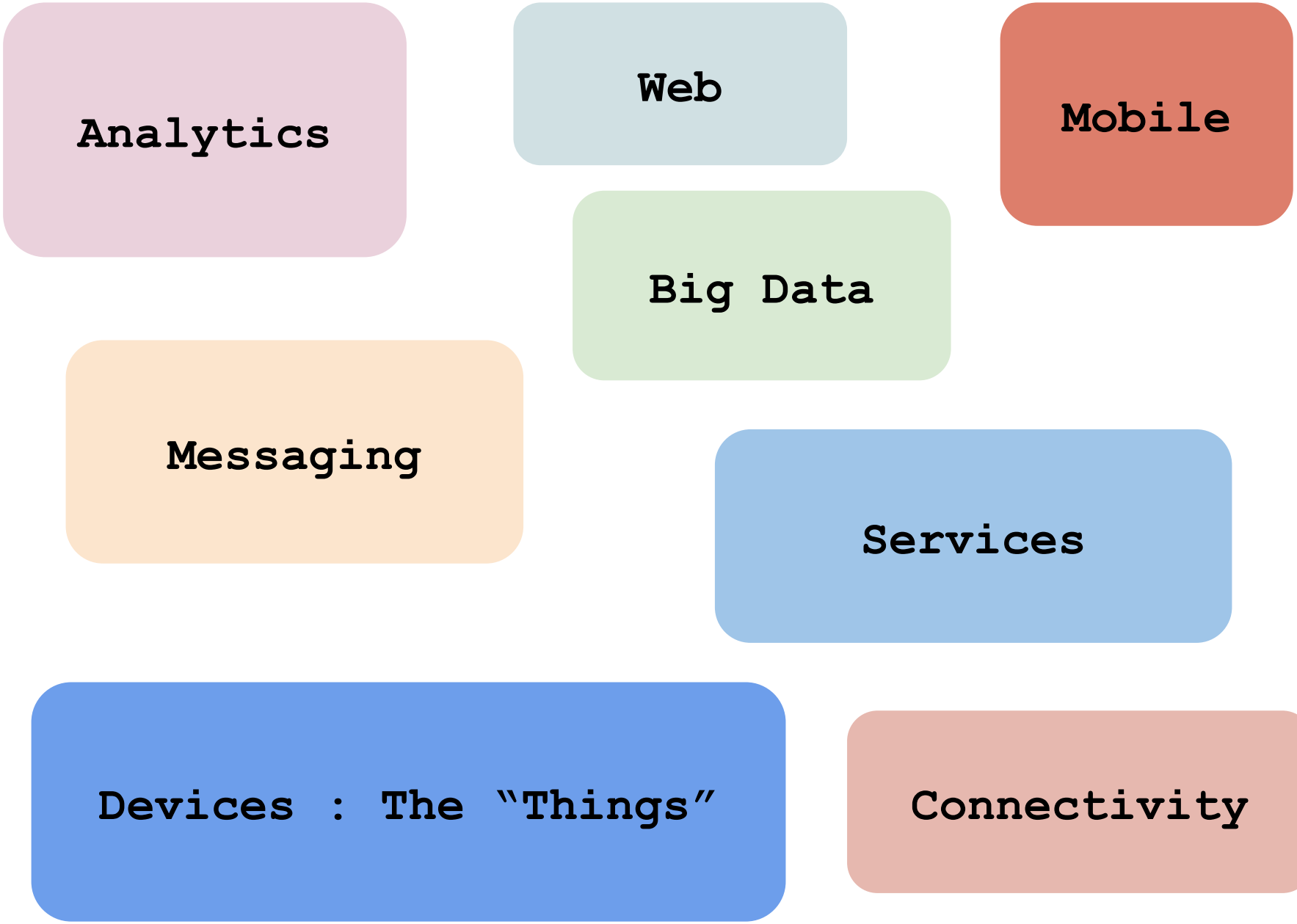
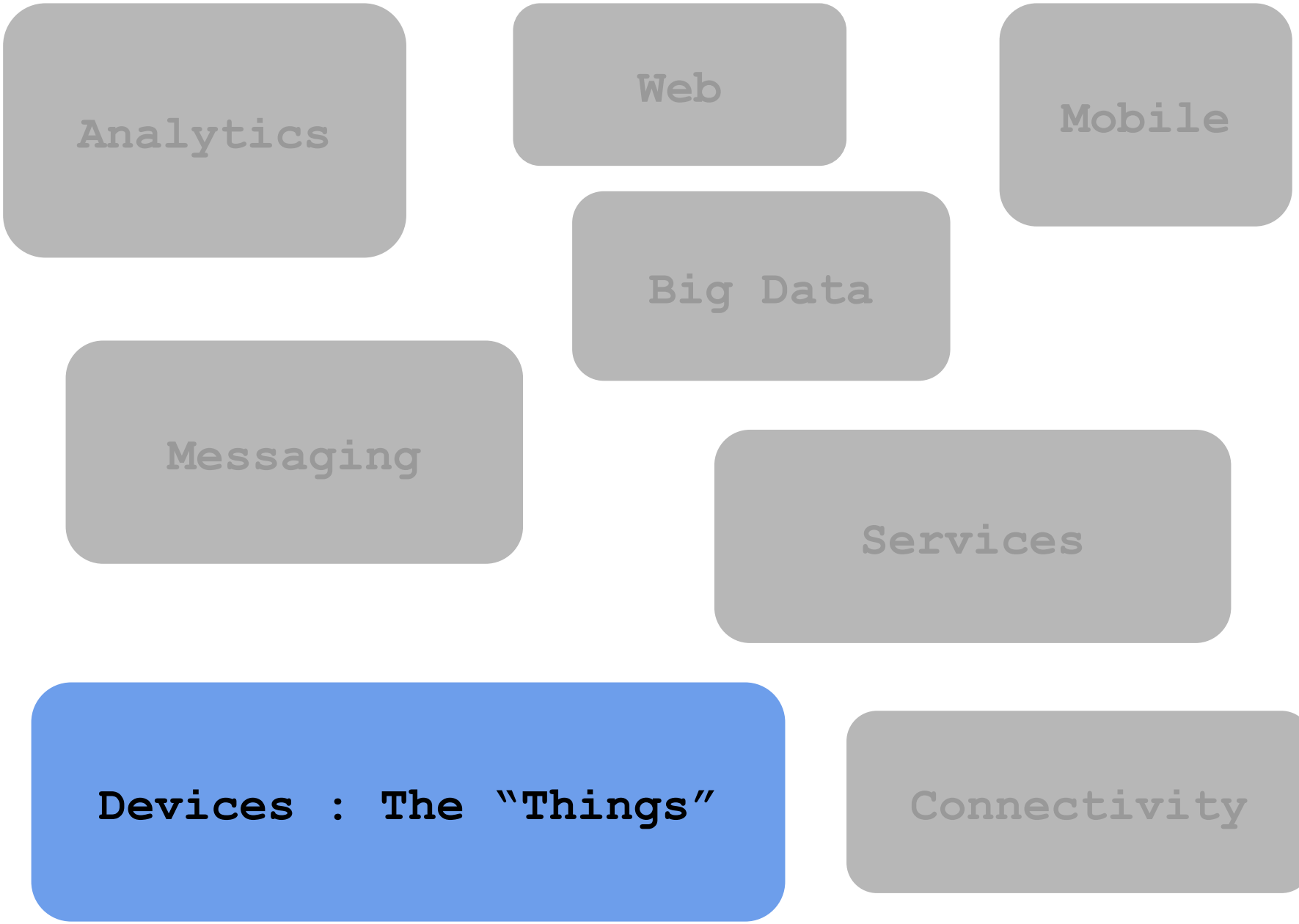
The Code Craft Thing for the Internet Of Things Thing

