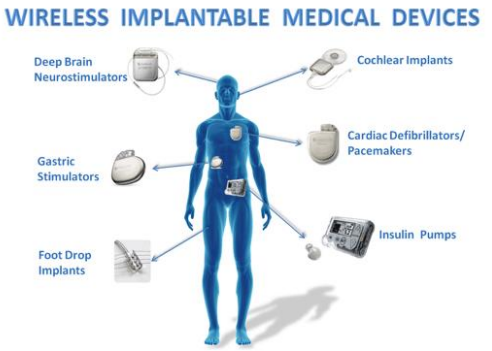
Remote monitoring en troubleshooting van IoT en internet-connected applicaties

Gerard Fianen
INDES – Integrated Development Solutions BV

IoT en Internet connected Devices

- Kleine sensors in grote aantallen
  - Goedkoop, laag stroomverbruik, beperkt geheugen
- Industriële robots en andere fout tolerante applicaties
  - Non stop



Over-The-Air updates zijn (maar) een deel van de oplossing

- Wat is de lastigste probleemomschrijving ?

  “Hij doet het niet …”

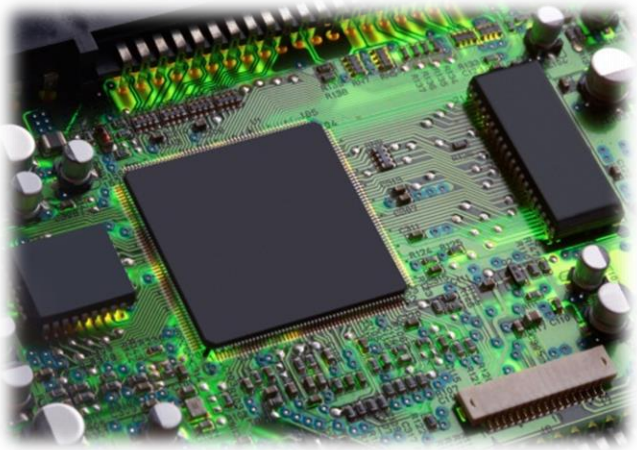
  “ eeeh… … Soms … ”

- Als je een probleem kunt reproduceren is het zo goed als opgelost

- Root cause analysis / Post Mortem analysis

Feedback loop

Deployment/update



Feedback loop

Device Metrics:

- Resource usage
- Counters
- Durations

Debug data:

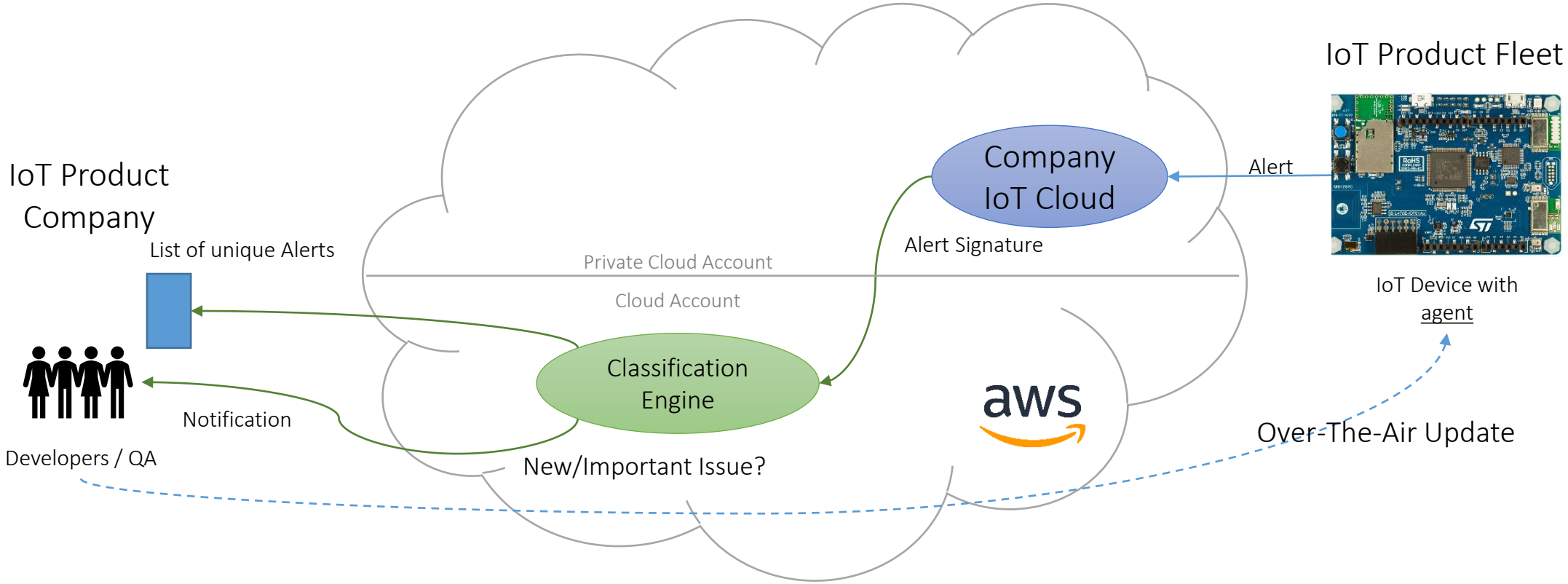
- Problem alerts
- Debug data
- RTOS event trace

