



Zephyr™ Project

Developer Summit

June 8-10, 2021 • @ZephyrIoT

Best Practices for Debugging Connected Applications running Zephyr

Chris Coleman
Luka Mustafa

Chris Coleman



- Co-Founder & CTO, Memfault
- Previously a Firmware Engineer @ Sun Microsystems, Pebble, & Fitbit
- Zephyr TSC member

Luka Mustafa



- Founder & CEO, IRNAS
- Multidisciplinary engineer with EE background
- Designing IoT solutions for industrial applications

Connected Applications

- 22 billion connected devices as of 2018, 50 billion projected by 2030!*
- Connectivity stacks are **complex**
- Many classes of issues
 - Faults / Hangs
 - Performance
 - Security
 - Connectivity interoperability



*Source: <https://www.statista.com/statistics/802690/worldwide-connected-devices-by-access-technology/>

Example project at IRNAS to set the scene

Industrial Solutions



IoT in Power
Transmission Lines



Real-time infrastructure
monitoring



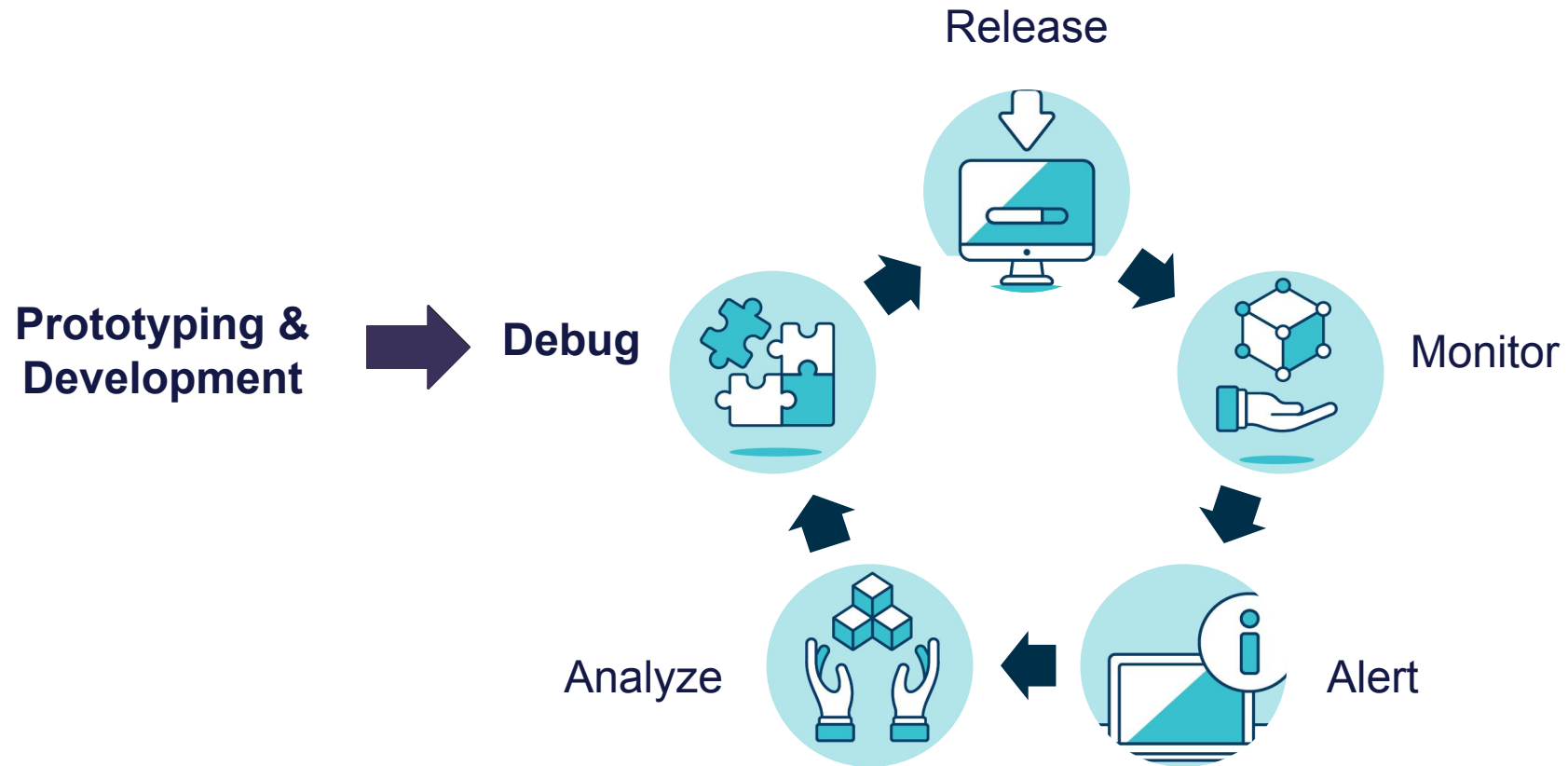
Autonomous drainage
maintenance system



Zephyr™ Project
Developer Summit

Device lifecycle

Debug Setup



1 Local Debug Setup

2 Zephyr Debug (K)Config Tips

3 Remote Monitoring Best Practices with Examples



Zephyr™ Project
Developer Summit

Local Debug Setup

1. **Reliable** JTAG setup
 2. Ability to read/write memory
 3. Ability to script common operations
- I use SEGGER J-Link + JLinkGDBServer + GDB

Starting GDB

```
$ west --verbose debug --runner jlink --gdb  
arm-none-eabi-gdb-py
```

```
-- runners.jlink: J-Link GDB server running on port 2331  
runners.jlink: JLinkGDBServer -select usb -port 2331 -if  
swd -speed 4000 -device nRF9160_xxAA -silent -singlerun
```

```
(gdb) continue
```

With west:

```
$ west flash
```

Directly via JLinkGDBServer / GDB!

```
(gdb) mon reset
Resetting target
(gdb) load
`build/zephyr/zephyr.elf' has changed; re-reading symbols.
Start address 0x00015df0, load size 130437
Transfer rate: 25475 KB/sec, 4207 bytes/write.
(gdb)
```

pyserial

```
$ pip install pyserial
$ pyserial-miniterm - 115200 --raw
--- Available ports:
--- 3: /dev/cu.usbmodem0009600050801 'J-Link - CDC DATA interface'
--- 4: /dev/cu.usbmodem0009600050803 'J-Link - CDC DATA interface'
--- 5: /dev/cu.usbmodem0009600050805 'J-Link - CDC DATA interface'

$ pyserial-miniterm /dev/cu.usbmodem0009600050801 115200 --raw
uart:~$ *** Booting Zephyr OS build v2.4.99-ncs1-3525-g4d068de3f50f ***
```



Zephyr™ Project
Developer Summit

Zephyr Debug (K)Config Tips

```
(gdb) info threads
  Id   Target Id                                     Frame
*  2   Thread 536956136 (idle 00 UNKNOWN PRI0 15)   arch_cpu_idle () at
  3   Thread 536956408 (main PENDING PRI0 0)        arch_swap (key=0) at
  4   Thread 536955312 (shell_uart PENDING PRI0 14)  arch_swap (key=0) at
  5   Thread 536956696 (sysworkq PENDING PRI0 255)   arch_swap (key=0) at
  6   Thread 536955648 (at_cmd_socket_thread PENDING PRI0 10) arch_swap (key=0) at
```

- CONFIG_DEBUG_THREAD_INFO=y
 - (Originally CONFIG_OPENOCD_SUPPORT=y)

Debug printing with printk

CONFIG_PRINTK=y

```
void main(void) {  
    printk("System Started!\n");  
    // ...  
}
```

```
uart:~$ System Started!  
// ...
```

- Bypasses logging subsystem by default and prints directly to console
- Useful for minimal overhead and guaranteed printing

Console Printing with Logging Subsystem

- CONFIG_LOG=y
- CONFIG_SHELL=y
- Deferred Mode (default)
 - logs are buffered and flushed process on low priority task
 - CONFIG_LOG_MODE_DEFERRED=y
- Immediate Mode (recommend for debug)
 - Logs are flushed from running task.
 - CONFIG_LOG_IMMEDIATE=y
- ⚠ Leaving logging impacts power consumption
 - Should be disabled for low power applications in production