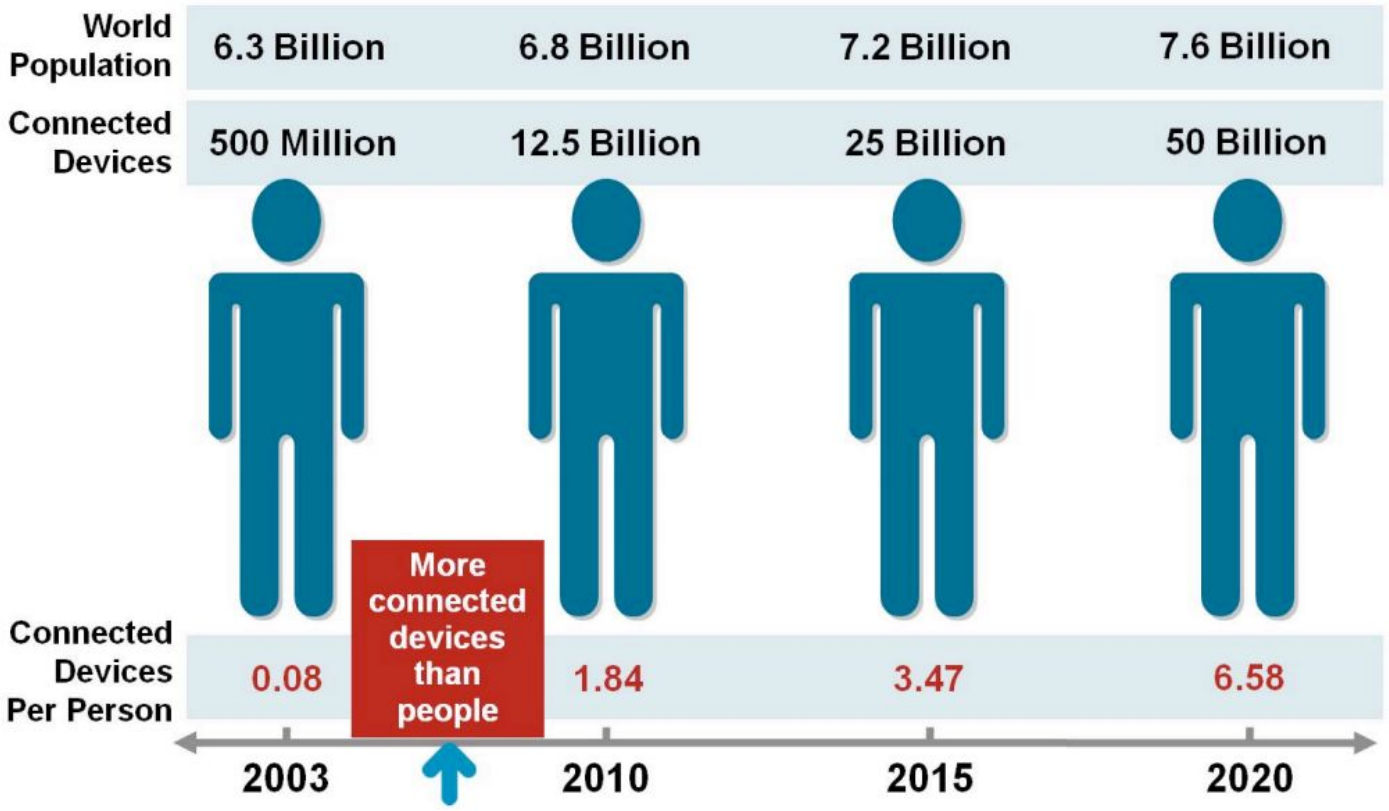
13coders

How test-driven development and design helps make embedded and IoT development safer + Better for users, and a happier experience for developers

Mike Ritchie | @13coders | mike@13coders.com


Analytics
Web
Mobile
Big Data
Messaging
Services
Devices : The “Things”
Connectivity


Analytics
Web
Mobile
Big Data
Messaging
Services
Devices : The "Things"
Connectivity



Source: Cisco IBSG, April 2011

Professional healthcare, transport, smart home, connected car, fitness, wearables, industrial robotics, military...

Systems Defined By Software : The Ubiquitous Chip

- A long-established trend
- Flexibility in product design
- Often licensed cores
- Specialised for applications
- Everywhere. Every. Where.

A relatively small number of companies design intellectual property for chip cores

They often don't manufacture Silicon themselves (ARM, as the most notable example)

Core design IP is licensed on to manufacturers and OEMs

Who produce devices tailored to specific markets

Some Manufacturers license multiple core IPs

Acquisition happens a lot

Pervasive Connectivity, Cloud Back-ends

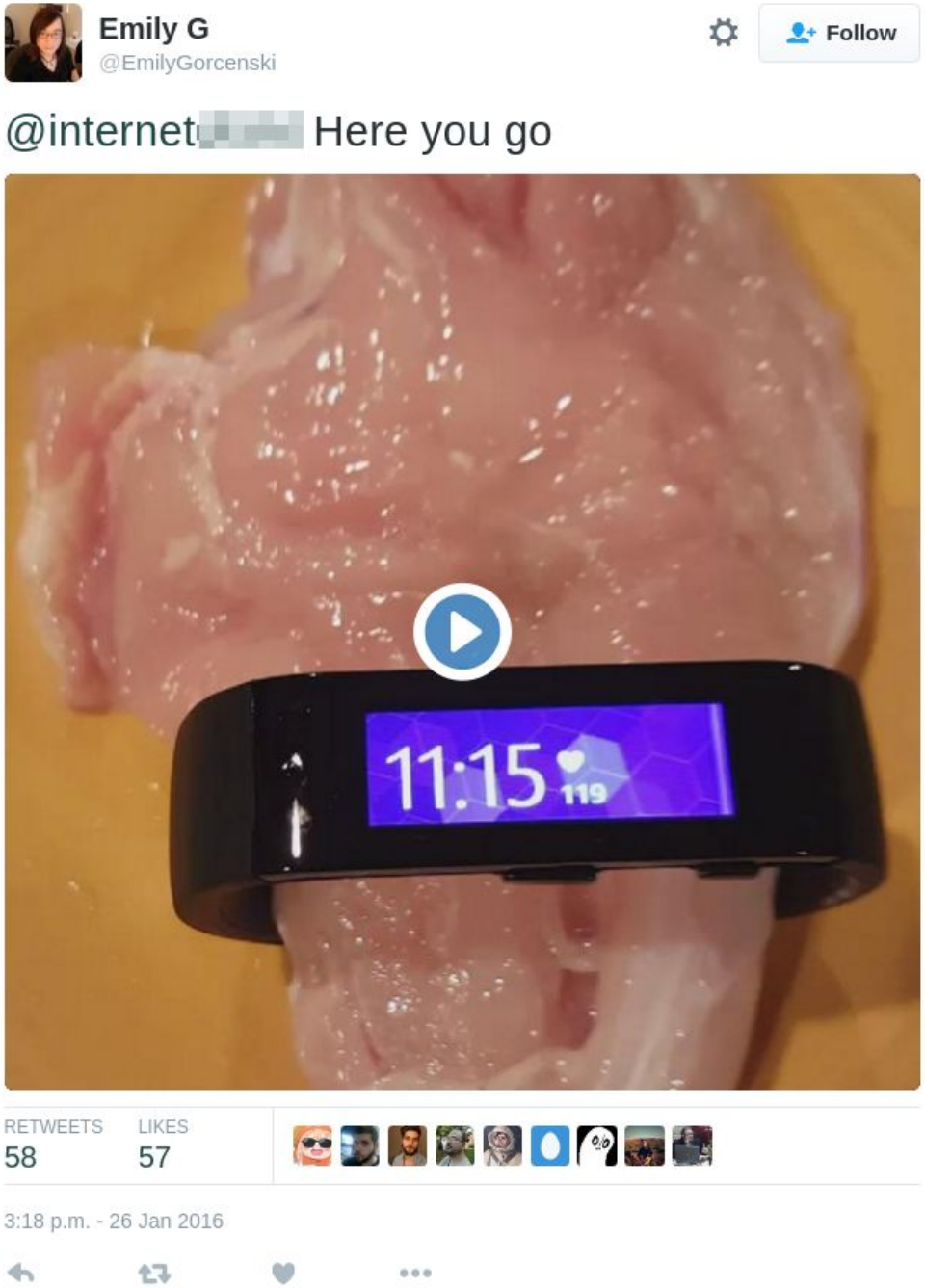
4G

In-Situ upgradeable devices

Yay for light bulb Updates !

Firmware updates

Update the software in your LIFX bulbs to unlock our latest features and improvements. Find out what's new.



