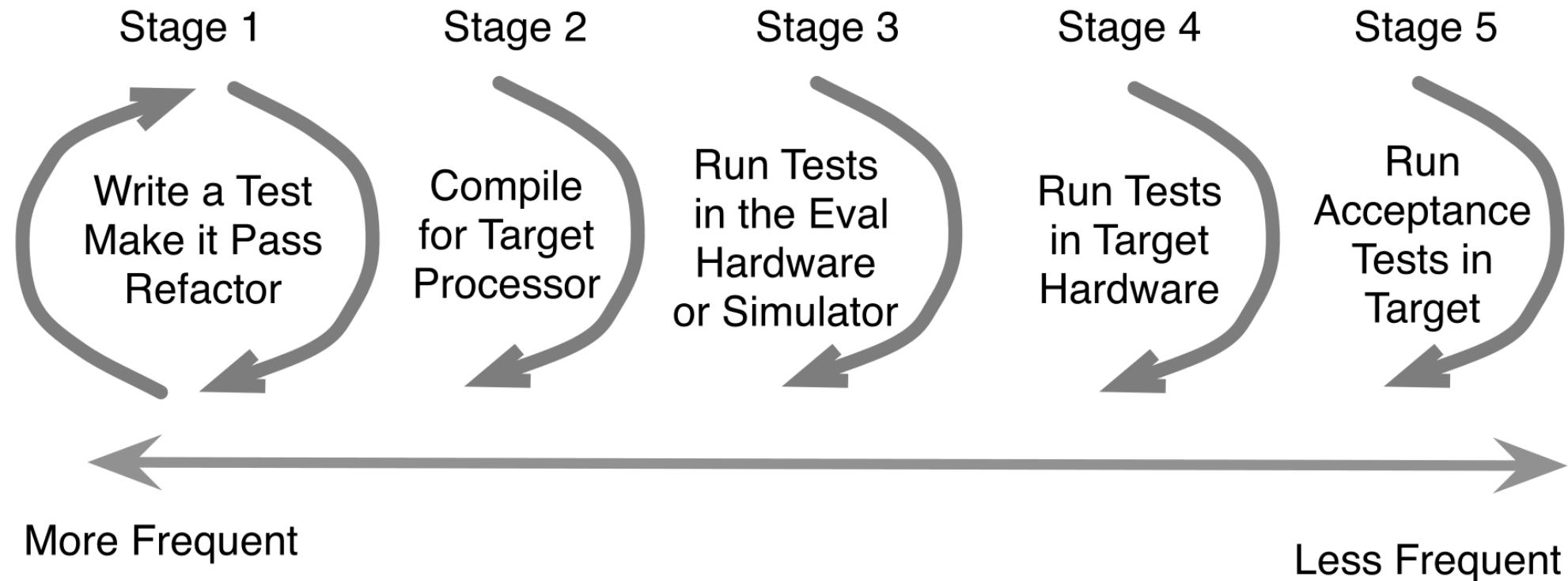
Simulators

Allow testing compiled target code in CI.

Run entire test suite.

Go hand-in-hand with Test Doubles.

TDD Test Cycles



Craftsmanship
fundamentals still matter

SOLID

1. Single Responsibility Principle
2. Open Closed Principle
3. Liskov Substitution Principle
4. Interface Segregation Principle
5. Dependency Inversion Principle

KEEP CALM

A close-up photograph of a man with light brown hair, wearing a flight suit and a communication headset with a microphone. He is sweating profusely, with sweat running down his forehead and chin. His expression is focused and slightly weary. The background is dark and out of focus.

IT'S DEMO TIME

The Katas

LED Driver Kata

Look at `KATA.md` in `Code/src/leddriver`

Recap

What we learnt

1. TDD is possible and *useful* for embedded software.
2. Embedded TDD strategies make the process easier.
3. Dual targeting is worth it.

(C is fun, maybe?)

Further Reading

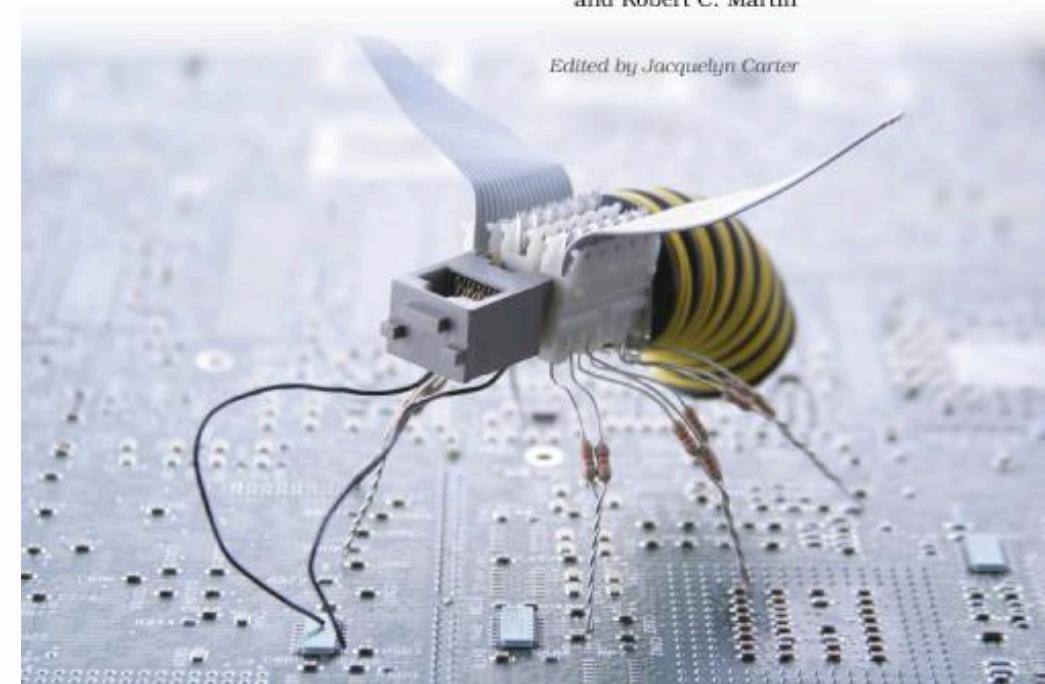
- TDD for Embedded C
- ThrowTheSwitch.org
- Unity Test Framework

Test-Driven Development for Embedded C

James W. Grenning

Forewords by Jack Ganssle
and Robert C. Martin

Edited by Jacquelyn Carter



Thank you 🙏

Q&A ?

I'm available for contracting
brice@fractallambda.com

Attributions

Arrange Act Assert - Own work - CC Attribution-Sharealike

Demo Time - Still frame taken from the film "Airplane!" 1980 - © Paramount Pictures - Used under fair use for teaching.

Party Time by Irtiza Haider - CC Attribution-Sharealike - - [Wikimedia Commons](#)

Ping-pong TDD - Own work - CC Attribution-Sharealike - Created using [Mermaid.js](#)

Red-green-refactor - © [Kodeco](#) - Used under fair use for teaching.

Replit Screenshots - Own work - Created under fair use for teaching.

TDD for Embedded C Book Cover - © [2011 Pragmatic Bookshelf](#) - Used under fair use for teaching.

TDD Cycles - © [2011 Pragmatic Bookshelf](#) - Used under fair use for teaching.

Mockingbird by Ryan Hagerty - Public Domain - [National Digital Library of the United States Fish and Wildlife Service](#)