Czego możemy nauczyć się od programistów wysokopoziomowych?

O mnie

Solw'IT Let's Solve It

UCGOSU.PL

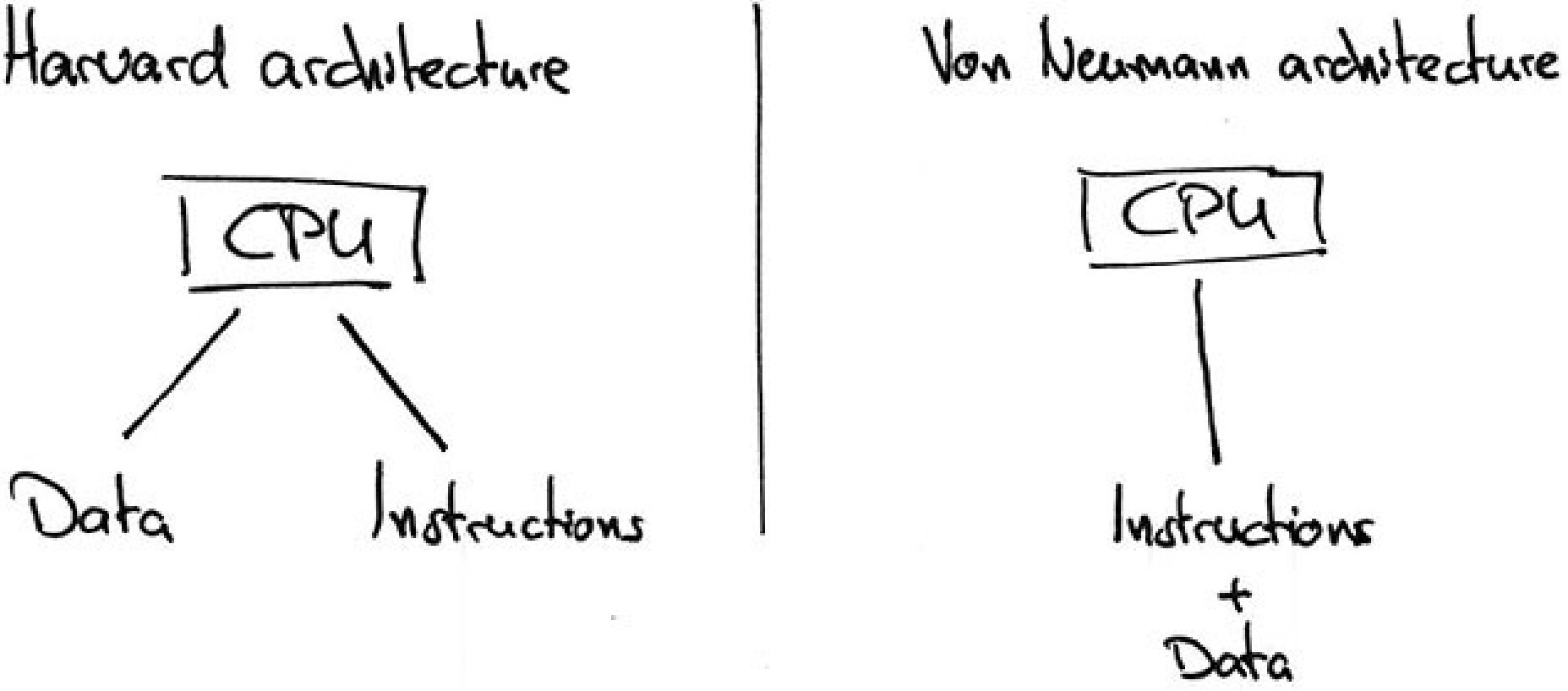
Programowanie i Robotyka

@MaciekGajdzica

Atmega

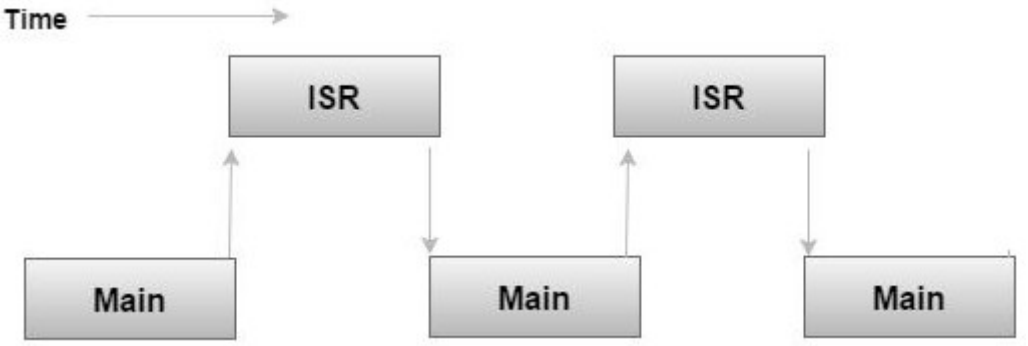
- Clock 16 MHz
- FLASH 32 kB
- RAM 2 kB

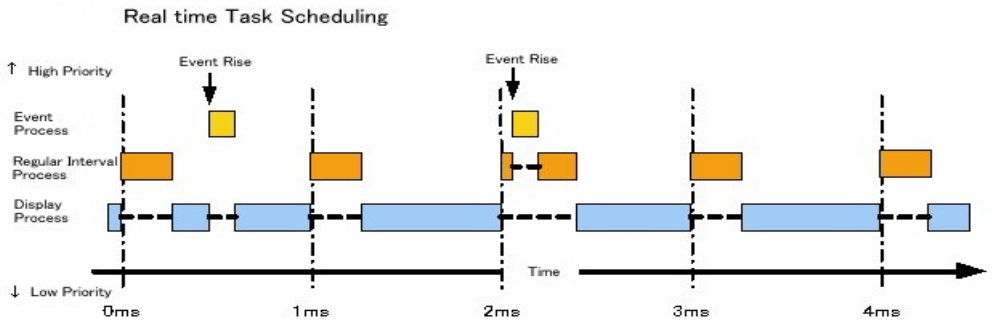
Architektura embedded?



Architektura embedded?

```c
#include <avr/io.h>

int main(void)
{
   DDRD  = 0xff;

   DDRC  = 0x00;
   PORTC = 0x03;

   while(1)
   {
     if(!(PINC & 0x01))
     {
         PORTD = 0xF0;
     }

     if(!(PINC & 0x02))
     {
         PORTD = 0x0F;
     }
   }
}
```