Zephyr Logging Modules

```
# Kconfig
module = MY_MODULE
module-str = My module
source "${ZEPHYR_BASE}/subsys/logging/Kconfig.template.log_config"

// my_module.c
LOG_MODULE_REGISTER(my_module, CONFIG_MY_MODULE_LOG_LEVEL);

# prj.conf - Choose one of the following:
CONFIG_MY_MODULE_LOG_LEVEL_OFF=y # 0
CONFIG_MY_MODULE_LOG_LEVEL_ERR=y # 1
CONFIG_MY_MODULE_LOG_LEVEL_WRN=y # 2
CONFIG_MY_MODULE_LOG_LEVEL_INF=y # 3 (default)
CONFIG_MY_MODULE_LOG_LEVEL_DBG=y # 4
```

Zephyr Logging Level Options

1. Autogenerated “autoconf.h file contains all active settings:
 - See “build/zephyr/include/generated/autoconf.h”
2. Grep through file for LOG_LEVEL, i.e

```
$ rg "LOG_LEVEL " build/zephyr/include/generated/autoconf.h
```

```
60:#define CONFIG_MPSL_LOG_LEVEL 3
68:#define CONFIG_MGMT_FMFU_LOG_LEVEL 3
84:#define CONFIG_MEMFAULT_INTEGRATION_LOG_LEVEL 3
86:#define CONFIG_AGPS_LOG_LEVEL 3
97:#define CONFIG_NRF_MODEM_LIB_LOG_LEVEL 3
// ...
```



Zephyr™ Project
Developer Summit

Remote Monitoring Best Practices

GPS tracker on an animal

- Mobile connectivity issues to be observed and resolved
- Hardware performance monitored
- Track and monitor all issues over time



Static sensor with long lifetime

- All faults must be handled to conserve power
- Operation to be optimized based on the use-case
- Validate upgrades in the field



Remote Monitoring Zephyr with Memfault

- Works on any ARM-based MCU with Zephyr OS
- C-SDK with connectivity agnostic data transport
- Cloud based issue analysis, alerting and deduplication on both device level and fleetwide trends



**Remotely debug issues with
coredumps, events and logs**



**Continuously monitor devices
with Metrics**



**Deploy OTA updates safely with
staged rollouts and targeted
device groups**

Memfault Zephyr Integration

```
# west.yml
[ ... ]
  - name: memfault-firmware-sdk
    url: https://github.com/memfault/memfault-firmware-sdk
    path: modules/memfault-firmware-sdk
    revision: master

# prj.conf
CONFIG_MEMFAULT=y
CONFIG_MEMFAULT_HTTP_ENABLE=y
```



Zephyr™ Project
Developer Summit

Core Properties To Track

1 Reboot Reasons

2 Watchdogs

3 Faults & Asserts

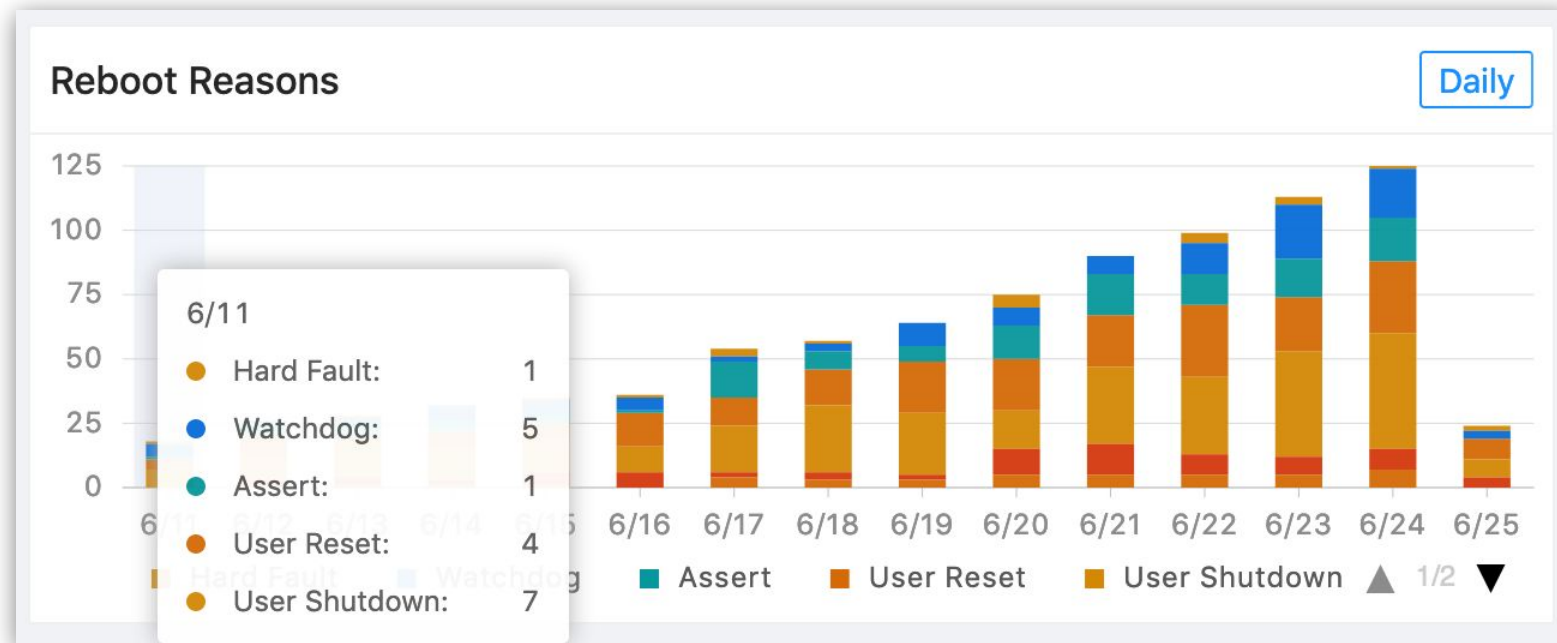
4 Connectivity Metrics



Zephyr™ Project
Developer Summit

Reboot Reasons

Tracking Device Resets



Leading indicator of fleet health

Hardware Resets

- Examples
 - PLL & Clock Failures
 - Brown Out
 - Hardware Watchdogs
- Can identify hardware defects

Software Resets

- Examples
 - Firmware Update / OTA
 - Assert
 - User initiated

Tracking Software Resets

1. Create “noinit” RAM region

```
/* memfault-no-init.ld */  
KEEP(*(.mflt_reboot_info));  
  
# CMakeLists.txt  
zephyr_linker_sources(NOINIT memfault-no-init.ld)
```

2. Place C variable in region

```
__attribute__((section(".noinit.mflt_reboot_info")))  
static uint8_t  
s_reboot_tracking[MEMFAULT_REBOOT_TRACKING_REGION_SIZE];
```

