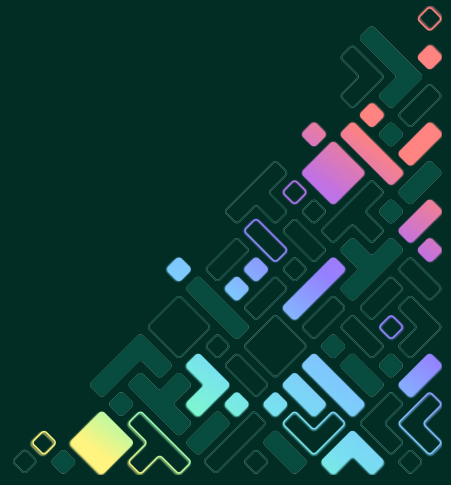


# Shell over Bluetooth LE

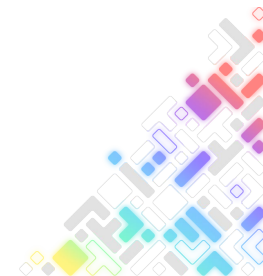
## Open Source Contribution

**Luis Ubieda**  
Lead FW Engineer



# Agenda

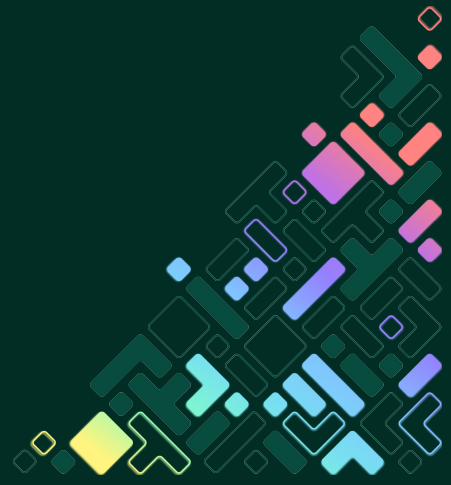
- Intro
- Bluetooth LE in a Nutshell
- UART over Bluetooth LE (NUS)
- The Zephyr Shell
- Using Shell over Bluetooth LE
- Live Demo
- Q&A



# Intro



EMBEDDED  
OPEN SOURCE  
SUMMIT



# About Me

Luis Ubieda

- Electrical Engineer
- Lead Firmware Engineer @ [Croxel](#)
- 7 Years developing Firmware
- Based in Melbourne, FL (Space Coast)
- Passion for Technology, Electronics and IoT
- First Zephyr Contribution on 2021
- Free-Time: Fitness and Sports
- Blogger: <https://embeddedtales.blog>
- First Time Speaker at EOSS

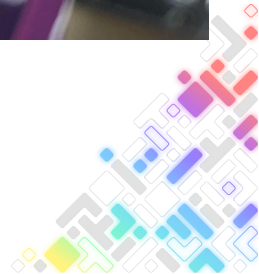
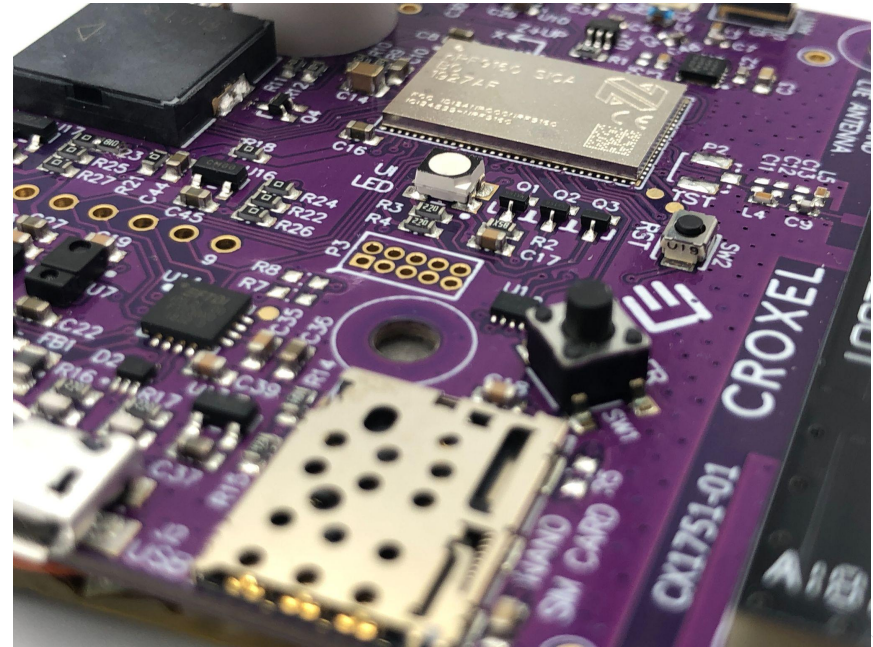


*Rocket Launch every other week at the Space Coast.  
Image Source: [visitspacecoast.com](https://visitspacecoast.com)*



# About Croxel

- Deep Zephyr RTOS expertise for tailored, technically sound solutions.
- End-to-end product development capabilities aligned with your goals.
- Rapid prototyping and low-power optimization for faster time-to-market.
- Seamless integration with your teams across all areas of development.
- Active contributions to the Zephyr Project and continuous community involvement.



# About the Contribution

- PR: [69881](#)
- Status: **Merged.**
- Fixes Issue: [12714](#) (open since 2019).

## serial: Add UART over Bluetooth LE #69881

 Merged carlescufi merged 8 commits into [zephyrproject-rtos:main](#) from [ubieda:feature/serial/bt-nus](#)

 Conversation 51  Commits 8  Checks 44  Files changed 32



ubieda commented on Mar 6 • edited ▾

Collaborator ⋮

### Description

This PR introduces a UART-compatible implementation of Bluetooth LE GATT, using NUS (Nordic UART Service), which allows users to utilize Bluetooth LE as serial endpoints with UART APIs and take advantage of existing support of subsystems that already support UART.

### Background

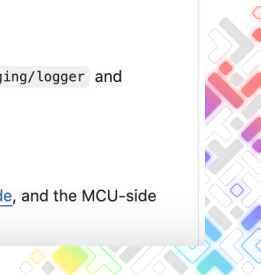
This PR fixes [#12714](#). Also, this PR is a follow-up of [#68076](#), with the agreed [changes captured on the Bluetooth WG](#).

### Changes

- Add NUS as a Bluetooth LE GATT Service.
- Add Sample demonstrating API usage of NUS.
- Add UART-compatible driver for Bluetooth LE, using NUS service.
- Add snippet for basic usage of UART over Bluetooth LE.
- Add sample support of UART over Bluetooth LE to `samples/subsys/logging/logger` and `samples/subsys/shell/shell_module`.

### Testing

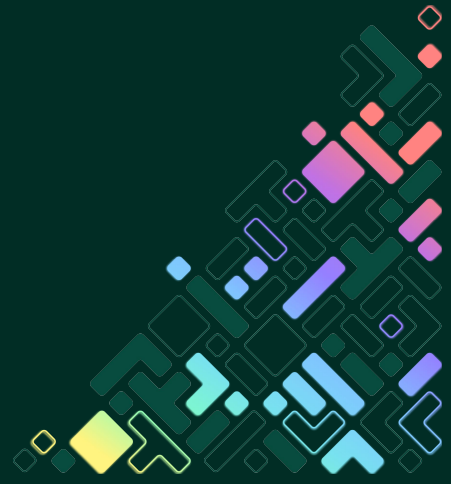
UART over Bluetooth LE has been tested using [this python-app on the PC-side](#), and the MCU-side running each of the following samples:



# Bluetooth LE in a Nutshell



EMBEDDED  
OPEN SOURCE  
SUMMIT



# Bluetooth LE in a Nutshell - GAP Profile

Generic Access Profile (GAP)

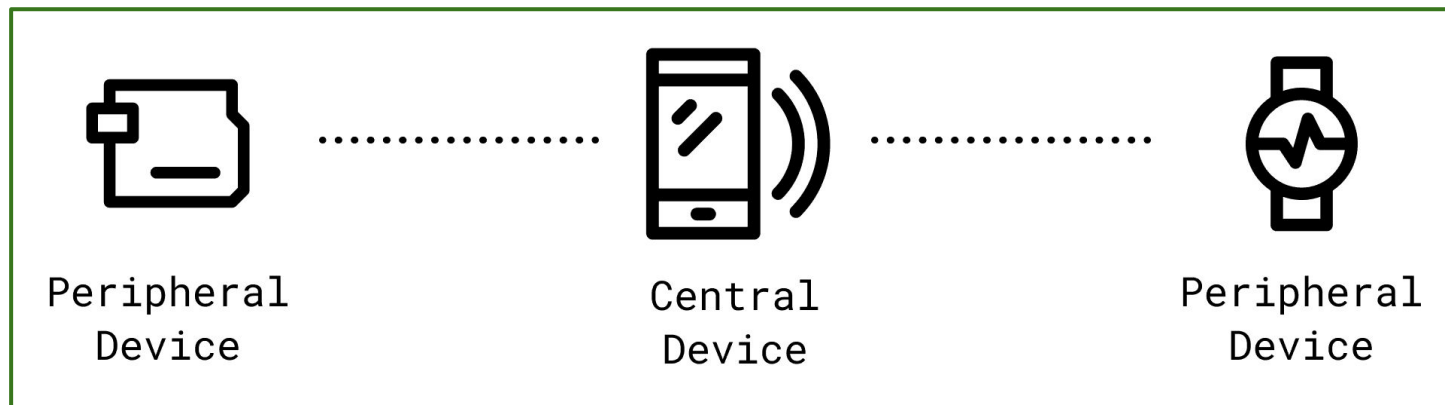
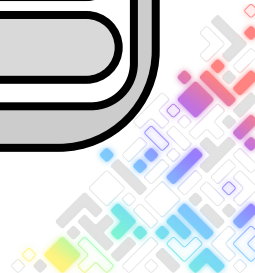
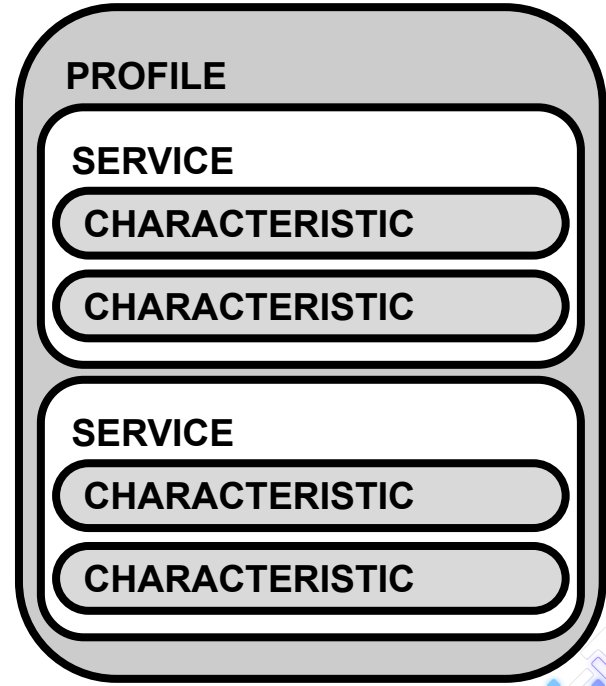
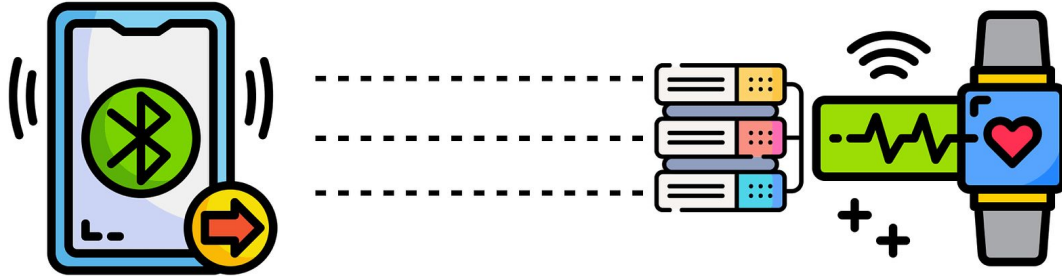


Image Source: <https://docs.arduino.cc/learn/communication/bluetooth/>





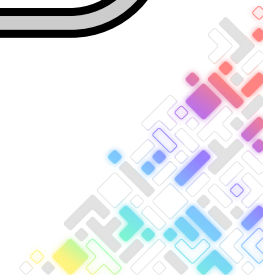
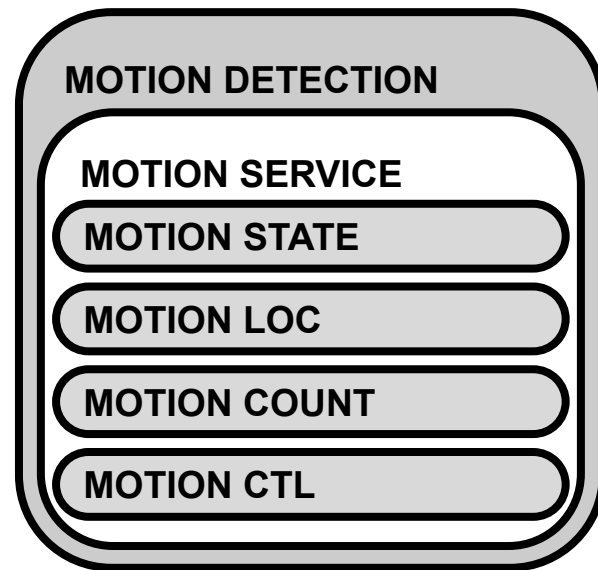
# Bluetooth LE in a Nutshell - GATT Profile



# Bluetooth LE in a Nutshell - Custom GATT

## Option 1: Bluetooth-centric GATT

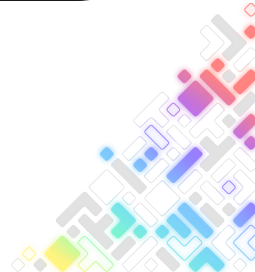
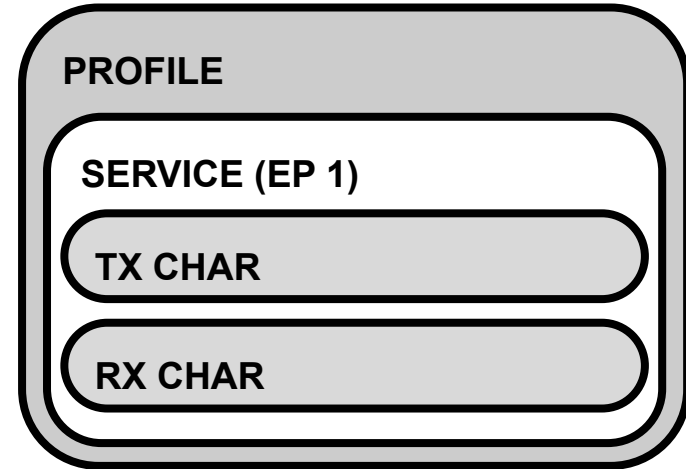
- One Characteristic per functionality (e.g: 1x for Button state, 1x for LED control, etc).
- Similar to Standard Bluetooth LE Services.



# Bluetooth LE in a Nutshell - Custom GATT

## Option 2: GATT as serial endpoints

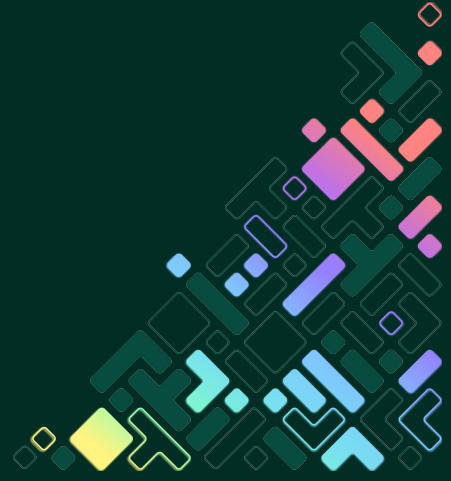
- Using GATT Services as a set of serial endpoints.
- Functionalities are controlled through set of endpoints, with a schema to encode/decode packets.
- Useful when many backends are used, Bluetooth LE being one.



# UART over Bluetooth LE (NUS)



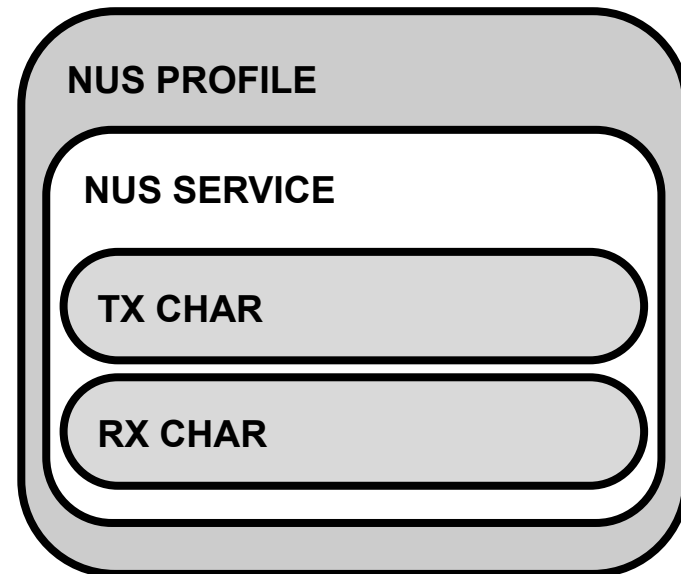
EMBEDDED  
OPEN SOURCE  
SUMMIT



# UART over Bluetooth LE (NUS) - GATT Service

## Nordic UART Service (Custom Service)

- Characteristics
  - TX: device sends data to the user.
  - RX: device receives data from the user.
- Similar to Serial Port Profile (SPP) in Bluetooth classic.
- Widely used in Bluetooth LE applications.

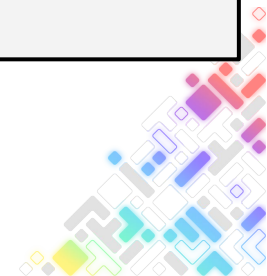


# UART over Bluetooth LE (NUS) - UART driver

- UART driver connects to NUS service (nus-uart).
- UART APIs abstract Bluetooth LE specifics.
- Multiple UARTs can be defined in the device-tree.
- Subsystems can be redirected to these NUS-UART endpoints.

## NUS instance defined in the device-tree

```
/ {  
    chosen {  
        zephyr,console = &bt_nus_console_uart;  
    };  
  
    bt_nus_console_uart: bt_nus_console_uart {  
        compatible = "zephyr,nus-uart";  
    };  
};
```



# Demo - Blinky w/ Console over Bluetooth LE

## 1. Build

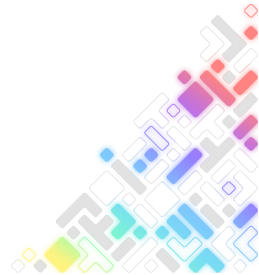
```
west build  
  -b nrf52840dk/nrf52840  
  -S nus-console  
    samples/basic/blinky
```

## 2. Flash

```
west flash --erase
```

## 3. Run User App

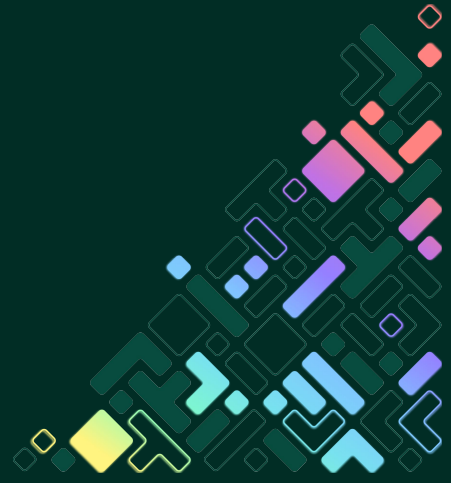
```
python3  
  ../modules/lib/etales-zephyr/scripts/bluetoothcli
```



# The Zephyr Shell



EMBEDDED  
OPEN SOURCE  
SUMMIT





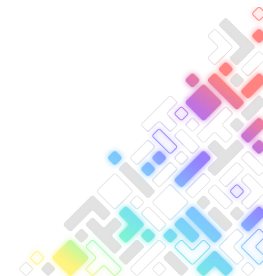
# The Zephyr Shell - Command Definition

## 1. Without Arguments

```
SHELL_CMD_REGISTER(syntax, subcmd, help, handler);
```

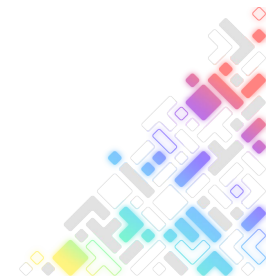
## 2. With Arguments

```
SHELL_CMD_ARG(cmd, subcmds, help, handler, mand-arg, opt-arg);
```



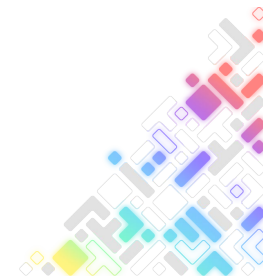
# The Zephyr Shell - Command Handler

```
static int func_handler(const struct shell *sh,  
                        size_t argc,  
                        char **argv)  
{  
    int result;  
  
    /* ... Execute action, gather result ... */  
    result = func_action();  
  
    return result;  
}
```



# The Zephyr Shell - Backends

1. Serial (UART, USB, Bluetooth LE)
2. RTT (J-Link)
3. RPMSG (IPC)
4. MQTT
5. Telnet
6. MCUmgr



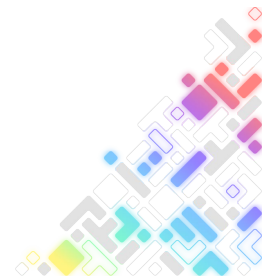
# The Zephyr Shell - Commands in Zephyr

## Drivers

1. I2C
2. ADC
3. GPIO
4. PWM
5. CAN
6. Flash
7. Sensors
8. LoRa

## Subsystems

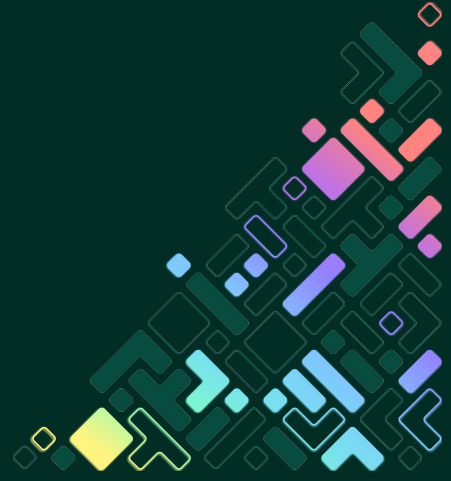
1. Bluetooth
2. Audio
3. Networking
4. MCUBoot
5. Logging
6. Storage/File-Systems
7. LLEXT



# Using Shell over Bluetooth LE



EMBEDDED  
OPEN SOURCE  
SUMMIT



# Steps to enable Shell over BLE - Kconfigs

## Required

# Enable Serial

CONFIG\_SERIAL=y

CONFIG\_UART\_CONSOLE=y

# Enable Shell

CONFIG\_SHELL=y

CONFIG\_SHELL\_BACKEND\_UART=y

# Enable Bluetooth + NUS

CONFIG\_BT=y

CONFIG\_BT\_PERIPHERAL=y

CONFIG\_BT\_ZEPHYR\_NUS=y

CONFIG\_UART\_BT=y

## Optionals

# For apps not using Bluetooth.

CONFIG\_BT\_ZEPHYR\_NUS\_AUTO\_START\_BLUETOOTH=y

# Throughput Optimizations (not a one-size fits all)

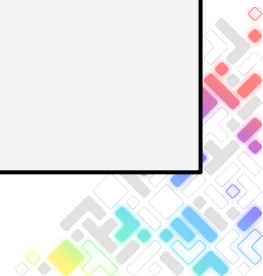
CONFIG\_BT\_RX\_STACK\_SIZE=2048

CONFIG\_BT\_L2CAP\_TX\_MTU=512

CONFIG\_BT\_BUF\_ACL\_RX\_SIZE=502

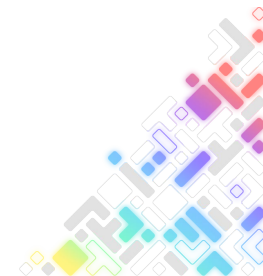
CONFIG\_BT\_BUF\_ACL\_TX\_SIZE=502

CONFIG\_BT\_CTLR\_DATA\_LENGTH\_MAX=251



# Steps to enable Shell over BLE - Device-tree

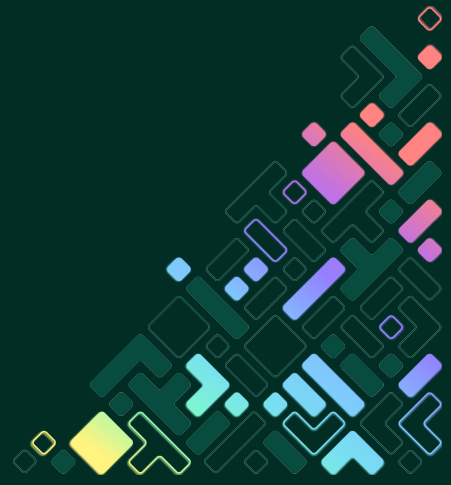
```
/ {  
    chosen {  
        zephyr,shell-uart = &bt_nus_shell_uart;  
    };  
  
    bt_nus_shell_uart: bt_nus_shell_uart {  
        compatible = "zephyr,nus-uart";  
  
        /* Tweak as needed for your app */  
        rx-fifo-size = <1024>;  
        tx-fifo-size = <1024>;  
    };  
};
```



# Live Demo



EMBEDDED  
OPEN SOURCE  
SUMMIT





# Demo - Shell over Bluetooth LE

## BLE NODE Sensor Board

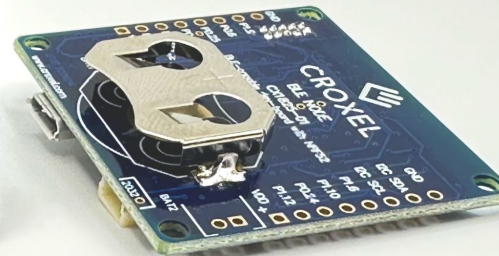
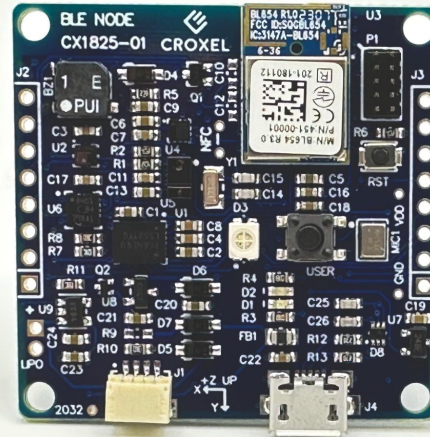
### Module

#### Ezurio (Laird) BL654

- Nordic nRF52840 BLE 5.2
- 1MB Flash + 256KB RAM
- ARM Cortex-M4F @64MHz
- Integrated antenna

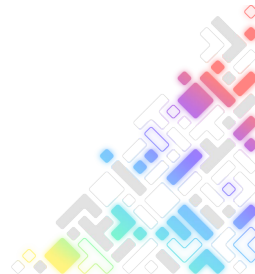
### Onboard Sensors

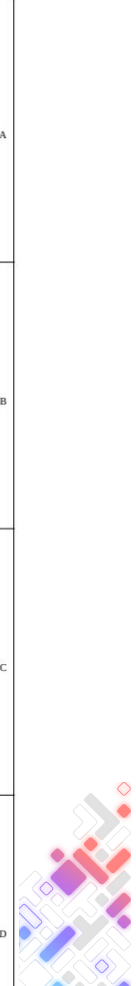
- ST HTS221TR: Temperature and humidity
- ST LPS22HBTR: Absolute pressure sensor
- Avago APDS-9960: Ambient light, RGB, proximity
- ST LIS3DHTR: 3-axis accelerometer
- Melexis MLX90248ESE: Hall effect sensor
- TDK Invensense ICS-41350: MEMS mic



### General Features

- USB-B for power and data
- Battery-powered Li-Po/coin cell
- Battery charger
- 3.3V LDO regulator
- RGB, discrete R&G LEDs, buzzer
- User and reset buttons
- QWIIC connector for I2C sensor expansion
- Expansion headers for GPIO, I2C, and power access
- 2x5 pin SWD headers
- Compact size 40mm x 40mm

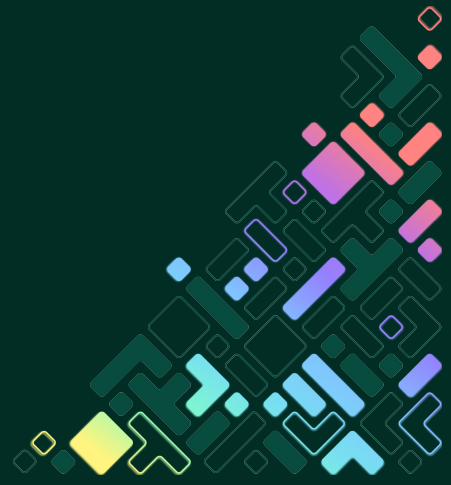




# Q&A



EMBEDDED  
OPEN SOURCE  
SUMMIT





# Zephyr<sup>®</sup> Project

## Developer Summit