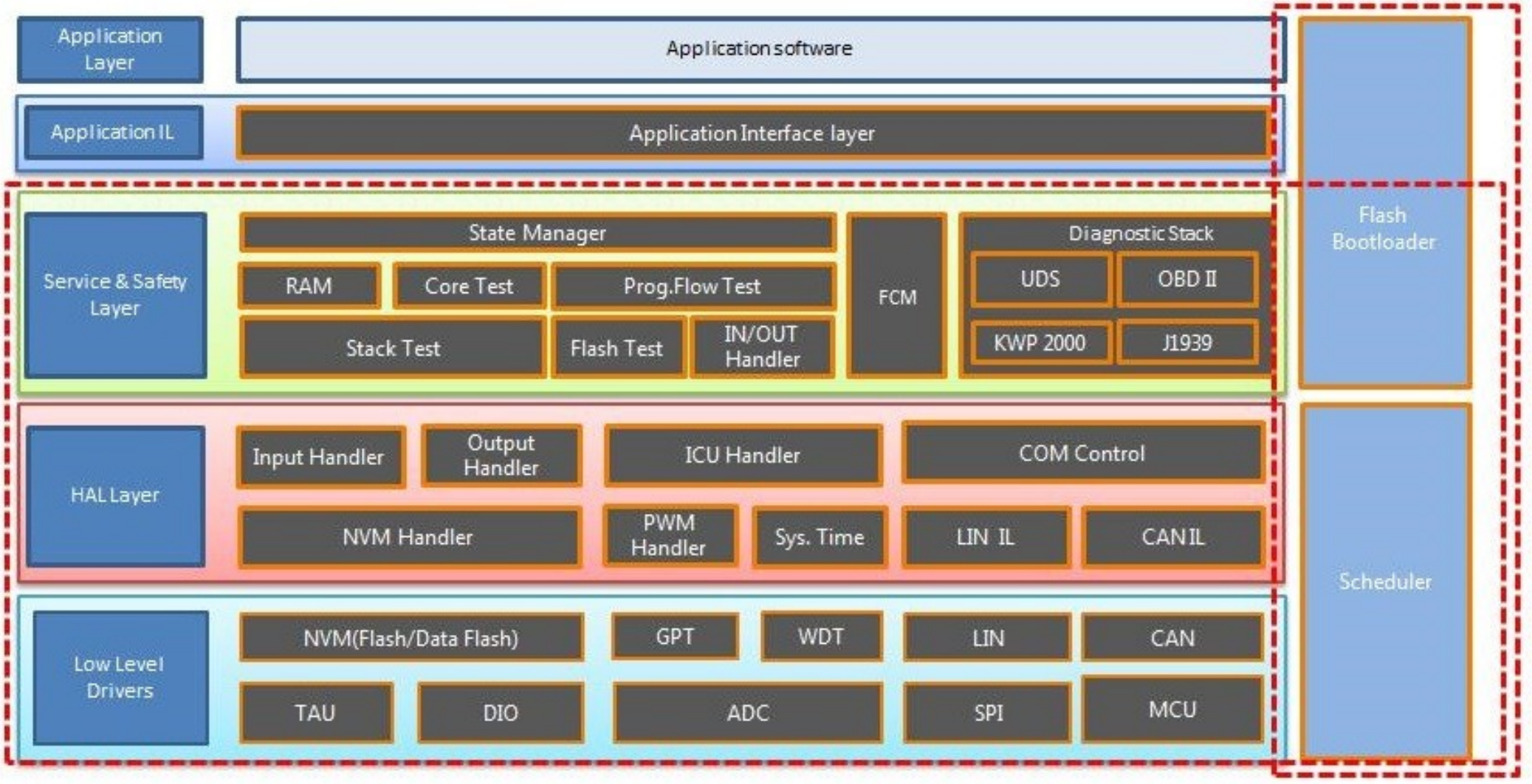


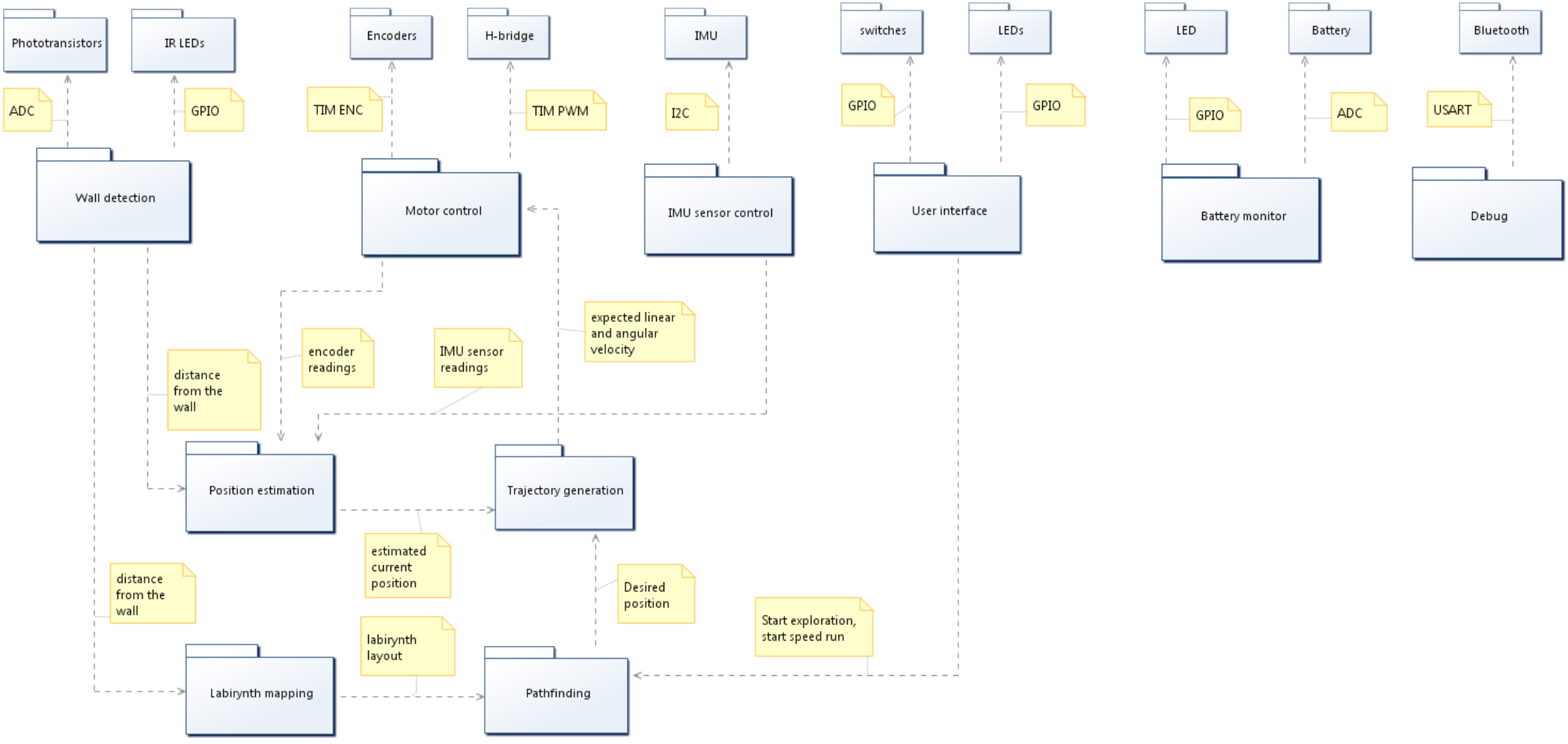
Architektura embedded

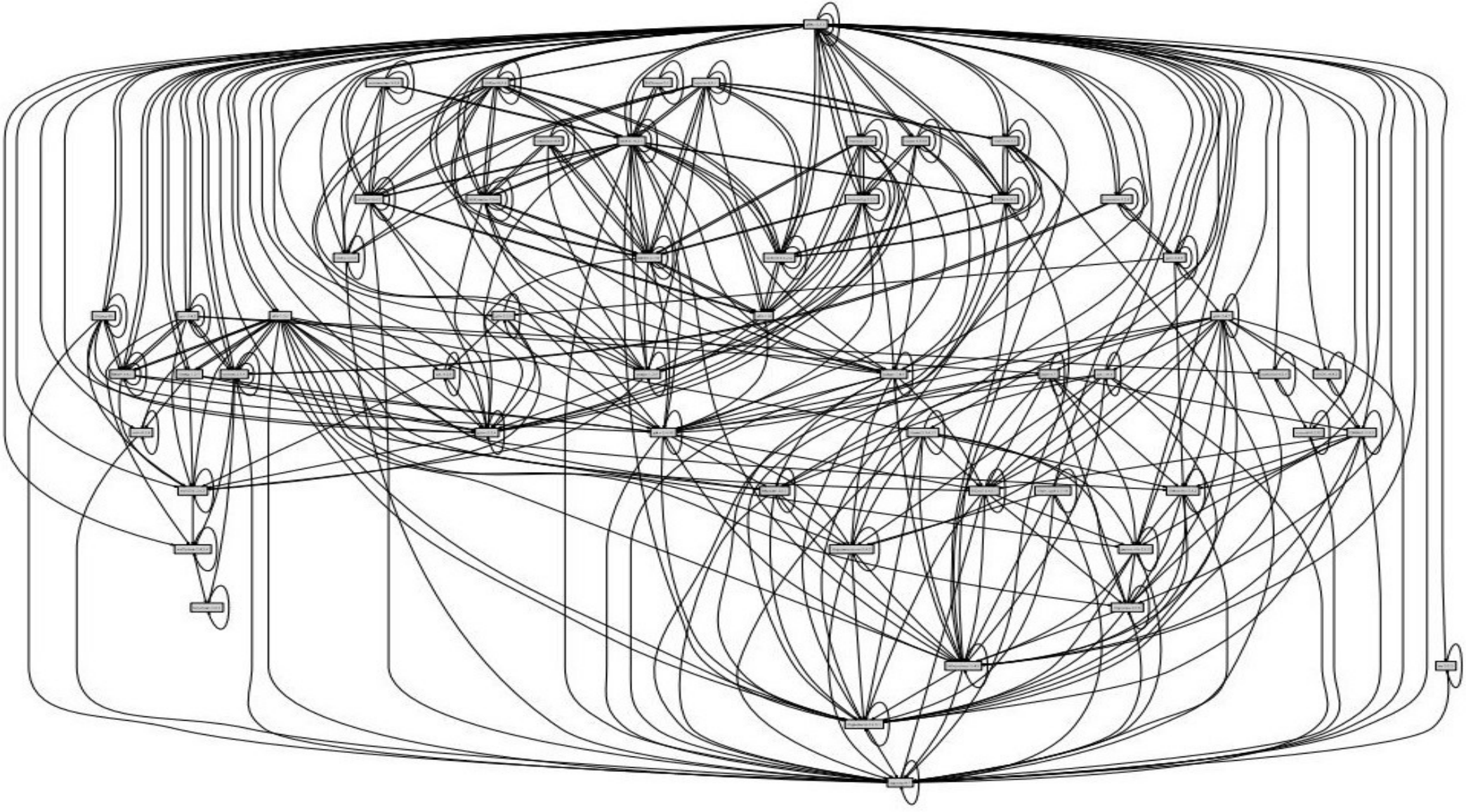
STM32

- Clock 180 MHz
- FLASH 2048 kB
- RAM 256 kB

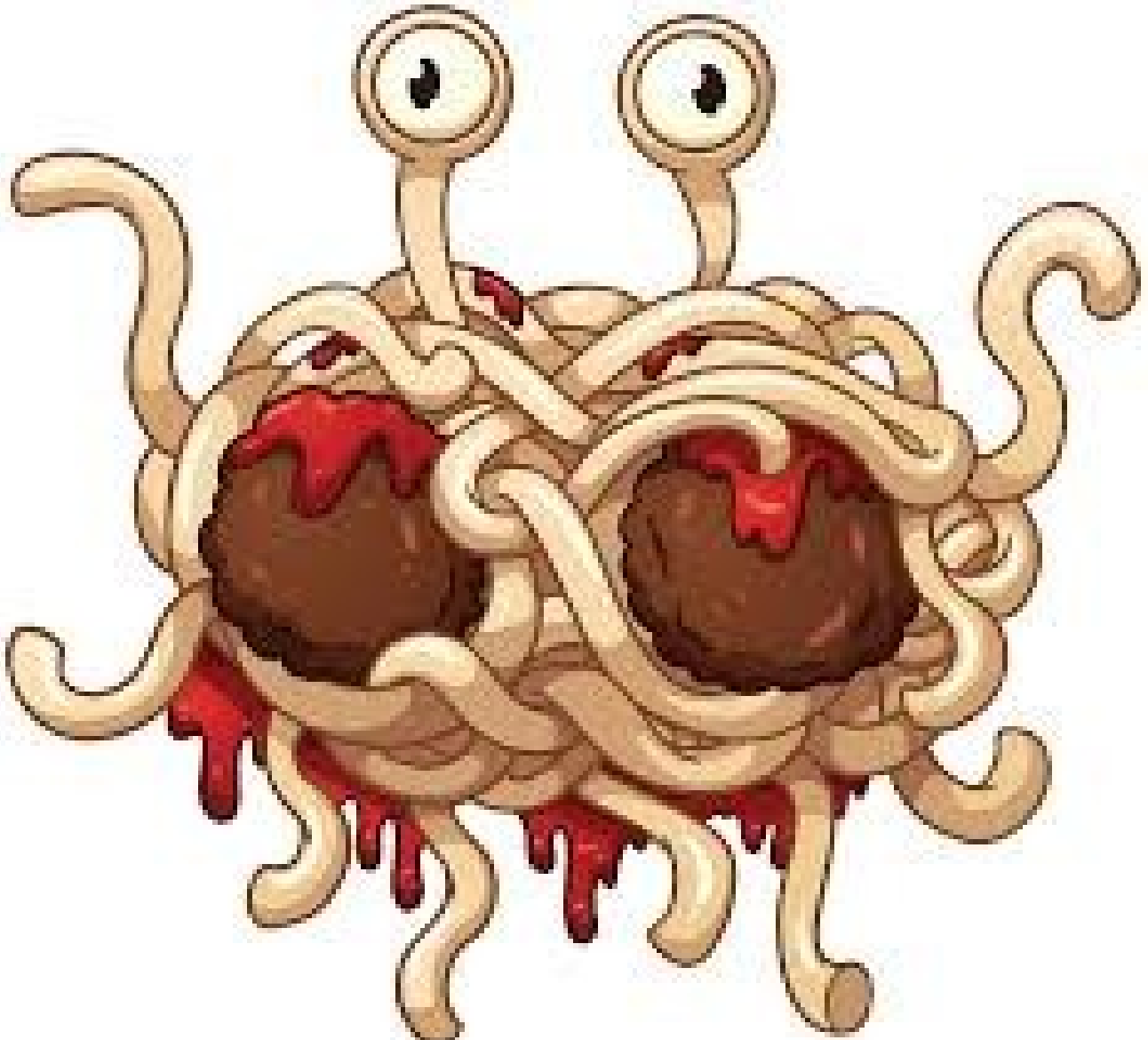
Software Architecture




Phototransistors
IR LEDs
Encoders
H-bridge
IMU
switches
LEDs
LED
Battery
Bluetooth
ADC
GPIO
TIM ENC
TIM PWM
I2C
GPIO
GPIO
GPIO
ADC
USART
Wall detection
Motor control
IMU sensor control
User interface
Battery monitor
Debug
expected linear and angular velocity
encoder readings
IMU sensor readings
distance from the wall
Position estimation
Trajectory generation
estimated current position
distance from the wall
Desired position
Start exploration, start speed run
labirynth layout
Labirynth mapping
Pathfinding

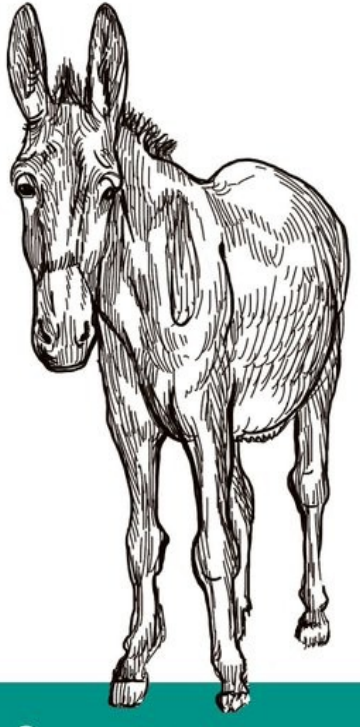



UniCredit
SONY
Ford
PS3
UEFA.com
POR QUÉ?



```
void  Sim80x_SetPower(bool TurnOn)