3. Record reason for reboot

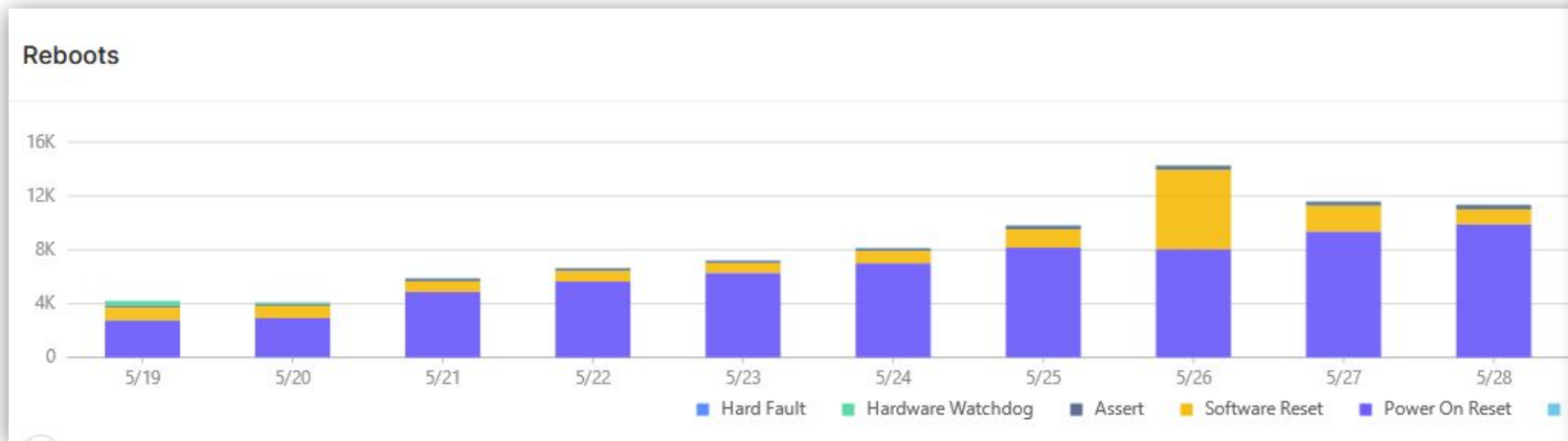
```
void fw_update_finish(void) {  
    // ...  
  
    memfault_reboot_tracking_mark_reset_imminent(kMfltRebootReason_F  
    irmwareUpdate, ...);  
    sys_reboot(0);  
}
```

Capturing Device Resets on Zephyr

Register init handler that to read bootup information:

```
static int record_reboot_reason() {  
    // 1. Read hardware reset reason register. (Check MCU data sheet for register name)  
    // 2. Capture software reset reason from noinit RAM  
    // 3. Send data to server for aggregation  
}  
  
SYS_INIT(record_reboot_reason, APPLICATION, CONFIG_KERNEL_INIT_PRIORITY_DEFAULT);
```

Example: Power supply issue



- 12K device reboots a day - *way too much*
- 99% of reboots contributed by 10 devices
- Bad mechanical part contributing to device constant reboots

Recent Resets By Device (Last 72 Hours)

| device_serial | reboot_reason | reset_count |
|---------------|----------------|-------------|
| power | | |
| 92 | Power on Reset | 1,899 |
| 08 | Power on Reset | 1,409 |
| 91 | Power on Reset | 1,269 |
| 01 | Power on Reset | 1,254 |
| 62 | Power on Reset | 1,030 |
| 03 | Power on Reset | 973 |
| 33 | Power on Reset | 890 |
| 17 | Power on Reset | 866 |
| 93 | Power on Reset | 850 |
| 22 | Power on Reset | 810 |
| 74 | Power on Reset | 764 |
| 73 | Power on Reset | 759 |
| 13 | Power on Reset | 716 |

< 1 2 3 4 5 ... 115 >

an hour ago



Zephyr™ Project
Developer Summit

Watchdogs

Defending against Hangs

- Last line of defense against a hung system!
- Can happen for many reasons:
 - Connectivity Stack Blocks on send()
 - Infinite Retry Loop talking to system
 - Deadlock between tasks
 - Corruption
- Two pieces:
 - Hardware Watchdog
 - Built in and/or external peripheral to reset device
 - Software Watchdog
 - Interrupt that fires ahead of hard reset so watchdog can be root caused

Zephyr - Hardware Watchdog API

```
// ...
void start_watchdog(void) {
    // consult device tree for available hardware watchdog
    s_wdt = device_get_binding(DT_LABEL(DT_INST(0, nordic_nrf_watchdog)));

    struct wdt_timeout_cfg wdt_config = {
        /* Reset SoC when watchdog timer expires. */
        .flags = WDT_FLAG_RESET_SOC,

        /* Expire watchdog after max window */
        .window.min = 0U,
        .window.max = WDT_MAX_WINDOW,
    };

    s_wdt_channel_id = wdt_install_timeout(s_wdt, &wdt_config);

    const uint8_t options = WDT_OPT_PAUSE_HALTED_BY_DBG;
    wdt_setup(s_wdt, options);
    // TODO: Start a software watchdog
}

void feed_watchdog(void) {
    wdt_feed(s_wdt, s_wdt_channel_id);
    // TODO: Feed software watchdog
}
```

See Zephyr API for more details:
[zephyr/include/drivers/watchdog.h](https://zephyrproject.dev/docs/api/drivers/watchdog/)

Zephyr Software Watchdog

```
static void prv_software_watchdog_timeout(struct k_timer *dummy) {
    MEMFAULT_ASSERT(0);
}

K_TIMER_DEFINE(s_watchdog_timer, prv_software_watchdog_timeout, NULL);
static uint32_t s_software_watchdog_timeout_ms = MEMFAULT_WATCHDOG_SW_TIMEOUT_SECS * 1000;

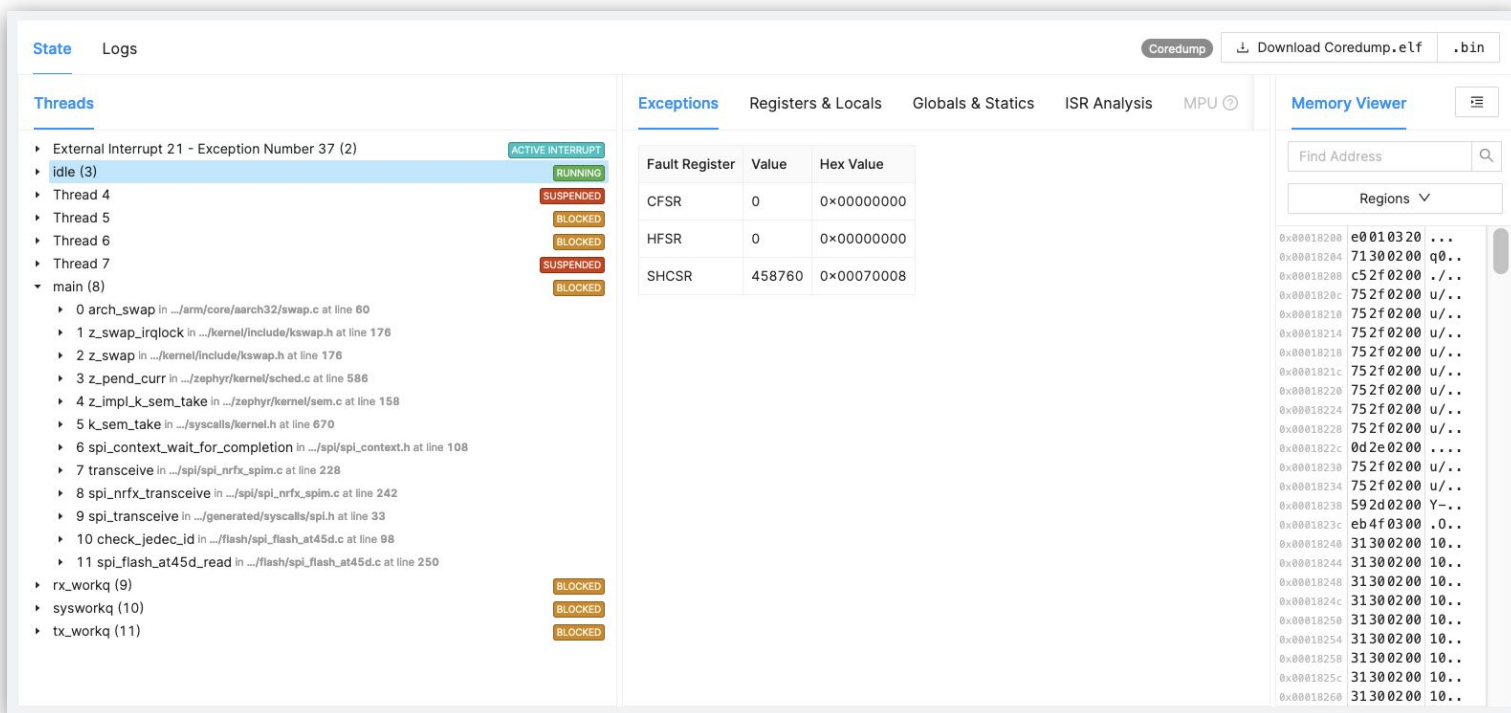
static void prv_start_or_reset(uint32_t timeout_ms) {
    k_timer_start(&s_watchdog_timer, K_MSEC(timeout_ms), K_MSEC(timeout_ms));
}

int memfault_software_watchdog_enable(void) {
    prv_start_or_reset(s_software_watchdog_timeout_ms);
    return 0;
}

int memfault_software_watchdog_feed(void) {
    prv_start_or_reset(s_software_watchdog_timeout_ms);
    return 0;
}
```

New built in “[Task Watchdog](#)”
API in 2.6 Release.

