Application Examples

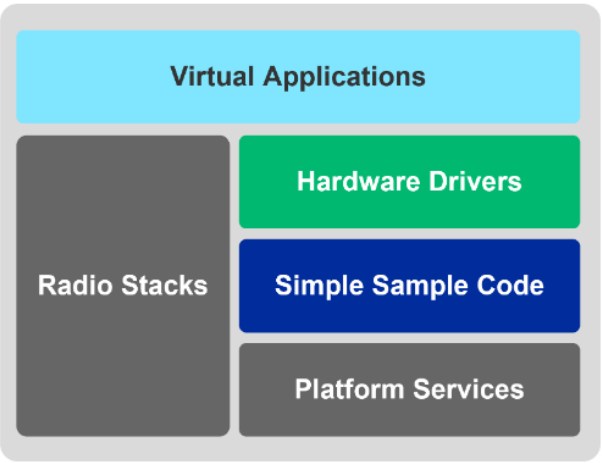
Overview

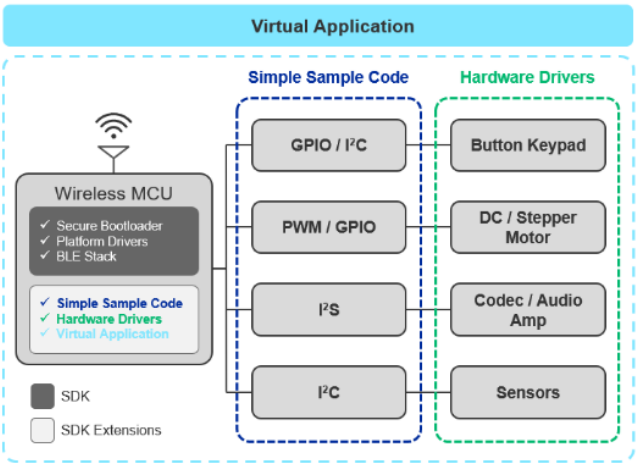
Overview

The **Application Examples Program (AEP)** is an umbrella term that includes the different promotion and enablement HW drivers, virtual applications, or reference designs.

The application examples are hosted on Github here.

Strong emphasis is put on a building block concept, where these can be combined and people can experiment with even more complex applications by reusing these sample codes. The aim is to extend the operation to 3rd party HW platforms/ecosystems as well, not just supporting our development kits.





As the go-to provider for IoT solutions, we provide developers at all levels with easy-to-use, accessible application examples they can use to speed up their developments and get to market faster.

Simple Sample Code

Simple sample code is the most basic product from the AEP program portfolio. Those mainly intend to demonstrate peripherals and low-level platform components (NVM3, crypto, BL).

The goal is to provide a comprehensive set of examples demonstrating their features from different angles for different purposes.

Hardware Driver

Hardware drivers are basic drivers for external hardware such as sensors, displays, or transmitters that would commonly be used with Silicon Labs products.

The scope of the hardware driver development is to provide feature-rich basic drivers for the market-leading development shields (Sparkfun Qwiic, MikroE Click platforms) commonly used for rapid prototyping by professionals and hobbyists for university and hobby projects.

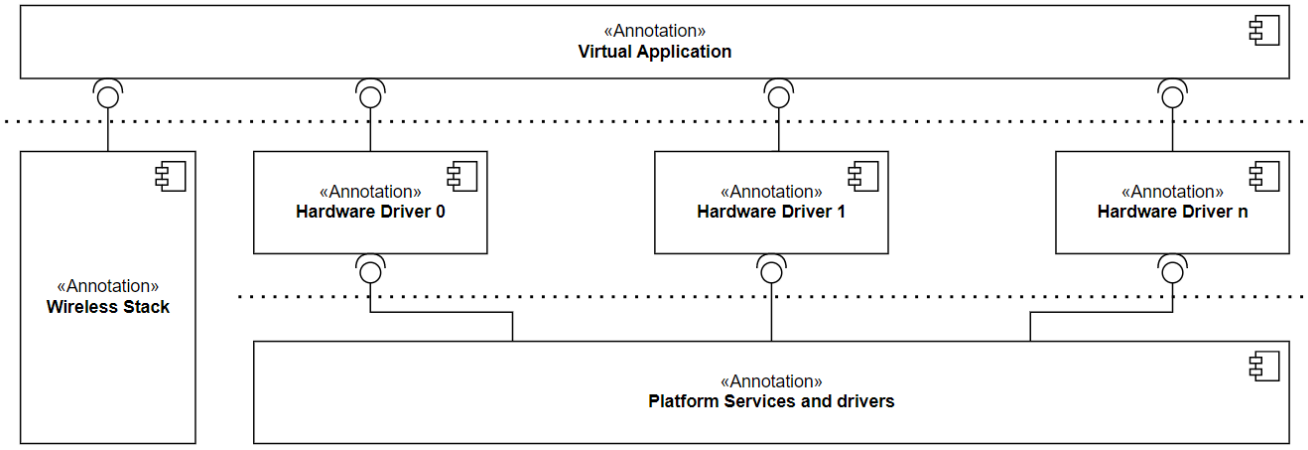
Those mainly intend to provide basic building blocks for application development.