{
  if(TurnOn==true)
  {
    if(Sim80x_SendAtCommand("AT\r\n",200,1,"AT\r\r\nOK\r\n") == 1)
    {
      osDelay(100);
      #if (_SIM80X_DEBUG==1)
      printf("\r\nSim80x_SetPower(ON) ---> OK\r\n");
      #endif
      Sim80x.Status.Power=1;
      Sim80x_InitValue();
    }
    else
    {
      #if (_SIM80X_USE_POWER_KEY==1)
      HAL_GPIO_WritePin(_SIM80X_POWER_KEY_GPIO,_SIM80X_POWER_KEY_PIN,GPIO_PIN_RESET);
      osDelay(1200);
      HAL_GPIO_WritePin(_SIM80X_POWER_KEY_GPIO,_SIM80X_POWER_KEY_PIN,GPIO_PIN_SET);
      #endif
      osDelay(3000);
      if(Sim80x_SendAtCommand("AT\r\n",200,1,"AT\r\r\nOK\r\n") == 1)
      {
        osDelay(10000);
        #if (_SIM80X_DEBUG==1)
        printf("\r\nSim80x_SetPower(ON) ---> OK\r\n");
        #endif
        Sim80x.Status.Power=1;
        Sim80x_InitValue();
      }
```

Where's the fun in just knowing what the code is supposed to do?