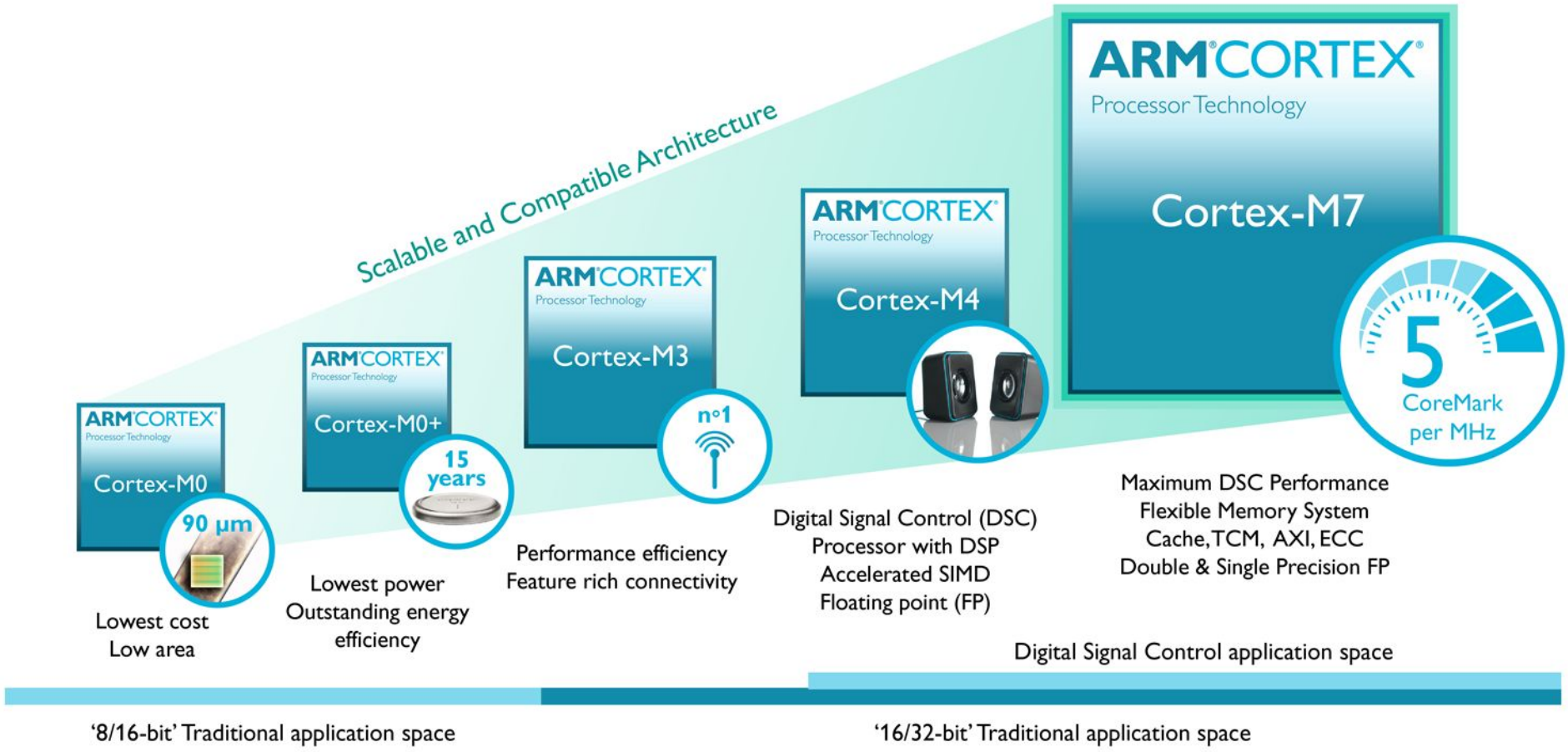
https://twitter.com/EmilyGorcenski/status/692003645437677568

Checkpoint #1

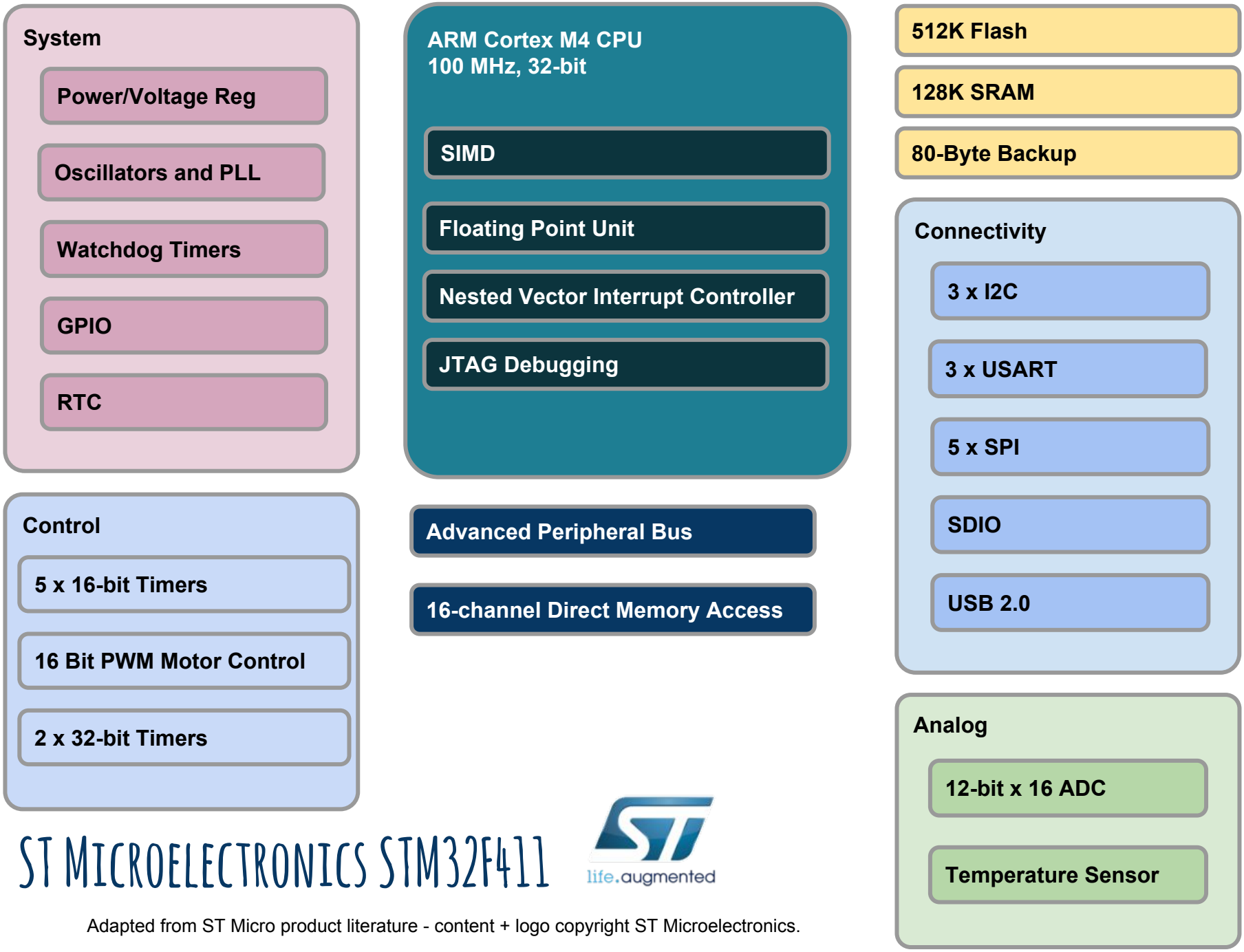
Are we cool with this so far?