Virtual Application

Virtual application is a higher abstraction layer, and it intends to promote the capabilities of hardware drivers integrated into a wireless stack like BLE.

Virtual applications are either wireless or mostly wireless stack applications using simple sample codes and hardware drivers to promote the platform and the AEP component features. Those applications form ready-to-build projects like Simplicity Studio projects.

Complex virtual applications typically provide solutions for popular IoT products, like smart door lock, speech controlling, and so on.

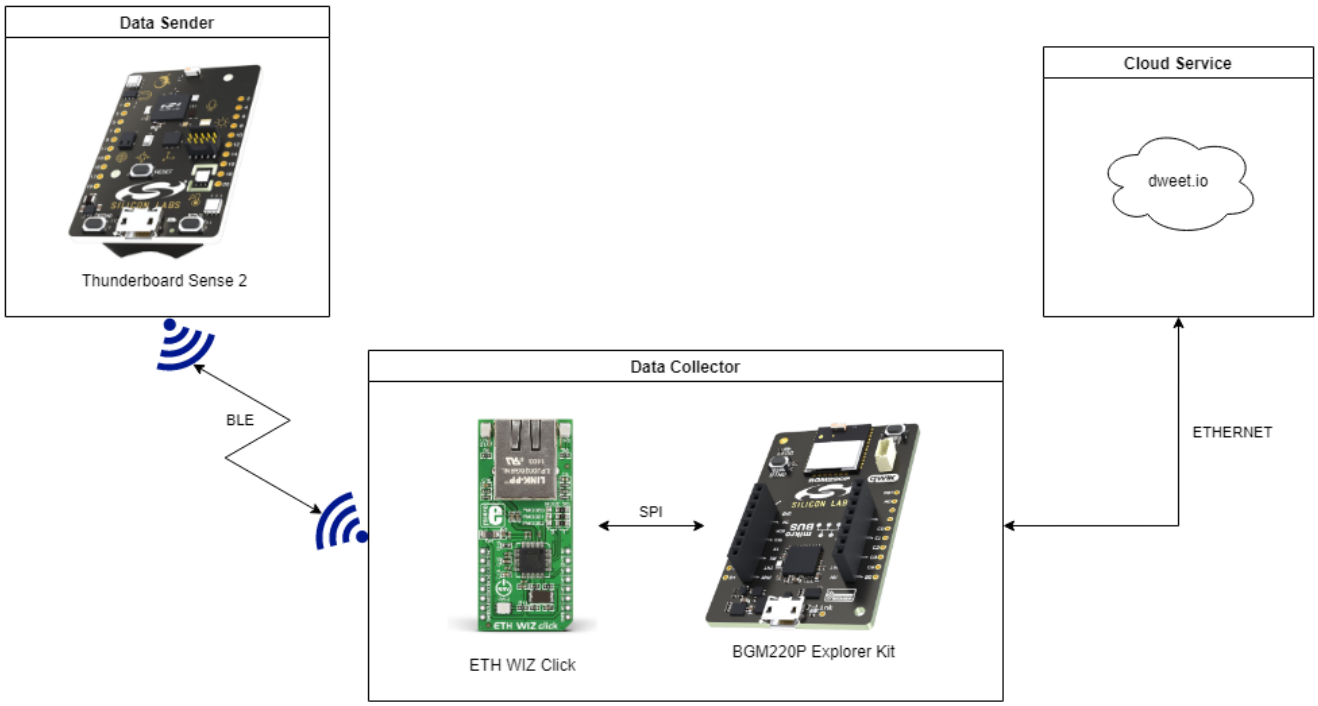


Example - Bluetooth Ethernet Gateway Application

Overview

This project aims to implement a simple Bluetooth-Ethernet Thin Gateway, wherein the sensor measures and collects data from the device's environment, and the gateway request the results via BLE.

