Zephyr® Project
Developer Summit

Open Source Ultra-Wideband RTLS with Zephyr — Developent Experience

Aleksander Wójtowicz, *AVSystem*

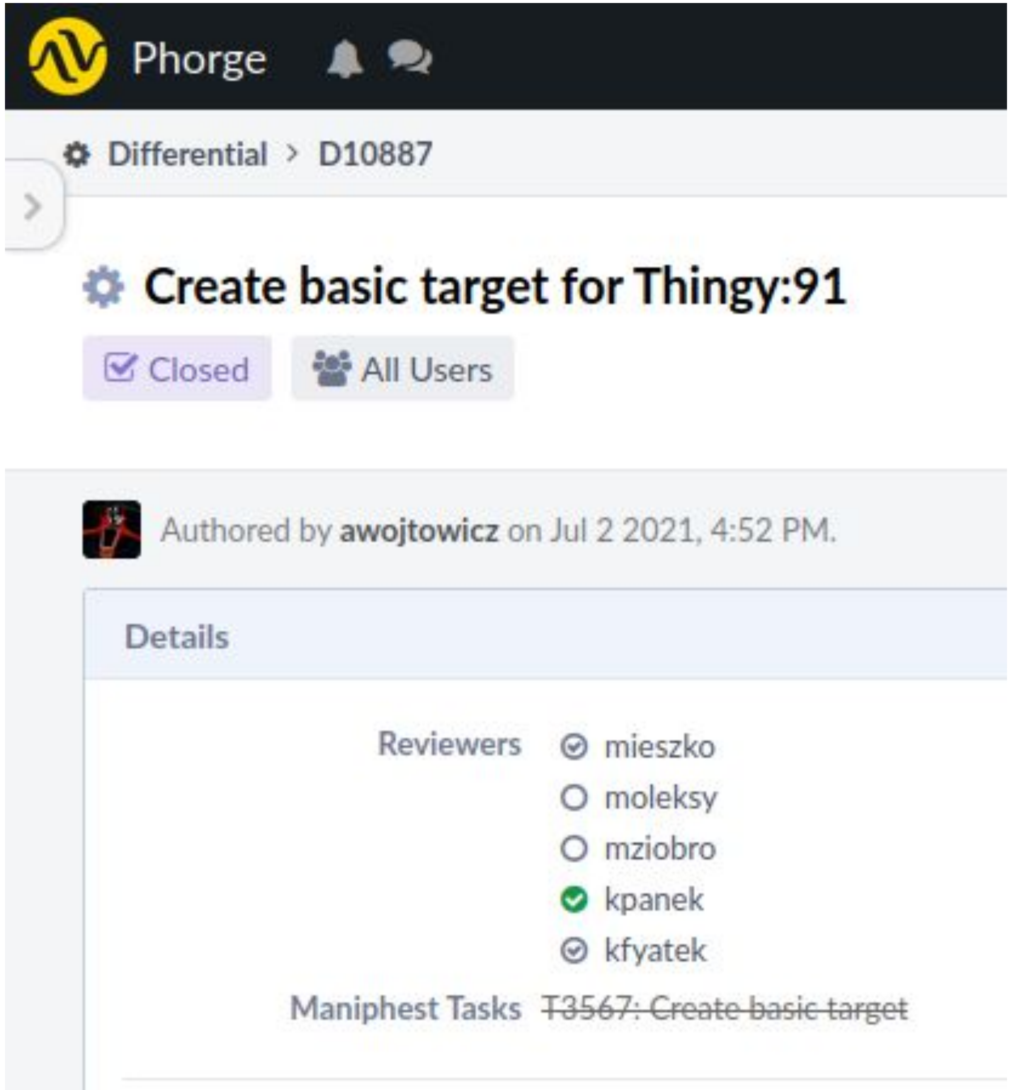
#EMBEDDEDOSSUMMIT

About me

- Aleksander Wójtowicz [vooytovitch]
  - https://github.com/anuar2k
- Fresh CS graduate @ AGH UST, Kraków
- Embedded Software Engineer @ AVSystem
  - we do IoT device management, based on LwM2M
  - I work on Anjay, our LwM2M client and its ports, including Zephyr

About me

- I didn’t know about Zephyr before joining AVSystem
- It was also the very first thing I worked on!



Talk plan

- what are Real Time Locating Systems?
- what’s Ultra-Wideband? how UWB-based RTLSs work?
- project showcase
- implementation details and my experience with Zephyr
  - how it helps with quick development?
  - how it helps with making generic software?
  - problems I’ve encountered
  - many tips & tricks

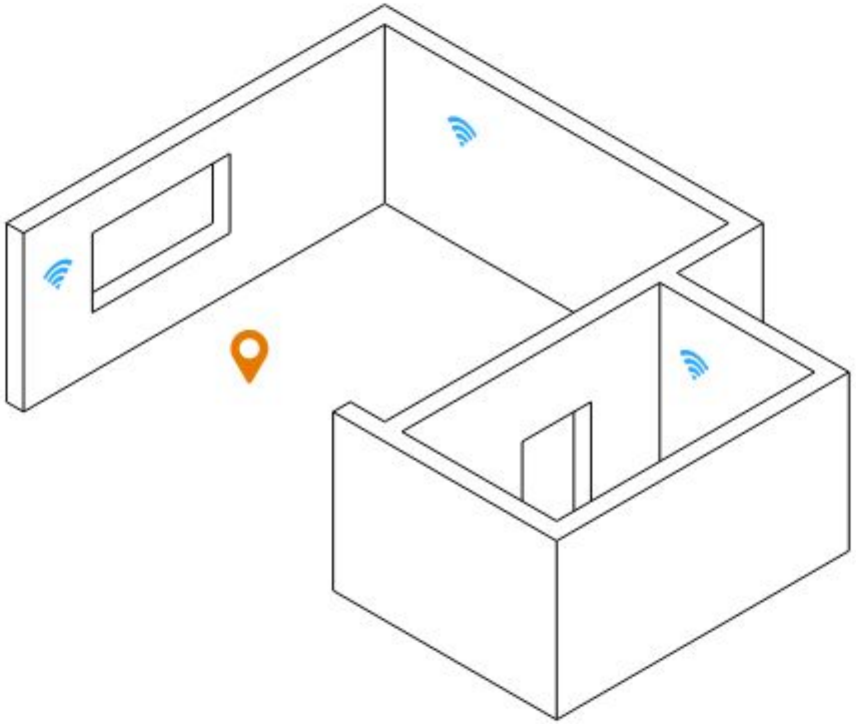
Zephyr® Project
Developer Summit

Real Time Locating Systems and Ultra-Wideband

#EMBEDDEDOSSUMMIT

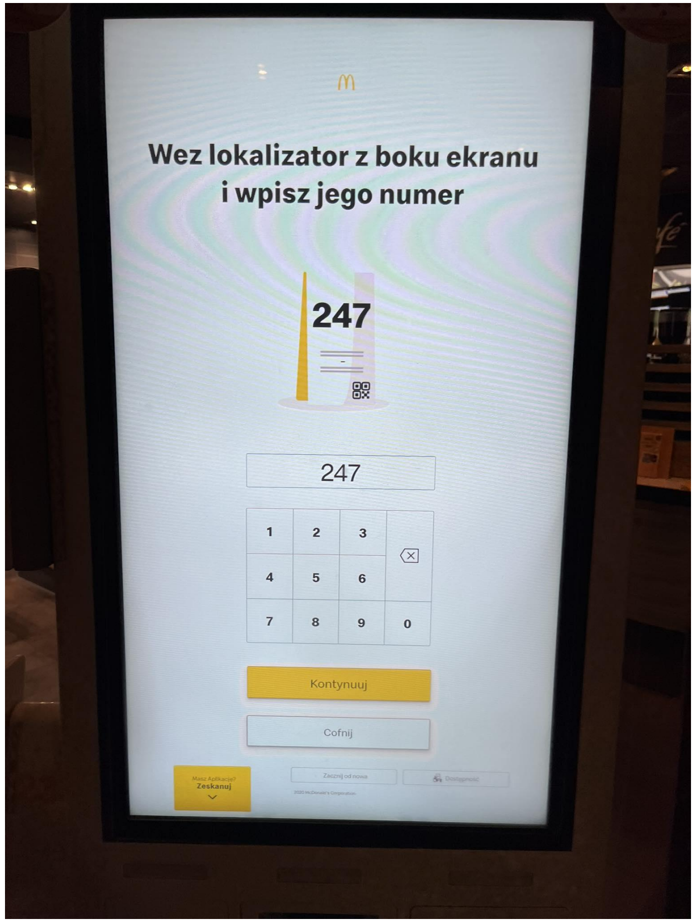
What are RTLSs for

- real time asset tracking
  - vehicles,
  - tools,
  - workforce…
- location data is logged to some service
- indoor setting
  - can’t use GPS trackers



In practice


Dodaj produkty
247


Wez lokalizator z boku ekranu
i wpisz jego numer
247
247
1
2
3
4
5
6
7
8
9
0
Kontynuuj
Cofnij