- Software-defined system behaviour + connectivity enables product evolution
- We need code we can live with

MCU Example : ARM Cortex "M" family

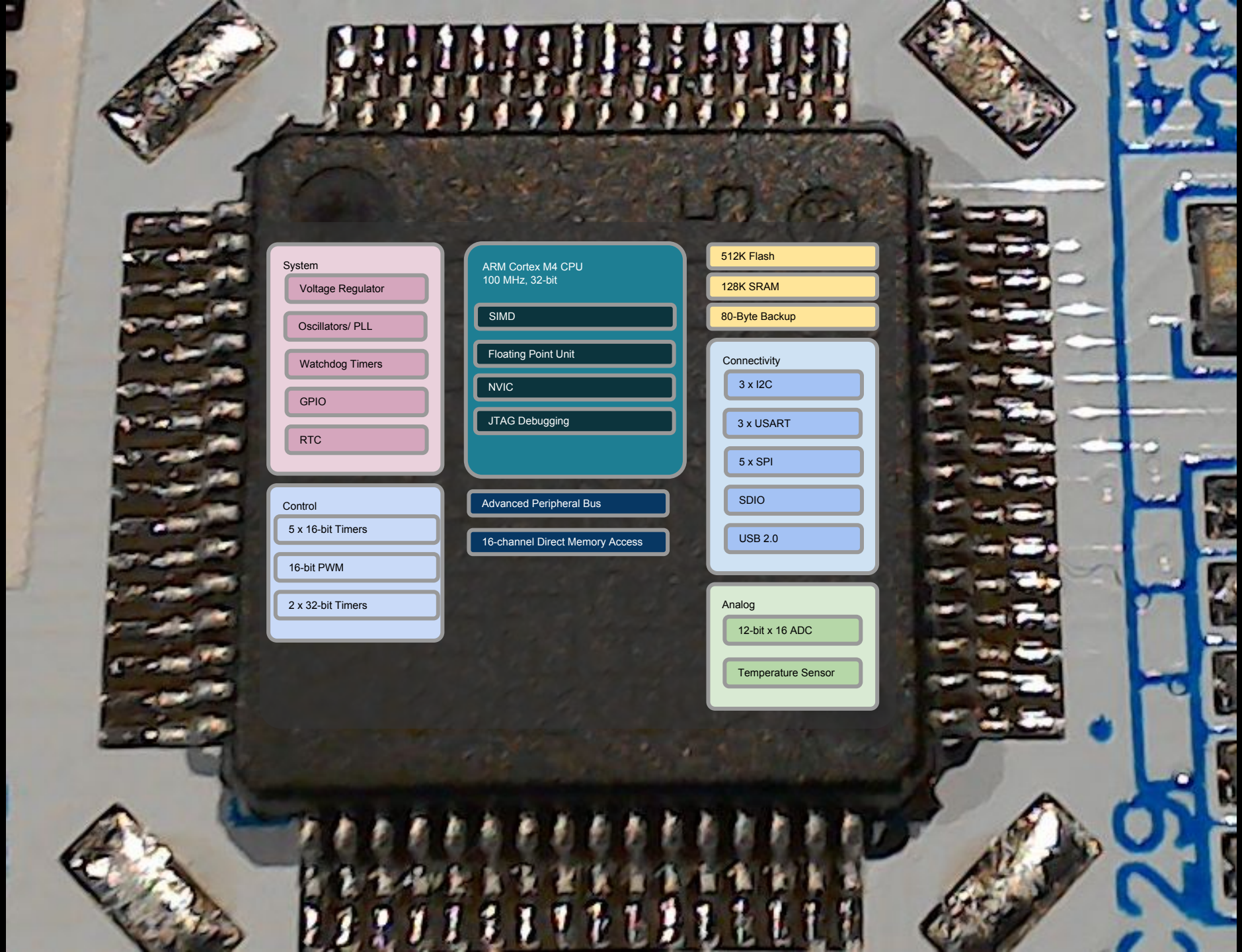


http://www.arm.com/assets/images/tpl/compare-Cortex-M-diagramLG.png, Copyright ARM,



ST Microelectronics STM32F411

life.augmented


System
Voltage Regulator
Oscillators/ PLL
Watchdog Timers
GPIO
RTC
Control
5 x 16-bit Timers
16-bit PWM
2 x 32-bit Timers
ARM Cortex M4 CPU
100 MHz, 32-bit
SIMD
Floating Point Unit
NVIC
JTAG Debugging
Advanced Peripheral Bus
16-channel Direct Memory Access
512K Flash
128K SRAM
80-Byte Backup
Connectivity
3 x I2C
3 x USART
5 x SPI
SDIO
USB 2.0
Analog
12-bit x 16 ADC
Temperature Sensor

OS Options, both free and proprietary

- Zephyr (Linux Foundation), FreeRTOS, RTEMS
- ARM mBed - Open License, but ARM-specific
- VxWorks (Wind River), QNX (BlackBerry)
- Embedded Linux (+Yocto Build, often)
- Windows ? (Embedded compact, Windows 10 IoT)