- Watchdog reset
- Memory errors
- I/O errors

Aandachtspunten

- Hoe ga ik om meer te veel alerts ?
  - Meerdere devices in het veld die eenzelfde error geven
  - Te veel irrelevante data in log
- Data security / privacy !
- Beperkte resources ?

Secure Remote Device Monitoring



High-level dashboard

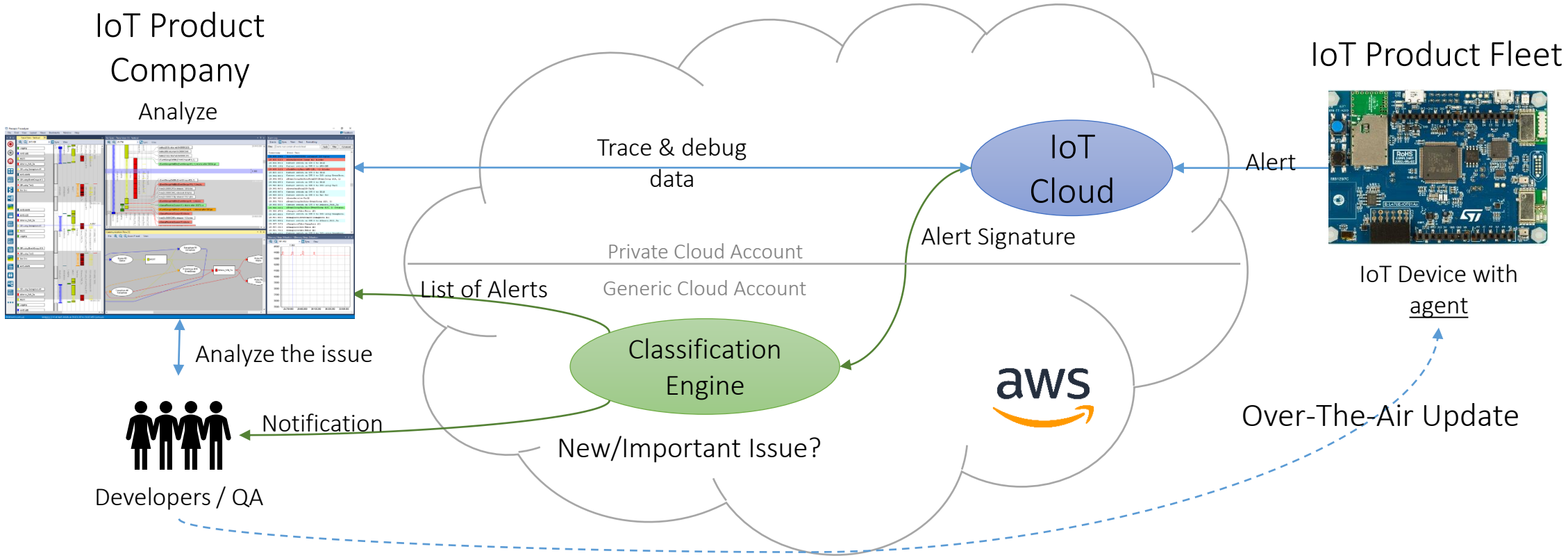
Issue Overview

Description	Count	Latest Occurrence	Details	Alerts	Latest Trace
CPU Overload	3	2022-08-02 13:19:12	View	Alerts	Open
Assert Failed	1	2022-08-02 13:18:50	View	Alerts	Open
Heartbeat failure	4	2022-08-02 13:18:32	View	Alerts	Open
Assert Failed	1	2022-07-26 14:20:44	View	Alerts	Open

Each alert can provide multiple types of diagnostics :

- System traces – Timeline of tasks, API calls, etc.
- Crash dumps – Registers, stack trace, memory
- Device logs – Existing application logs?
- Other device data? Anything can be included

Remote Device Monitoring



Register dump

crash.txt - Anteckningar

Arkiv Redigera Format Visa Hjälp

```
Reading symbols from devalert-f407-elevator-controller.elf...

Remote debugging using |  C:/DevAlertDispatcher/CrashDebug.exe --elf devalert-f407-elevator-contrc
prvIdleTask (pvParameters=0x8005791 <xTaskIncrementTick+524>)
    at ../freertos/tasks.c:3481
Backtrace
#0  prvIdleTask (pvParameters=0x8005791 <xTaskIncrementTick+524>)
    at ../freertos/tasks.c:3481
#1  0x00000000 in ?? ()

Fault status registers
0xe000ed28:     0x00000000      0x00000000      0x00000000
0xe000ed38:     0x00000000
Registers
r0              0x0                 0
r1              0xa5a5a5a5          -1515870811
r2              0x2001df9c          536993692
r3              0x0                 0
r4              0xa5a5a5a5          -1515870811
```

Rad 4, kol 1

Debug

DevAlertCrashDump [GDB Hardware Debugging]

latest.elf

Thread #1 <main> (Suspended : User Request)