Example: SPI driver stuck



The screenshot displays the Zephyr IDE's debugger interface. The 'Threads' panel on the left shows a list of threads, with 'main (8)' expanded to show its call stack. The 'Exceptions' panel in the center shows a table of fault registers. The 'Memory Viewer' panel on the right shows a memory dump.

Threads

- External Interrupt 21 - Exception Number 37 (2) **ACTIVE INTERRUPT**
- idle (3) **RUNNING**
- Thread 4 **SUSPENDED**
- Thread 5 **BLOCKED**
- Thread 6 **BLOCKED**
- Thread 7 **SUSPENDED**
- main (8) **BLOCKED**
 - 0 arch_swap in .../arm/core/aarch32/swap.c at line 60
 - 1 z_swap_irqlock in .../kernel/include/kswap.h at line 176
 - 2 z_swap in .../kernel/include/kswap.h at line 176
 - 3 z_pend_curr in .../zephyr/kernel/sched.c at line 586
 - 4 z_impl_k_sem_take in .../zephyr/kernel/sem.c at line 158
 - 5 k_sem_take in .../syscalls/kernel.h at line 670
 - 6 spi_context_wait_for_completion in .../spi/spi_context.h at line 108
 - 7 transceive in .../spi/spi_nrfx_spi.c at line 228
 - 8 spi_nrfx_transceive in .../spi/spi_nrfx_spi.c at line 242
 - 9 spi_transceive in .../generated/syscalls/spi.h at line 33
 - 10 check_jedec_id in .../flash/spi_flash_at45d.c at line 98
 - 11 spi_flash_at45d_read in .../flash/spi_flash_at45d.c at line 250
- rx_workq (9) **BLOCKED**
- sysworkq (10) **BLOCKED**
- tx_workq (11) **BLOCKED**

Exceptions

| Fault Register | Value | Hex Value |
|----------------|--------|------------|
| CFSR | 0 | 0x00000000 |
| HFSR | 0 | 0x00000000 |
| SHCSR | 458760 | 0x00070008 |

Memory Viewer

Find Address:

Regions:

Memory dump (hex):

```
0x00018200 e0010320 ...
0x00018204 71300200 q0..
0x00018208 c52f0200 ./..
0x0001820c 752f0200 u/..
0x00018210 752f0200 u/..
0x00018214 752f0200 u/..
0x00018218 752f0200 u/..
0x0001821c 752f0200 u/..
0x00018220 752f0200 u/..
0x00018224 752f0200 u/..
0x00018228 752f0200 u/..
0x0001822c 0d2e0200 ....
0x00018230 752f0200 u/..
0x00018234 752f0200 u/..
0x00018238 592d0200 Y-..
0x0001823c eb4f0300 .0..
0x00018240 31300200 10..
0x00018244 31300200 10..
0x00018248 31300200 10..
0x0001824c 31300200 10..
0x00018250 31300200 10..
0x00018254 31300200 10..
0x00018258 31300200 10..
0x0001825c 31300200 10..
0x00018260 31300200 10..
```

- SPI flash degrading over time, incorrect timing of communication
- Traced this on 1% of devices after 16 months of field deployment
- Driver fix and roll-out with next release



Zephyr™ Project
Developer Summit

Faults & Asserts

Fault Handler - Register Dump

```
[00:26:12.826,782] <err> os: ***** BUS FAULT *****  
[00:26:12.832,153] <err> os:   Instruction bus error  
[00:26:12.837,738] <err> os: r0/a1:  0x00000001  r1/a2:  0x200150c1  r2/a3:  0x00000000  
[00:26:12.846,343] <err> os: r3/a4:  0x0badcafe  r12/ip:  0x00000001  r14/lr:  0x0001a6cb  
[00:26:12.854,919] <err> os: xpsr:  0x60000000  
[00:26:12.860,107] <err> os: s[ 0]:  0x00000001  s[ 1]:  0x00000001  s[ 2]:  0x00000001  s[ 3]:  0x00000001  
[00:26:12.870,422] <err> os: s[ 4]:  0x00000001  s[ 5]:  0x00000001  s[ 6]:  0x00000001  s[ 7]:  0x00000001  
[00:26:12.880,737] <err> os: s[ 8]:  0x00000001  s[ 9]:  0x00000001  s[10]:  0x00000001  s[11]:  0x00000001  
[00:26:12.891,052] <err> os: s[12]:  0x00000001  s[13]:  0x00000001  s[14]:  0x00000001  s[15]:  0x00000001  
[00:26:12.901,367] <err> os: fpscr:  0x00000000  
[00:26:12.906,524] <err> os: r4/v1:  0x00000001  r5/v2:  0x000135af  r6/v3:  0x2001abf8  
[00:26:12.915,130] <err> os: r7/v4:  0x2001ac00  r8/v5:  0xffffffff  r9/v6:  0x00000001  
[00:26:12.923,736] <err> os: r10/v7: 0x00000001  r11/v8: 0x00029f38    psp:  0x2001ab38  
[00:26:12.932,342] <err> os: EXC_RETURN: 0xffffffff  
[00:26:12.937,835] <err> os: Faulting instruction address (r15/pc): 0x0badcafe
```

Zephyr Fault Handler - Cortex M

```
void network_send(void) {  
    const size_t packet_size = 1500;  
    void *buffer = z_malloc(packet_size);  
    // missing NULL check!  
    memcpy(buffer, 0x0, packet_size);  
    // ...  
}
```



```
// zephyr/arch/arm/core/aarch32/cortex_m/fault.c  
void z_arm_fault(uint32_t msp, uint32_t psp,  
    uint32_t exc_return,  
    _callee_saved_t *callee_regs)  
{  
    // ...  
}
```

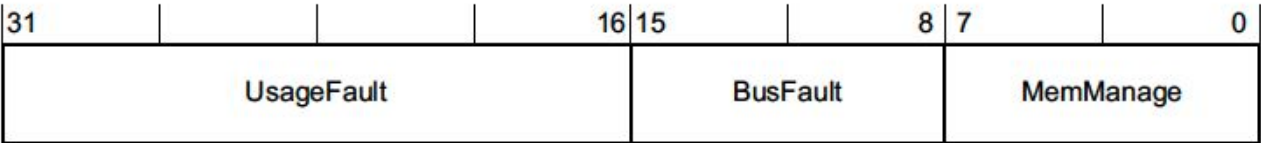


```
bool memfault_coredump_save(const  
    sMemfaultCoredumpSaveInfo  
*save_info) {  
    // Save register state  
    // Save _kernel and task contexts  
    // Save selected .bss & .data regions  
}
```



```
void sys_arch_reboot(int type) {  
    // ...  
}
```

Configurable Fault Status Register (CFSR)



Memfault Analysis

Configurable Fault (i.e UsageFault, BusFault, MemManage) escalated to HardFault

BusFault detected at 0x50008158

Precise BusFault detected! Triggered by Instruction: 'ldr r1, [r3, #0]' pc=0x00026fb8

| Fault Register | Value | Hex Value |
|----------------|------------|------------|
| CFSR | 33280 | 0x00008200 |
| HFSR | 1073741824 | 0x40000000 |
| SHCSR | 458884 | 0x00070084 |

Zephyr Fault Handler - Stacks

- ▼ SVCcall (2) ACTIVE INTERRUPT
 - ▶ 0 __wrap_z_fatal_error in .../memfault_fault_handler.c at line 52
 - ▶ 1 z_do_kernel_oops in .../arm/core/aarch32/fatal.c at line 113
 - ▶ 2 _oops in .../aarch32/swap_helper.S at line 482
 - ▶ 3 0xffffffff