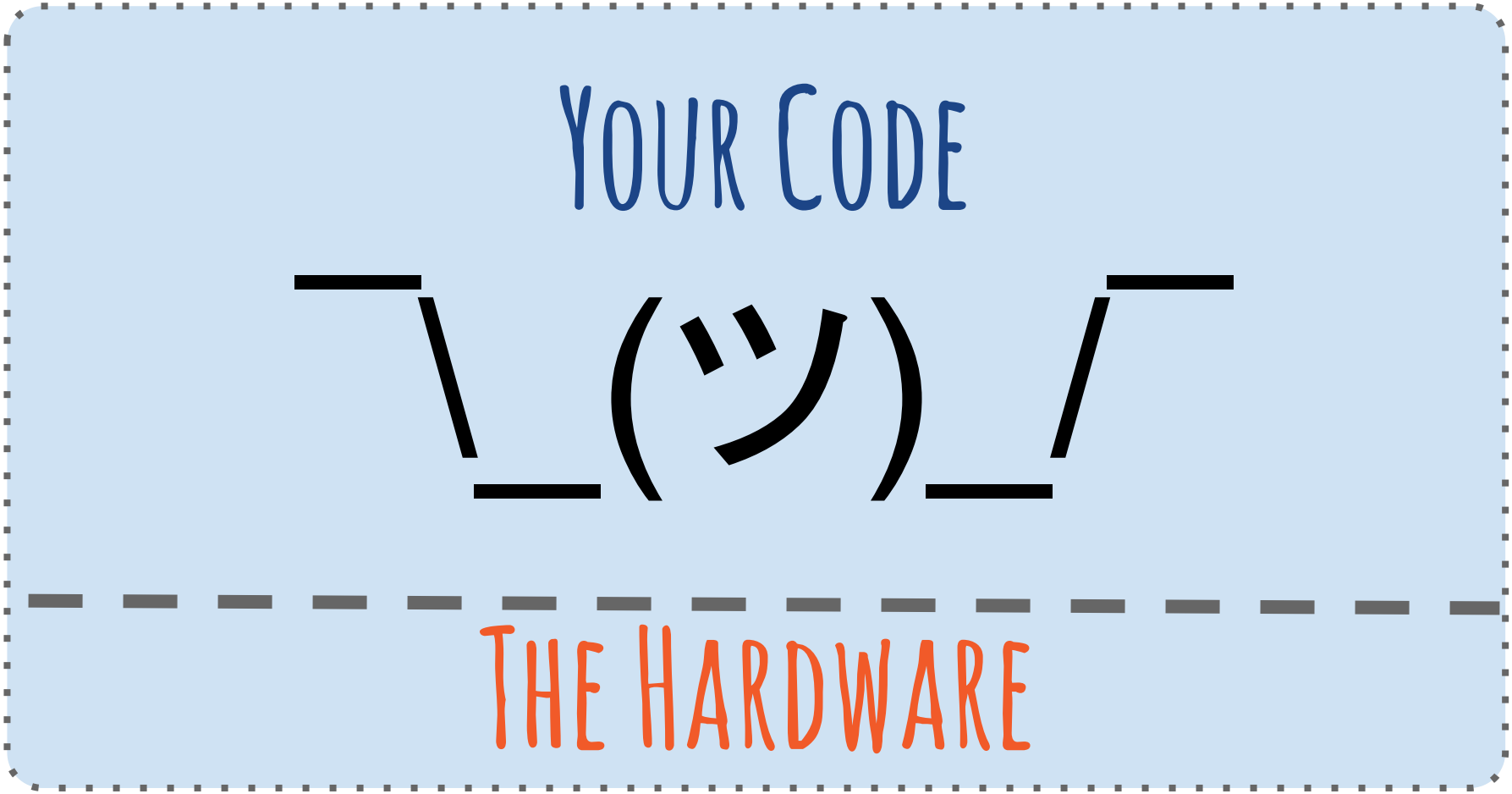
13coders

The Ever-So-Complex Architecture of Bare Metal Builds

Adapted from presentation on BitBox by Xavier Moulet, FOSDEM 2015

13coders

The Ever-So-Complex Architecture of Bare Metal Builds



Adapted from presentation on BitBox by Xavier Moulet, FOSDEM 2015

13coders

The Ever-So-Complex Architecture of Bare Metal Builds

Your Code

```
HAL_GPIO_Write
Pin()
```

The Hardware

Adapted from presentation on BitBox by Xavier Moulet, FOSDEM 2015

13coders

The Ever-So-Complex Architecture of Bare Metal Builds

Your Code

```
HAL_GPIO_Write
    Pin()
```

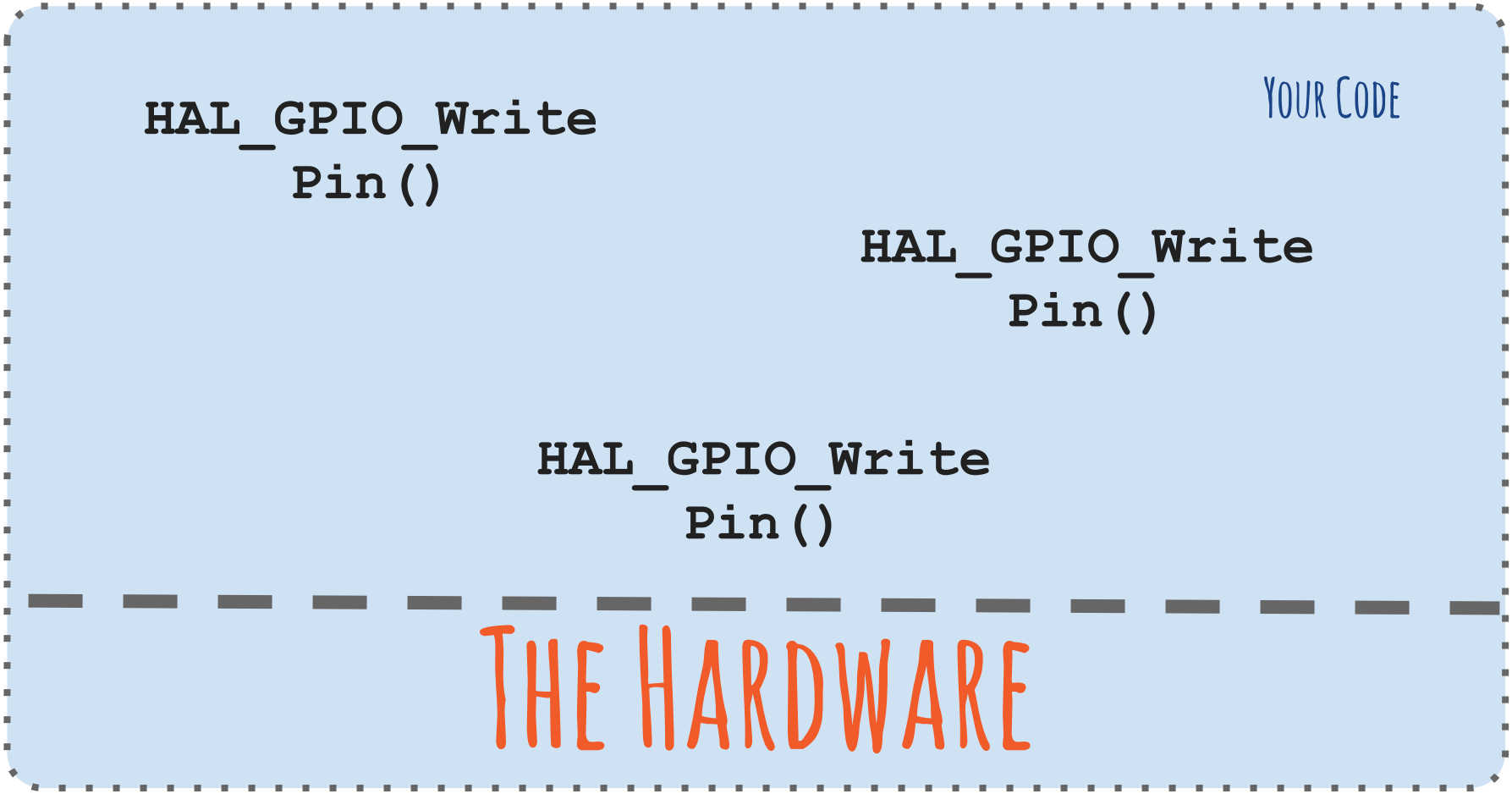
```
HAL_GPIO_Write
    Pin()
```

The Hardware

Adapted from presentation on BitBox by Xavier Moulet, FOSDEM 2015

13coders

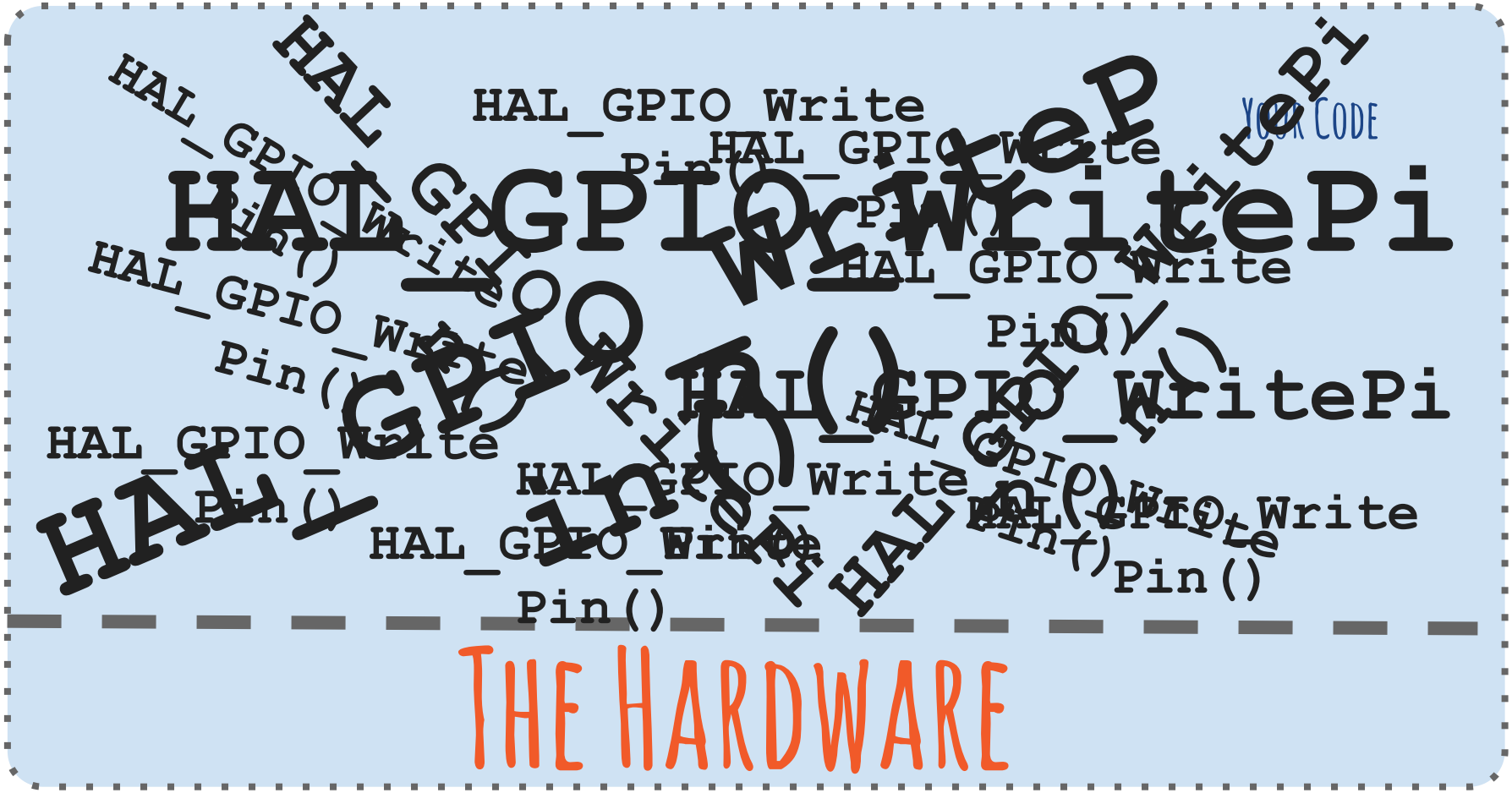
The Ever-So-Complex Architecture of Bare Metal Builds