247

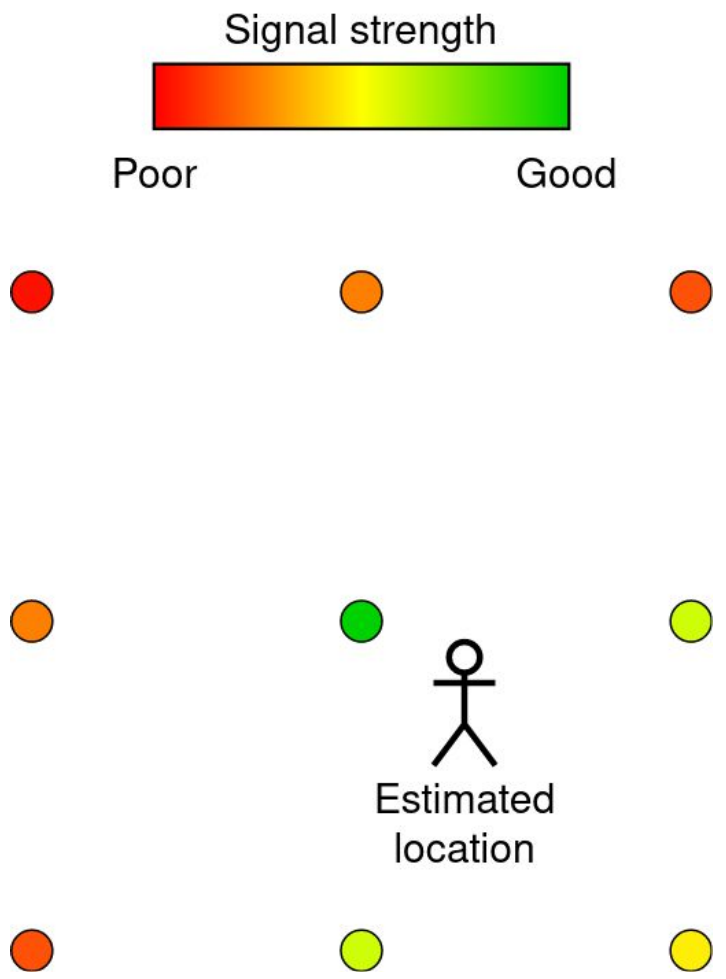
In practice


RadBeacon Locator Tent
Model: RBT-006
Radius Networks, Inc.
radiusnetworks.com Assembled in China
200401

Zephyr® Project
Developer Summit

In practice

- various methods
- RSSI-based is the most popular
  - array of “anchors” around the area
  - “tag” on moving object
  - stronger signal -> closer you are to anchor
  - multiple measurements -> location estimation



Signal strength with Bluetooth/Wi-Fi

pros:

- cheap trackers
- works indoors

cons:

- RSSI changes due to many factors, not just distance
- that ~1 m accuracy may be still not good enough

What's Ultra-Wideband?

- yet another radio technology
- recent growth of use in consumer electronics
- since 2007, also a PHY in IEEE 802.15.4
    - merged to main spec in 2011
- **unique physical properties for accurate distance measurement**
    - works in non-line-of-sight scenarios
    - this can be used to build an accurate RTLS

Apple AirTag

Photo: Apple

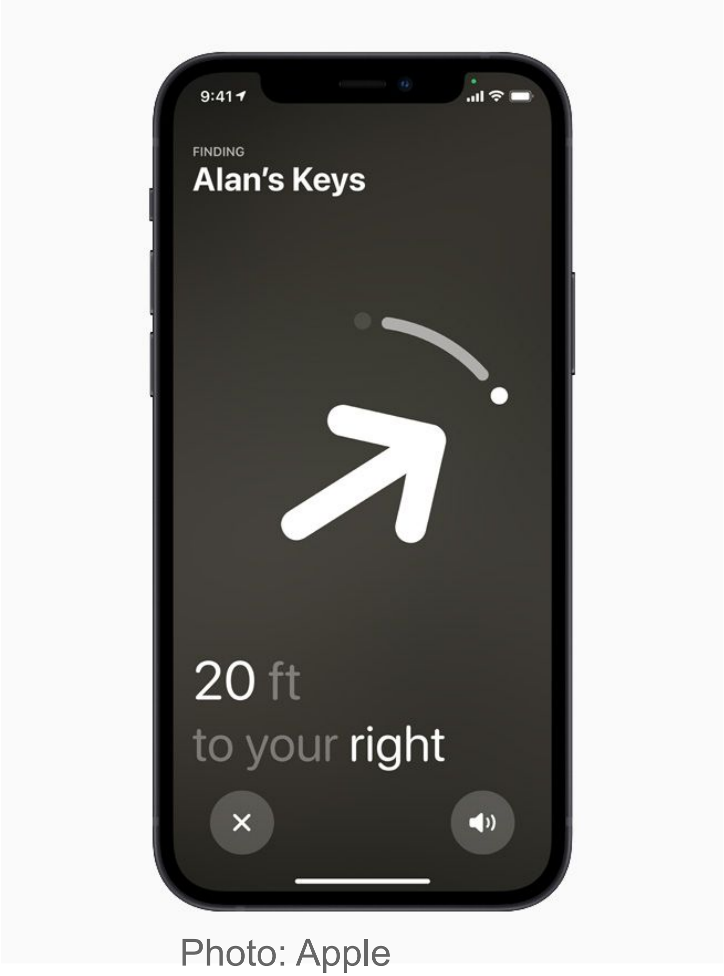


Photo: Apple

BMW Digital Key Plus

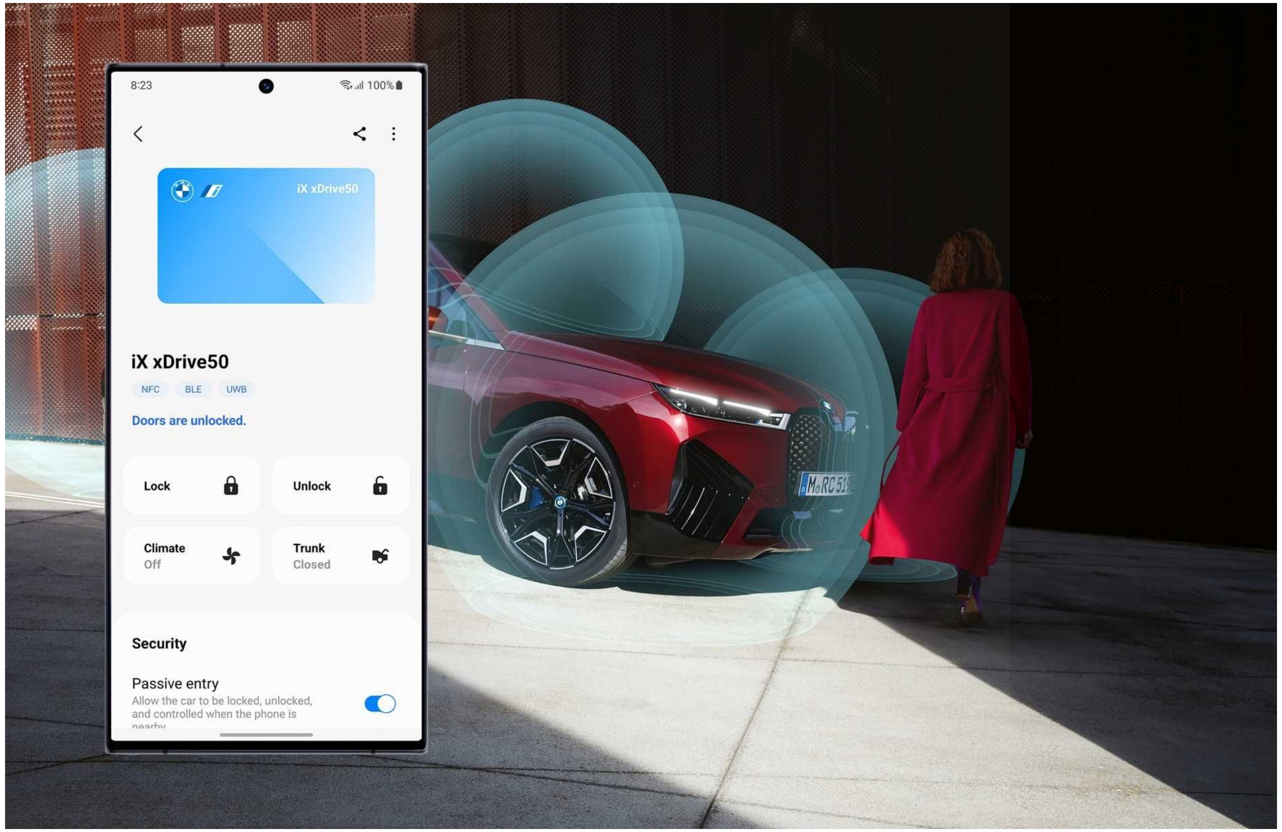


Photo: https://www.press.bmwgroup.com/usa/article/detail/T0414019EN_US/bmw-digital-key-plus-now-available-on-compatible-android-devices

UWB use in RTLS

bands: ~3 to 10 GHz, bandwidth: >= 500 MHz (!!!)