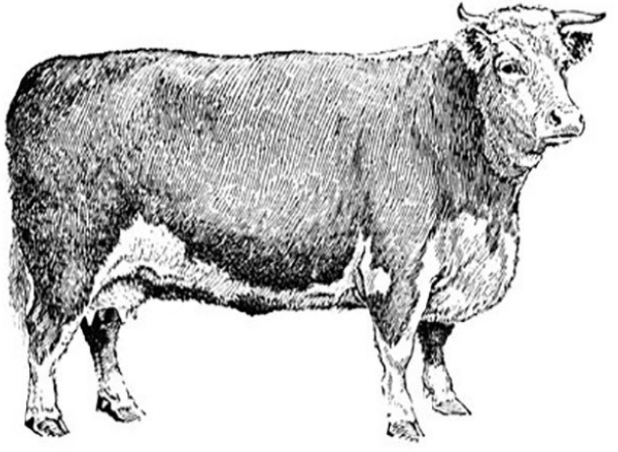
Essential

Excuses for Not Writing Documentation

O RLY?

@ThePracticalDev

No comments, no documentation but 20 tickets



The Guy Who Wrote This Is Gone

It's running everywhere

O RLY?

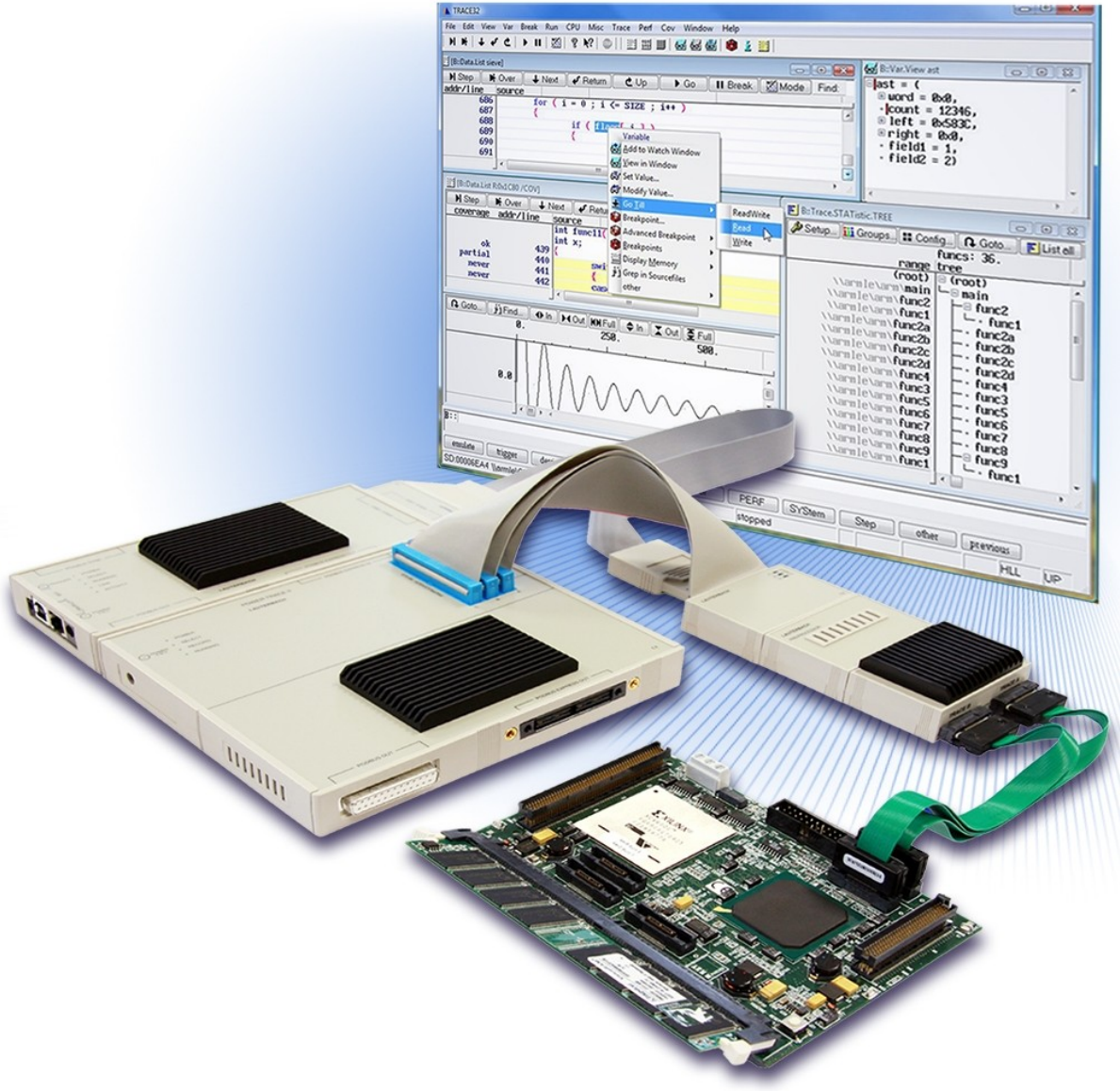
FML

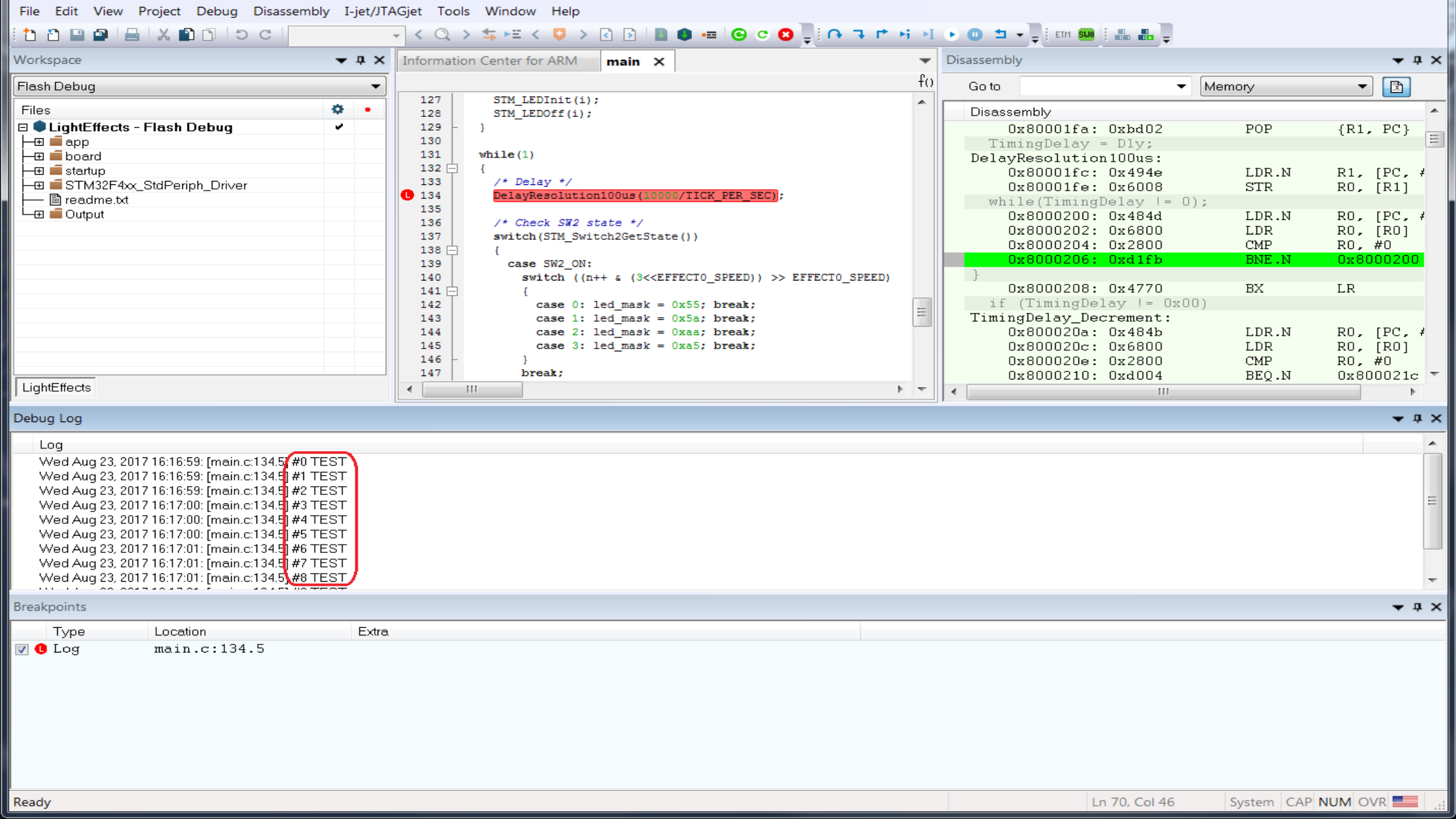
Ok it compiles, so let’s get to the real work - debugging