Adapted from presentation on BitBox by Xavier Moulet, FOSDEM 2015

13coders

The Ever-So-Complex Architecture of Bare Metal Builds



Adapted from presentation on BitBox by Xavier Moulet, FOSDEM 2015

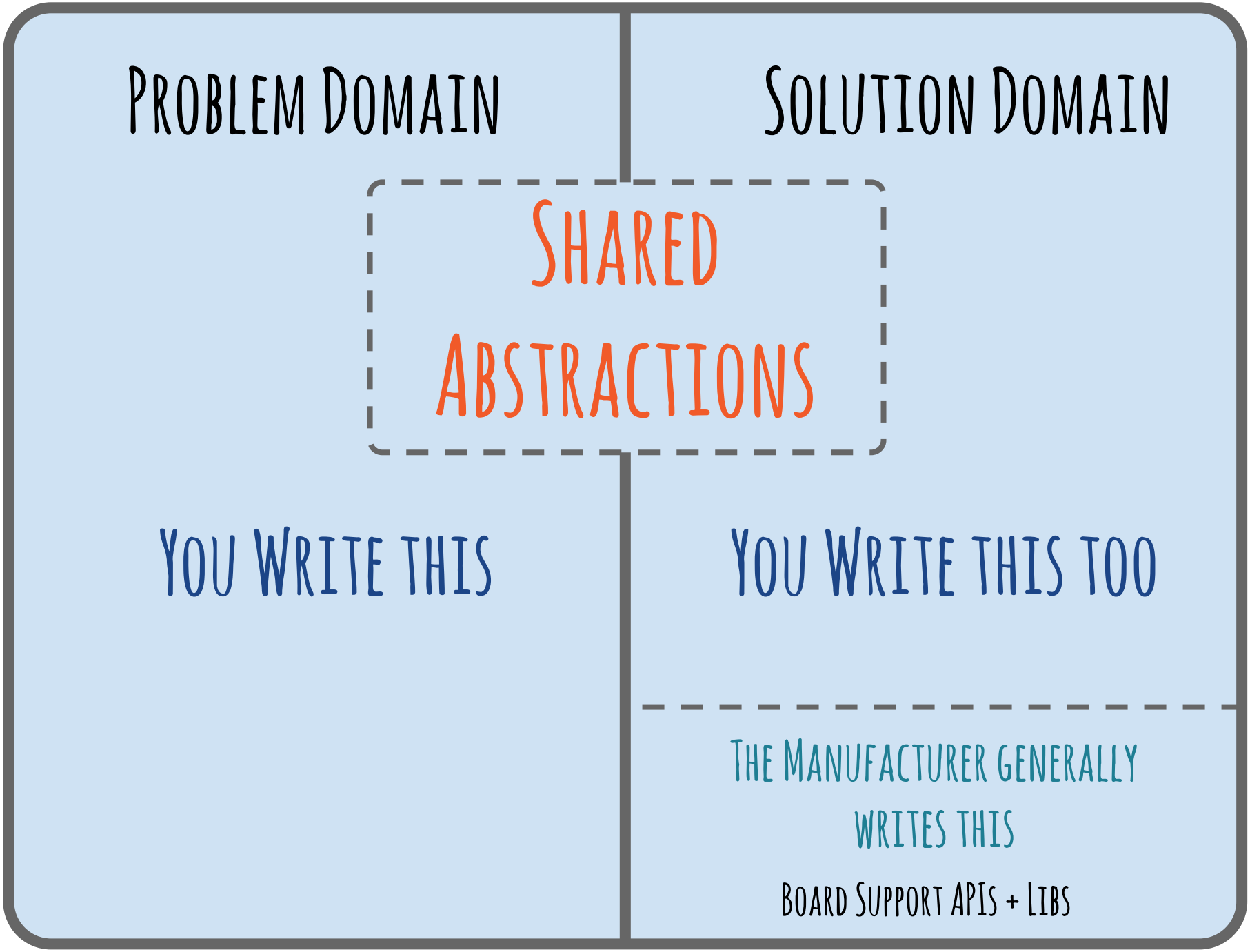
13coders

```
  /*Configure GPIO pins : USART_TX_Pin USART_RX_Pin */
  GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */
```

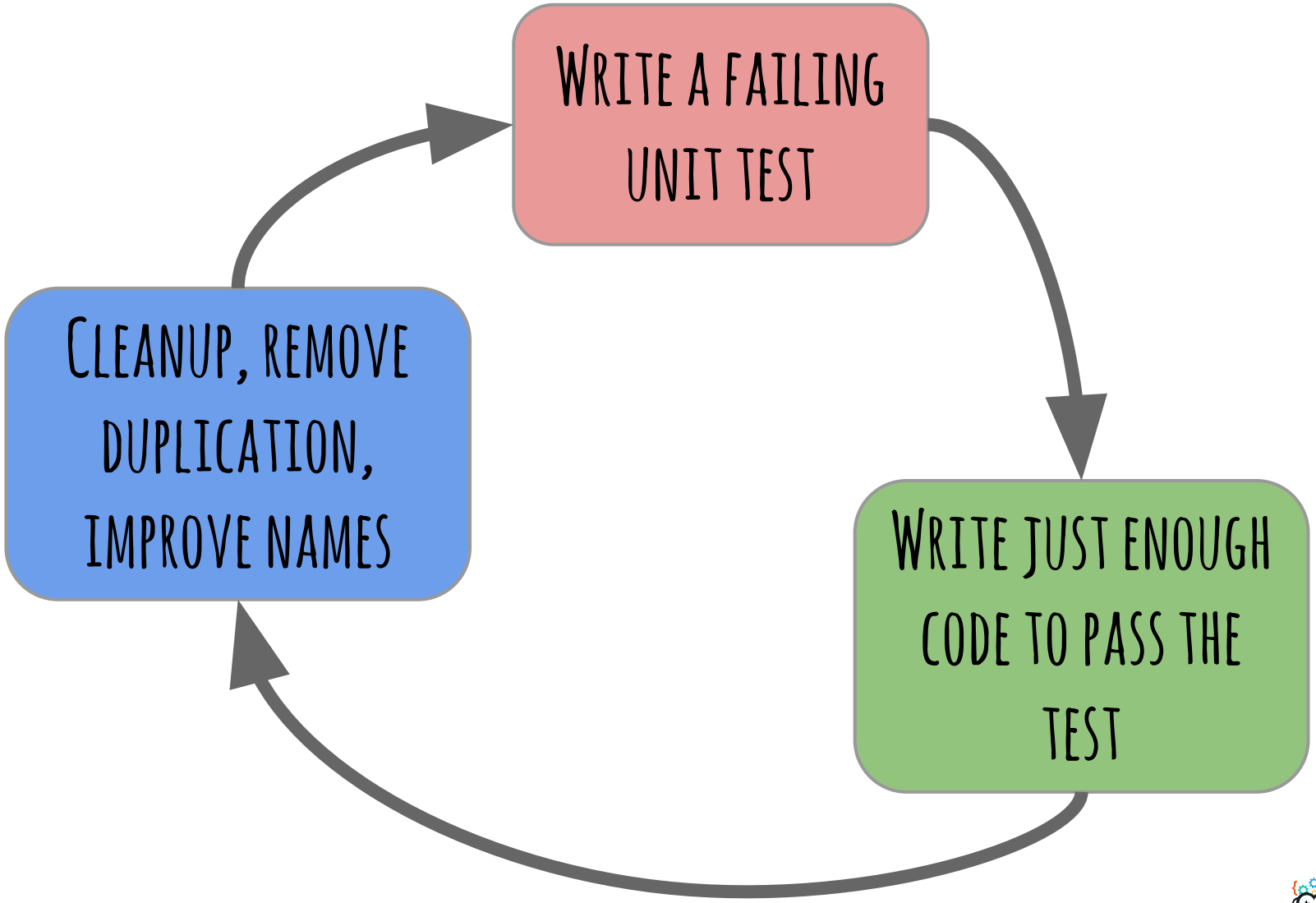
```
  /*Configure GPIO pins : USART_TX_Pin USART_RX_Pin */
  GPIO_InitStruct.Pin = USART_TX_Pin|USART_RX_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Con                 p      :  D2      */
  GPIO_                t      ru      in      2      in,
  GPIO_                t      r      t      oc      P      E      P      ;
  GPIO_                t      r      p      d      G      O.      E      LO
  HAL_G                                                                uct);

  /*Configure GPIO pin O      tput Level */
  HAL_GPIO_WritePin(      GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */
```

A fundamental Principle : Dependency Inversion

High-level modules should not depend on low-level modules.
Both should depend on abstractions.