↓

high RX time measurement resolution + multipath detection

↓

time of flight calculation

↓

distance = ToF * speed of light

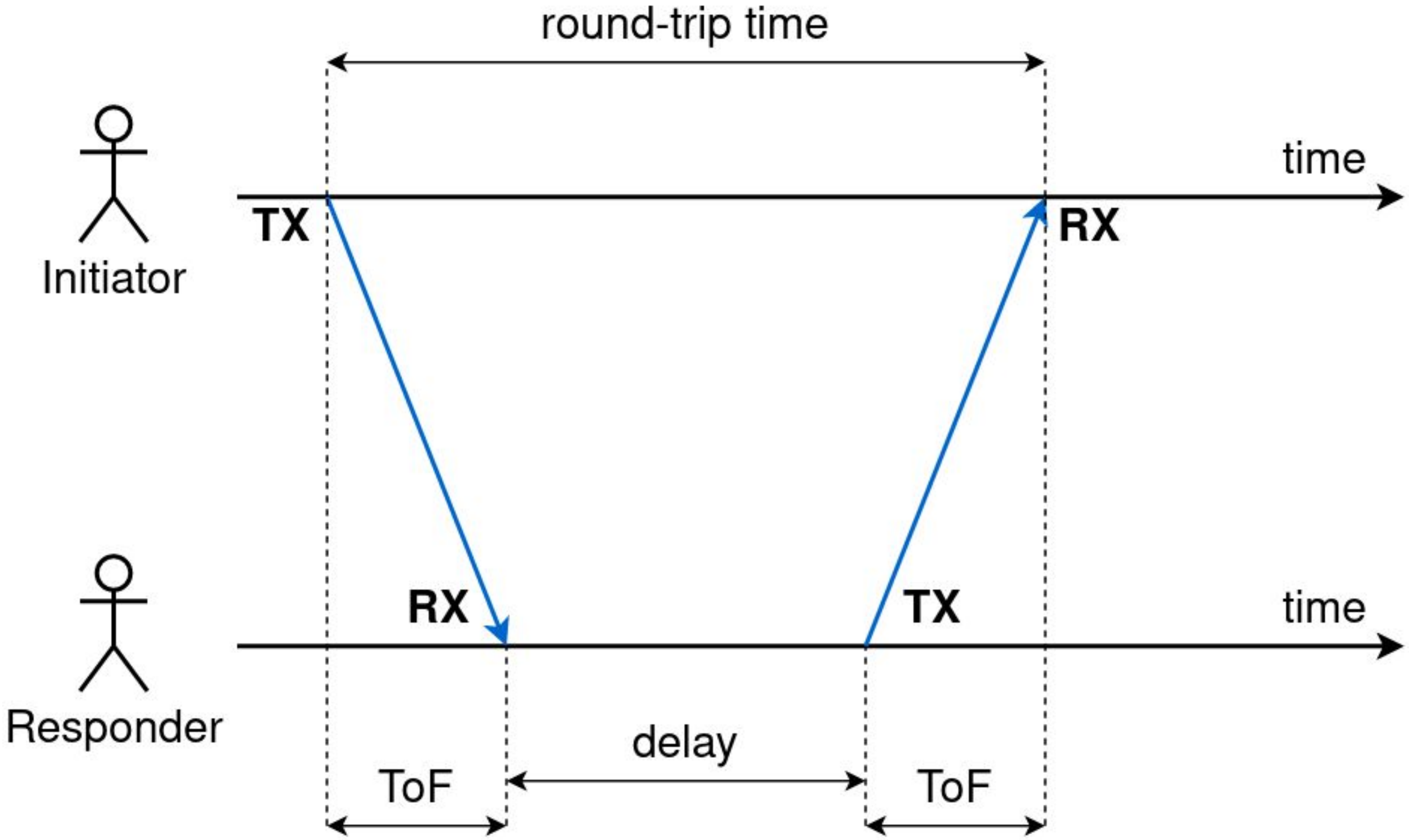
↓

multiple distances to anchors

↓

location

Single-Sided Two-Way Ranging



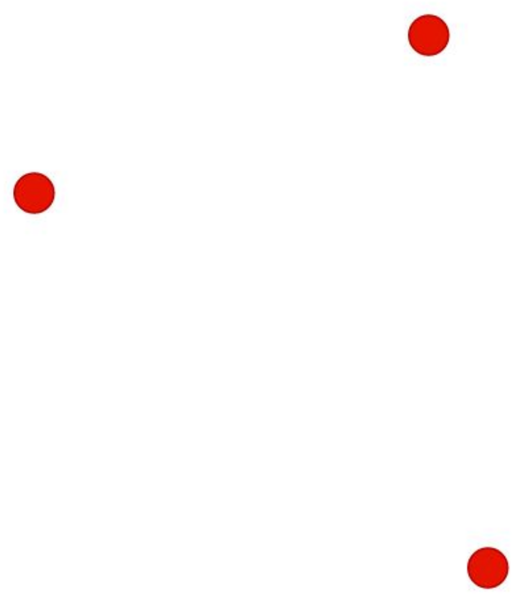
$$ToF = \frac{round\,trip\,time - delay}{2}$$

$$d = c * ToF$$

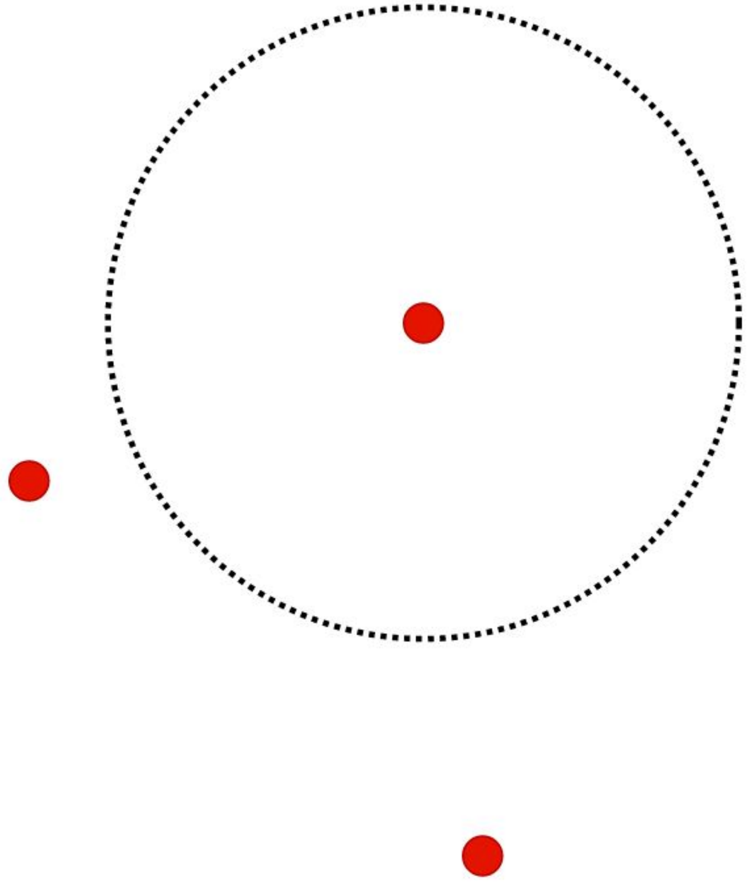
Accuracy

- ~20 cm easily
  - for SS-TWR clock drift correction is needed
- with additional processing, antenna calibration, etc.
  up to 2 cm

