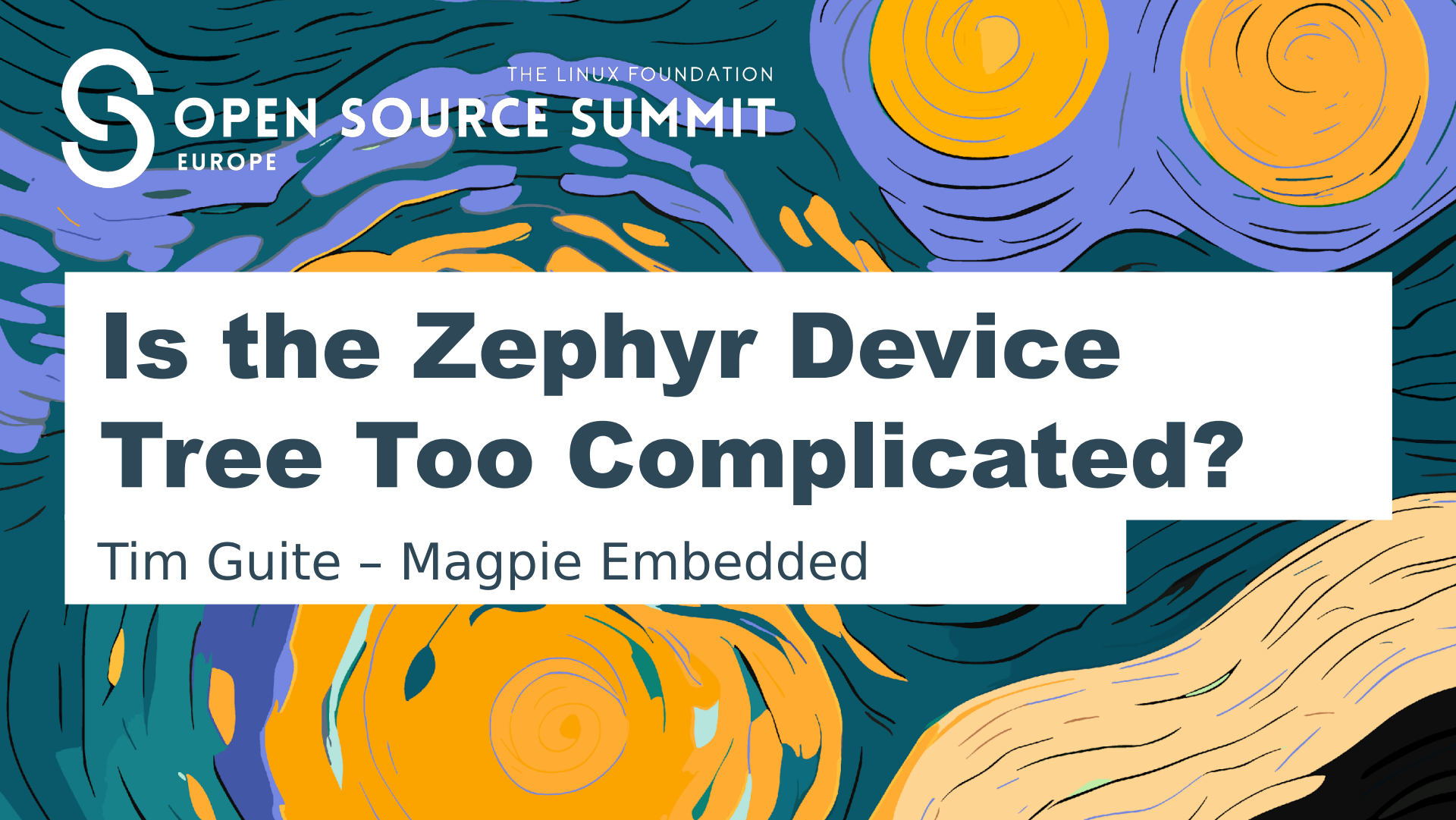


Is the Zephyr Device Tree Too Complicated?

Tim Guite – Magpie Embedded

About the Speaker

Tim Guite is an experienced embedded systems engineer who has worked across medical devices, sub-sea robotics and particle accelerators at multiple layers of the stack. He is happy when his desk is covered in development boards, jumper wires and various sensors. He is passionate about open source and sharing knowledge in the hopes that will lead to better, more robust systems and have a positive impact on the world.

Betteridge’s Law of Headlines:

Any headline that ends in a question mark can be answered by the word

NO

Maybe

(for a lot of users)

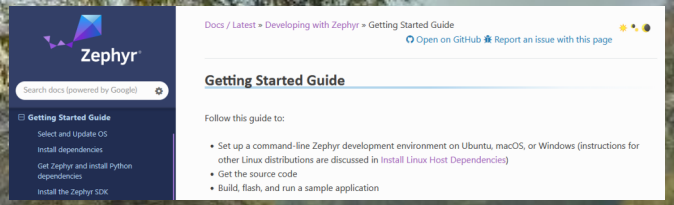
(Useful tools have arrived!)