Zaawansowane debuggery sprzętowe
- Call stack
- Trace
- Step back
- Print log

Nowoczesne kompilatory
- Dodatkowe optymalizacje
- Wsparcie MISRA
- Dodatkowe warningi

Dodatkowy sprzęt
- Oscyloskop
- Analizator logiczny
- Sniffer
- Generator sygnałów




File Edit View Project Debug Disassembly I-jet/JTAGjet Tools Window Help
Workspace
Flash Debug
Files
LightEffects - Flash Debug
app
board
startup
STM32F4xx_StdPeriph_Driver
readme.txt
Output
LightEffects
Information Center for ARM
main
127 STM_LEDInit(i);
128 STM_LEDOff(i);
129 }
130
131 while(1)
132 {
133 /* Delay */
134 DelayResolution100us(10000/TICK_PER_SEC);
135
136 /* Check SW2 state */
137 switch(STM_Switch2GetState())
138 {
139 case SW2_ON:
140 switch ((n++ & (3<<EFFECT0_SPEED)) >> EFFECT0_SPEED)
141 {
142 case 0: led_mask = 0x55; break;
143 case 1: led_mask = 0x5a; break;
144 case 2: led_mask = 0xaa; break;
145 case 3: led_mask = 0xa5; break;
146 }
147 break;
Disassembly
Go to
Memory
Disassembly
0x80001fa: 0xbd02 POP {R1, PC}
TimingDelay = Dly;
DelayResolution100us:
0x80001fc: 0x494e LDR.N R1, [PC, #
0x80001fe: 0x6008 STR R0, [R1]
while(TimingDelay != 0);
0x8000200: 0x484d LDR.N R0, [PC, #
0x8000202: 0x6800 LDR R0, [R0]
0x8000204: 0x2800 CMP R0, #0
0x8000206: 0xd1fb BNE.N 0x8000200
}
0x8000208: 0x4770 BX LR
if (TimingDelay != 0x00)
TimingDelay_Decrement:
0x800020a: 0x484b LDR.N R0, [PC, #
0x800020c: 0x6800 LDR R0, [R0]
0x800020e: 0x2800 CMP R0, #0
0x8000210: 0xd004 BEQ.N 0x800021c
Debug Log
Log
Wed Aug 23, 2017 16:16:59: [main.c:134.5] #0 TEST
Wed Aug 23, 2017 16:16:59: [main.c:134.5] #1 TEST
Wed Aug 23, 2017 16:16:59: [main.c:134.5] #2 TEST
Wed Aug 23, 2017 16:17:00: [main.c:134.5] #3 TEST
Wed Aug 23, 2017 16:17:00: [main.c:134.5] #4 TEST
Wed Aug 23, 2017 16:17:00: [main.c:134.5] #5 TEST
Wed Aug 23, 2017 16:17:01: [main.c:134.5] #6 TEST
Wed Aug 23, 2017 16:17:01: [main.c:134.5] #7 TEST
Wed Aug 23, 2017 16:17:01: [main.c:134.5] #8 TEST
Breakpoints
Type Location Extra
Log main.c:134.5
Ready
Ln 70, Col 46
System
CAP NUM OVR


WE NEED TO GO
DEEPER

Nie studiowaliśmy informatyki

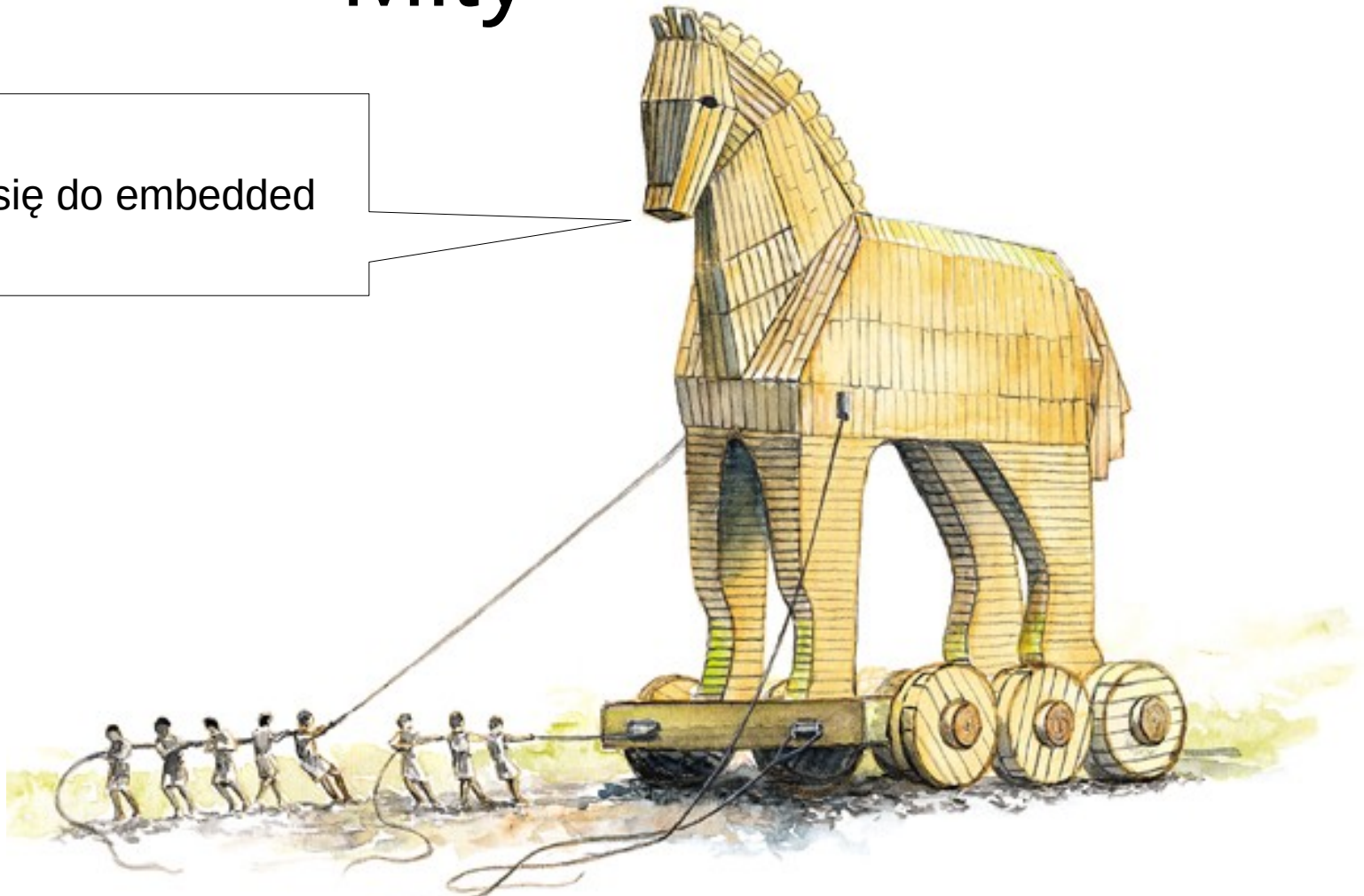
- Elektronika
- Automatyka
- Mechatronika
- Technikum