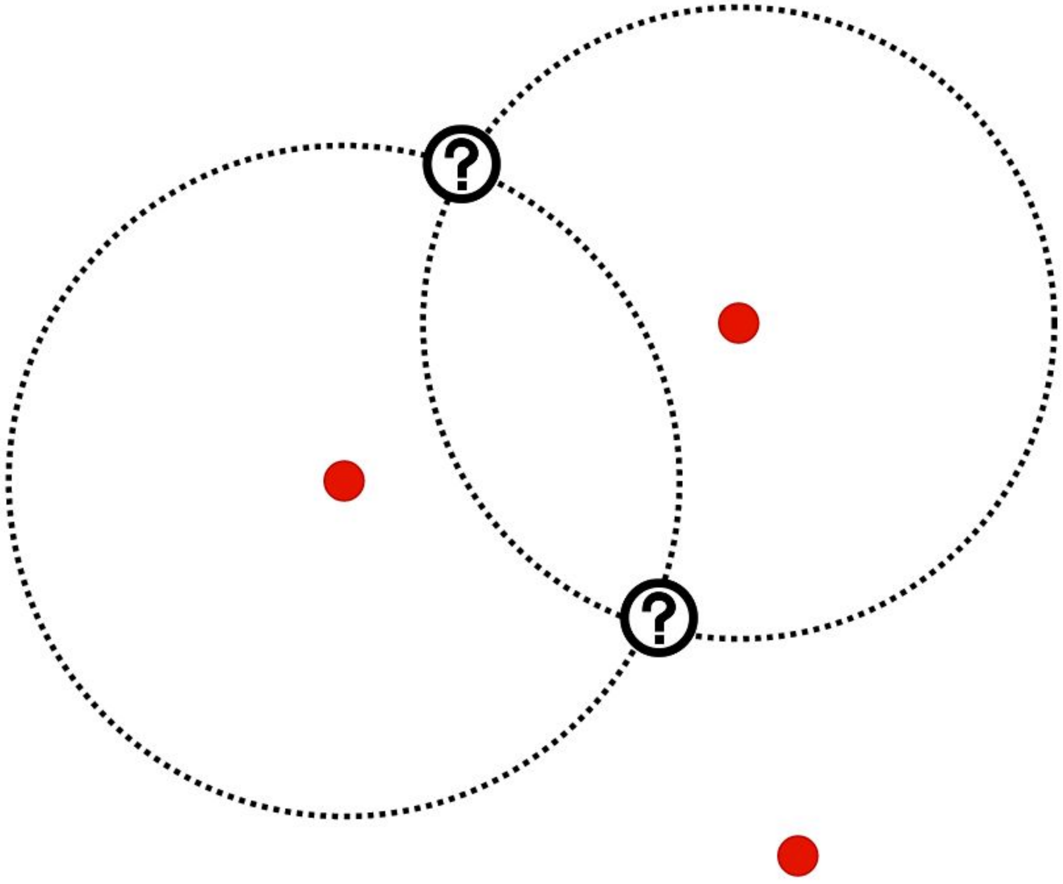
True-range multilateration 2D



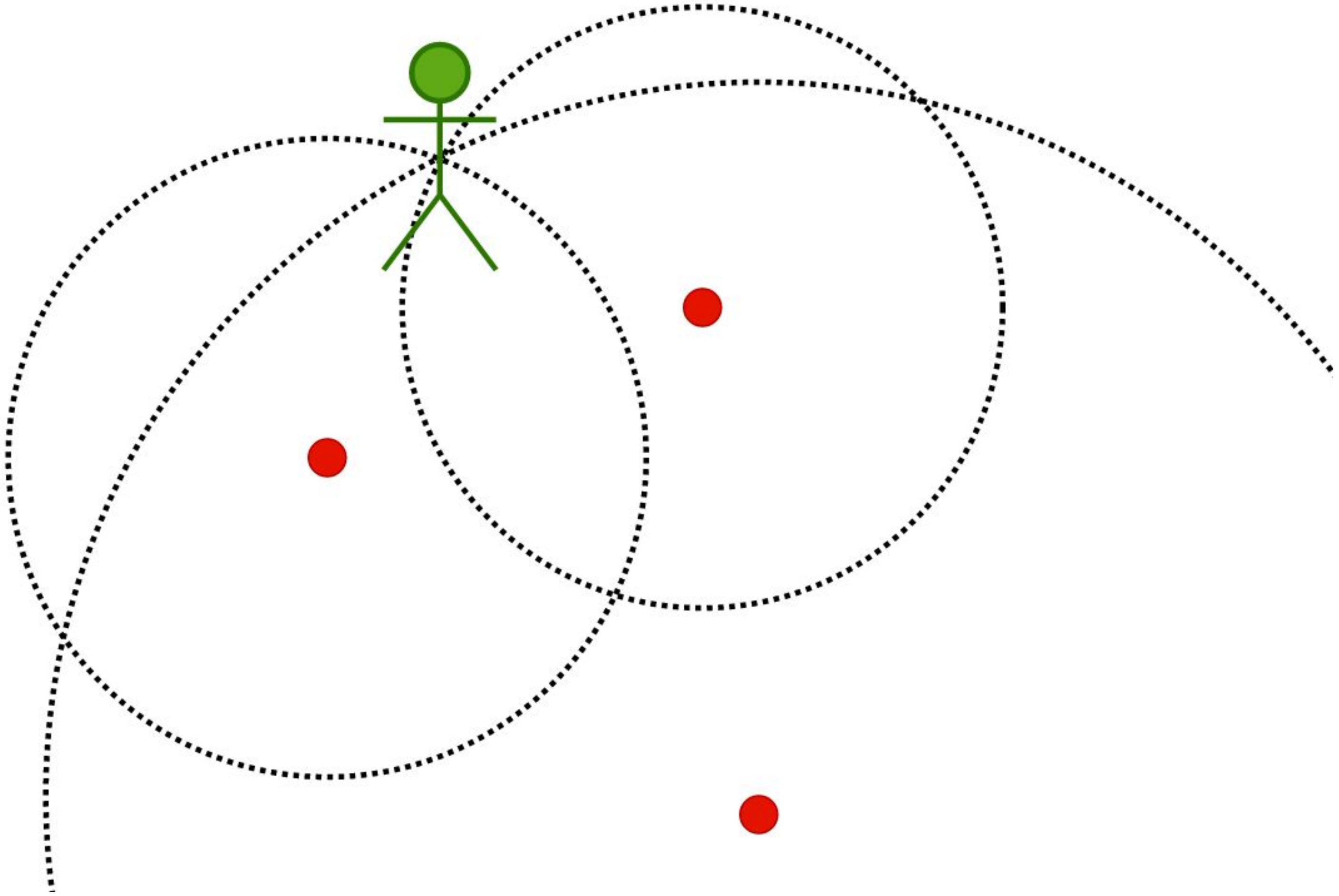
True-range multilateration 2D



True-range multilateration 2D



True-range multilateration 2D



True-range multilateration 3D

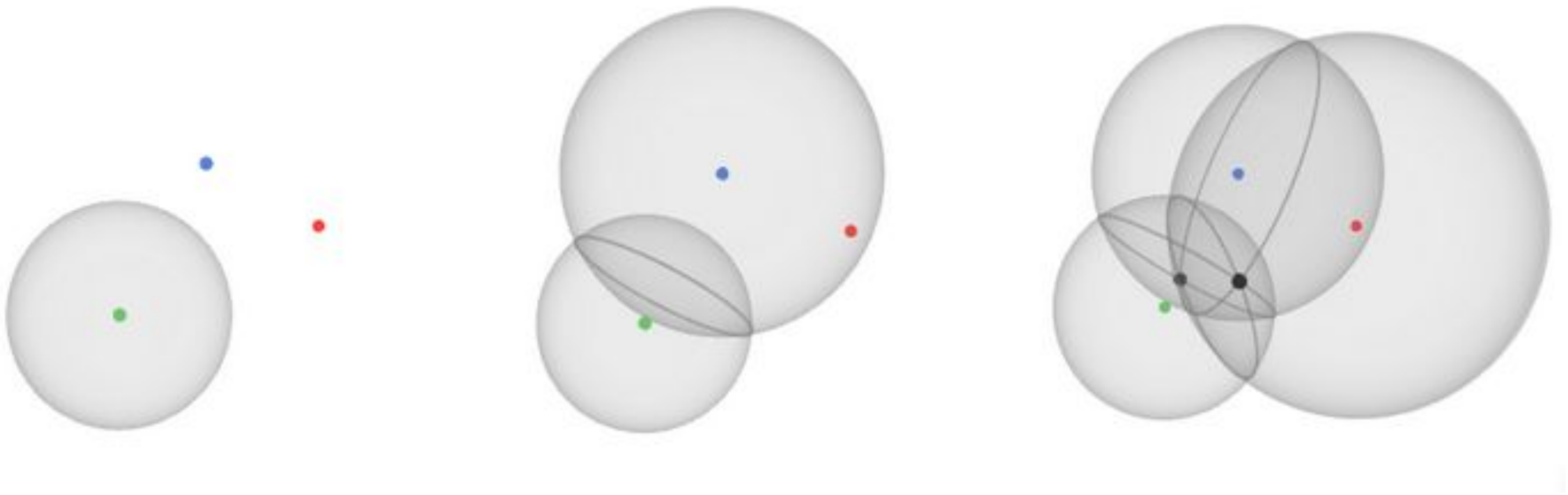


Photo: ciechanow.ski/gps

High accuracy RTLS use cases

- accident prevention
  - track workforce and heavy machines
- asset tracking
  - shorter access time
  - usage analysis
- shopping carts with navigation!

Zephyr® Project
Developer Summit

Project showcase

#EMBEDDEDOSSUMMIT

HyperRTLS

- open source UWB RTLS
- made as part of engineering thesis
  - coauthored with Sebastian Szczepański
  - supervised by professor Tomasz Szydło
- https://github.com/HyperRTLS/
  - you’ll find also thesis full text there

Features

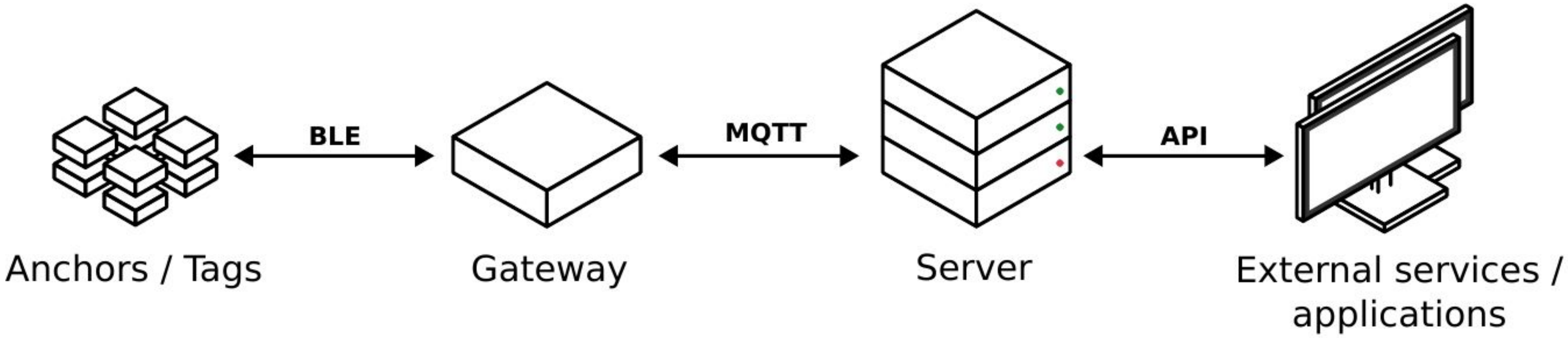
- Zephyr-based apps for hardware
  - tags and anchors
  - default target: Decawave MDEK1001
    - nRF52832 (with BLE), DW1000 UWB IC
  - gateway app
    - requires BLE and IP stack