50% Background, 50% Proposals

50% Background, 50% Proposals

50% Background, 40% Fixes, 10% Proposals

Getting Started



Next Steps

Copyright: Jason Priem (link)

Next Steps

KConfig

DeviceTree

alias/label/node

Where is the actual driver code?

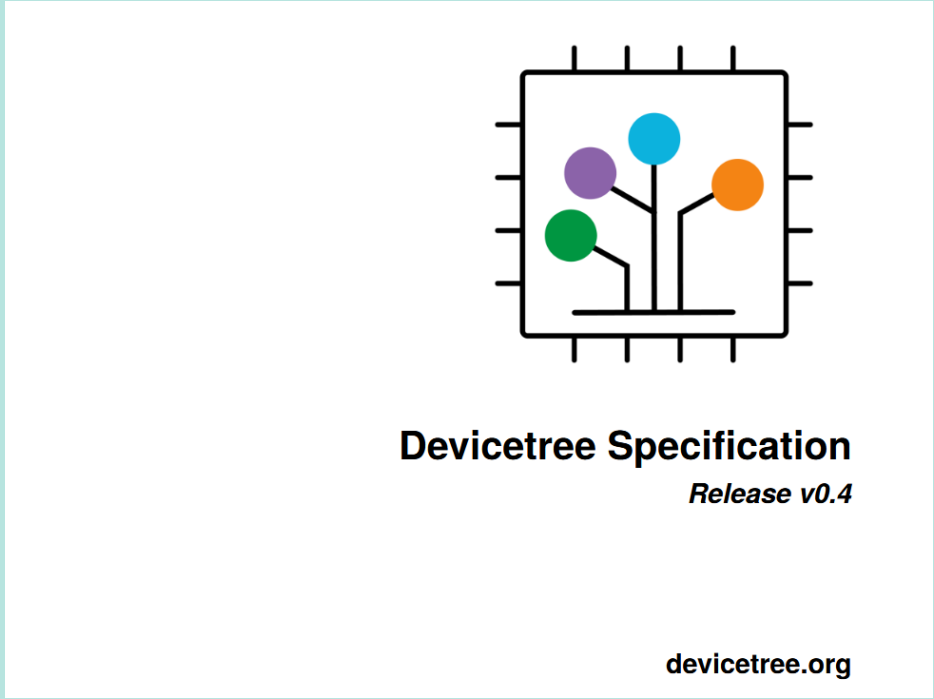
Where does the I2C address go?

Copyright: Jason Priem (link)

This will make a lot of sense if you have kernel development experience

What is it for?

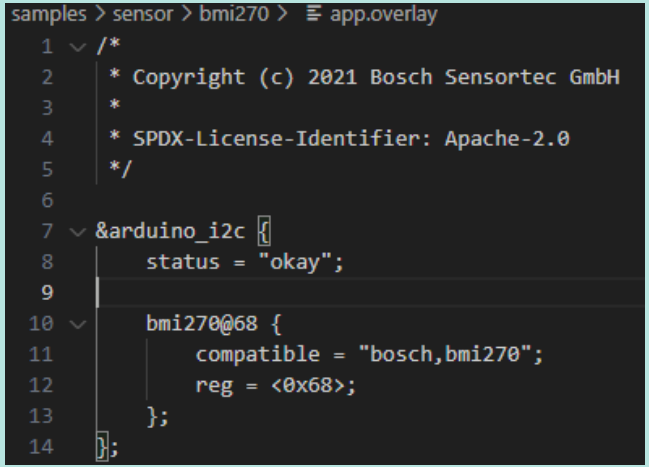
- “A devicetree is primarily a hierarchical data structure that describes hardware” – Zephyr Device Tree Docs
- Framework for specifying hardware for embedded systems
- Flexible and powerful for maintainers and power users
- Confusing for developers who just want to set up chains of I2C and SPI devices which are already supported in Zephyr – the majority?



https://www.devicetree.org/specifications

Sensor Samples

- Redundant information
- Device specific
- Convention driven?
- Hard to follow through to driver code
- Not well abstracted like **blinky**
- Less clear than custom HAL class in C or C++ (the competition)



Is there an easier way?