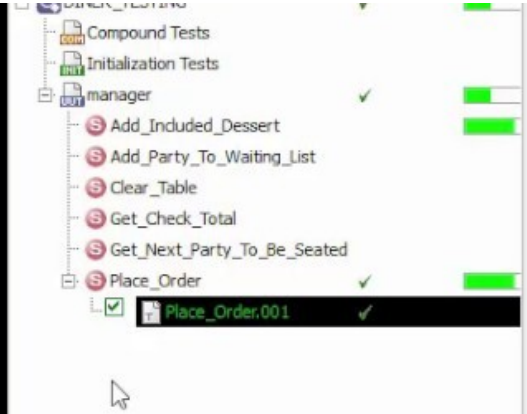
Różne obszary wiedzy

- Programowanie
- Elektronika
- Układy, peryferia
- Budowa procesora
- Wiedza domenowa

Mity

C++ nie nadaje się do embedded





Event 2			
Parameter	Type	Actual Value	Expected Value
Stub called: manager.c			
Subprogram: Get_Table_Record			

Event 3			
Parameter	Type	Actual Value	Expected Value
Stub called: manager.c			
Subprogram: Update_Table_Record			
Data			
Is_Occupied	enum	v_true	match
Number_In_Party	unsigned short	1	match
Check_Total	float	14	match

Event 4			
Parameter	Type	Actual Value	Expected Value
Returned from UUT: manager.c			
Subprogram: Place_Order			
Table	unsigned short	1	
Seat	unsigned short	1	
Order			
Entree	enum	STEAK	

-- End of Test Case --

Test Status	
Expected Results matched 100%	(3/3) PASS
Test Status	PASS

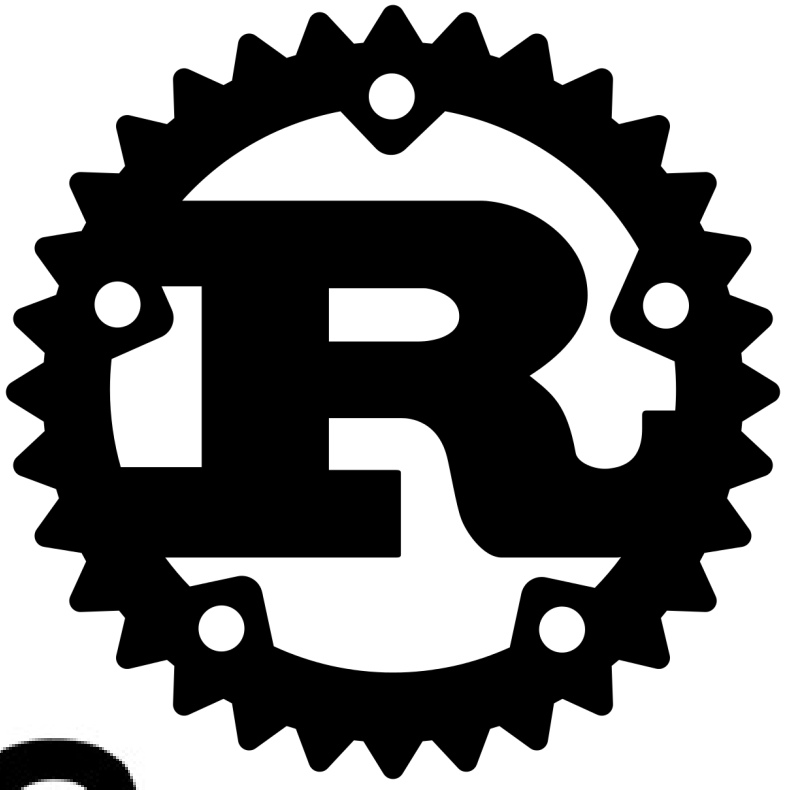
Col: 0 Line: 1

✓ Parameter Tree | Control Flow | ✓ Execution Report

Project Requirements

Test Case Requirements

ID	Title
DINE-79	Placing an order updates occupied status
DINE-80	Placing an order updates number in party
DINE-205	Steak Order is $14




Ada
2012


B1
B2
B3

First things first

- Solidna baza wiedzy
- Zrozumienie domeny, technologii, problemów
- Umiejętności projektowania systemu i jego części
- Jakość kodu – refactoring, review, clean code, wzorce
- Prowadzenie projektu – współpraca, planowanie, wymagania, testowanie, automatyzacja

BEST PRACTICES

CODE COMPLETE

Second Edition 2

McConnell

Microsoft Press

CODE COMPLETE

Microsoft

2 Second Edition

A practical handbook of software construction

Steve McConnell

Two-time winner of the Software Development Magazine Jolt Award

The C4 model

System Context

The system plus users and system dependencies

Containers

The overall shape of the architecture and technology choices

Components

Components and their interactions within a container

Classes (or Code)

Component implementation details

C4 – dodatkowe materiały

- https://c4model.com/
- https://structurizr.com/
- Simon Brown - Visualising software architecture with the C4 model
- Sławomir Sobótka - C4 - lekkie podejście do dokumentowania architektury

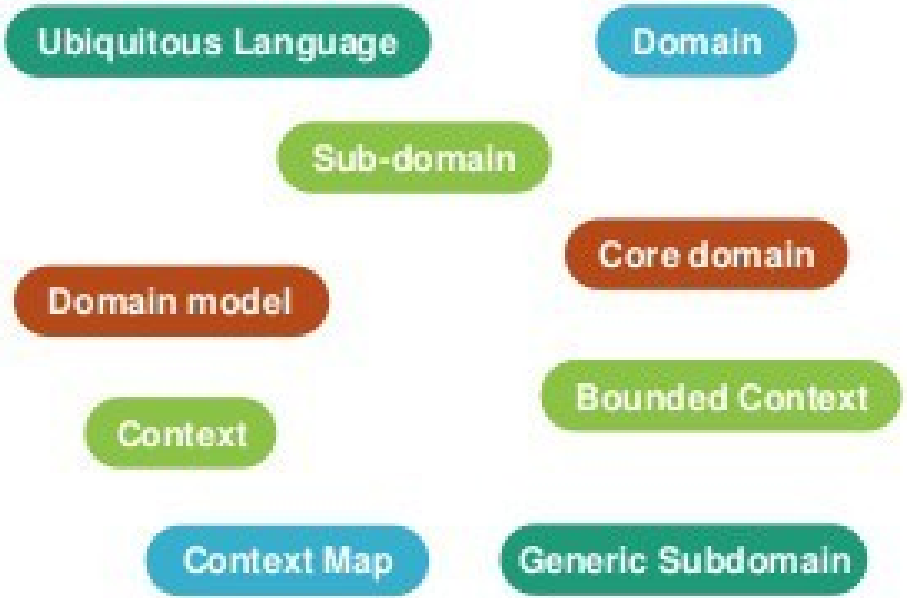
Event Storming

Event Storming - materiały

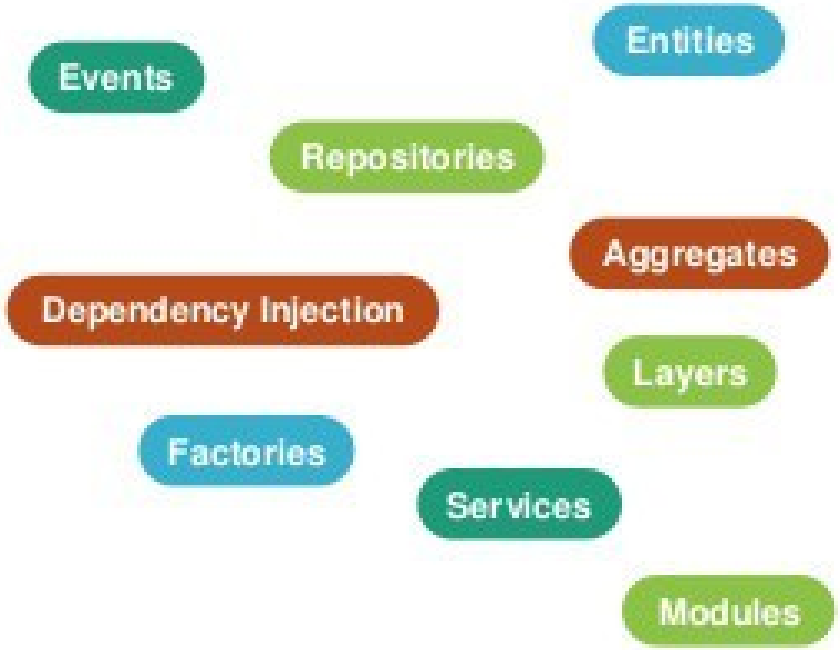
- https://www.eventstorming.com/
- Github – Awesome Event Storming
- Alberto Brandolini – Event Storming
- Mariusz Gil - Discovering unknown domain with Event Storming