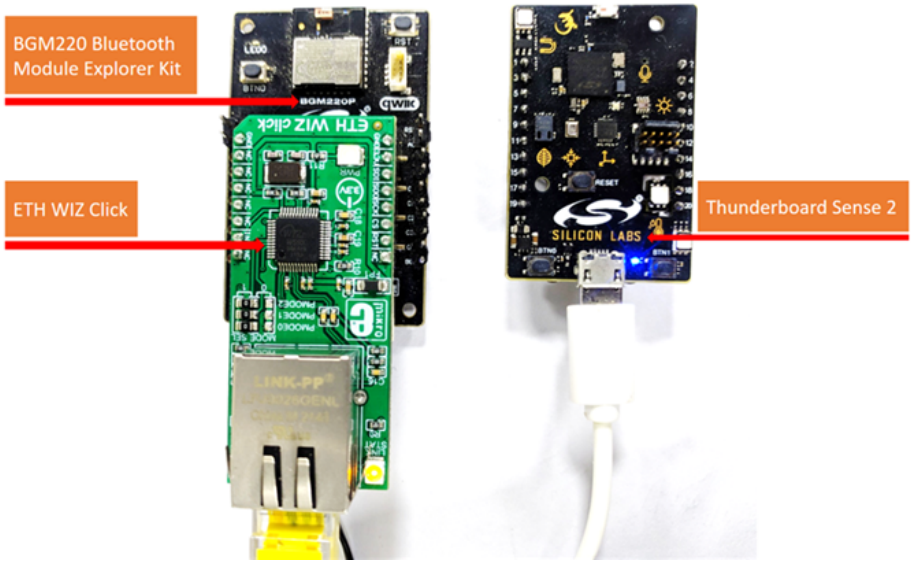
When the device is connected to a sensor peripheral, the gateway reads the BLE characteristics to retrieve the measured temperature and humidity. The measurement results are uploaded to dweet.io via the Ethernet Click board.



Bluetooth Ethernet Gateway

Hardware Setup

The ETH WIZ Click can be plugged into the BGM220 Bluetooth Module Explorer Kit via the mikroBus socket.