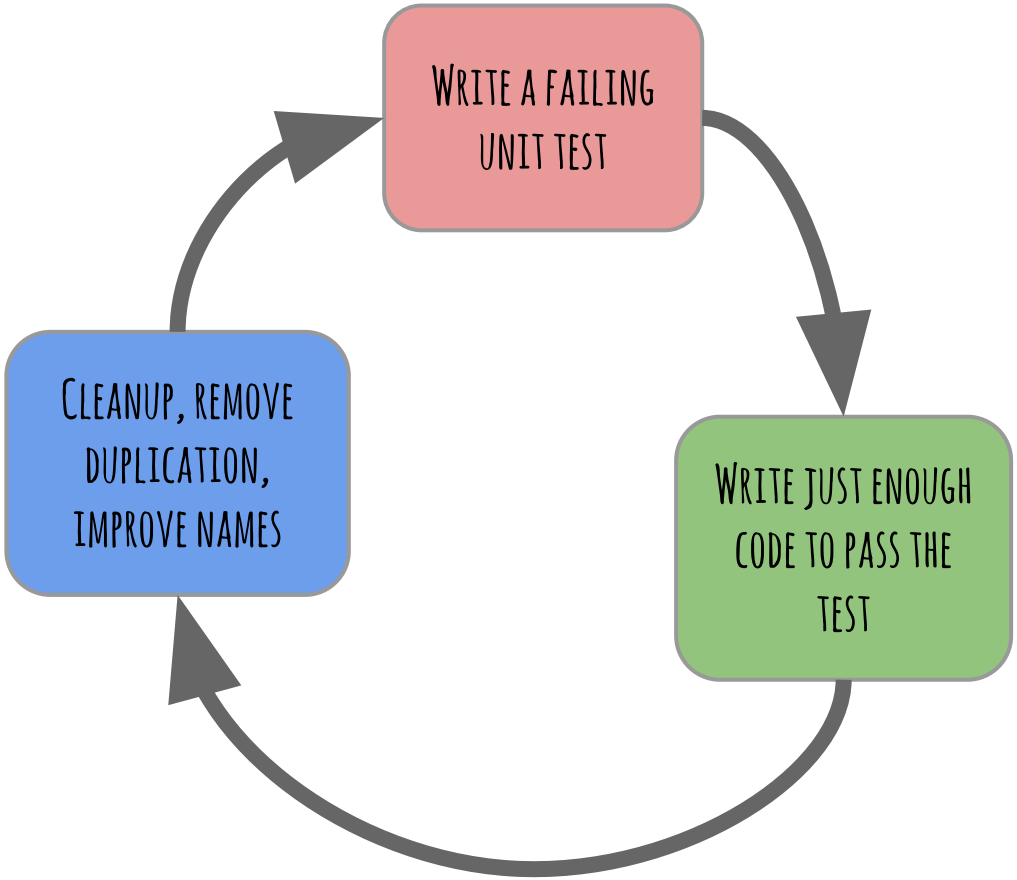

Problem Domain
Solution Domain
Shared Abstractions
You Write this
You Write this too
The Manufacturer generally writes this
Board Support APIs + Libs

The TDD Microcycle



13coders

On your development machine, constantly



On target hardware, less frequently

Build for the target

Run acceptance and unit tests on the target

Continuous Integration + TDD + Tools => Happy Devs

- "Continuous integration" without tests isn't Really continuous integration
- You don't get the same feedback with test-last
- Use dynamic analysis tools to leverage tests

Your Build system really wants to help : Let it

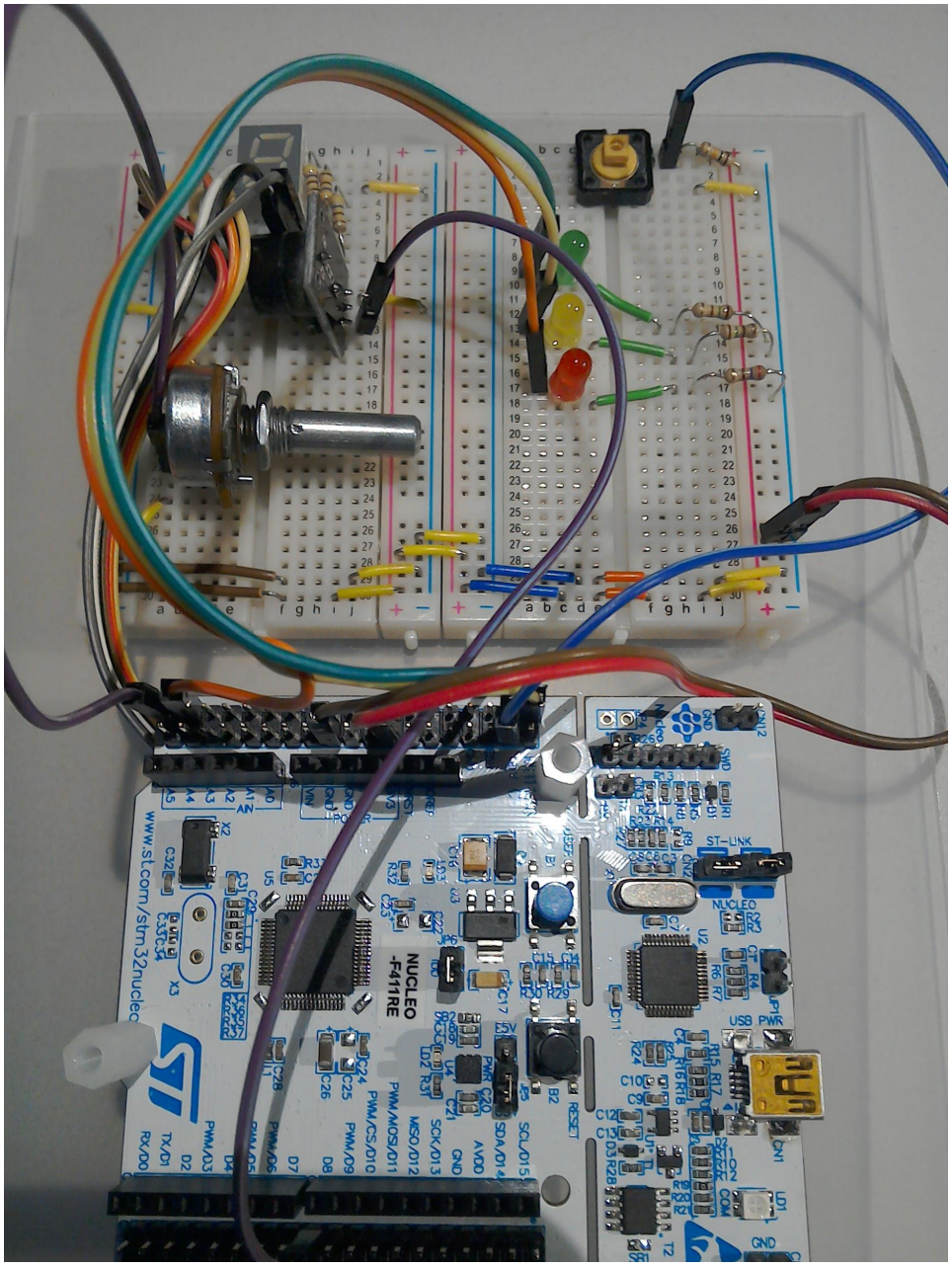
- Split your build to cleanly separate concerns
- Limit include paths : if you can't see it, you can't depend on it
  - Especially, don't make hardware headers available to your core "problem" code
- constrain dependencies via linker settings

13coders

Checkpoint #2

You're still here! That's amazing!

- We're looking at MCUs and bare-metal
- The design is whatever we make it
- Principles help

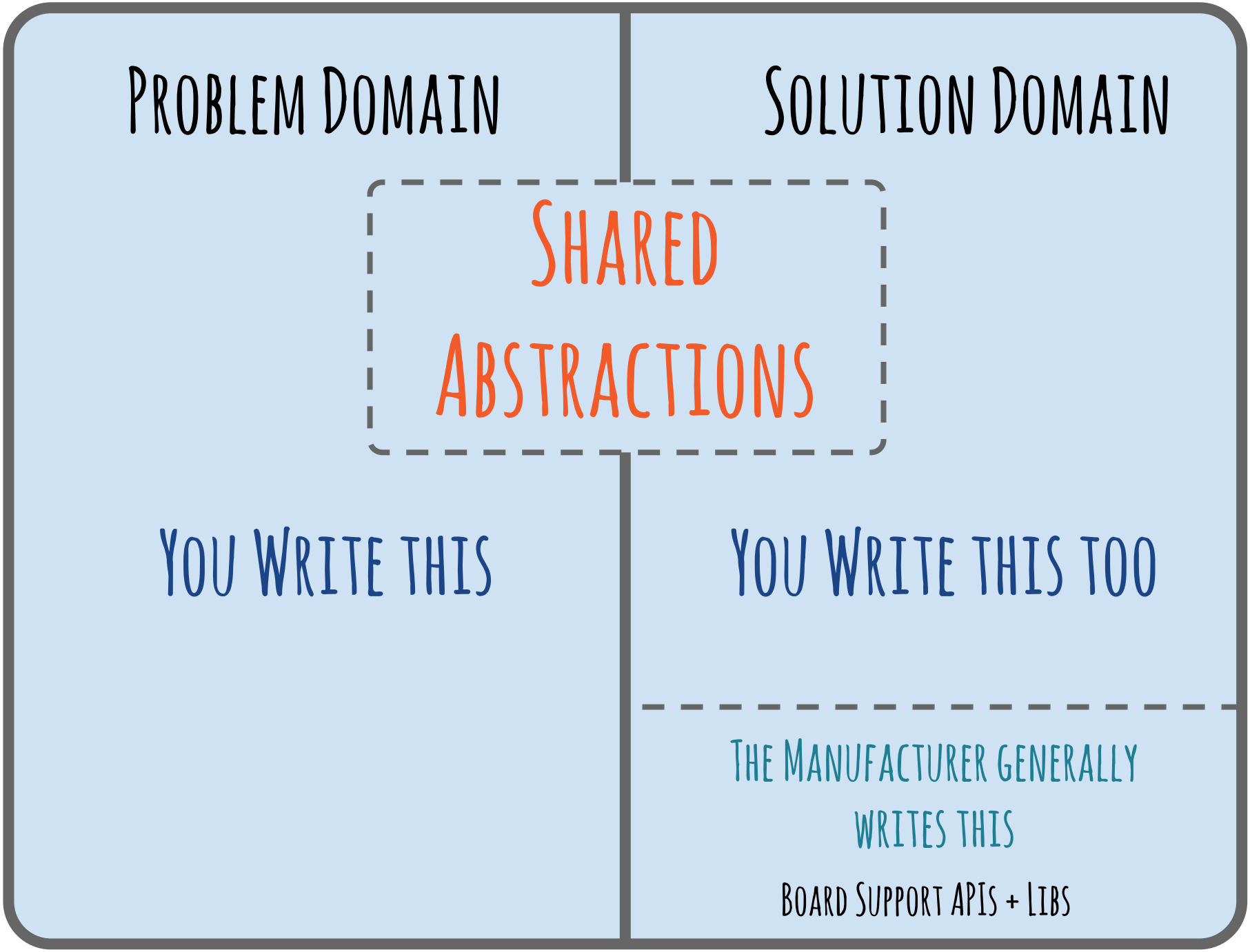


Exclusively for Tech Meetup

The only Fire Alarm system you will ever need.

Going on Kickstarter any day now. Seriously.

13coders


Problem Domain
Solution Domain
Shared Abstractions
You Write this
You Write this too
The Manufacturer generally writes this
Board Support APIs + Libs

Checkpoint #3

Well, that was...codey....

- TDD works in any language...yes, even C++
- It's dependencies all the way down
- There are always obstacles. Overcome them.

Safety critical, deterministic

C & assembler
Few or no libs
zero dynamic memory use

Subset of C++
Some Standard library
dynamic memory, but not "after the wheels leave the ground"

Low CPU/memory

High CPU/memory

C++
Little dynamic memory use

C++
Standard Library
3rd Party libs
Dynamic memory
Garbage coll. language?

"Trivial Purpose"

13coders

When Embedded Goes Wrong - a tale of woe

An Embedded automotive application cited as a possible cause of "unintended acceleration". A NASA study found:

11,253 read/write global variables

Cyclomatic Complexity of 146 in a key function

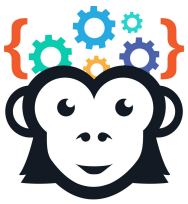
...With no unit test plan...

"It is far, far easier to make a correct program fast than it is to make a fast program correct"

-- Herb Sutter & Andrei Alexandrescu

13coders

Thanks for Listening!



13coders

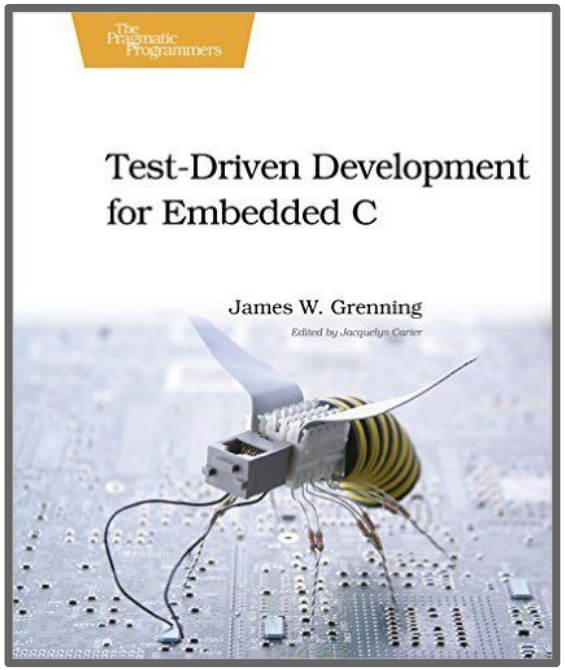
https://uk.linkedin.com/in/13coders

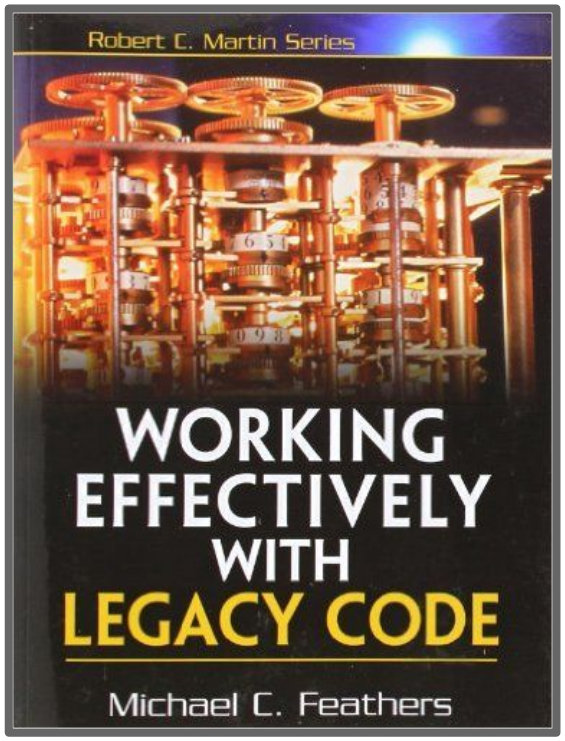
@13coders

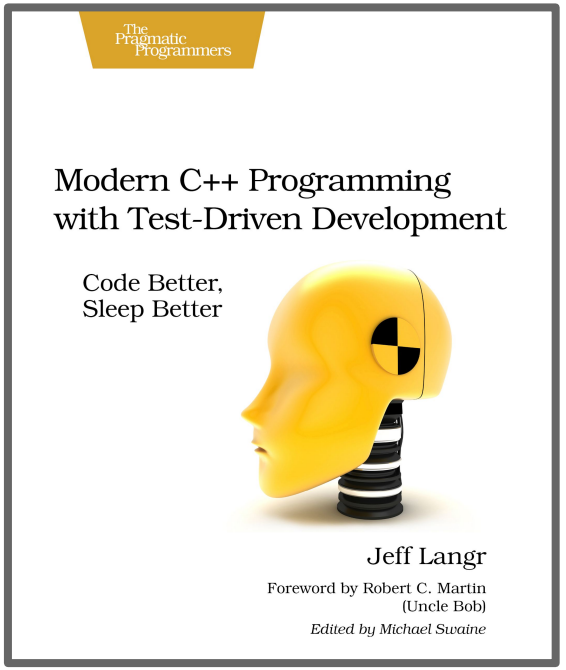
mike@13coders.com

+44 7808 480387

"Oh, *finally*...I thought this guy would never stop talking"


The Pragmatic Programmers
Test-Driven Development for Embedded C
James W. Grenning
Edited by Jacquelyn Carter


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers


The Pragmatic Programmers
Modern C++ Programming with Test-Driven Development
Code Better, Sleep Better
Jeff Langr
Foreword by Robert C. Martin (Uncle Bob)
Edited by Michael Swaine