Domain Driven Design

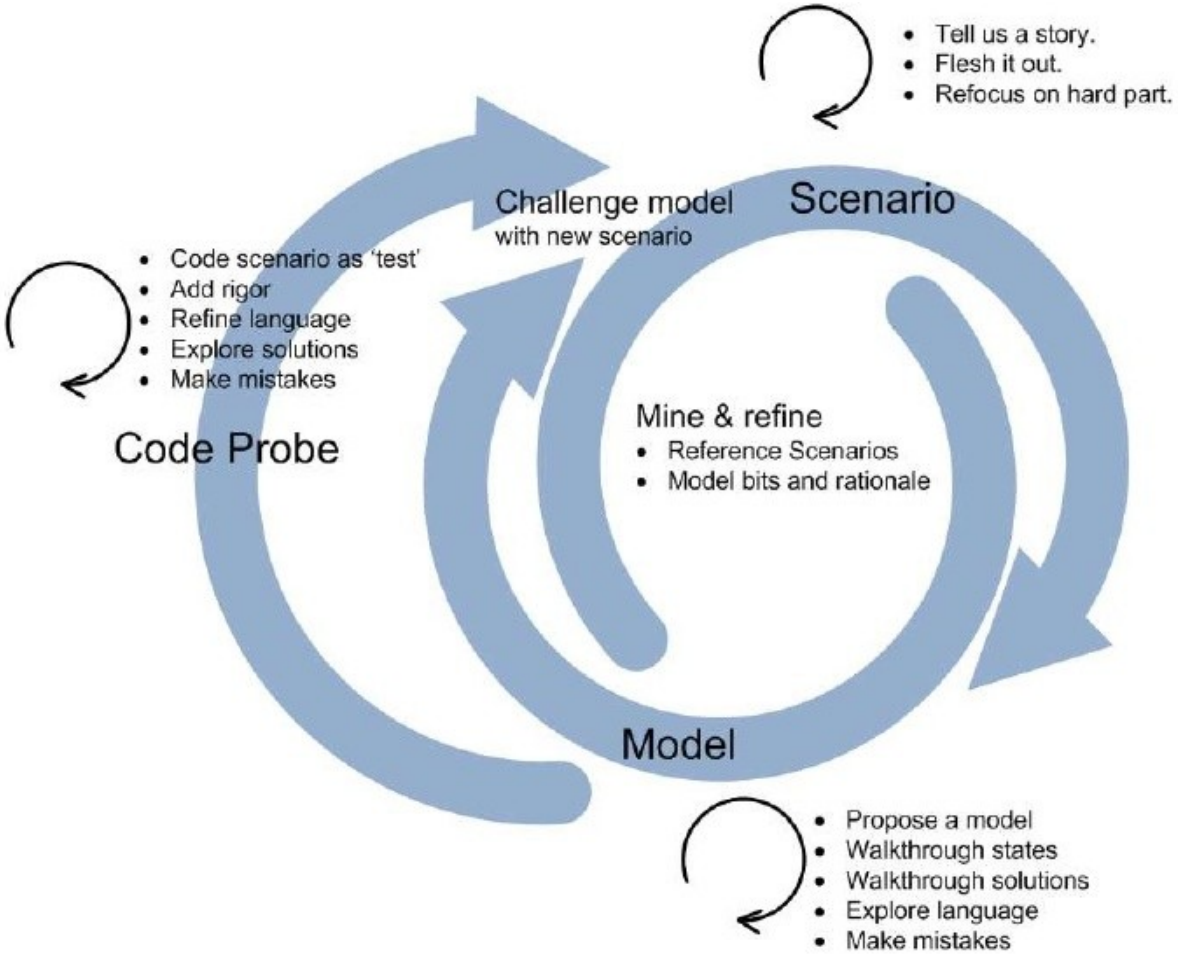
Strategic design



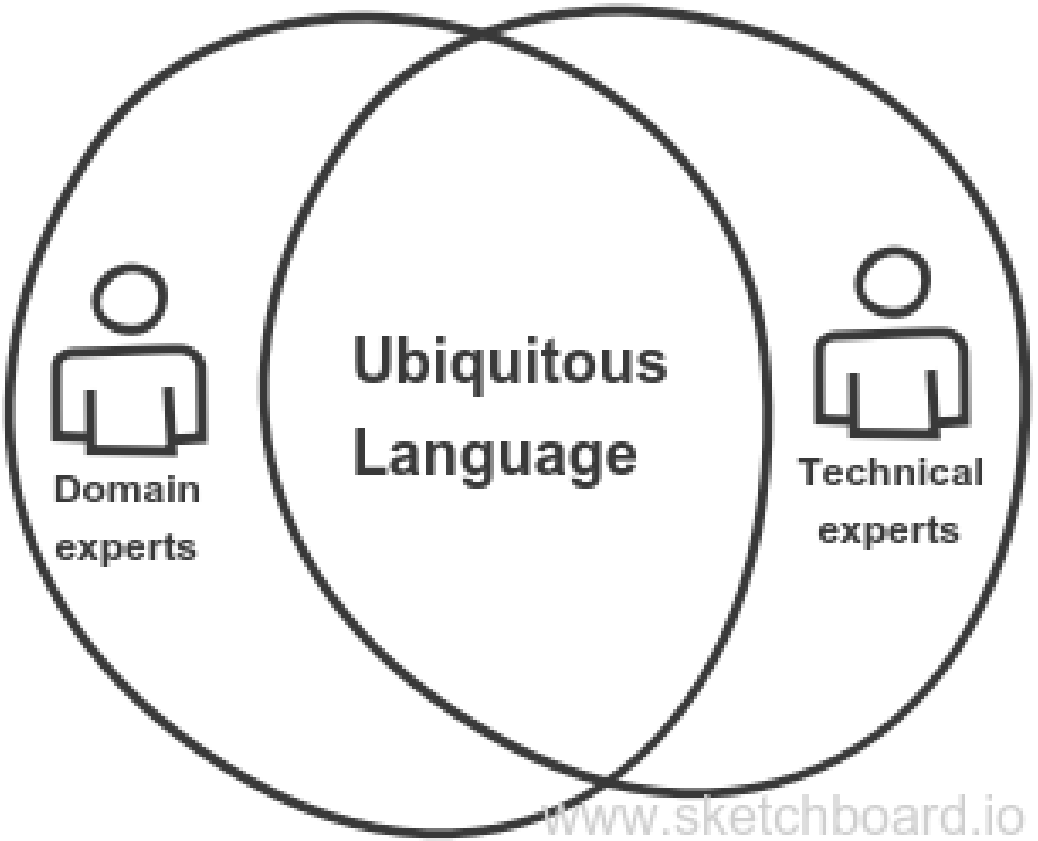
Tactical patterns



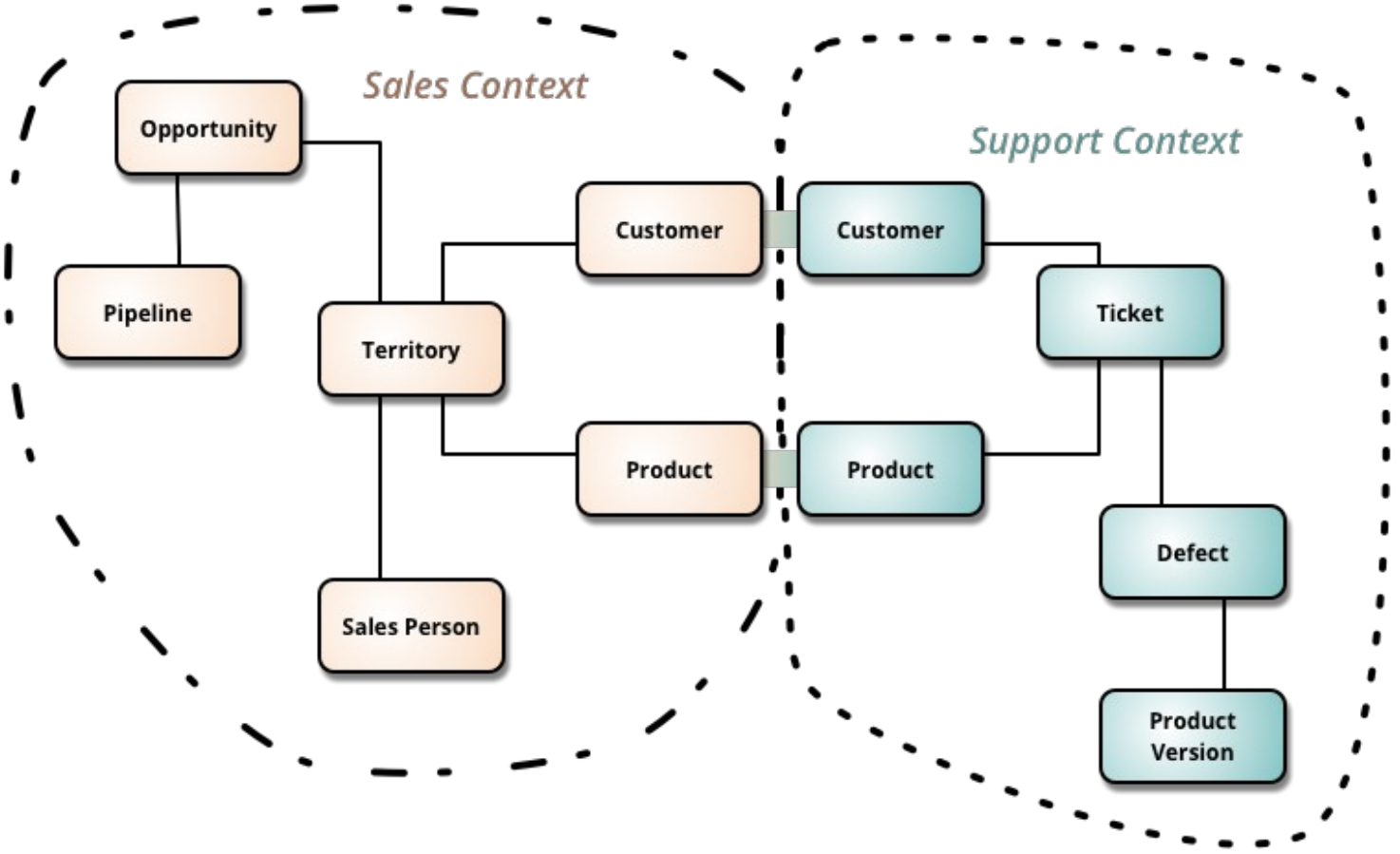
DDD – Design Whirlpool



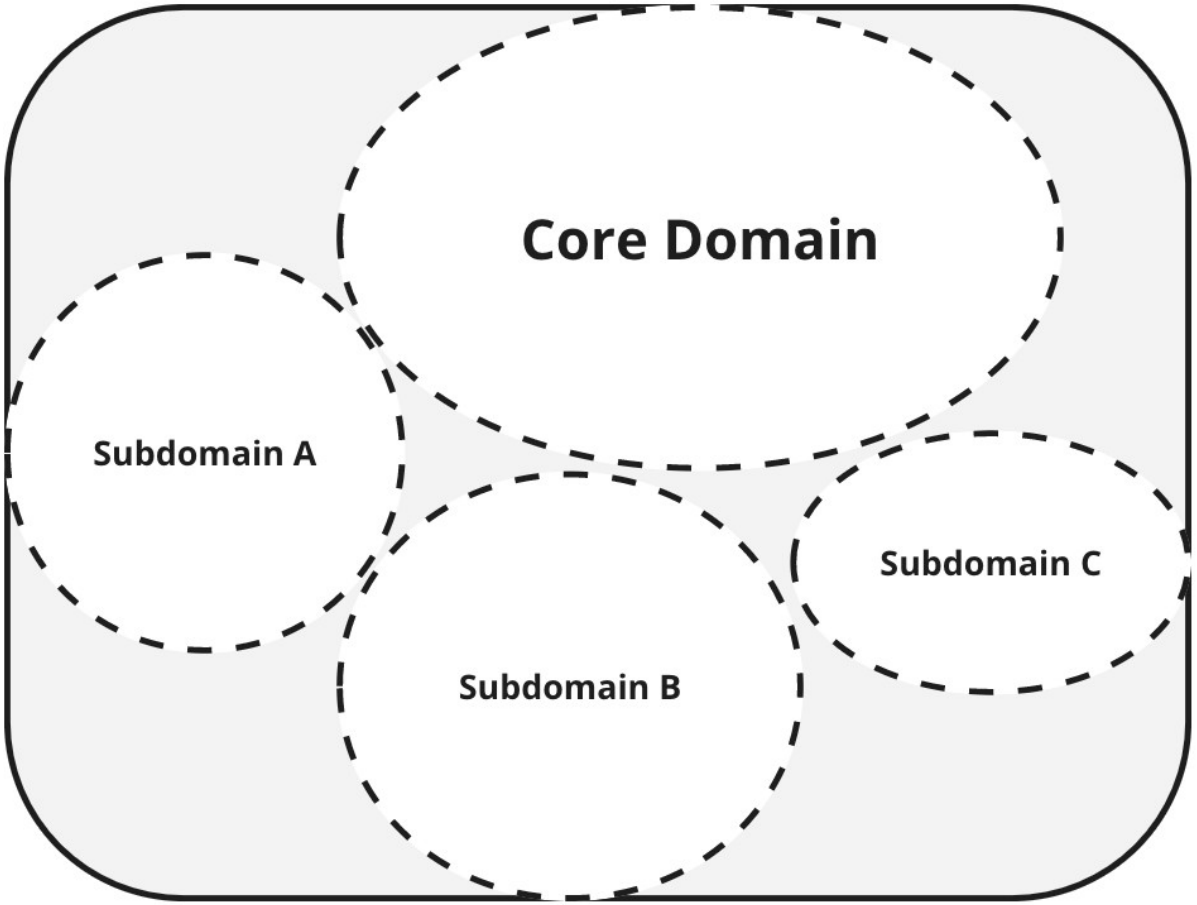