- ▼ main (3) RUNNING
 - ▶ 0 __chk_fail in .../libc/newlib/libc-hooks.c at line 308
 - ▶ 1 __memcpy_chk
 - ▶ 2 __memcpy_ichk in .../include/ssp/string.h at line 83
 - ▶ 3 network_send in .../example_app/src/main.c at line 135

▶ at_cmd_socket_thread (4)

READY

▶ idle 00 (5)

READY

▶ shell_uart (6)

BLOCKED

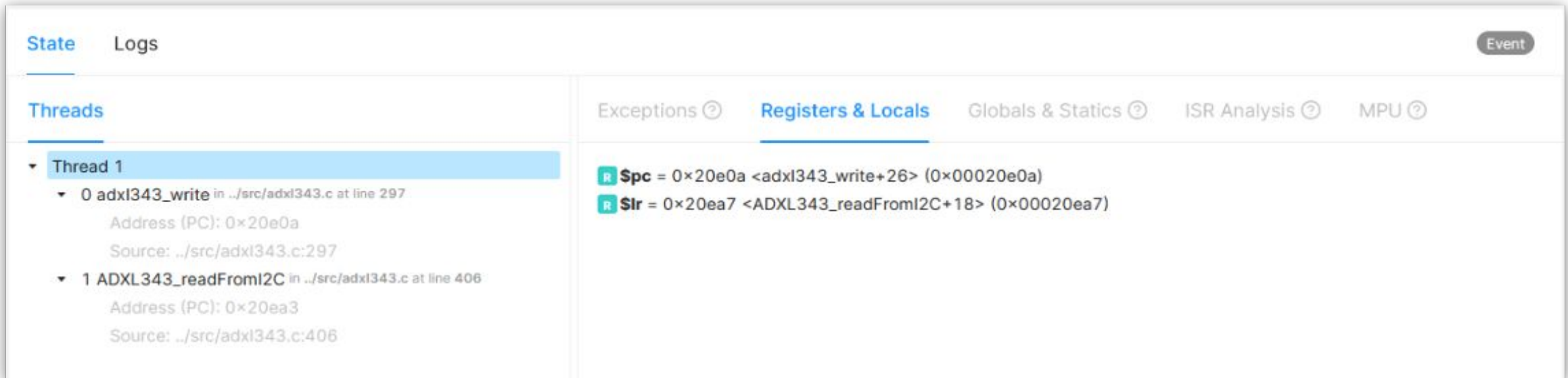
▶ sysworkq (7)

BLOCKED

Zephyr Fault Handler - Globals & Statics

| | | | |
|--|--|---------------------------------------|--|
| <input type="text" value="_kernel"/> | | <input type="text" value="Order by"/> | <input type="text" value="Memory Location"/> |
| ▼ _kernel = z_kernel {...} | | | @ 0x200162fc |
| ▼ cpus = _cpu[1] {...} | | | @ 0x200162fc |
| ▼ [0] = _cpu {...} | | | @ 0x200162fc |
| nested = uint32_t 0 | | | @ 0x200162fc |
| ▶ irq_stack = char* {...} | | | @ 0x20016300 |
| ▶ current = k_thread* {...} | | | @ 0x20016304 |
| ▶ idle_thread = k_thread* {...} | | | @ 0x20016308 |
| slice_ticks = int 0 | | | @ 0x2001630c |
| id = uint8_t 0 | | | @ 0x20016310 |
| ▶ timeout_q = sys_dlist_t {...} | | | @ 0x20016314 |
| idle = int32_t 0 | | | @ 0x2001631c |
| ▶ ready_q = _ready_q {...} | | | @ 0x20016320 |
| ▶ current_fp = k_thread* {...} | | | @ 0x2001632c |
| ▼ threads = k_thread* {...} | | | @ 0x20016330 |
| ▶ * = k_thread {...} | | | @ 0x20014aa0 |
| z_sys_post_kernel = _Bool 1 | | | @ 0x2001a567 |

Example: Accelerometer fault



The screenshot shows a debugger window with two main panes. The left pane, titled 'State', has a 'Threads' tab selected, showing a list of threads for 'Thread 1'. The right pane, titled 'Registers & Locals', shows the current register values.

Threads:

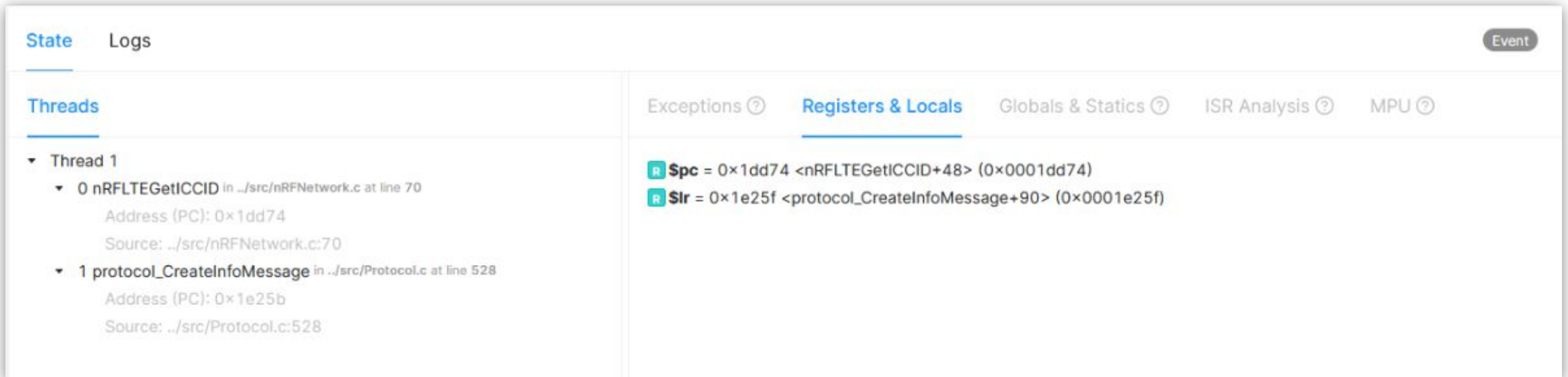
- Thread 1
 - 0 adxl343_write in ../src/adxl343.c at line 297
 - Address (PC): 0x20e0a
 - Source: ../src/adxl343.c:297
 - 1 ADXL343_readFromI2C in ../src/adxl343.c at line 406
 - Address (PC): 0x20ea3
 - Source: ../src/adxl343.c:406

Registers & Locals:

- R Spc = 0x20e0a <adxl343_write+26> (0x00020e0a)
- R Sir = 0x20ea7 <ADXL343_readFromI2C+18> (0x00020ea7)

- Non-critical fault - asserting trace to see
- Traced this on 3% of devices - non-critical but good to fix
- Either HW related or race-condition related

Example: SIM card fault



The screenshot shows a debugger window with two main panes. The left pane, titled 'Threads', shows a list of threads for 'Thread 1'. It contains two entries: '0 nRFLTEGetICCID in ../src/nRFNetwork.c at line 70' and '1 protocol_CreateInfoMessage in ../src/Protocol.c at line 528'. The right pane, titled 'Registers & Locals', shows the values of registers 'R Spc' and 'R \$lr'. 'R Spc' is 0x1dd74, and 'R \$lr' is 0x1e25f. The top of the window has tabs for 'State', 'Logs', 'Exceptions', 'Registers & Locals', 'Globals & Statics', 'ISR Analysis', and 'MPU'. An 'Event' button is in the top right corner.

State Logs Event

Threads

▼ Thread 1

- ▼ 0 nRFLTEGetICCID in ../src/nRFNetwork.c at line 70
Address (PC): 0x1dd74
Source: ../src/nRFNetwork.c:70
- ▼ 1 protocol_CreateInfoMessage in ../src/Protocol.c at line 528
Address (PC): 0x1e25b
Source: ../src/Protocol.c:528

Exceptions ? Registers & Locals Globals & Statics ? ISR Analysis ? MPU ?