_stack_chk_fail() at main.c:215 0x80016bc

testStackSmash() at main.c:224 0x80016fc

ButtonTask() at main.c:266 0x800176a

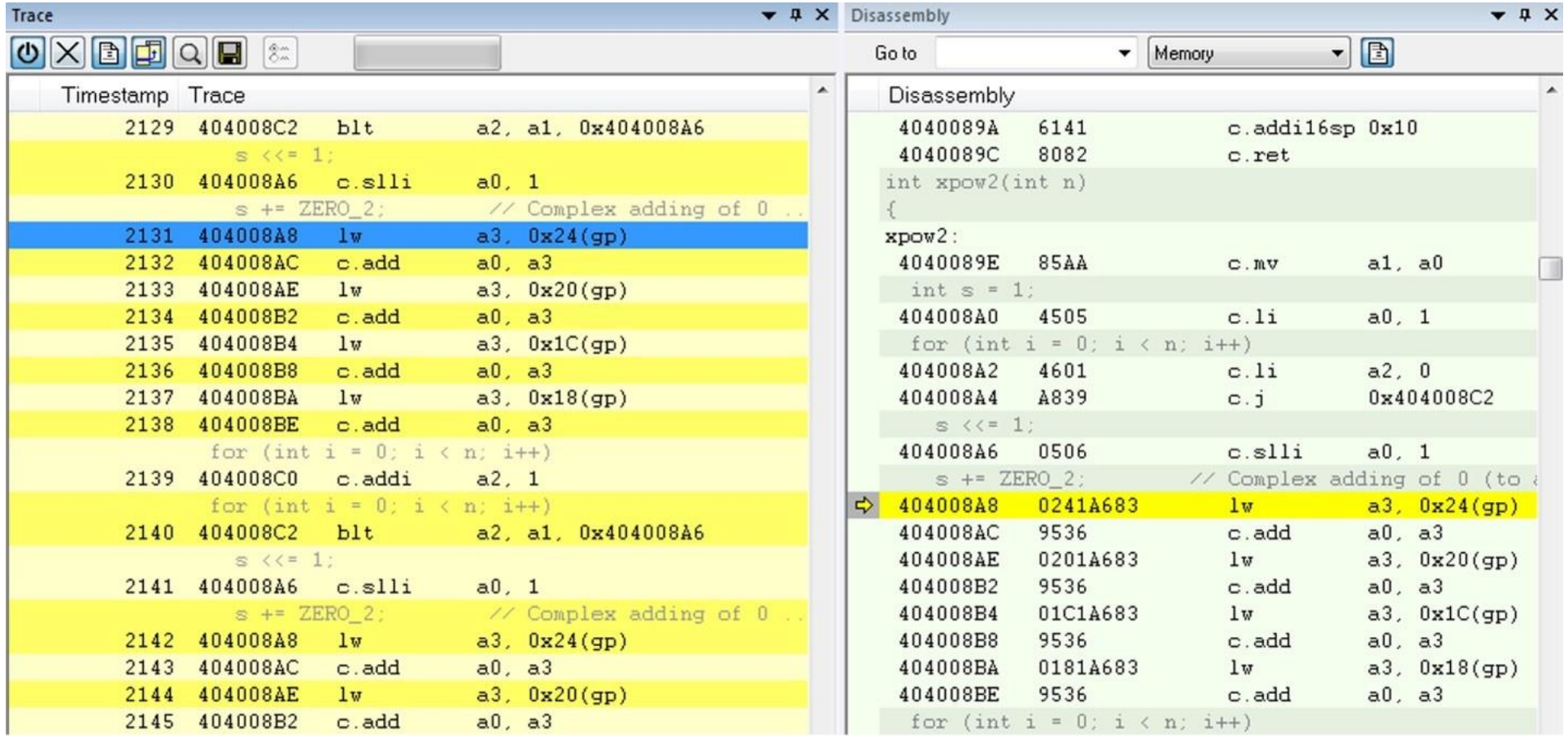
C:/DevAlertDispatcher/arm-none-eabi-gdb.exe (8.3.1.20191211)

main.c | tasks.c | startup_stm3... | aws_dev_mod... | iot_wifi.c | dfmAlert.c

```
209
210 void *__stack_chk_guard = (void *)0xdeadbeef;
211
212 void __stack_chk_fail(void)
213 {
214       // If you get a hard fault here, the stack has been corrupted by the calling function.
215       volatile int stackcheckfailed = *(unsigned int*)0x00100000;
216 }
217
218 void testStackSmash(void)
219 {
220     volatile char buf[10];
221
222     // Will trigger the above fault handler on return (using -fstack-protector-strong flag)
223     sprintf(buf, "This will corrupt the stack");
224 }
225
```

Debug dump

Trace data ..



Trace met beperkte resources ?

- Wireless Bandbreedte
- Geheugen
- Stroomverbruik

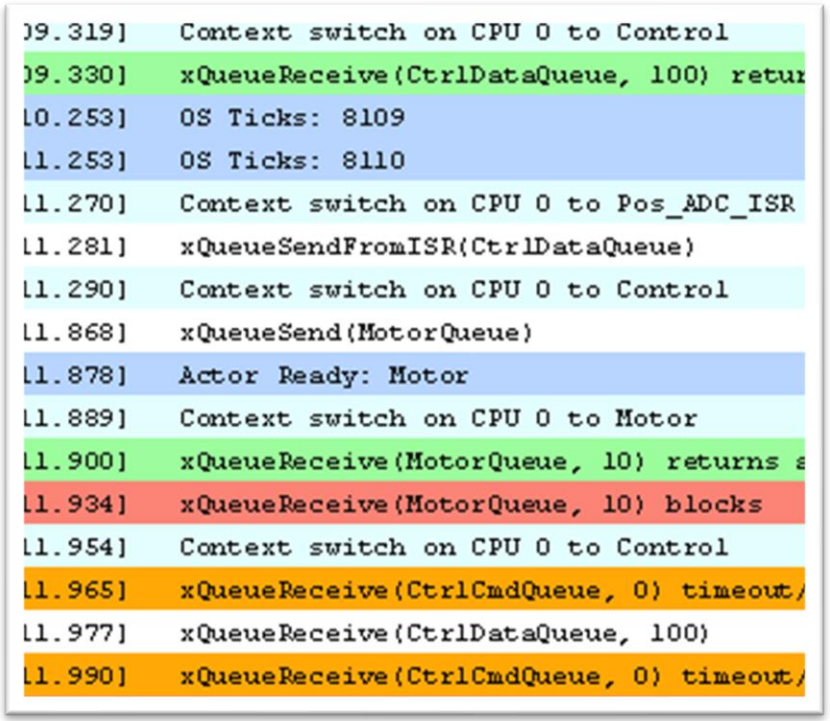
Instruction Trace

```
ime             info
 184140         000100fc -> write r31
 184141  100a8    IncrementCounterBy1  add          %r0,%
 184141         0000002e -> write r0
 184142  100ac    IncrementCounterBy1+0x04  extb
 184142         0000002e -> write r0
 184143  100b0    IncrementCounterBy1+0x08  j_s
 184144  100fc    main+0x30       stb          %r0,[%r1]
 184144             2e -> write mem [0x011000]
 184145  10100    main+0x34       ldb          %r0,[%r2]
 184145             5a <- read  mem [0x011001] -> write
 184146         0000005a -> write r0
 184147  10104    main+0x38       bl_s         IncrementC
 184147         00010106 -> write r31
 184148  100b4    IncrementCounterBy2  add          %r0,%
 184148         0000005c -> write r0
 184149  100b8    IncrementCounterBy2+0x04  extb
 184149         0000005c -> write r0
 184150  100bc    IncrementCounterBy2+0x08  j_s
 184151  10106    main+0x3a       stb          %r0,[%r2]
```

Printf / Application logging

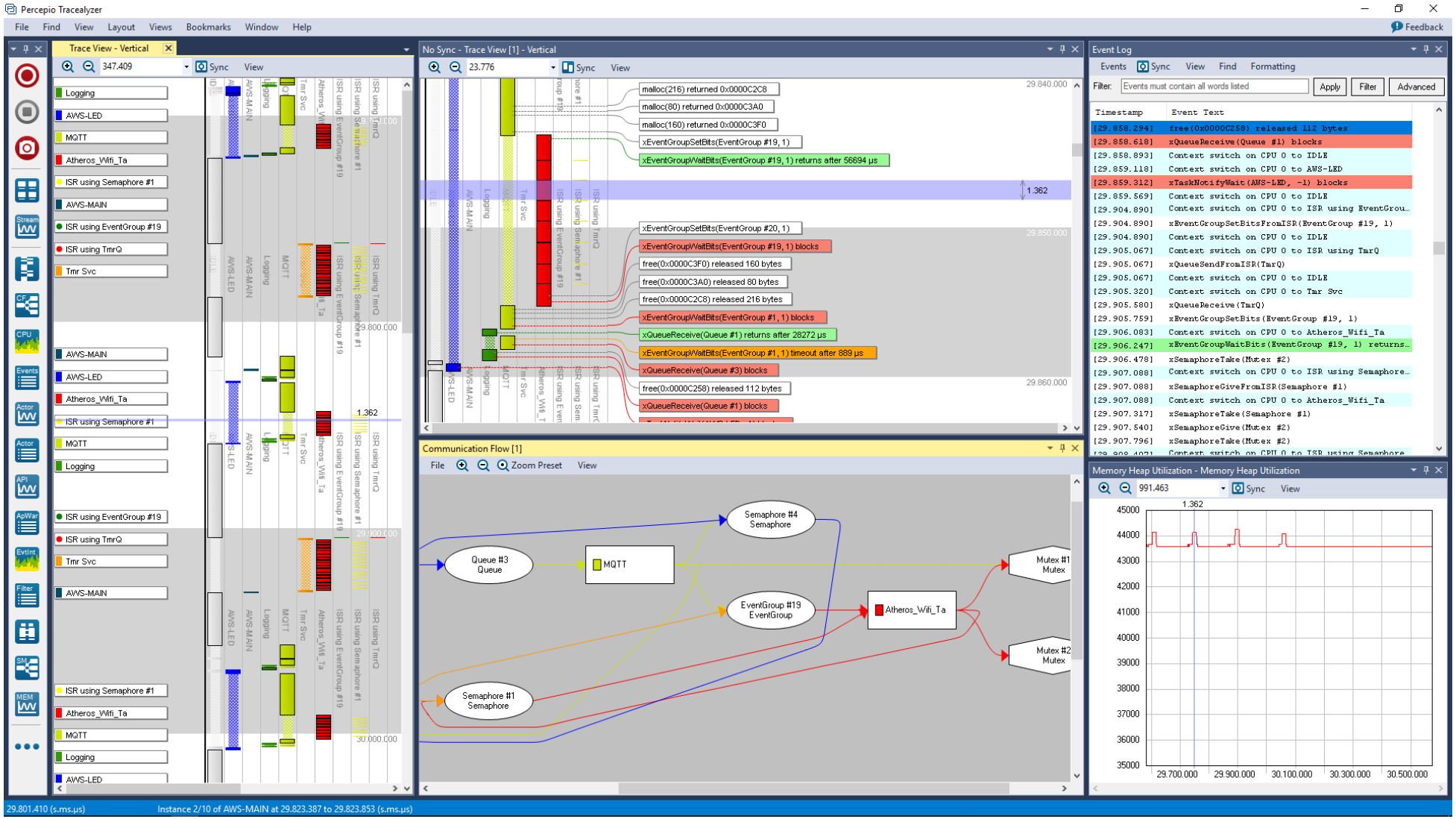
```
Svc] Starting key provisioning...
Svc] Write root certificate...
Svc] Write device private key...
 Svc] Write device certificate...
 Svc] Key provisioning done...
 Svc] Starting WiFi...
mr Svc] WiFi module initialized.
WS-MAIN] WiFi connected to AP AndroidAP.
AWS-MAIN] Attempt to Get IP.
AWS-MAIN] IP Address acquired 192.168.0.51
AWS-LED] [Shadow 0] MQTT: Creation of dedicated MQT
AWS-LED] Sending command to MQTT task.
MQTT] Received message 10000 from queue.
MQTT] Looked up a7sw0r7rvpirn.iot.us-east-1.amazona
[MQTT] MQTT Connect was accepted. Connection establ
[MQTT] Notifying task.
[AWS-LED] Command sent to MQTT task passed.
```

Trace on another level : Event Trace



	Instruction Trace	**(RTOS) Event Trace**	**Application Logging**
Producer	Processor core	Software (API or Kernel)	Software (application)
Abstraction Level	Low	Medium	High
Overhead	None	Some	More
System Requirements	High	Low	Low
Flexibility	Low	High	High

Typical example : With just 5 KB of trace data, 350 ms trace (during a busy period, with many details)



What can be analyzed ? - 1

Some examples:

- Multi-threading and timing
  - Context switches, internal kernel events
  - Execution time, response time, periodicity…
- API calls (OS, Middleware stacks, Drivers…)
  - Call sequences and timing
  - Parameters and return values
  - Blocking and timeouts
  - Object dependencies
- Application logging
  - Debug messages, variable values…
- Time between important events
- State changes over time

What can be analyzed ? - 2

RTOS Analytics

- Service Call Block Time
- Object Utilization
- Message Receive Time
- Priority Changes
- Event Intensity (three types)

Application Analytics

- Communication Flow
- State Machine Graph
- User Events Signal Plot
- I/O Intensity and I/O Plot

CPU Usage

- CPU Load Graph
- Actor Statistics Report
- Actor Instance Graphs
- Interval Coverage Graph

Memory Usage

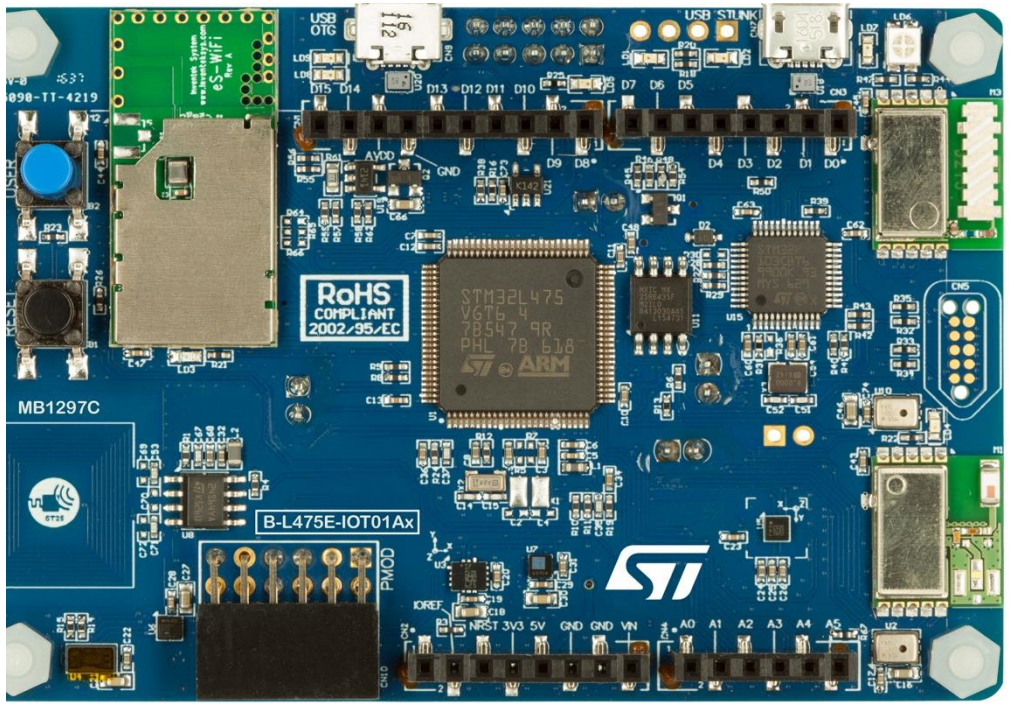
- Stack Usage
- Memory Heap Utilization

All views are **interconnected** in clever ways, so you can click on a data point in one view and see the corresponding location in another related view.

Demonstratie

- IoT STM Discovery IoT node
- Wifi naar onze AWS cloud

- Introductie van run-time fouten



Cross Compilers, Debuggers, IDE
RTOS, Middleware, Protocol stacks, GUI, Database
Debug & Trace probes
Real-Time Trace, RTOS-Event Trace
Static Analysis, Timing Analysis, Stack Analysis
Unit Test, Code Coverage
System-level Test

Tessy

Test-As-A-Service
Verification-As-a-Service

PoE conformance

www.indes.com info@indes.com gerard@indes.com