Photo: https://www.qorvo.com/products/p/MDEK1001

Features

- Backend software
  - Node.js app
    - connects to Mosquitto and PostgreSQL
    - serves REST API for RTLS management, location retrieval
      - supposed to be used by end products with business logic
  - example app using the REST API

Data flow


BLE
MQTT
API
Anchors / Tags
Gateway
Server
External services / applications

Motivation

- engineering thesis, obviously…
- there's no end-to-end open source UWB RTLS solution
- there's tens of companies which offer commercial systems
  - they're pricey; entry barrier is high, while UWB modules are rather cheap
    - MDEK1001 (bundle of 12 devices) is ~300 USD
- learning purposes
  - even Decawave's positioning stack is a binary blob…

Demo

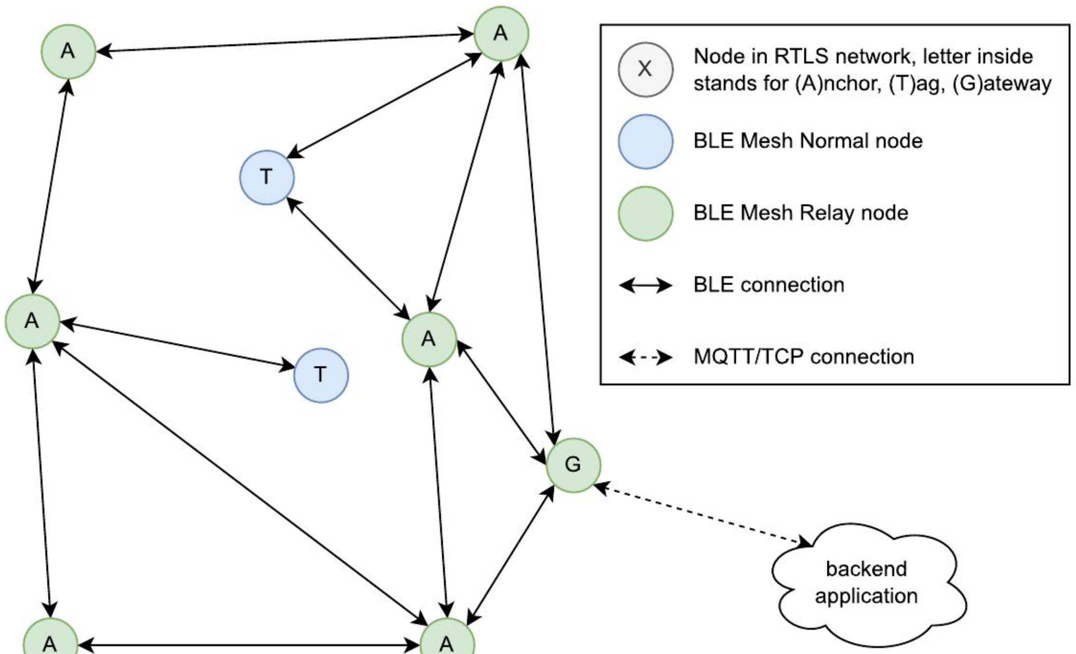
Zephyr® Project
Developer Summit

Implementation and development experience

#EMBEDDEDOSSUMMIT

Gateway

- no IP stack on devkits, so no MQTT
- observation: anchors and tags are always in Bluetooth range
- idea: use mesh networking
  - anchors are backbone
- we'd like to use OpenThread but no support on nRF52832 :(



Options

- Linux SBC (like Raspberry Pi) and script talking to BlueZ through dbus
  - cool, since it could run on dev PC as well
  - …but both dbus and BlueZ API are too complex =(
    - wrappers didn’t help
- Zephyr?
  - use the same API everywhere!
  - …but rest of development experience will be worse

Or won’t it?

- turns out: we still can get away with bringing no real HW
- Zephyr has many emulators
  - QEMU (closest to running on real HW),
  - native_posix (Zephyr as Linux process),
- at first they sound useless
  - emulators are supposed to *emulate*, right…?
- **but: you can proxy real peripherals to them**
  - we went with QEMU

IP stack on QEMU

- TUN or TAP interface
  - SLIP
    - proxied over Unix socket
  - Intel E1000
    - virtualized over TAP iface by QEMU
      - the preferred way

In practice

```
anuar2k:~/zephyrproject/hyperrtls-gateway$ cat boards/qemu_x86_gw.conf
### Networking
CONFIG_PCIE=y
CONFIG_ETH_E1000=y

CONFIG_NET_CONFIG_SETTINGS=y
CONFIG_NET_CONFIG_NEED_IPV4=y
CONFIG_NET_CONFIG_MY_IPV4_ADDR="192.0.2.1"
CONFIG_NET_CONFIG_MY_IPV4_GW="192.0.2.2"
CONFIG_NET_CONFIG_MY_IPV4_NETMASK="255.255.0.0"
CONFIG_NET_L2_ETHERNET=y
CONFIG_NET_QEMU_ETHERNET=y
```

In practice

```
git clone https://github.com/zephyrproject-rtos/net-tools.git
sudo ./net-tools/net-setup.sh start
sudo ~/hyperrtls-gateway/setup_networking.sh wlp0s20f3
```

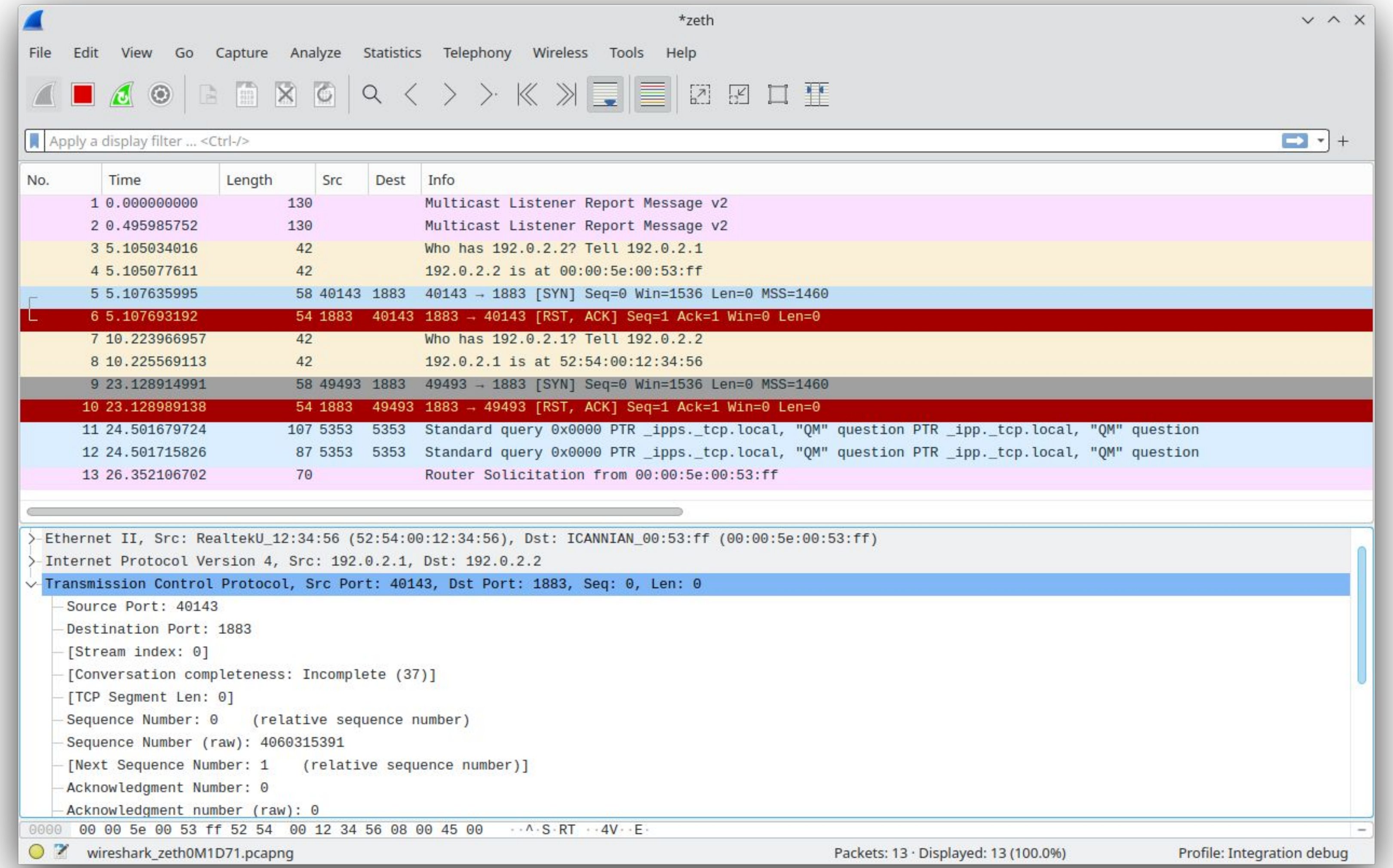
Target-specific hacks

```c
void main(void) {
    // omitted for brevity...






    gw_mqtt_client_run(&client_config);
    LOG_ERR("MQTT client unexpectedly returned");
    hrtls_fail();
}
```

Target-specific hacks

```c
void main(void) {
    // omitted for brevity...

    #ifdef CONFIG_BOARD_QEMU_X86
    // TODO: why this is needed on qemu_x86? WTF
    k_sleep(K_SECONDS(5));
    #endif // CONFIG_BOARD_QEMU_X86

    gw_mqtt_client_run(&client_config);
    LOG_ERR("MQTT client unexpectedly returned");
    hrtls_fail();
}
```

pcap!


*zeth
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Length Src Dest Info
1 0.000000000 130 Multicast Listener Report Message v2
2 0.495985752 130 Multicast Listener Report Message v2
3 5.105034016 42 Who has 192.0.2.2? Tell 192.0.2.1
4 5.105077611 42 192.0.2.2 is at 00:00:5e:00:53:ff
5 5.107635995 58 40143 1883 40143 → 1883 [SYN] Seq=0 Win=1536 Len=0 MSS=1460
6 5.107693192 54 1883 40143 1883 → 40143 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7 10.223966957 42 Who has 192.0.2.1? Tell 192.0.2.2
8 10.225569113 42 192.0.2.1 is at 52:54:00:12:34:56
9 23.128914991 58 49493 1883 49493 → 1883 [SYN] Seq=0 Win=1536 Len=0 MSS=1460
10 23.128989138 54 1883 49493 1883 → 49493 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11 24.501679724 107 5353 5353 Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question
12 24.501715826 87 5353 5353 Standard query 0x0000 PTR _ipps._tcp.local, "QM" question PTR _ipp._tcp.local, "QM" question
13 26.352106702 70 Router Solicitation from 00:00:5e:00:53:ff
Ethernet II, Src: RealtekU_12:34:56 (52:54:00:12:34:56), Dst: ICANNIAN_00:53:ff (00:00:5e:00:53:ff)
Internet Protocol Version 4, Src: 192.0.2.1, Dst: 192.0.2.2
Transmission Control Protocol, Src Port: 40143, Dst Port: 1883, Seq: 0, Len: 0
Source Port: 40143
Destination Port: 1883
[Stream index: 0]
[Conversation completeness: Incomplete (37)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 4060315391
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
Acknowledgment number (raw): 0
0000 00 00 5e 00 53 ff 52 54 00 12 34 56 08 00 45 00 ··^·S·RT ··4V··E·
wireshark_zeth0M1D71.pcapng
Packets: 13 · Displayed: 13 (100.0%)
Profile: Integration debug

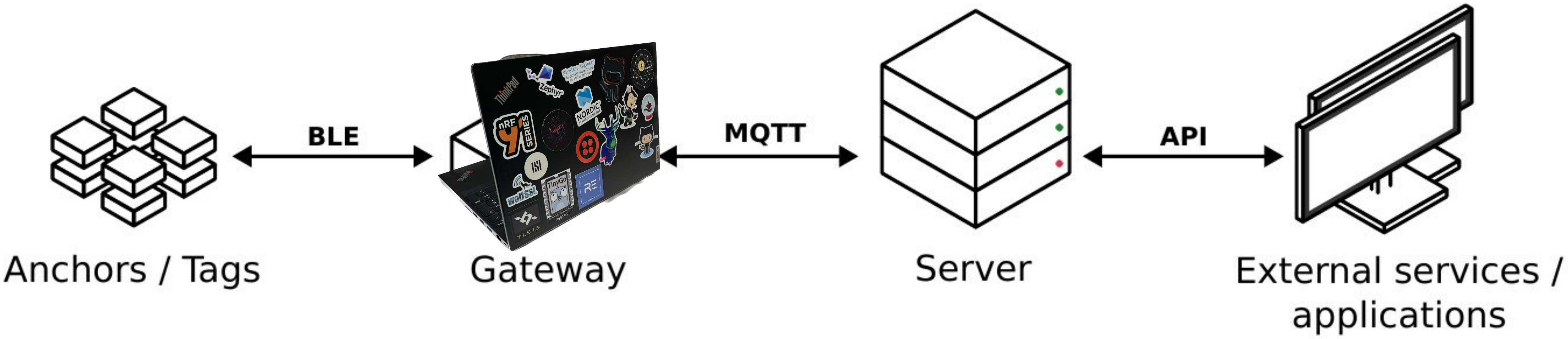
Bluetooth Low Energy

- Host and Mesh implementation are just software
- Controller is linked using HCI and runs on
  - the same chip (as in nRF52), uses RAM
  - external chip (as in nRF9160DK), uses UART
  - **a peripheral BLE controller on host PC**
    - in case of QEMU: btproxy forwards Linux Bluetooth socket to Unix socket, forwarded to Zephyr as a serial device

In practice

```
hciconfig
sudo hciconfig hci<idx> down
sudo <bluez_dir>/tools/btproxy -u -i <idx>
```

Data flow - my setup



Side note: nRF Connect SDK

- NCS is a fork of Zephyr for Nordic Semi products
  - most APIs compatible with upstream Zephyr
- Bluetooth well documented, ton of examples
  - some things don't work OOTB because of e.g. macros unavailable in upstream
    - just inline them ;)

- Bluetooth: Central and Peripheral HRS
- Bluetooth: Central BAS
- Bluetooth: Central HIDS
- Bluetooth: Central Heart Rate Monitor with Coded PHY
- Bluetooth: Central NFC pairing
- Bluetooth: Central SMP Client
- Bluetooth: Central UART
- Bluetooth: Direct Test Mode
- Bluetooth: Direction finding central
- Bluetooth: Direction finding connectionless locator
- Bluetooth: Direction finding connectionless beacon
- Bluetooth: Direction finding peripheral
- Bluetooth: EnOcean
- Bluetooth: HCI low power UART
- Bluetooth: LLPM
- Bluetooth: Multiple advertising sets
- Bluetooth: nRF Distance Measurement with Bluetooth LE discovery
- Bluetooth: Peripheral AMS client
- Bluetooth: Peripheral ANCS client
- Bluetooth: Peripheral Bond Management Service (BMS)
- Bluetooth: Continuous Glucose Monitoring Service (CGMS)
- Bluetooth: Peripheral CTS client
- Bluetooth: Fast Pair
- Bluetooth: Peripheral GATT Discovery Manager
- Bluetooth: Peripheral HIDS keyboard
- Bluetooth: Peripheral HIDS mouse
- Bluetooth: Peripheral Heart Rate Monitor with Coded PHY
- Bluetooth: Peripheral LBS
- Bluetooth: Peripheral Memfault Diagnostic Service (MDS)
- Bluetooth: NFC pairing
- Bluetooth: Peripheral power profiling
- Bluetooth: Peripheral Running Speed and Cadence Service (RSCS)
- Bluetooth: Peripheral Status
- Bluetooth: Peripheral UART
- Bluetooth: External radio coexistence using 1-wire interface
- Bluetooth: Host for nRF RPC Bluetooth Low Energy
- Bluetooth: NUS shell transport
- Bluetooth: Throughput
- Bluetooth: Mesh and peripheral coexistence
- Bluetooth: Mesh chat
- Bluetooth: Mesh light
- Bluetooth: Mesh light fixture
- Bluetooth: Mesh light dimmer and scene selector
- Bluetooth: Mesh light switch
- Bluetooth: Mesh sensor observer
- Bluetooth: Mesh sensor
- Bluetooth: Mesh Silvair EnOcean
- Bluetooth: Mesh Device Firmware Update (DFU) distributor
- Bluetooth: Mesh Device Firmware Update (DFU) target

UWB on Zephyr: DW1000 driver

- Zephyr has 802.15.4 API and driver for DW1000
- at first it seemed useless
  - used just to send data, no control over TX/RX timestamps
- turns out: you can, but the API is ~~weird~~ flexible
  - net_pkt API has many optional functions behind a Kconfig option (`CONFIG_NET_PKT_TIMESTAMP`)
- takeaway: complex APIs without examples are useless :(

Porting the driver

```
int writetospi(uint16 headerLength,
               const uint8 *headerBuffer,
               uint32 bodyLength,
               const uint8 *bodyBuffer) {
    decaIrqStatus_t stat = decamutexon();

    while (HAL_SPI_GetState(&hspi1) != HAL_SPI_STATE_READY);

    HAL_GPIO_WritePin(DW_NSS_GPIO_Port, DW_NSS_Pin, GPIO_PIN_RESET);
    HAL_SPI_Transmit(&hspi1, (uint8_t *)&headerBuffer[0],
            headerLength, HAL_MAX_DELAY);
    HAL_SPI_Transmit(&hspi1, (uint8_t *)&bodyBuffer[0], bodyLength,
            HAL_MAX_DELAY);
    HAL_GPIO_WritePin(DW_NSS_GPIO_Port, DW_NSS_Pin, GPIO_PIN_SET);

    decamutexoff(stat);

    return 0;
}
```

```
int writetospi(uint16 headerLength,
               const uint8 *headerBuffer,
               uint32 bodyLength,
               const uint8 *bodyBuffer) {
    decaIrqStatus_t irq_stat = decamutexon();

    int res = spi_write(
        spi_device,
        spi_selected_config,
        &(struct spi_buf_set) {
            .buffers = (struct spi_buf[]) {
                {
                    .buf = (uint8_t *)headerBuffer,
                    .len = headerLength
                },
                {
                    .buf = (uint8_t *)bodyBuffer,
                    .len = bodyLength
                }
            },
            .count = 2
        }
    );

    decamutexoff(irq_stat);

    return res;
}
```

Beware

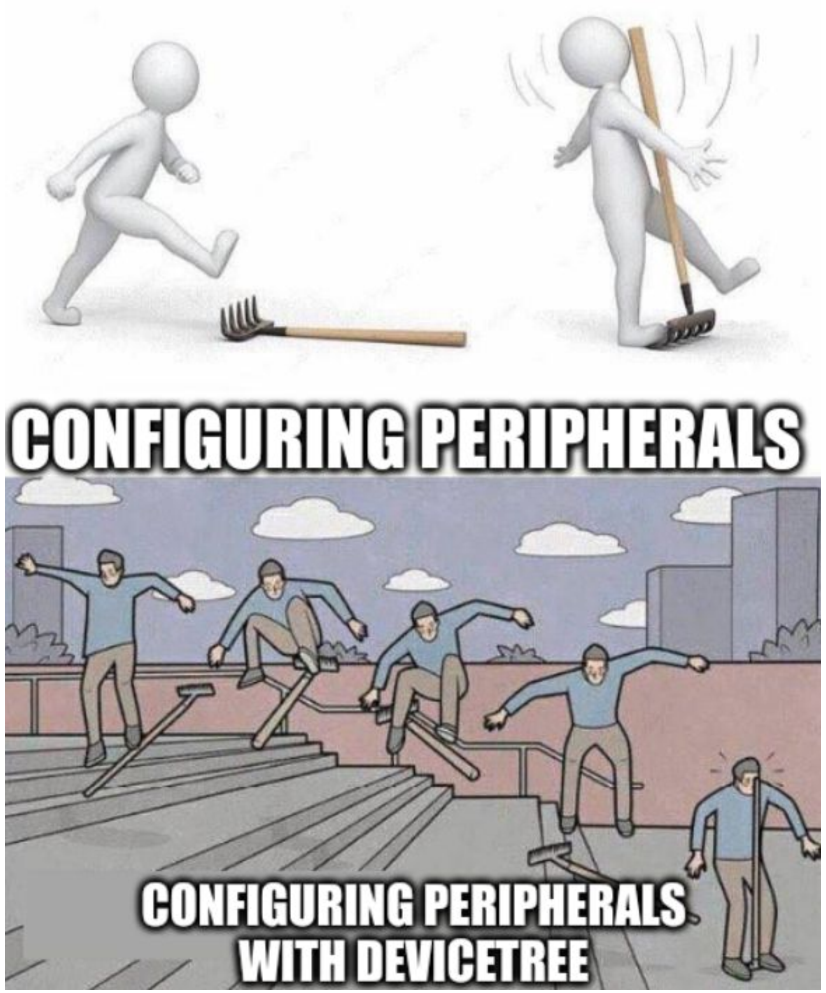
hyperrtls-gateway : west — Konsole

File Edit View Bookmarks Plugins Settings Help

```
Symbol information
Name: SPI_STM32_USE_HW_SS
Prompt: STM32 Hardware Slave Select support
Type: bool
Value: n

Help:

  Use Slave Select pin instead of software Slave Select.

Direct dependencies (=n):
     SPI_STM32(=n)
  && SPI(=n)

Default:
  - y

Kconfig definition, with parent deps. propagated to 'depends on'
================================================================

At drivers/spi/Kconfig.stm32:27
Included via /home/anuar2k/zephyrproject/hyperrtls-gateway/Kconfig:32 -> Kconfig.zephyr:42 -> drivers/Kconfig:40 ->
Menu path: (Top) -> Device Drivers -> SPI hardware bus support -> STM32 MCU SPI controller driver

  config SPI_STM32_USE_HW_SS
        bool "STM32 Hardware Slave Select support"
        default y
        depends on SPI_STM32(=n) && SPI(=n)
        help
          Use Slave Select pin instead of software Slave Select.

[ESC/q] Return to menu      [/] Jump to symbol
```

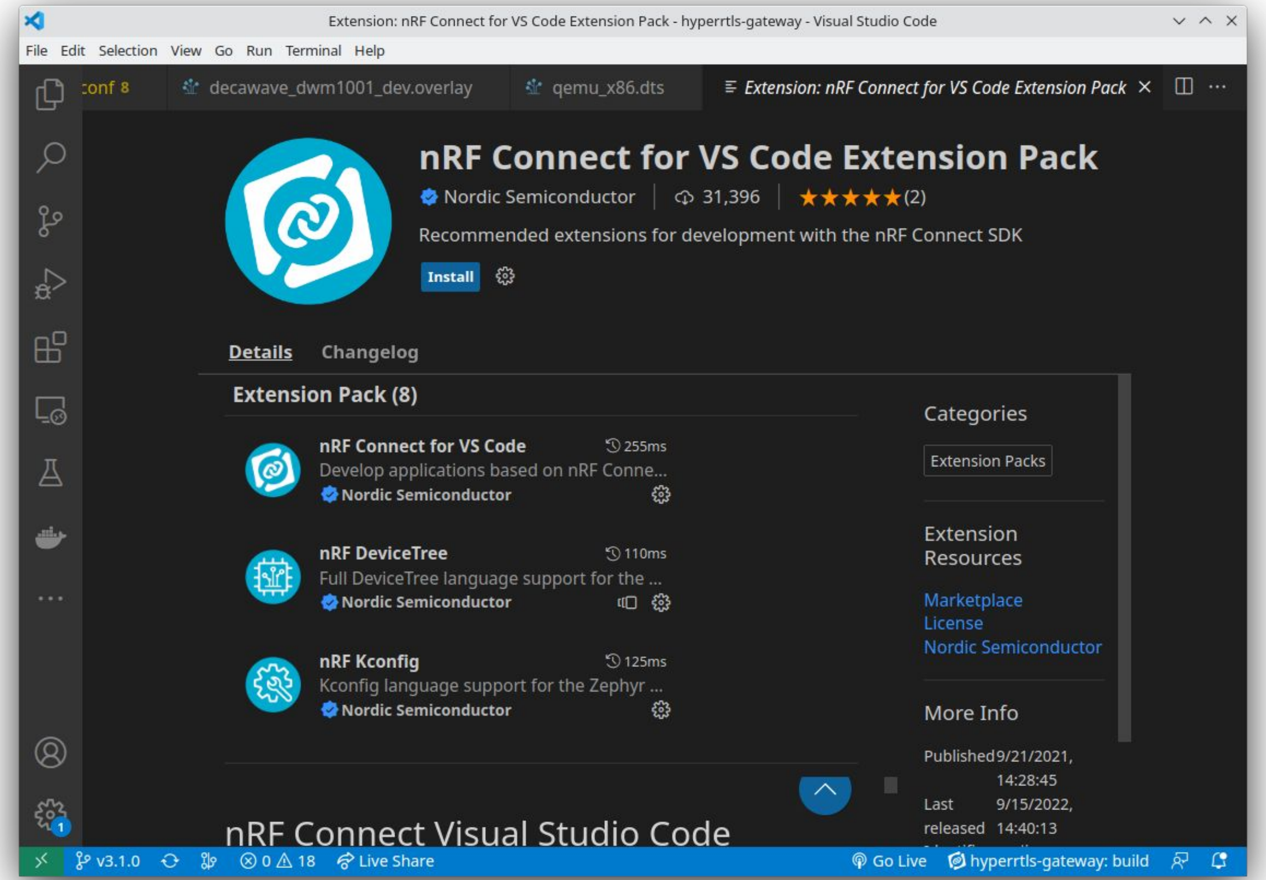
Devicetree: the double-edged sword

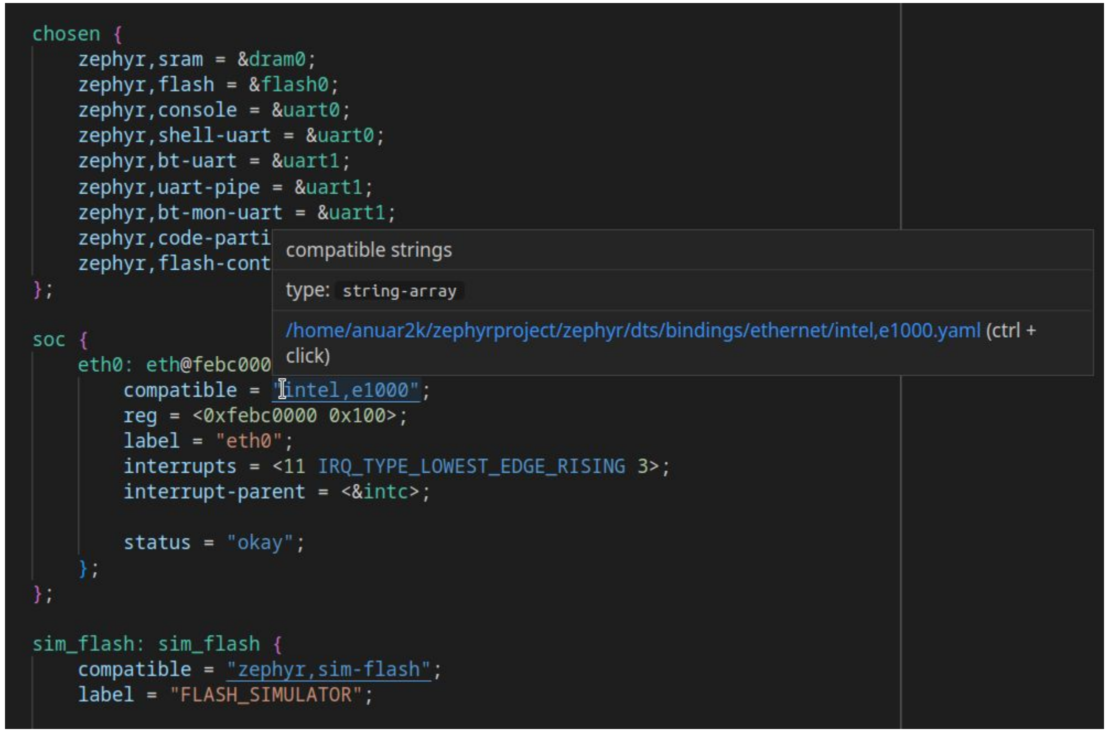
```
In file included from /home/anuar2k/zephyrproject/zephyr/include/zephyr/arch/arm/aarch32/arch.h:20,
                 from /home/anuar2k/zephyrproject/zephyr/include/zephyr/arch/cpu.h:19,
                 from /home/anuar2k/zephyrproject/zephyr/include/zephyr/kernel_includes.h:33:
/home/anuar2k/zephyrproject/zephyr/include/zephyr/devicetree.h:81:17: error: 'DT_N_S_zephyr_user_P_s
nal_gpios_IDX_0_VAL_pin' undeclared here (not in a function); did you mean 'DT_N_S_leds_S_led_3_P_gp
s_IDX_0_VAL_pin'?
   81 | #define DT_ROOT DT_N
      |                 ^~~~
/home/anuar2k/zephyrproject/zephyr/include/zephyr/devicetree.h:3043:24: note: in definition of macro
DT_CAT'
 3043 | #define DT_CAT(a1, a2) a1 ## a2
      |                        ^~
/home/anuar2k/zephyrproject/zephyr/include/zephyr/devicetree.h:1049:9: note: in expansion of macro '
_PROP'
 1049 |         DT_PROP(node_id, pha##_IDX_##idx##_VAL_##cell)
      |         ^~~~~~~
/home/anuar2k/zephyrproject/zephyr/include/zephyr/devicetree/gpio.h:158:9: note: in expansion of mac
 'DT_PHA_BY_IDX'
  158 |         DT_PHA_BY_IDX(node_id, gpio_pha, idx, pin)
      |         ^~~~~~~~~~~~~
/home/anuar2k/zephyrproject/zephyr/include/zephyr/drivers/gpio.h:339:24: note: in expansion of macro
DT_GPIO_PIN_BY_IDX'
  339 |                 .pin = DT_GPIO_PIN_BY_IDX(node_id, prop, idx),                  \
      |                        ^~~~~~~~~~~~~~~~~~
/home/anuar2k/zephyrproject/zephyr/include/zephyr/drivers/gpio.h:374:9: note: in expansion of macro
PIO_DT_SPEC_GET_BY_IDX'
  374 |         GPIO_DT_SPEC_GET_BY_IDX(node_id, prop, 0)
      |         ^~~~~~~~~~~~~~~~~~~~~~~
/home/anuar2k/zephyrproject/hyperrtls-gateway/src/dw1000/platform/deca_platform.c:13:46: note: in ex
nsion of macro 'GPIO_DT_SPEC_GET'
   13 | static const struct gpio_dt_spec reset_pin = GPIO_DT_SPEC_GET(DT_PATH(zephyr_user), signal_g
os);
      |                                              ^~~~~~~~~~~~~~~~
/home/anuar2k/zephyrproject/zephyr/include/zephyr/sys/util_internal.h:98:26: note: in expansion of m
ro 'UTIL_PRIMITIVE_CAT'
   98 | #define UTIL_CAT(a, ...) UTIL_PRIMITIVE_CAT(a, __VA_ARGS__)
```



Zephyr® Project Developer Summit

Making devicetree less painful





Zephyr® Project
Developer Summit

Making devicetree less painful

https://youtu.be/w8GgP3h0M8M

Quick maths

- location is calculated from distances
  - they’re measured by tags
- two options:
  - send measurements to central server, calculate location, optionally send it back to tag
  - calculate location on tag, optionally send it to the server
- we went with doing the calculation on the tags

Linear algebra on Zephyr

- implementing all functions from ground-up is NOT an option
  - (pseudo)-inverse is the hardest to implement
- many options on full-size computers
  - Python: NumPy?
  - C: GSL (GNU Scientific Library)
    - good luck with trying to compile that for embedded…
- Zephyr has Zephyr Scientific Library!
  - for some reason not ZSL but *zscilib*

Deja vu

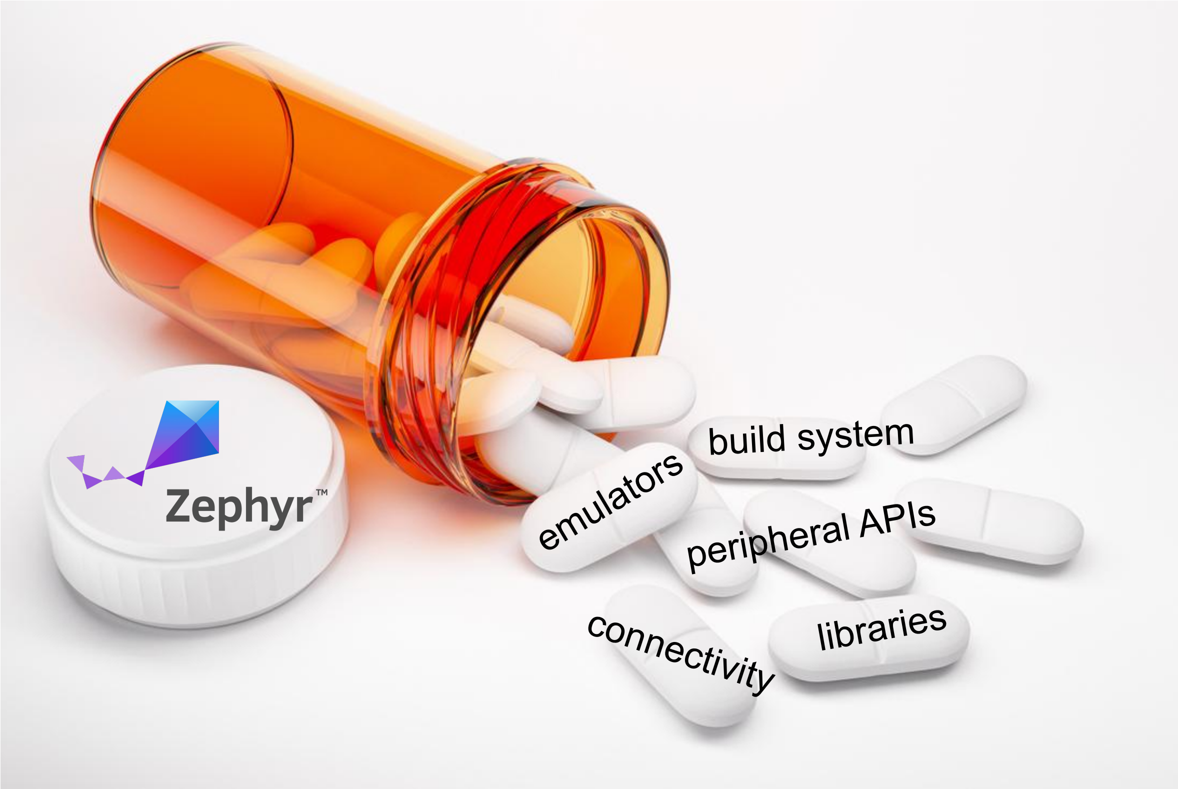
```
ZSL_MATRIX_DEF(A, 4, 4);
for (size_t row = 0; row < 4; row++) {
    zsl_mtx_set(&A, row, 0, 1);
    zsl_mtx_set(&A, row, 1, -2 * measurements[row].anchor_pos.x);
    zsl_mtx_set(&A, row, 2, -2 * measurements[row].anchor_pos.y);
    zsl_mtx_set(&A, row, 3, -2 * measurements[row].anchor_pos.z);
}

zsl_real_t A_det;
zsl_mtx_deter(&A, &A_det);
if (A_det == 0) {
    // det == 0 <=> anchors are coplanar
    return -EINVAL;
}
```

zscilib

- easy to add
  - entry in .yml manifest
  - `west update`
  - `CONFIG_ZSL=y`
- abuses VLAs
  - some functions are stack heavy
    - `CONFIG_MAIN_STACK_SIZE=32768` to the rescue
- multithreading? use `CONFIG_FPU_SHARING=y`

Wrap up

- Zephyr is not the silver bullet
  - but certainly a good painkiller!
- is the project successful?
  - we graduated, so I guess yes!



Zephyr® Project
Developer Summit

Thank you!

questions?