R Spc = 0x1dd74 <nRFLTEGetICCID+48> (0x0001dd74)
R \$lr = 0x1e25f <protocol_CreateInfoMessage+90> (0x0001e25f)

- Failing to read SIM card upon boot
- Traced this on <0.1% of devices - non-critical as devices retry successfully
- HW related

Example: GPS fix failed

Gps Fix Error at nRFGPSWaitForFix [🔗](#)

[Resolve](#) [Merge](#) 👁️ LM

[Details](#) [All traces](#) [Comments 0](#) [Merged issues](#)

First Seen 5 days ago Last Seen 35 minutes ago Devices 1 Traces 4

Device :68
Cohort default
Software 1 prod (nrf9160_ns)
Hardware v3.2

⏮ Older Newer ⏭

Captured 35 minutes ago 📄

State [Logs](#) [Event](#)

Search ⬆ ⬇ ⌵ Filter 🗑

Lvl Message

73,0,0,26|72,0,0,30|13,0,0,20|7,0,0,25|30,0,0,24|

- Device GPS fix failing in certain cases
- Understand state of device when that happens
- Have option to log values, for example which satellites have been seen at what signal level

Example: NB-IoT modem GPS wait

Gps Error at wait_LTE_state [🔗](#)

Resolve Merge

Details All traces Comments 0 Merged issues

Device 451
Cohort default
Software (nrf9160_ns)
Hardware v2.0

First Seen 21 days ago Last Seen 12 minutes ago Devices 266 Traces 299

Older Newer

Captured 12 minutes ago

State Logs

Threads

Thread 1

- 0 wait_LTE_state in ./src/nrfGPS.c at line 433
Address (PC): 0x1fa16
Source: ./src/nrfGPS.c:433
- 1 nRFGPSStart in ./src/nrfGPS.c at line 384
Address (PC): 0x1fa5f
Source: ./src/nrfGPS.c:384

Registers & Locals

Spc = 0x1fa16 <wait_LTE_state+54> (0x0001fa16)
Slr = 0x1fa63 <nRFGPSStart+46> (0x0001fa63)

- nRF9160 modem and GPS can not be used at the same time
- Mechanism implemented to prevent this, asserting issue to track how often these events happen
- FW related

Example: Prioritizing Fixes

Sort by Trace Count: High to Low

UnresolvedAll

Filter

Titlee.g. Assert

ReasonMem Fault x Watchdog x Assert x

Cohorte.g. default

Devicee.g. ABCD1234

Software Typee.g. firmware-main

Software Versione.g. 1.0.0-alpha

Hardware Versione.g. EVT

| Title | Count | Devices |
|--|-------|---------|
| <div>Assert</div> <div>Assert at prv_check1</div> <div>proto-software 1.0.1 – 0.9.0 a day ago – 4 months ago</div> | 24202 | 378 |
| <div>Assert</div> <div>Assert at cli_execute</div> <div>proto-software 1.0.1 – 0.0.3 8 hours ago – 3 months ago</div> | 5412 | 332 |
| <div>Assert</div> <div>Assert at timeout_handler_exec</div> <div>proto-software 1.0.0 8 hours ago – 3 months ago</div> | 4025 | 444 |
| <div>Assert</div> <div>Assert at prv_recursive_crash</div> <div>proto-software 1.0.1 – 1.0.0 4 hours ago – 3 months ago</div> | 2822 | 386 |
| <div>Assert</div> <div>Assert at _esp_error_check_failed</div> <div>main 1.0.0-md5+f46b8e5d a day ago – 3 months ago</div> | 1351 | 154 |
| <div>Watchdog</div> <div>Watchdog at MemfaultWatchdog_Handler</div> <div>proto-software 1.0.2-beta1 5 days ago – 3 months ago</div> | 1411 | 193 |
| <div>Mem Fault</div> <div>Mem Fault at compute_fft [Stack Overflow in accel-workq]</div> <div>main 1.0.0-md5+a1c641ba a day ago – 3 months ago</div> | 1427 | 203 |

1–7 of 7 records < 1 >



Zephyr™ Project
Developer Summit

Connectivity Metrics

Using Metrics to Monitor Performance

- Not all issues result in resets!
- Many factors can impact connectivity
 - location / RF environment
 - antenna efficiency
 - data being transferred
 - CPU & task utilization, time sleeping
- Enables health comparisons across all devices and between firmware releases

Adding Metrics to Zephyr with Memfault

1. Define metric

```
MEMFAULT_METRICS_KEY_DEFINE(  
    LteDisconnect,  
    kMemfaultMetricType_Unsigned)
```

2. Update metric in code

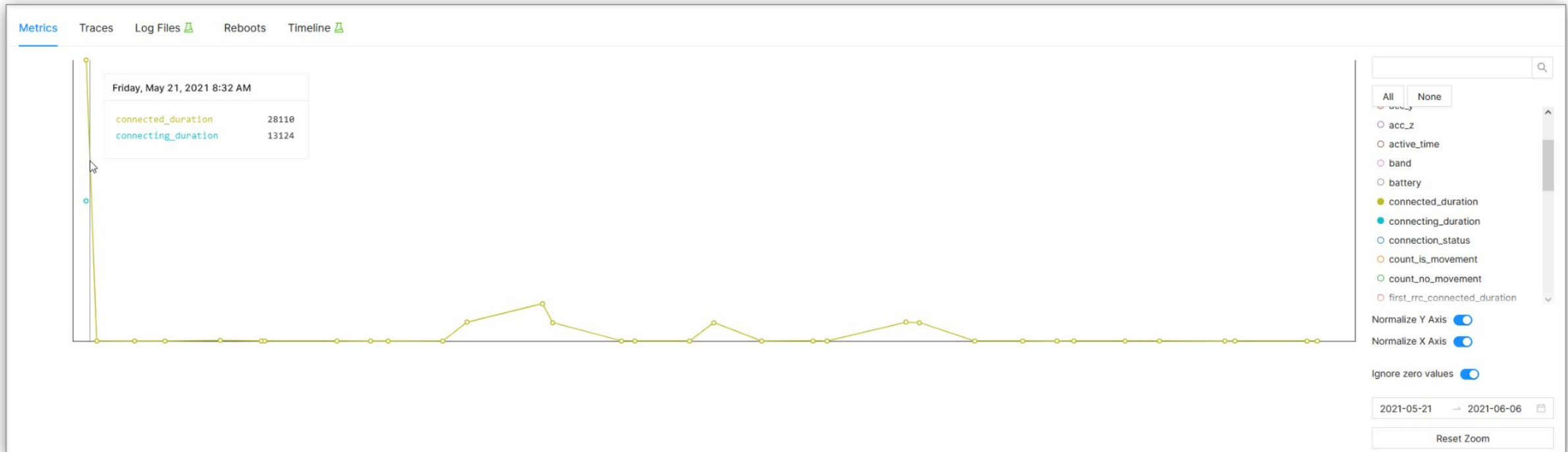
```
void lte_disconnect(void) {  
    memfault_metrics_heartbeat_add(  
        MEMFAULT_METRICS_KEY(LteDisconnect), 1);  
    //...  
}
```

Memfault SDK + Cloud



- Serializes and compresses metrics for transport
- Indexes Metrics by device and firmware version
- Exposes web interface for browsing metrics by device and across Fleet

Example: NB-IoT/LTE-M basic connectivity



- **Connected:** Time modem is actively communicating with mobile network
- **Connecting:** Time modem requires to connect to mobile network
- Track activity and power consumption

Example: base stations and PSM in NB-IoT/LTE-M

Metrics

Traces

Log Files

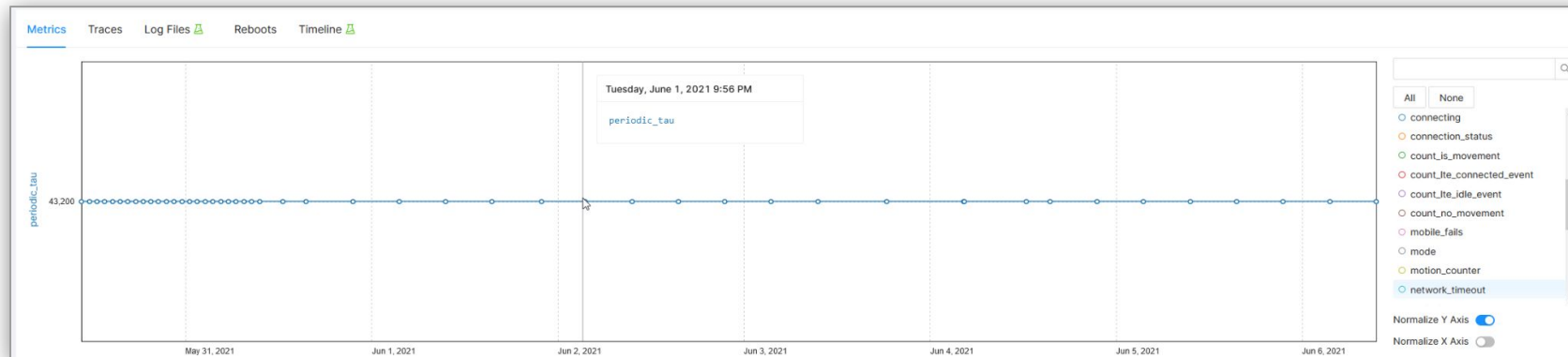
Reboots

Timeline

Software Version

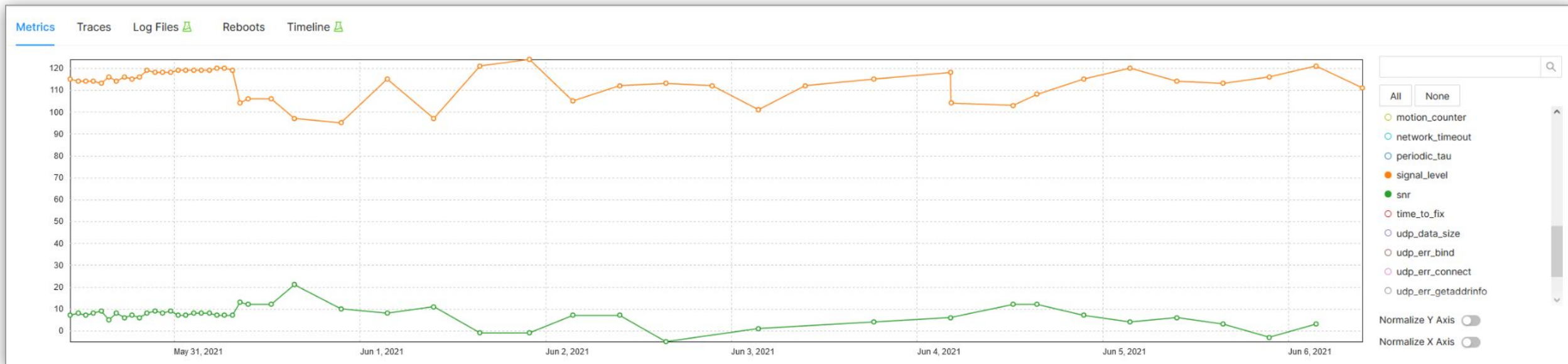
Start date → End date

| Issue | Captured | Source Type | Device Serial | Software Version |
|--|-------------|-------------|---------------|------------------|
| <div>Network Status</div> <div>parseXMonitor: %XMONITOR: 1,"", "", "24201", "76C1", 7, 20, "01333A00", 435, 6400, 23, 30, "", "00000010", "00101100", "01011001"</div> | 11 days ago | Event | 58 | 5 |
| <div>Network Status</div> <div>parseXMonitor: %XMONITOR: 1,"", "", "24201", "76C1", 7, 20, "01333A00", 435, 6400, 36, 37, "", "00000010", "00101100", "01011001"</div> | 12 days ago | Event | 58 | 4 |



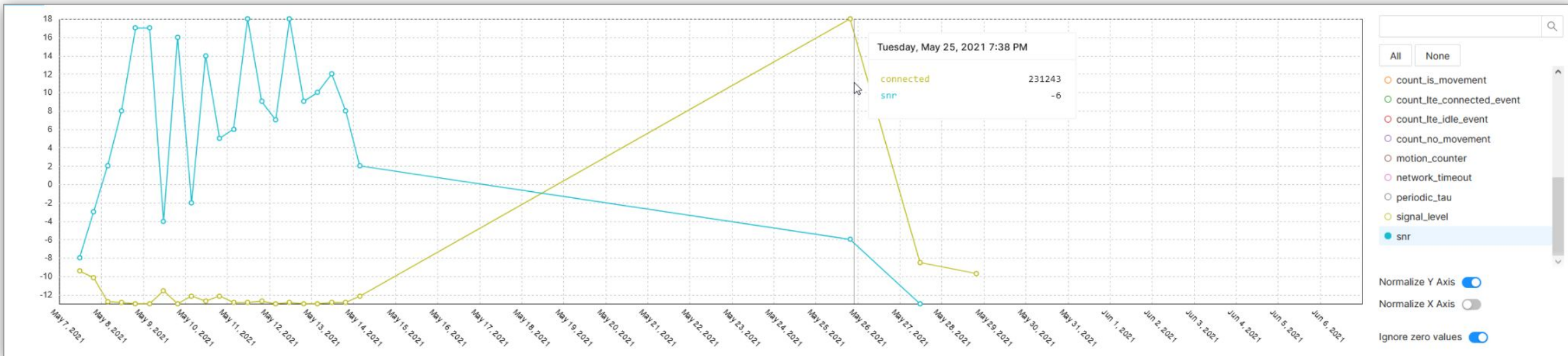
- Tracking base-station response upon connect
 - Check timer responses for PSM/eDRX
 - Check IDs and rough locations
- Correlate issues with particular base-stations or networks

Example: Mobile network signal quality



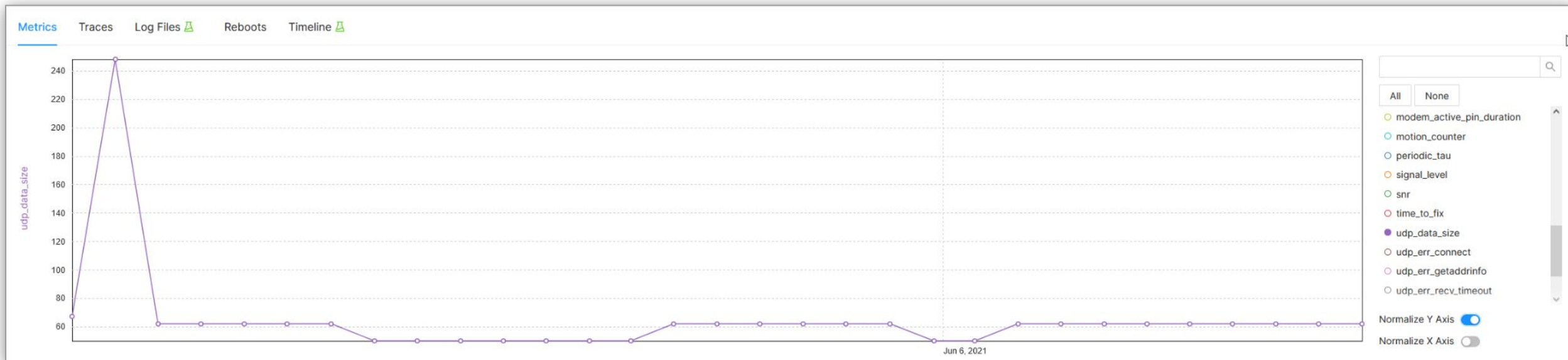
- Signal level: Monitoring quality of coverage for moving device
- SNR: Link quality
- Track what is the average value across fleet

Example: NB-IoT/LTE-M bad coverage



- Connected: Time spend sending data, SNR: Link quality
- Most of the time connected time is low, on bad SNR it significantly increases. 15s ---> 250s, same amount of data to send.
- Introduced a timeout based on SNR, better to skip sending

Example: NB-IoT/LTE-M data size



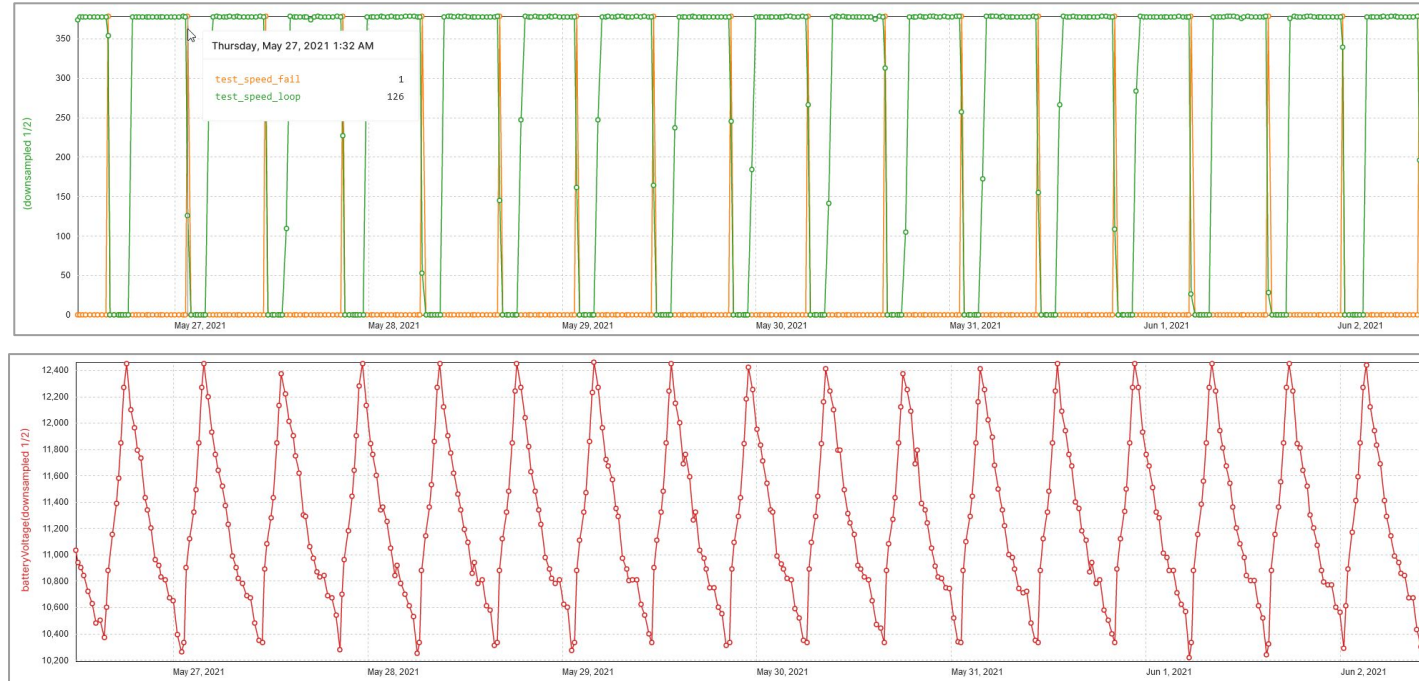
- UDP data size: Track bytes per send interval
- Post-reboot more data is sent
- Some packets are bigger due to more info or traces
- Track issue of data consumption



Zephyr™ Project
Developer Summit

Automated testing

Example: Device cyclic testing



- Track automated tests progress
- On-device metrics: battery, runtime, number of inputs/output...
- Test-jig metrics: test pass/fail count, number of requested inputs...
 - via REST API from jig
- Compare on-device and test system results to track issues

At Institute IRNAS, we strive to apply the vast **scientific knowledge to everyday reality**, by creating **hardware products and IoT systems** that are:

- *effective,*
- *affordable,*
- *well-tailored,*
- *future-proof.*

We believe in an open-source world and sharing.

We aim to empower the world with technologies that improve lives, let that be an advanced communication system, an open, affordable medical device, 3D bioprinting or a simple everyday utensil.

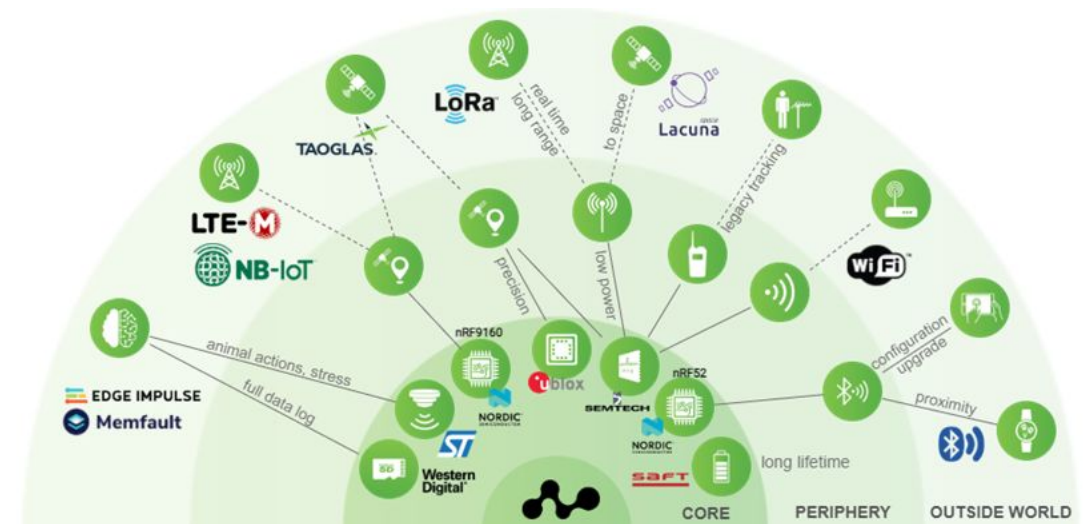
6-in-One Complete Service



- **Electronics Engineering**
- **Software Engineering**
- **Mechanical Engineering**
- **Rapid Prototyping**
- **Small to medium-size series manufacturing**
- **Experimental testing for scientific applications**

Why IRNAS for Zephyr Devices

- **Product Development** - Offer a complete development service, taking your project from the idea to the finished product. Focusing on industrial IoT applications primarily on BLE, NB-IoT/LTE-M, LoRaWAN based on Nordic Semiconductor solution and running Zephyr.
- **In-house Manufacturing** - In-house fabrication lab is fully equipped for prototyping & manufacturing, and it includes an electronics PnP line, 3D printers, a laser cutter, a CNC workstation, a CNC mill, and more.
- **Cross-Disciplinary Team** - Highly-skilled team of scientists and engineers with expertise in mechanical, electronic and software engineering, data analysis and numerical control, acoustical, medical and bio-engineering.



IRNAS technology map 2021

Why Memfault for Zephyr Devices



Memfault

Fault Debugging

- Zephyr integrations for 1.14 LTS - 2.6
- Automatic Issue Deduplication
- Zephyr RTOS Task Awareness
- Fault handler provided as part of C-SDK
- Full stacktrace and variable recovery

Device Monitoring

- Easily scale up or down
- Add custom metrics with 2 lines of code (battery level, connectivity stats, RTOS Statistics, etc)
- Device and fleet-level metrics in one dashboard

OTA Firmware Updates

- Send bug fixes from the same platform
- Deploy and schedule cohort-based and staged rollouts
- Stop faulty updates with one click

IRNAS

- [IRNAS Website](#)
- IRNAS Blog: [ElephantEdge tracker: Designing the firmware and first prototype solution](#)
- IRNAS Blog: [RAM-1: Remote monitoring of smart power grids with cellular IoT- and Bluetooth LE-powered device](#)

Memfault

- [Memfault Free Trial](#)
- Interrupt Blog: [How to debug a HardFault on an ARM Cortex-M MCU](#)
- Interrupt Blog: [Fix Bugs and Secure Firmware with the MPU](#)
- Interrupt Blog: [A Practical guide to ARM Cortex-M Exception Handling](#)
- Interrupt Blog: [A Guide to Watchdog Timers for Embedded Systems](#)



Zephyr™ Project
Developer Summit

Questions?



ZephyrTM Project

Developer Summit

June 8-10, 2021 ▪ @ZephyrIoT