DDD - Wspólny język



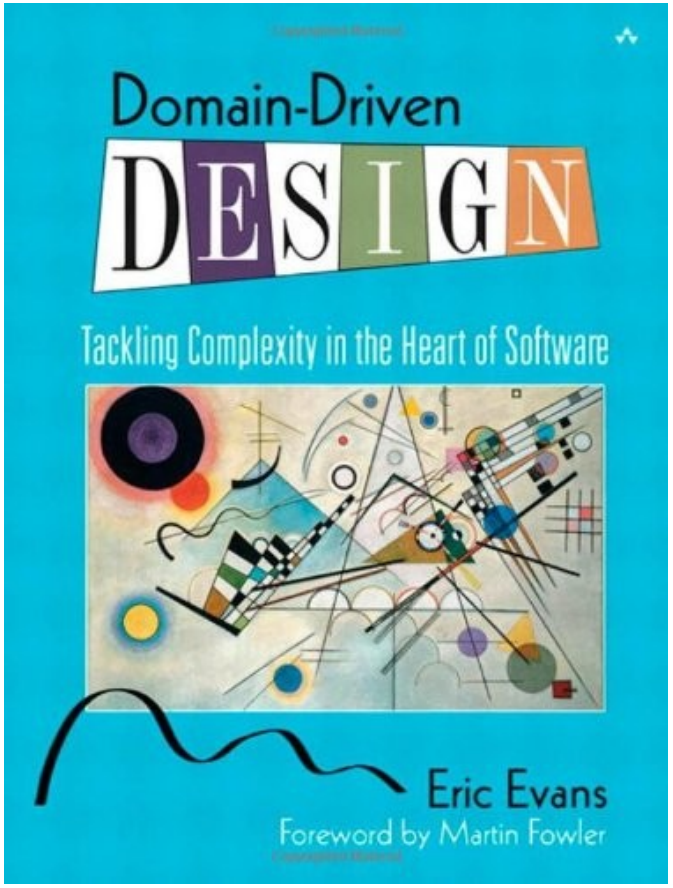
DDD – bounded context



DDD – domeny w systemie



DDD - materiały



DDD - materiały

- Bottega - https://bottega.com.pl/artykuly-i-prezentacje


The End