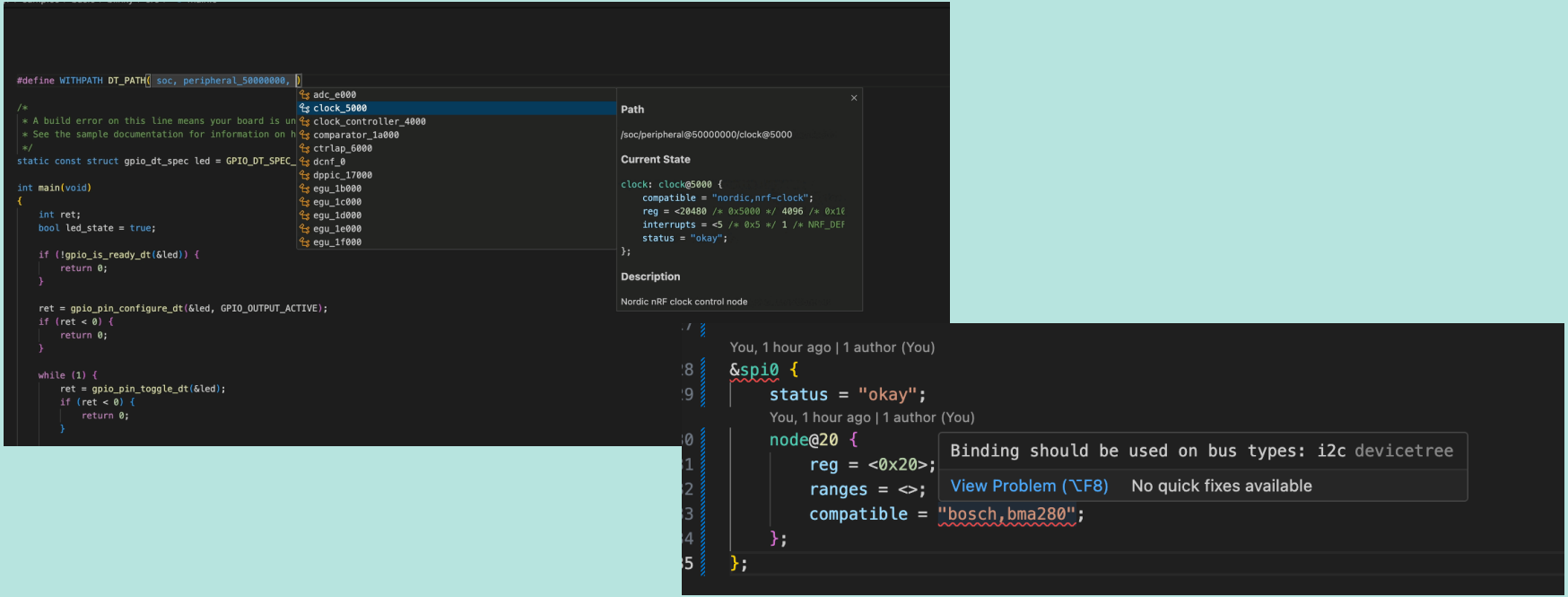
- Simple definitions for I2C and SPI devices
- Combine with info from bindings
- Faster feedback on DT errors
- Easier to build portable systems
- Compile to .dts files

- Reduces flexibility
- Another tool may add complexity
- Unforeseen interactions between DT and KConfig

Flattening the Learning Curve

- Additional info given in generated **zephyr.dts** as of 4.2
- Amazing DTS LSP work!
- DTS formatting PRs open now: #92805
- Graphical viewer from Nordic
- DTSH tool for interacting with Device Tree

DTS LSP


#define WITHPATH DT_PATH(soc, peripheral_50000000,)
adc_e000
clock_5000
clock_controller_4000
comparator_1a000
ctrlap_6000
dcnf_0
dppic_17000
egu_1b000
egu_1c000
egu_1d000
egu_1e000
egu_1f000
Path
/soc/peripheral@50000000/clock@5000
Current State
clock: clock@5000 {
compatible = "nordic,nrf-clock";
reg = <20480 /* 0x5000 */ 4096 /* 0x1
interrupts = <5 /* 0x5 */ 1 /* NRF_DEF
status = "okay";
};
Description
Nordic nRF clock control node
int main(void)
{
int ret;
bool led_state = true;
if (!gpio_is_ready_dt(&led)) {
return 0;
}
ret = gpio_pin_configure_dt(&led, GPIO_OUTPUT_ACTIVE);
if (ret < 0) {
return 0;
}
while (1) {
ret = gpio_pin_toggle_dt(&led);
if (ret < 0) {
return 0;
}
You, 1 hour ago | 1 author (You)
&spi0 {
status = "okay";
You, 1 hour ago | 1 author (You)
node@20 {
reg = <0x20>;
ranges = <>;
compatible = "bosch,bma280";
};
};
Binding should be used on bus types: i2c devicetree
View Problem (⌥F8)
No quick fixes available

Could It Be Easier?

- Abstract device trees for portability
- Connection between source code and device tree
- zephyr\scripts\dts\python-devicetree\src\devicetree\edtlib.py
- zephyr\scripts\dts\python-devicetree\src\devicetree\dtlib.py
- Have classes for working with device trees and bindings

Resources

- Practical Zephyr by Martin Lampacher
- Using the Device Tree by Cirbuit Dojo
- Zephyr Tutorial by Javad Rahimipetroudi at Mind
- Device Tree Tutorial by Shawn Hymel
- DTS LSP by Kyle Bonnici at Nordic Semiconductor
- Nrf DeviceTree Editor by Nordic Semiconductor
- dtsh by Chris Duf and Luca Burelli
- Abstract Hardware Interfaces by Chris Wilson at Golioth


Thank You!
magpieembedded.co.uk/content/2025_08_27_is_zephyr_device_tree_too_complicated.html
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE