

# MSH-202: Accelerating Bluetooth Mesh Development

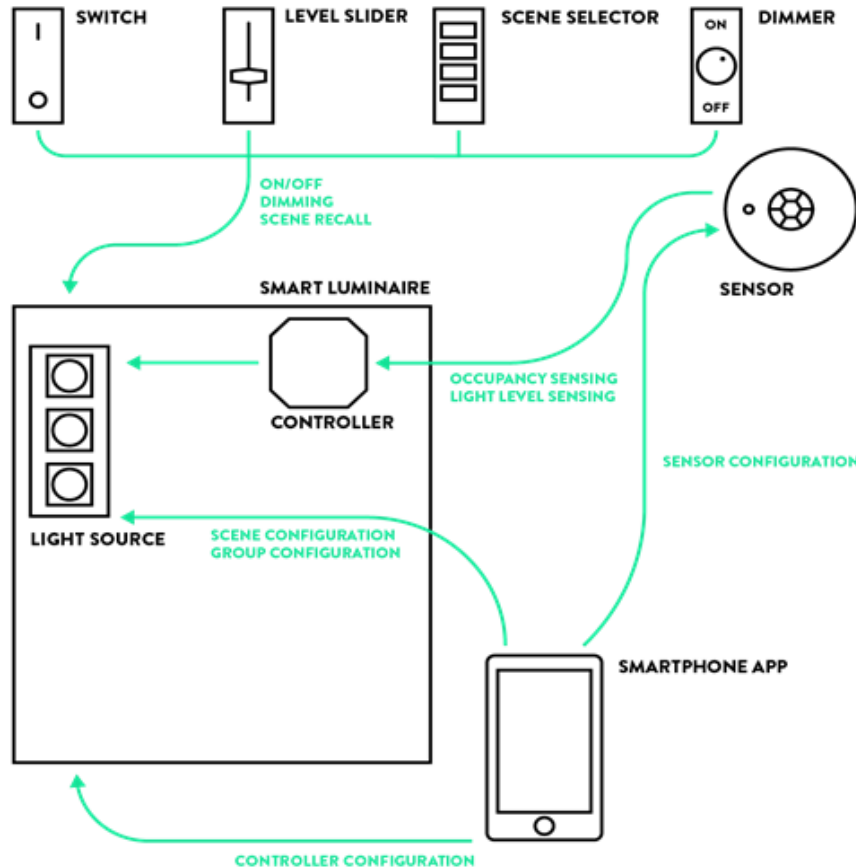
Tiago Monte | 14 September 2021



# Accelerating Bluetooth Mesh Development

- An overview of Silicon Labs' development tools and their unique and differentiating value to accelerate Bluetooth Mesh development
- Outline of what will be covered
  - What does a typical wireless lighting solution consist of
  - Where to get started – introduction to the SDK sample apps that map into each of those wireless lighting solution components (lights, controls, sensors)
  - How to customize to your specific applications needs – Studio's Project Configurator, SW components, Bluetooth Mesh Configurator
  - Additional tools for SW development: pyBGAPI and Mobile ADK
  - Advanced Debugging Tools: Energy Profiler and Network Analyzer
  - Hardware Tools
- Hands-on demo showcasing Energy Profiler and Network Analyzer

# Components of a Wireless Lighting Solution



- **Lighting controls** (mains or battery power)
  - On/Off switches
  - Dimmers
  - Scene selectors
- **Sensors** (mains or battery power)
  - Occupancy
  - Ambient Light Level
- **Light sources and controllers**
  - Controlled by lighting controls and/or sensors
- **Optional smart phone app or gateway**
  - Provisioning and configuration
  - Device control
  - Life cycle management (OTA, removing devices etc.)
  - Value added services

As discussed on MSH-102



---

## Project Configuration and Software Development



# Getting Started with Demos and Sample Apps

- Rich set of sample apps to get you started quickly and effectively
- In addition to embedded sample apps (below) also host sample apps are available

The screenshot shows the 'EXAMPLE PROJECTS & DEMOS' section of the EFR32xG21 2.4 GHz 10 dBm Radio Board (BRD4181B) page. The page lists 8 resources found, categorized by Technology Type, Provider, and Quality. The 'Bluetooth Mesh' category is selected under Technology Type. The 'Gecko SDK Suite v3.2.0' is selected under Provider. The 'ALPHA' quality is selected under Quality. The list of resources includes:

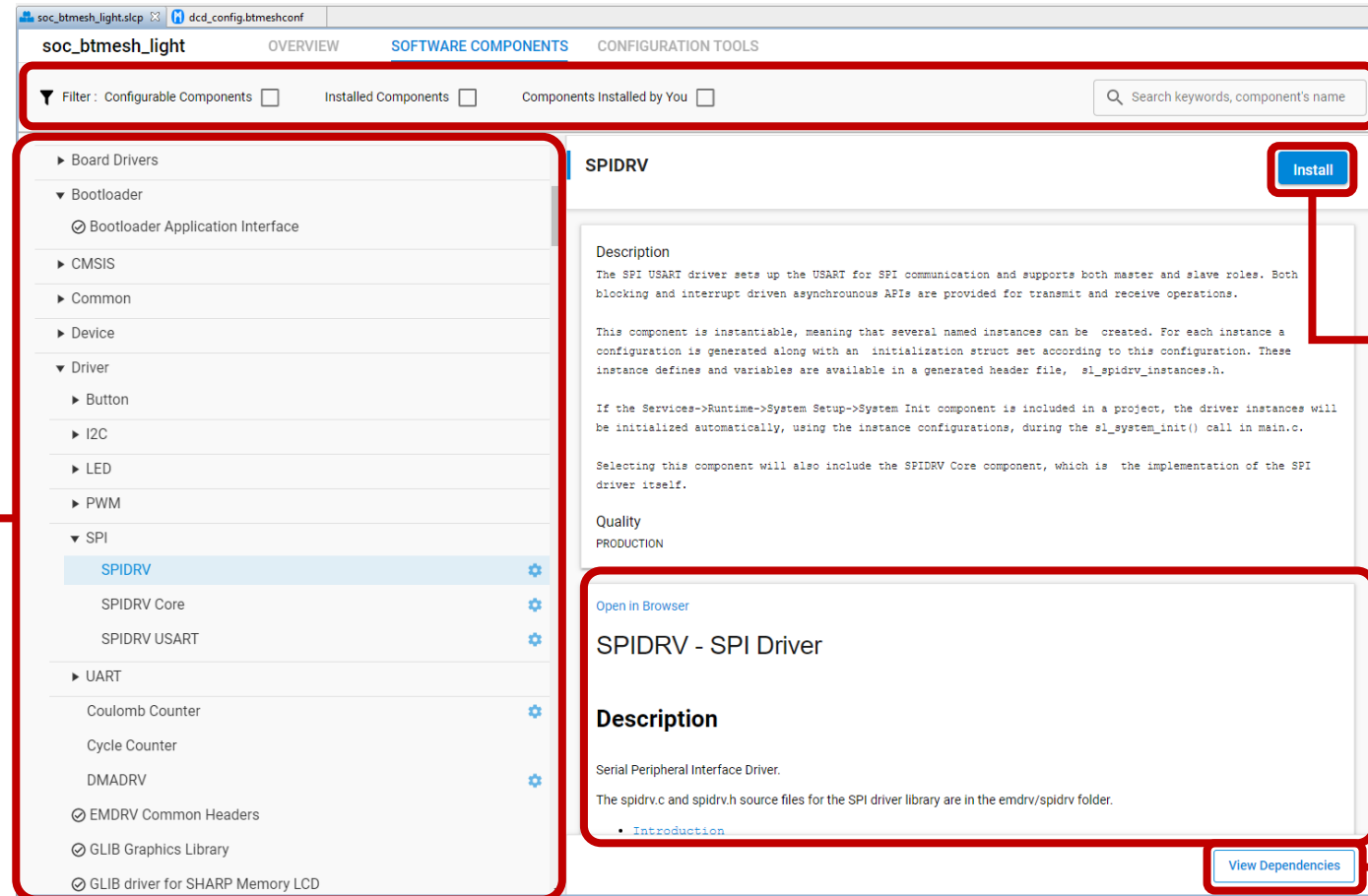
- Bluetooth Mesh - NCP Empty**: Bluetooth Mesh NCP (Network Co-Processor) target demonstrates the bare minimum needed for a Bluetooth Mesh NCP Target C application, that makes it possible for the NCP Host Controller to access the Bluetooth Mesh stack via UART. It provides access to the Host Controller via the link layer via HCI. The communication between the Host Controller and the target can be secured by installing the Secure NCP component. This example requires the BIGAPI UART DFU Bootloader. **Generic**
- Bluetooth Mesh - SoC Empty**: This example demonstrates the bare minimum needed for a Bluetooth Mesh C application that allows Over-the-Air Device Firmware Upgrading (OTA DFW). The application starts Unprovisioned Device Beacons after boot waiting to be provisioned to a Mesh Network. This example can be used as a starting point for an SoC project and it can be customized by adding new components from the Component Browser or by modifying the application (app.c). This example requires one of the Internal Storage Bootloader (single image) variants depending on device memory. **Generic**
- Bluetooth Mesh - SoC HSL Light**: This example is an out-of-the-box Software Demo where the LEDs of the WSTK can be controlled by push button presses on another device (soc\_btmesh\_switch). Beside switching on and off the LEDs, their lighting intensity can also be set. Hue and saturation (only displayed on the LCD) can be set by hsl client. The example also tries to establish friendship as Friend node and prints its status to the LCD. The example is based on the Bluetooth Mesh Generic On/Off Model, the Light Lightness Model, HSL Model and LC Model. This example requires one of the Internal Storage Bootloader (single image) variants depending on device memory. **Lights**
- Bluetooth Mesh - SoC Light**: This example is an out-of-the-box Software Demo where the LEDs of the WSTK can be controlled by push button presses on another device (soc\_btmesh\_switch). Beside switching on and off the LEDs, their lighting intensity can also be set. Hue and saturation (only displayed on the LCD) can be set by hsl client. The example also tries to establish friendship as Friend node and prints its status to the LCD. The example is based on the Bluetooth Mesh Generic On/Off Model, the Light Lightness Model, CTL Model and LC Model. This example requires one of the Internal Storage Bootloader (single image) variants depending on device memory.
- Bluetooth Mesh - SoC Sensor Client**: This example demonstrates the Bluetooth Mesh Sensor Client Model. It collects and displays sensor measurement data from remote device(s) (eg soc\_btmesh\_sensor\_server). The current status is displayed on the LCD and also sent to UART. This example requires one of the Internal Storage Bootloader (single image) variants depending on device memory. **Sensors**
- Bluetooth Mesh - SoC Sensor Server**: This example demonstrates the Bluetooth Mesh Sensor Server Model and Sensor Setup Server Model. It collects and displays sensor measurement data from remote device(s) (eg soc\_btmesh\_sensor\_client). The current status is displayed on the LCD and also sent to UART. This example requires one of the Internal Storage Bootloader (single image) variants depending on device memory.
- Bluetooth Mesh - SoC Switch**: This example is an out-of-the-box Software Demo optimized for user experience where the device acts as a switch. Push Button presses on the WSTK or CLI commands can control the state, lightness and color temperature of the LEDs and also scenes on a remote device (soc\_btmesh\_light). The example also acts as a LPN and tries to establish friendship. The status messages are also displayed on the LCD and sent to UART. The example is based on the Bluetooth Mesh Generic On/Off Client Model, the Light Lightness Client Model, CTL Client Model and Scene Client Model. This example requires one of the Internal Storage Bootloader... **Switches**
- Bluetooth Mesh - SoC Switch Low Power**: This example is an out-of-the-box Software Demo optimized for low current consumption where the device acts as a switch. It has disabled CLI, logging and LCD. Push Button presses on the development board can control the state, lightness and color temperature of the LEDs and also scenes on a remote device (soc\_btmesh\_light). The example also acts as a LPN and tries to establish friendship. The example is based on the Bluetooth Mesh Generic On/Off Client Model, the Light Lightness Client Model, CTL Client Model and Scene Client Model. This example requires one of the Internal Storage Bootloader (single image) variants...

Annotations on the image:

- A red box on the left contains the text: "Run EFR32 as NCP and move application to host". An arrow points from this box to the 'Bluetooth Mesh - NCP Empty' resource.
- A red box on the right contains the text: "Build your own application from scratch". An arrow points from this box to the 'Bluetooth Mesh - SoC Empty' resource.

# Application Customization – Project Configurator

- Need more? Project Configurator allows the inclusion of SW components into your project effortlessly and across all supported parts



User filters and search to find functionality

One click to install component and its dependencies

Built-in documentation

Component dependencies

SW Components are grouped in logical categories

# Application Customization – Bluetooth Mesh Configurator

- Selecting functionality for your node is a breeze with the Bluetooth Mesh Configurator

The screenshot shows the Bluetooth Mesh Configurator interface. On the left, a list of models is categorized into 'Foundation models' and 'Generic models'. A red box highlights the 'Generic models' list, with an arrow pointing to a callout box labeled 'Add standard models'. The main area displays the configuration for 'Main 0. Element'. It includes fields for 'Company' (Silicon Laboratories), 'Company ID' (0x 02ff), 'Product ID' (0x 0005), and 'Version Number' (0x 0210). Below these, there are tables for 'SIG Models' and 'Vendor Models'. The 'Vendor Models' table is highlighted with a red box, and an arrow points from it to a callout box labeled 'Add vendor models'. A green plus button is also highlighted with a red box, with an arrow pointing to a callout box labeled 'Add elements'. At the bottom, a 'Problems (3 errors)' section is highlighted with a red box, with an arrow pointing to a callout box labeled 'Solve issues'.

**Bluetooth Mesh Configurator**

Company: Silicon Laboratories | Company ID: 0x 02ff | Product ID: 0x 0005 | Version Number: 0x 0210

**Foundation models**

- Configuration Client (0x0001)
- Health Server (0x0002)
- Health Client (0x0003)

**Generic models**

- Generic OnOff Server (0x1000)
- Generic OnOff Client (0x1001)
- Generic Level Server (0x1002) +
- Generic Level Client (0x1003)
- Generic Default Transition Time Server (0x1004)
- Generic Default Transition Time Client (0x1005)
- Generic Power OnOff Server (0x1006)
- Generic Power OnOff Setup Server (0x1007)
- Generic Power OnOff Client (0x1008)
- Generic Power Level Server (0x1009)
- Generic Power Level Setup Server (0x100a)
- Generic Power Level Client (0x100b)
- Generic Battery Server (0x100c)

**Main 0. Element** | Location: unknown | Location ID: 0x 0000

SIG Models	Model ID
Configuration Server	0x0000
Health Server	0x0002
Generic Level Client	0x1003
Scheduler Setup Server	0x1207
Sensor Server	0x1100
Sensor Setup Server	0x1101

Vendor Models	Model ID	Company ID
Vendor Model	0x0000	0x02ff
Vendor Model	0x0000	0x02ff

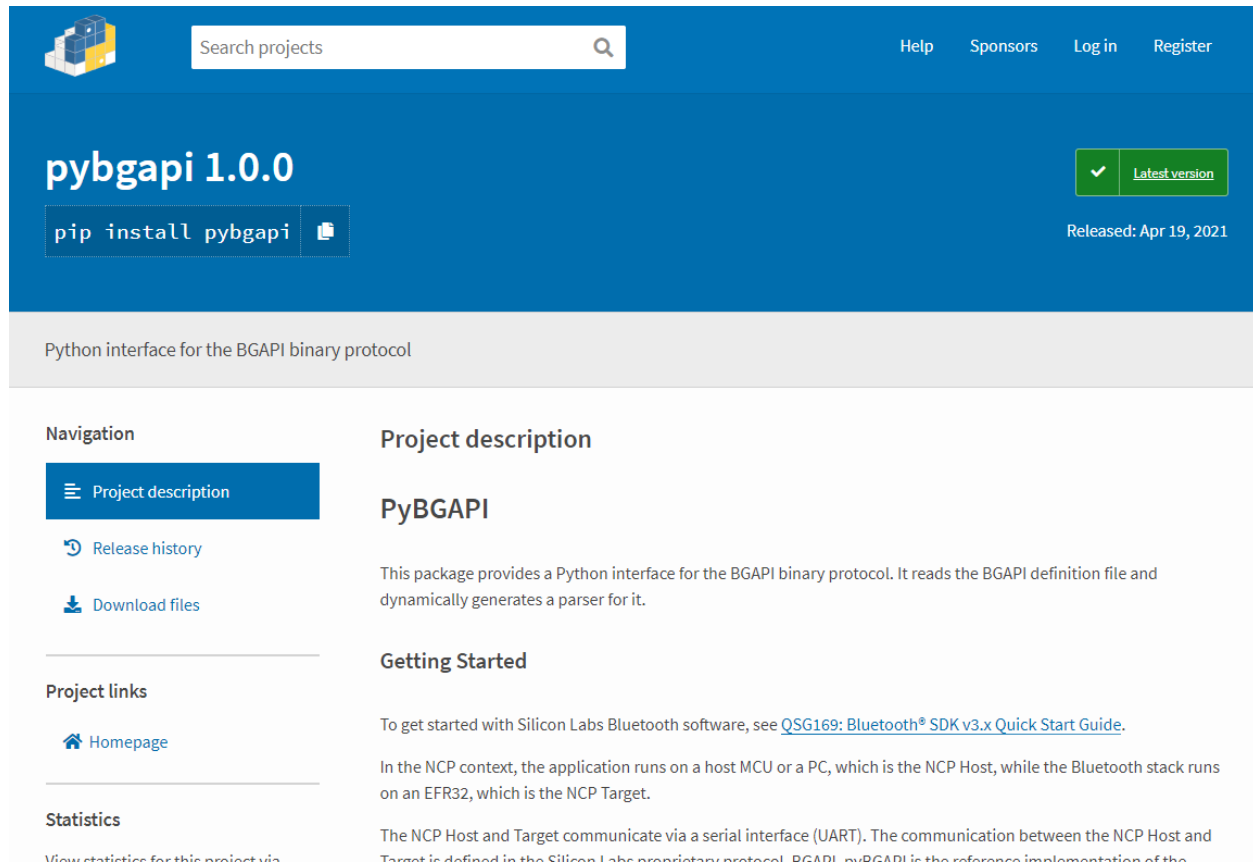
**Problems (3 errors)**

Description

- Errors (3)
- SceneSetupServer (0x1204): required 1 but only have 0!
- SchedulerServer (0x1206): required 1 but only have 0
- 0th element includes multiple models with same Model ID (0x0000) and Company ID (0x02ff)

# PyBGAPI Python Library

- Prototyping and developing host side applications has never been easier: <https://pypi.org/project/pybgapi/>
- Extensively used by customers as well as Silicon Labs' teams for test automation



The screenshot shows the PyPI project page for PyBGAPI 1.0.0. The header is blue with a search bar and links for Help, Sponsors, Log in, and Register. The main section features the package name 'pybgapi 1.0.0' in large white text, a green checkmark icon, and a 'Latest version' button. Below this is a code block showing 'pip install pybgapi' and a 'Released: Apr 19, 2021' date. The description states it is a 'Python interface for the BGAPI binary protocol'. The left sidebar contains navigation links: 'Project description' (selected), 'Release history', and 'Download files'. Below this are 'Project links' (Homepage) and 'Statistics'. The main content area has a 'Project description' section titled 'PyBGAPI' with a paragraph explaining its purpose, followed by a 'Getting Started' section with instructions on how to use the library.

pybgapi 1.0.0

pip install pybgapi

Released: Apr 19, 2021

Python interface for the BGAPI binary protocol

Navigation

- Project description
- Release history
- Download files

Project links

- Homepage

Statistics

View statistics for this project via

Project description

## PyBGAPI

This package provides a Python interface for the BGAPI binary protocol. It reads the BGAPI definition file and dynamically generates a parser for it.

### Getting Started

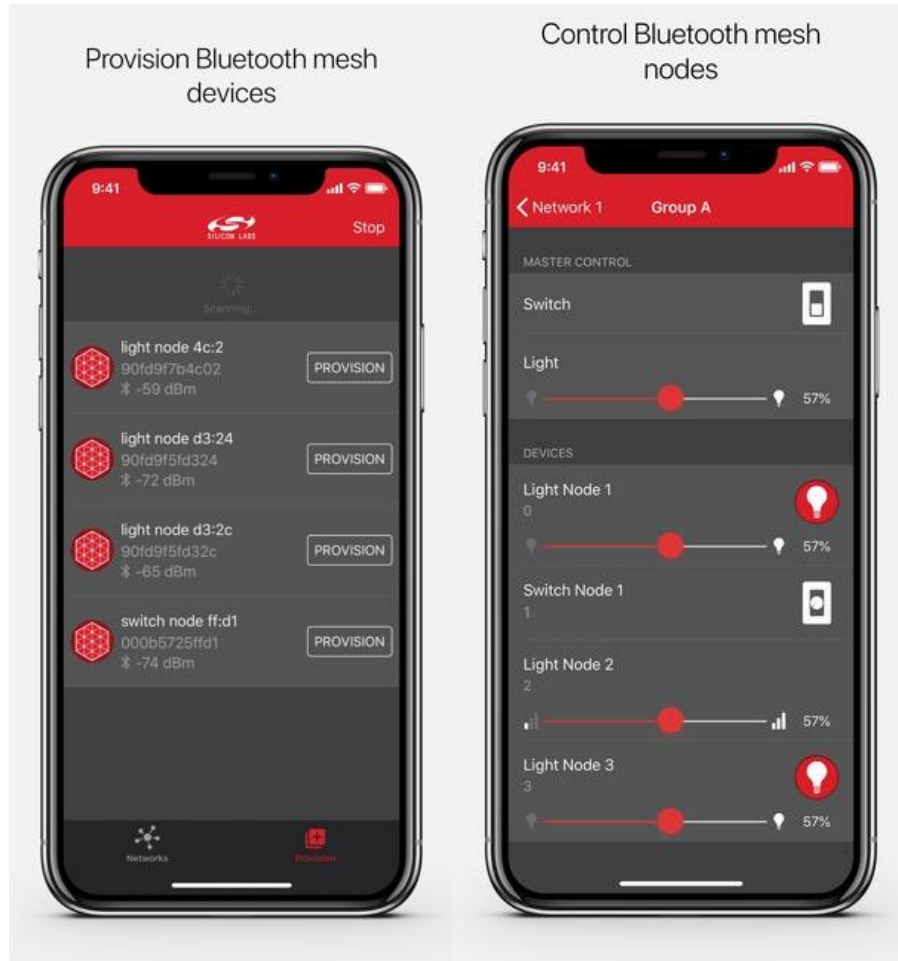
To get started with Silicon Labs Bluetooth software, see [QSG169: Bluetooth® SDK v3.x Quick Start Guide](#).

In the NCP context, the application runs on a host MCU or a PC, which is the NCP Host, while the Bluetooth stack runs on an EFR32, which is the NCP Target.

The NCP Host and Target communicate via a serial interface (UART). The communication between the NCP Host and Target is defined in the Silicon Labs proprietary protocol, BGAPI. PyBGAPI is the reference implementation of the



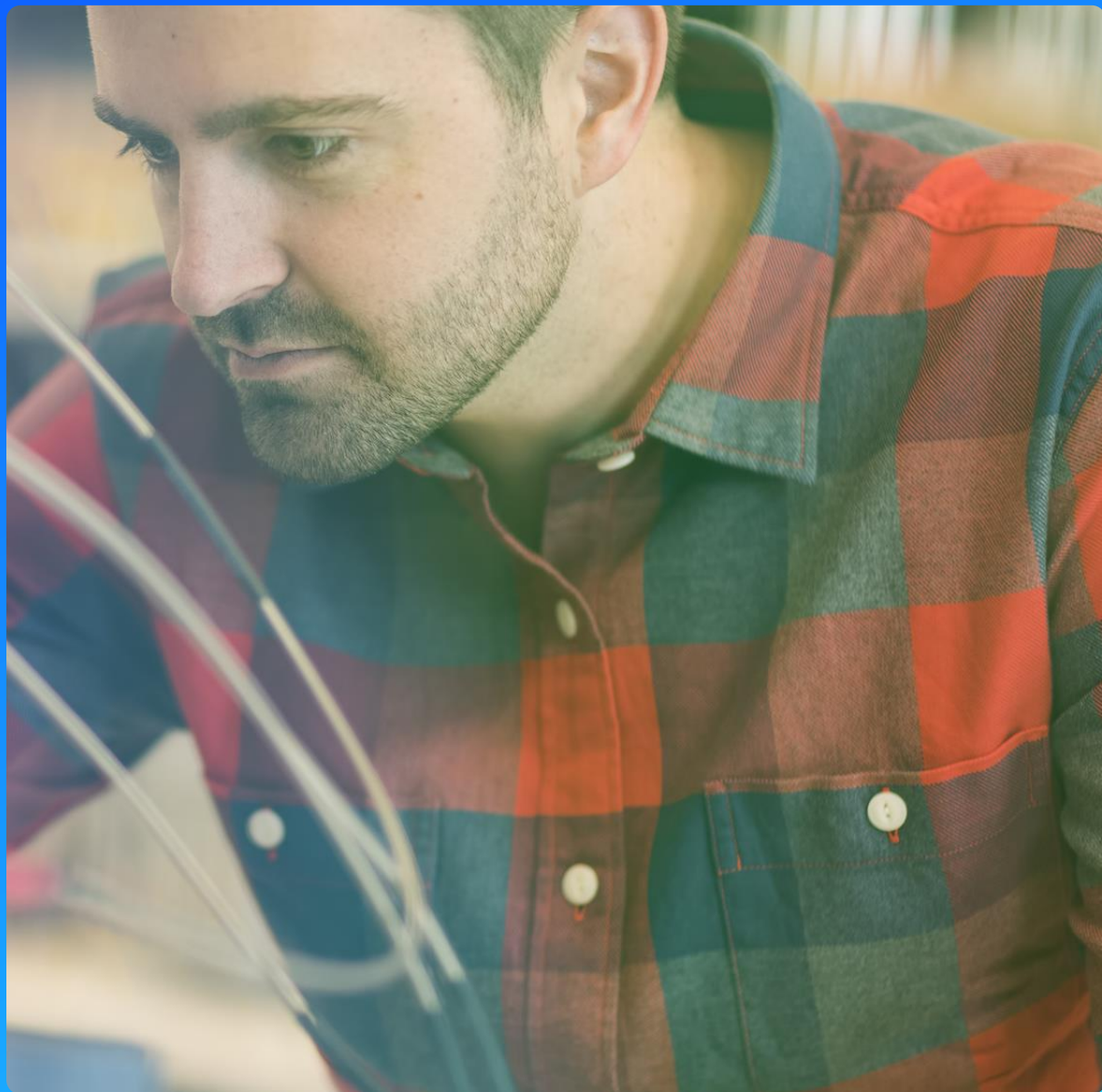
# Mobile Application Development Kit (ADK)



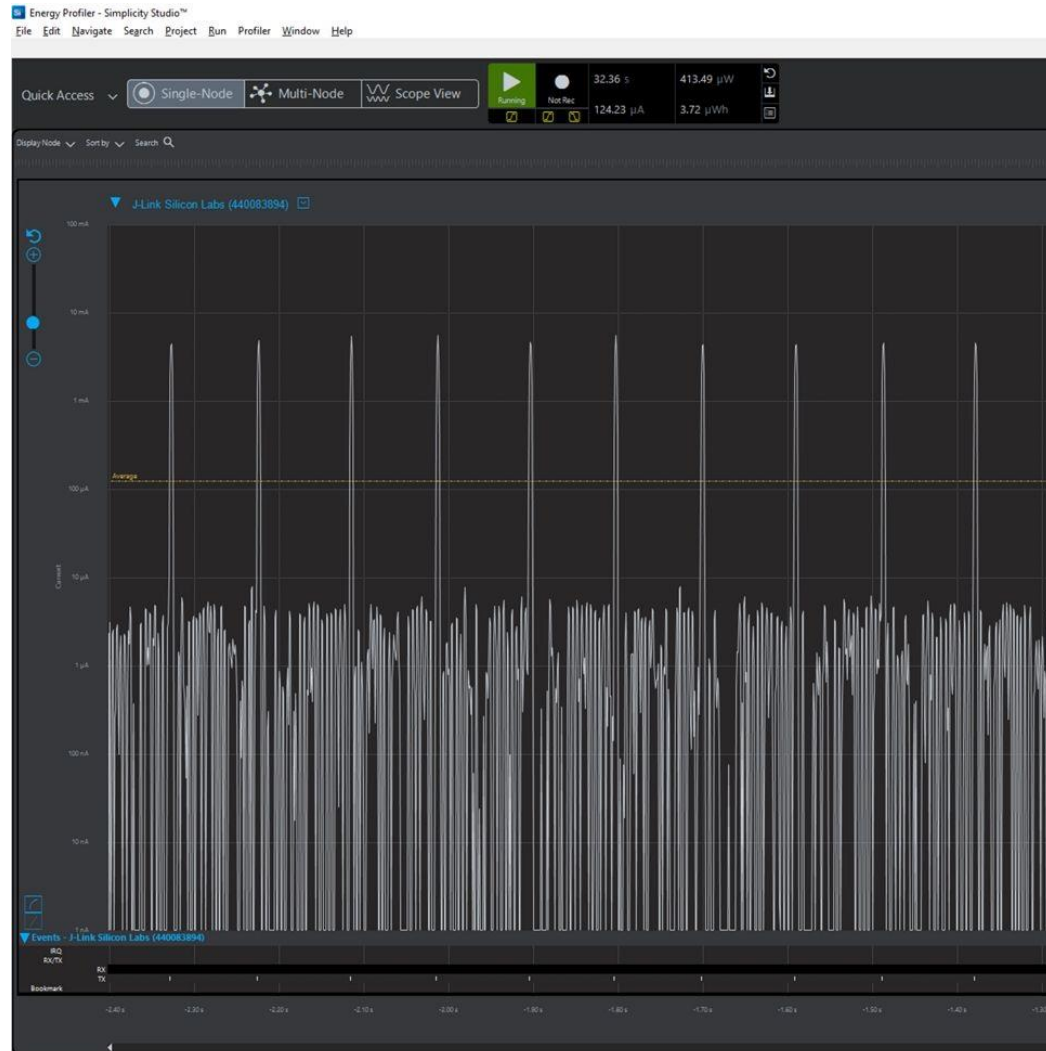
- **ADKs enable Bluetooth mesh application development for phones**
  - Both iOS and Android platform are supported
  - ADKs provide a Bluetooth mesh stack for both platforms
  - ADK contains example code how to provision, configure and control mesh devices
  - LE connectivity uses the underlying Bluetooth API provided by iOS or Android
  - Mobile app Source code available via Simplicity Studio
- **Features**
  - Node provisioning over LE connection and PB-GATT
  - Mesh node configuration
  - Mesh model configuration
  - Node control
  - Mesh database import/export
- **Download for evaluation from [iTunes](#) or [Google](#)**

---

## Advanced Debugging Tools

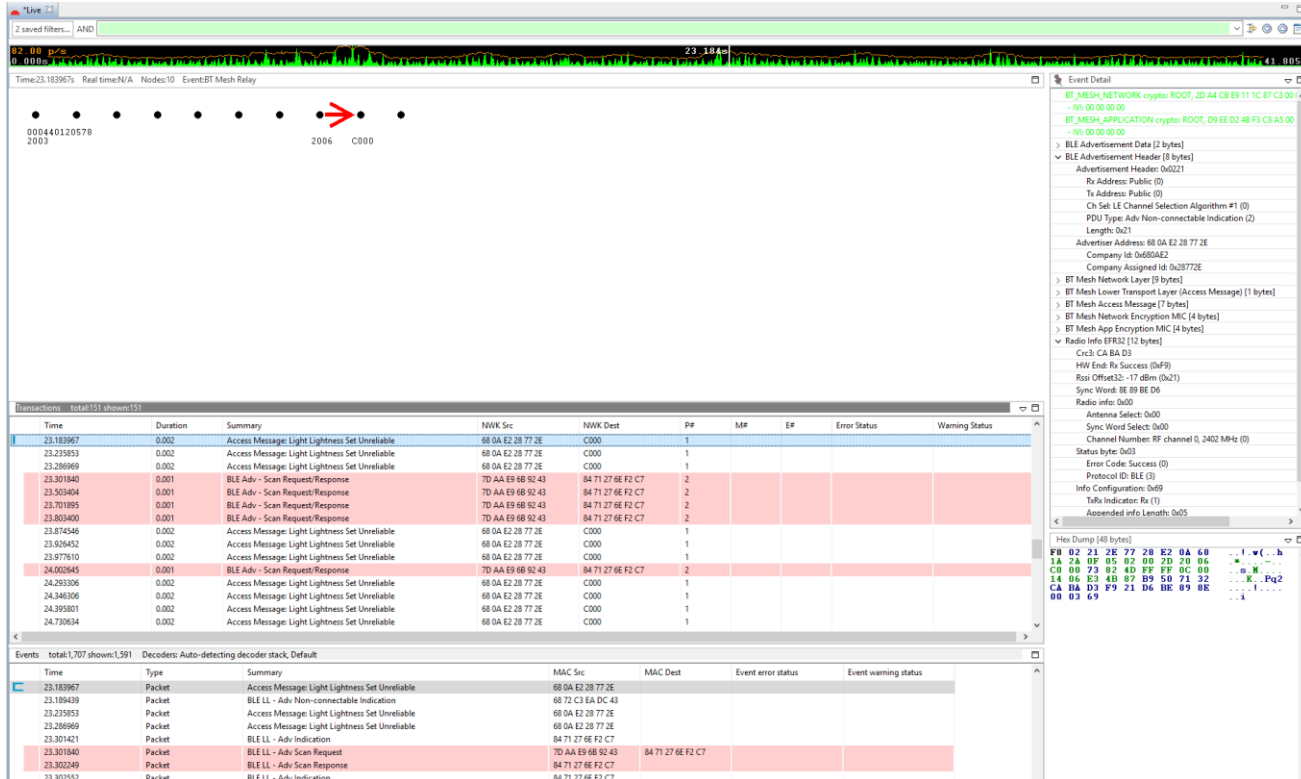


# Advanced Debugging – Energy Profiler



- **Energy Profiler measures current consumption**
  - Leverages AEM (Advanced Energy Monitoring) on Pro Kit
  - Can also be used on custom hardware
- **Identify misbehavior on your device based on energy signature**
  - Extract peak current, average and other metrics
  - High level of detail, comparable to bench instruments
- **Capture directly from WSTK's USB or Ethernet**
  - Live capture from multiple Ethernet networked Pro Kits from a single PC
  - Set triggers to start capturing only on specific activity

# Advanced Debugging – Network Analyzer

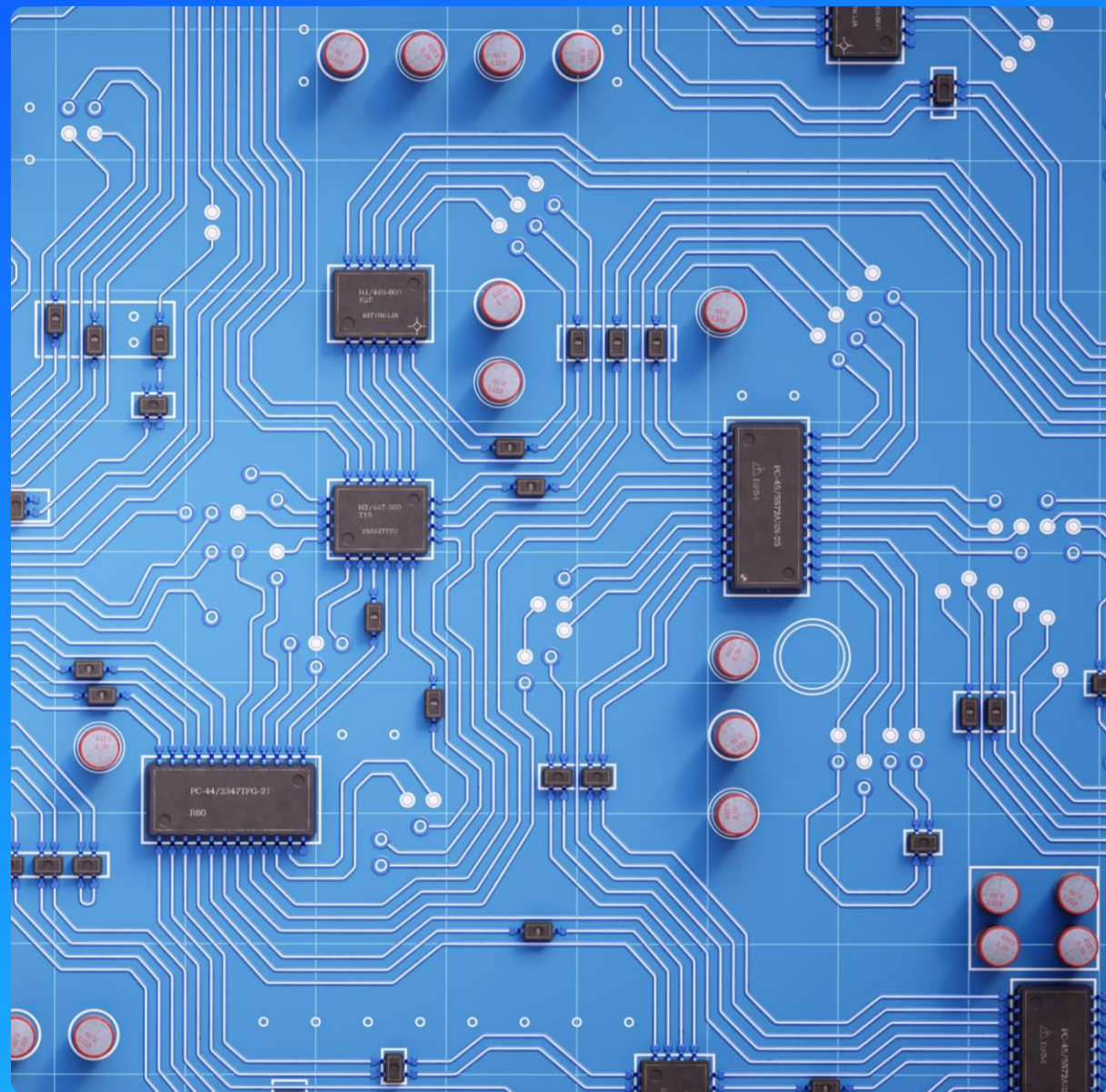


- **Network Analyzer captures and decodes Bluetooth LE and mesh packets**
  - Understand the network traffic easily
  - Debug connectivity or protocol issues
- **Packets are received from a dedicated PTI interface on EFR32**
  - PTI accurately captures what a device transmits or receives
  - A Bluetooth sniffer only captures what it hears
- **Capture directly from WSTK's USB or Ethernet**
  - Live capture from multiple Ethernet networked WSTKs from a single PC
  - No need to be in the range of all devices in a network



---

# Hardware Tools



# Hardware Tools – Kit Overview

	Explorer Kit	Dev Kit	Pro Kit
Debug Speed	1.6MHz	1.6MHz	8MHz
Debug USB	Full Speed	Full Speed	High Speed
Packet Trace Interface (PTI)	✓	✓	✓ 2x
Breakout Pads	✓	✓	✓
Pushbutton s & User LEDs	✓	✓	✓
Virtual COM	✓	✓	✓
Coin cell battery holder	–	✓	✓
On-board Sensors	–	✓	✓
Battery Pack Connector	–	✓	✓
Radio Board Connectors	–	–	✓
EXP Connector	–	–	✓
Display	–	–	✓
Debug OUT	–	–	EFM8/32, EFR32, EZR32
Debug Ethernet	–	–	100 Mbit/s
Energy Monitor (AEM)	–	–	✓
3 <sup>rd</sup> Party Hardware addons	✓	–	–



## Explorer Kit

- Lowest price point
- On-board debugger and signal breakouts
- Minimal on-board features
- 3<sup>rd</sup> part hardware support

**Available with BG22/BGM220**  
**Ideal for battery powered nodes (LPN)**

## Dev Kit

- Single device development board
- On-board debugger and signal breakouts
- On-board sensors
- Impressive out-of-the-box demos

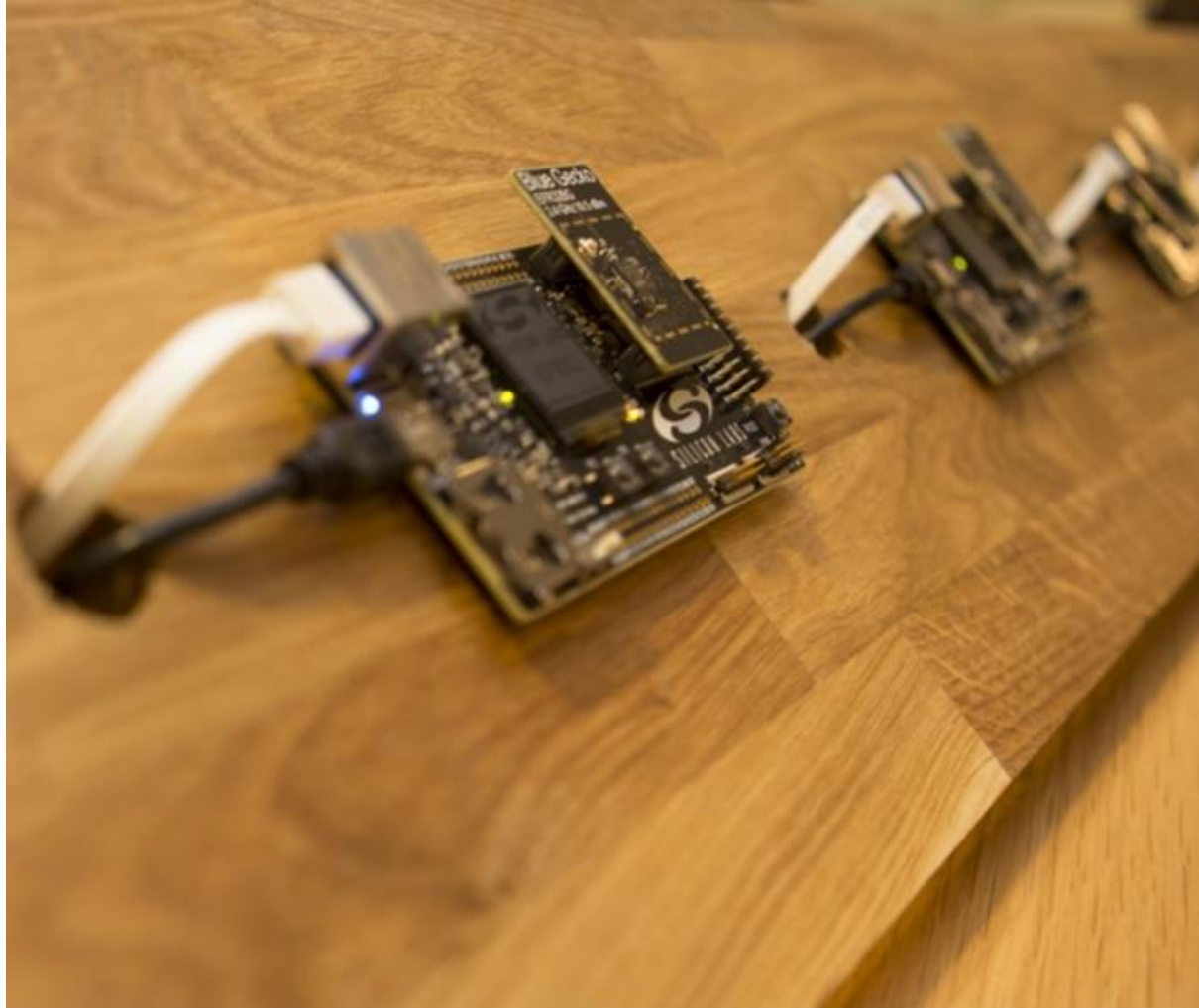
## Pro Kit

- Modular development platform
- Advanced development use cases
- Energy profiling and external device debug
- Designed to maximize reuse of EFR32 devices
- Ethernet for large network test

**Available with BG21**  
**Ideal for mains powered nodes**



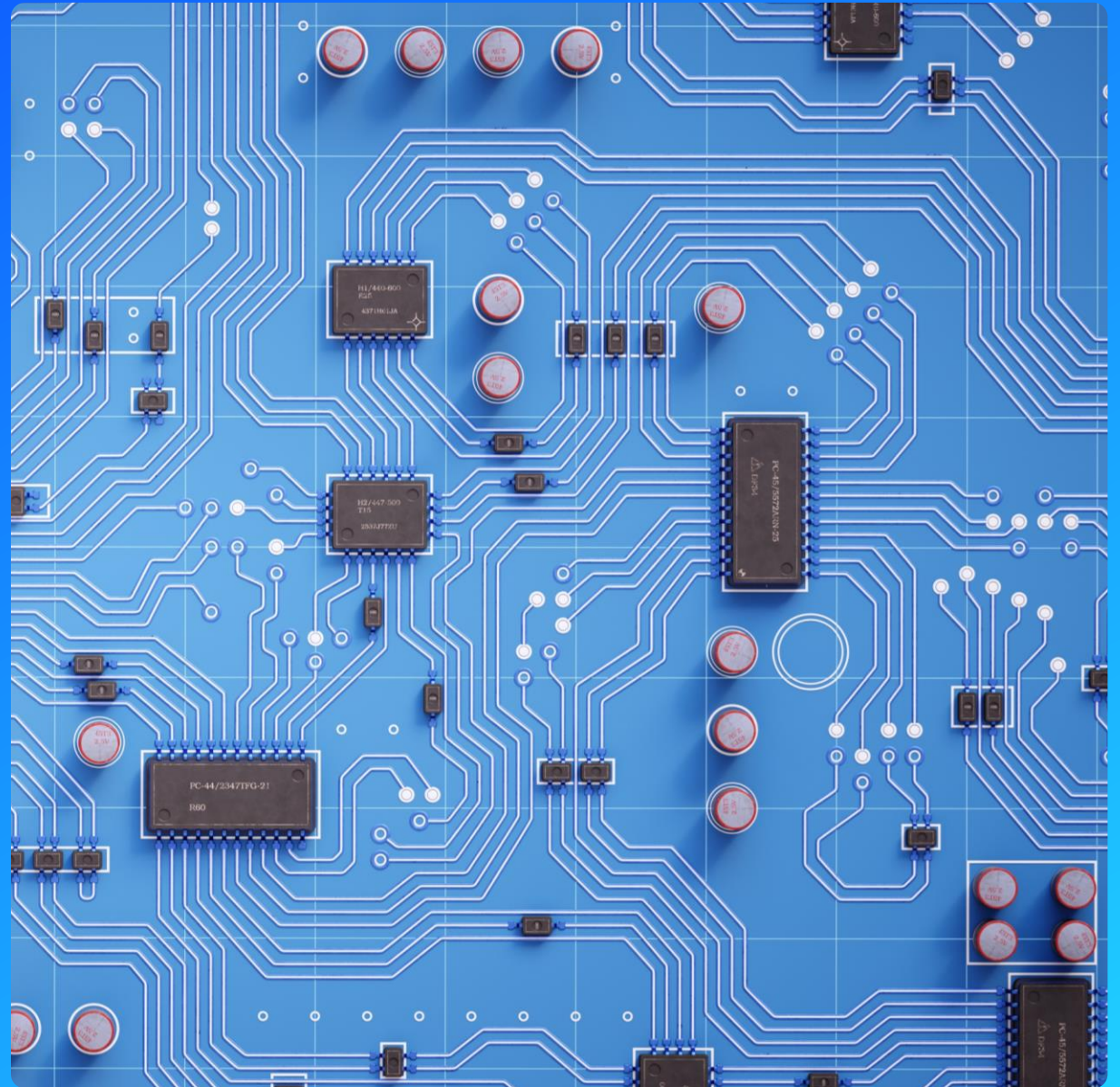
# Hardware Tools – Pro Kit



- **Remote Development and Debug**
  - WSTKs can be Ethernet connected
  - Enables debug over IP
  - Enables firmware updates over IP
  - Enables NCP over IP
  - USB also available
- **Protocol Capture and Analysis**
  - WSTKs have a Packet Trace Interface available
  - Can be used to capture all packets the radio sends and receives
  - Packet Trace is also accessible over IP
- **WSTKs can also be used to build large test networks**
  - Silicon Labs has WSTK networks up to 240 nodes
  - Used for Silicon Labs' own test networks for various mesh protocols such as Zigbee, Z-Wave and Wi-Sun

---

# Hands-on Demo





# Summary

- Silicon Labs' Bluetooth Mesh SDK **sample apps** offer optimal starting point
- **Project Configurator** and **Bluetooth Mesh Configurator** for application customization according to the customer requirements
- **PyBGAPI** for test automation and Bluetooth Mesh **ADK** for mobile app development
- **Energy Profiler** and **Network Analyzer** for advanced debugging
- **Hardware Kit** offering for various use cases and needs
- Energy Profiler and Network Analyzer **Hands-on Demo**

Thank you