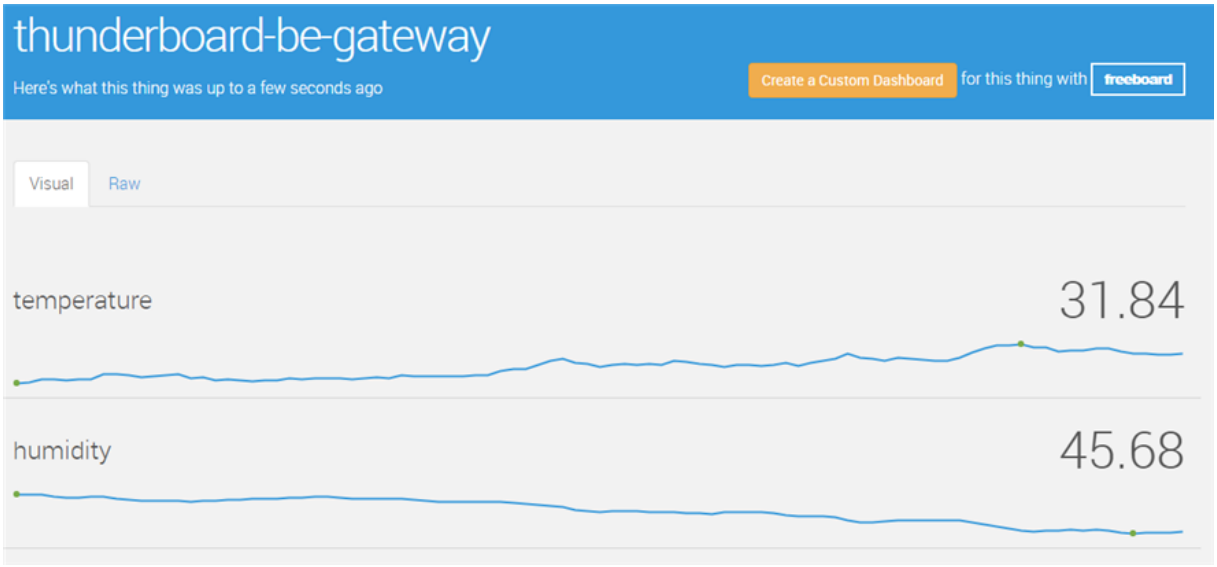


Cloud Service

The Thunderboard Sense 2 development kit measure the temperature and humidity values. These values are transmitted via BLE from the data sender device to the data collector environment.

The measured environment data are transferred to the cloud service via an Ethernet connection.

The transmitted values can be visualized on the graphical interface of the cloud service.


thunderboard-be-gateway
Here's what this thing was up to a few seconds ago
Create a Custom Dashboard
for this thing with
freeboard
Visual
Raw
temperature
31.84
humidity
45.68

Getting Started

What do you need to get started?

To get started with Application Examples, the following requirements have to be fulfilled.

How is it organized?

How do you use it?

Requirements

Requirements

Hardware

Development Kits

> The required development kit is defined for each application example. Information on the required development kit is described in the project's readme files in the repository of the example.

The most frequently used development, starter, and explorer kits are listed in the table below.

Description	Identifier	Documentation/Links
EFR32xG22 Wireless Gecko Starter Kit	SLWSTK6021A	SLWSTK6021A
BGM220 Bluetooth Module Explorer Kit	BGM220-EK4314A	BGM220-EK4314A
Thunderboard Sense 2	SLTB004A	SLTB004A
EFR32xG24 Dev Kit	xG24-DK2601B	xG24-DK2601B
EFR32xG24 Explorer Kit	xG24-EK2703A	xG24-EK2703A
SparkFun Thing Plus Matter - MGM240P	DEV-20270 (BRD2704A)	DEV-20270

Description	Identifier	Documentation/Links
SparkFun Qwiic Cable Kit	KIT-15081	KIT-15081
Silabs Click Shield	MIKROE-4464	MIKROE-44641

Third-Party Hardware

The hardware drivers and some of the virtual applications require third-party hardware. These third-party hardware boards can be purchased from the following suppliers.

> The documentation of each hardware driver and application example describes the required type of third-party board.

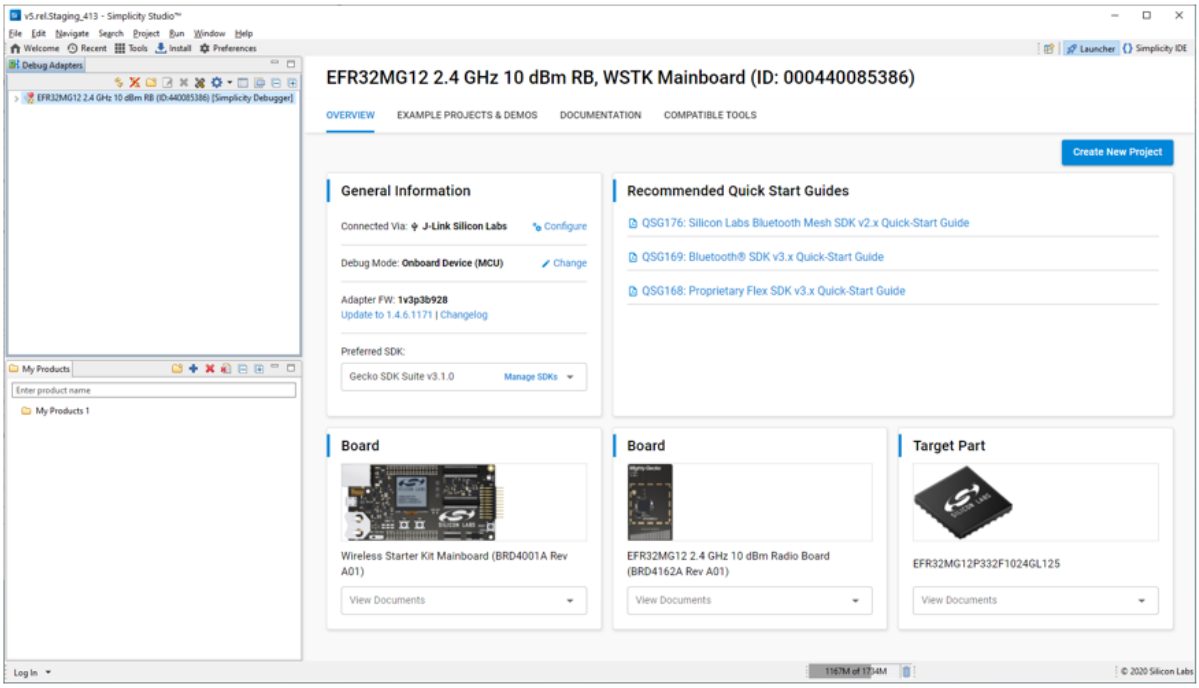
The most frequently used third-party suppliers

Supplier	Link
MIKROE	https://www.mikroe.com
sparkfun ELECTRONICS	https://www.sparkfun.com
adafruit	https://www.adafruit.com

Software

Development Environment

Simplicity Studio is the core development environment designed to support the Silicon Labs IoT portfolio of system-on-chips (SoCs) and modules. It provides access to target device-specific web and SDK resources; software and hardware configuration tools; an integrated development environment (IDE) featuring industry-standard code editors, compilers and debuggers, and advanced, value-add tools for network analysis and code-correlated energy profiling.



Download Simplicity Studio

Simplicity Studio Overview

Software Development Kit

The required GSDK version is described in the readme files for each project.

> In general, GSDK **v4.x.x** version or higher is required to compile and run the examples.

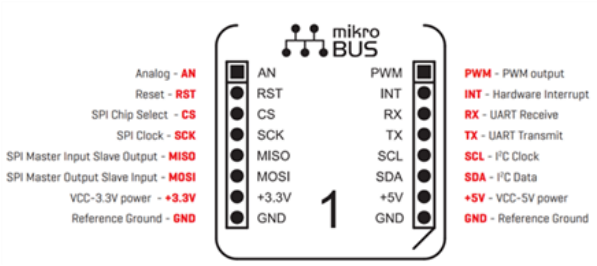
Some of the examples require Third Party Hardware Driver GSDK Extension.

Setup

Silabs development kits can be connected to any third-party shield via simple wiring, however, most of the boards support quick and easy connectivity via Qwiic and mikroBUS connectors.

Mikroe Click Boards

MikroElektronika Click boards can be connected to host controllers via the mikroBUS connectors; see the pinout specification below.



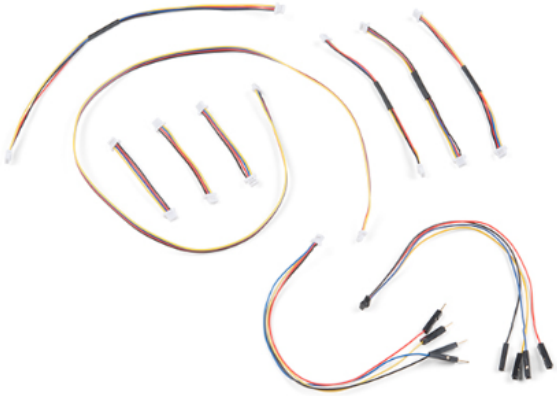
Sparkfun Qwiic or Adafruit STEMMA QT Boards

The sparkfun Qwiic and Adafruit STEMMA QT capable boards use the same 4 pins - JST SH 1.0mm pitch connectors to provide quick and easy I2C connectivity between development kits and third-party boards.

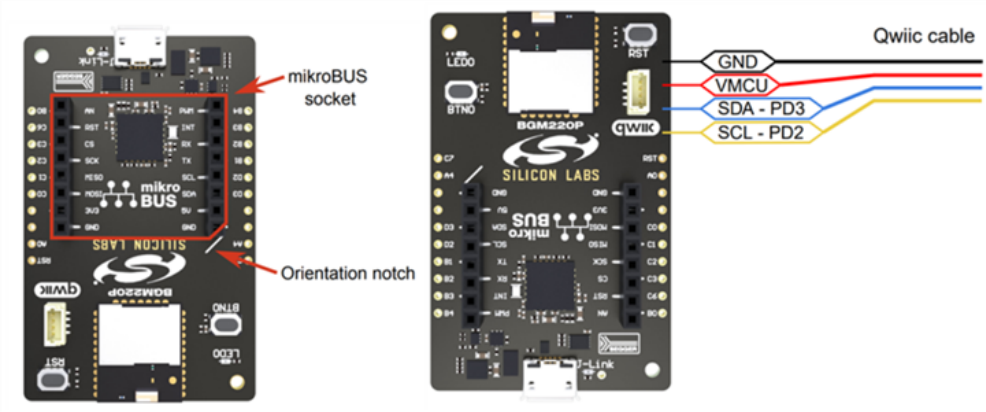
All Qwiic/STEMMA QT cables have the following color scheme and arrangement:

- Black = GND
- Red = 3.3V
- Blue = SDA
- Yellow = SCL

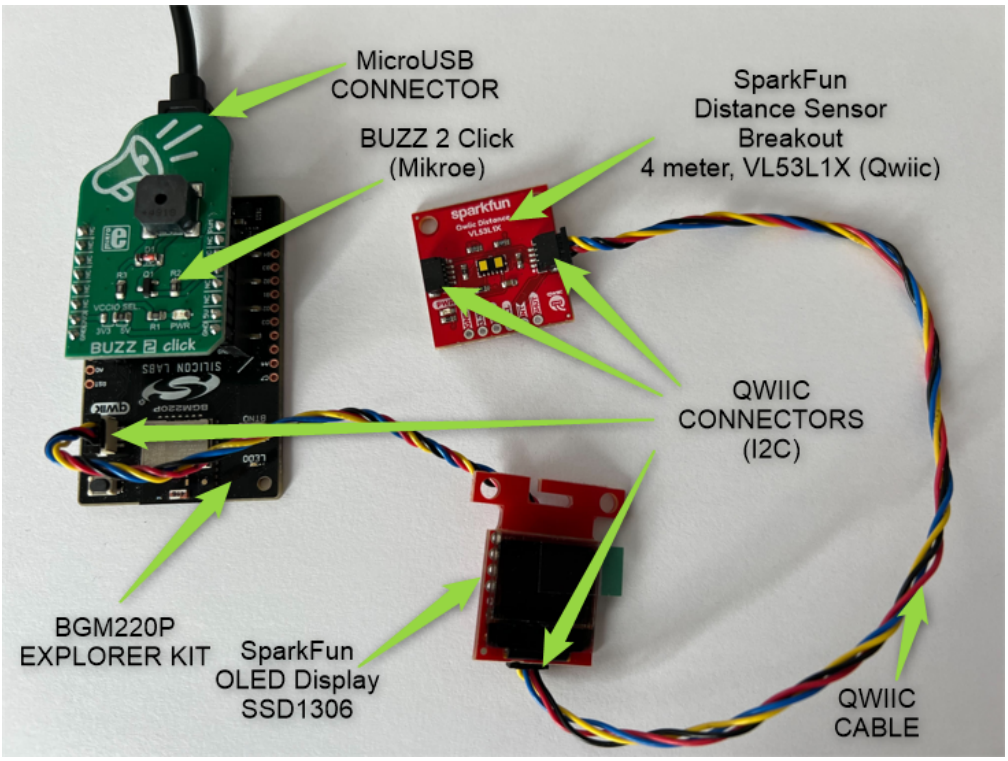
These are the most frequently used cables for Sparkfun Qwiic and Adafruit STEMMA QT connections.



BGM220 Bluetooth Module Explorer Kit / EFR32xG24 Explorer Kit



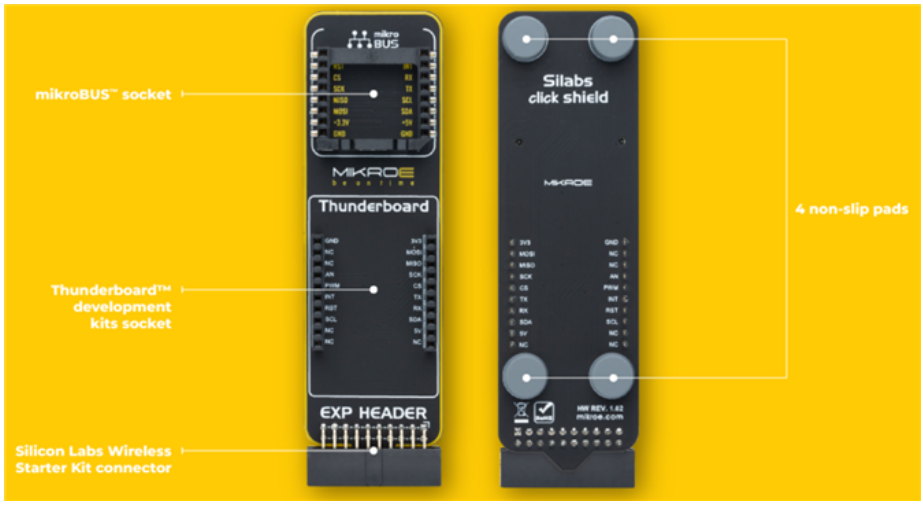
The kit features support for hardware add-on boards via a mikroBus socket and a Qwiic connector. The hardware add-on support allows developers to create and prototype applications using a virtually endless combination of off-the-shelf boards from mikroE, sparkfun, AdaFruit, and Seeed Studios.



EFR32xG22 Wireless Gecko Starter Kit / Thunderboard Sense 2 / EFR32xG24 Dev Kit

Mikroe provides a Silabs Click Shield for easily connecting Silabs development kits from the Thunderboard family or any other Silabs wireless or MCU starter kit through the expansion header (EXP) to the Mikroe Click boards.

Sparkfun Qwiic/Adafruit STEMMA QT boards also can be connected to these development kits through the expansion header.



Mikroe Click Temperature sensor board connected to a Thunderboard Sense 2 via a Silabs Click shield.

SparkFun Thing Plus Matter - MGM240P

SparkFun Thing Plus Matter - MGM240P is a development board from the Sparkfun Thing Plus development board family.

This board integrated the MGM240P module. The MGM240P wireless module from Silicon Labs® provides secure connectivity for both 802.15.4 with Mesh communication (Thread) and Bluetooth® Low Energy 5.3 protocols. The module comes ready for integration into Silicon Labs' Matter IoT protocol for home automation. SparkFun's Thing Plus development boards are Feather-compatible and include a Qwiic connector for easy integration into our Qwiic Connect System for solderless I2C circuits.

External boards can be easily connected via the onboard Qwiic connector.


MGM240P

Repositories

Repositories

The hardware drivers and application examples are stored on Github in public Git repositories.

Hardware Drivers

These are drivers for third-party devices such as sensors, displays, and interfaces for various types of hardware.

> Deprecated repository, the newly developed drivers for the application examples are stored in the Third-Party Hardware Drivers GSDK Extension.

https://github.com/SiliconLabs/platform_hardware_drivers

Third-Party Hardware Drivers - GSDK Extension

SDK Extension for Third-Party hardware drivers for EFM32 and EFR32.

https://github.com/SiliconLabs/third_party_hw_drivers_extension

Wireless Applications by technology

The Application Examples Program provides real-life application examples for most of the popular wireless technologies. These examples are stored in separate repositories.

https://github.com/SiliconLabs/application_examples

Bluetooth

- https://github.com/SiliconLabs/bluetooth_applications
- https://github.com/SiliconLabs/bluetooth_mesh_applications
- https://github.com/SiliconLabs/bluetooth_mesh_stack_features
- https://github.com/SiliconLabs/bluetooth_peripherals
- https://github.com/SiliconLabs/bluetooth_stack_features

Proprietary

- https://github.com/SiliconLabs/proprietary_rail
- https://github.com/SiliconLabs/proprietary_connect

Thread

- https://github.com/SiliconLabs/openthread_applications

Wi-Fi

- https://github.com/SiliconLabs/wifi_combo_applications

Zigbee

- https://github.com/SiliconLabs/zigbee_applications

Z-Wave

- https://github.com/SiliconLabs/z_wave_applications

Matter

- https://github.com/SiliconLabs/matter_applications

EFM32-EFR32 Platform

- https://github.com/SiliconLabs/peripheral_examples
- https://github.com/SiliconLabs/platform_applications

Utilities and Training Materials

Trainings

- https://github.com/SiliconLabs/training_examples

IoT Utilities

- https://github.com/SiliconLabs/host_utilities
- https://github.com/SiliconLabs/java_pcap_file_utilities

Usage

Usage

This section contains information on adding the Third-Party Hardware Drivers GSDK Extension and importing Application Example projects into Simplicity Studio.

Importing Simplicity Studio projects

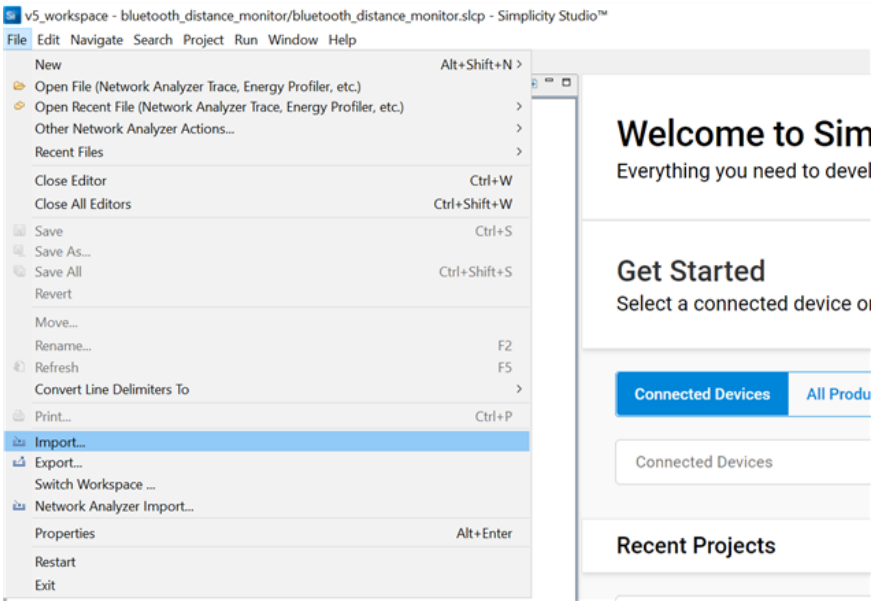
Simplicity Studio projects are typically stored under the SimplicityStudio folder inside each project's folder in the Git repositories.

File Formats

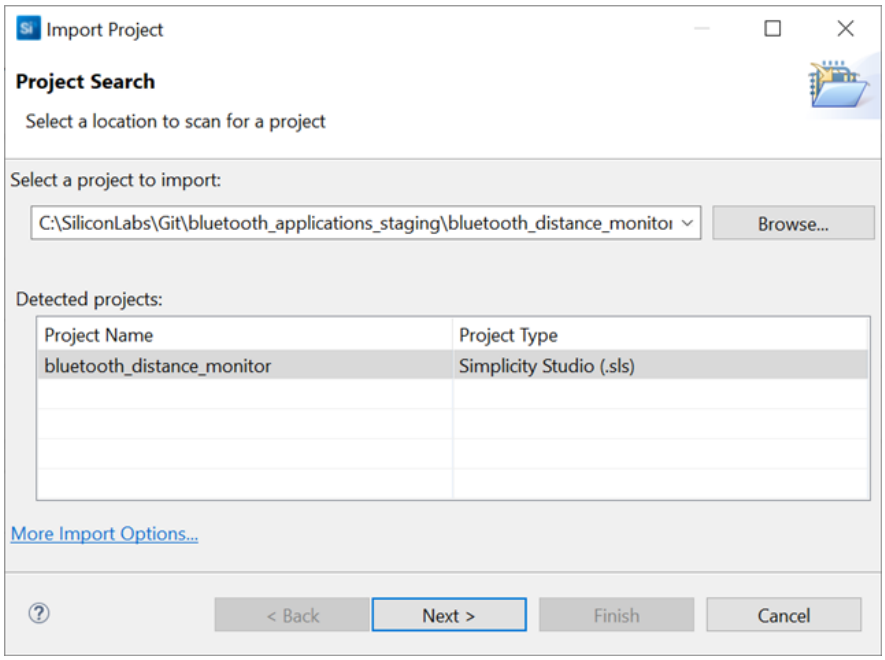
- *.sls
  Legacy projects provide *.sls files. These Simplicity Studio files contain each source file required to build the project (except GSDK files).
- *.slcp
  Newer projects provide only *.slcp files; these files contain the project configuration, such as the configured software components, source files, headers, includes paths, etc.
- *.btconf
  This file contains configuration interpreted by the Bluetooth GATT Configurator Tool, and it contains the GATT database with the configured services and characteristics.
- *.pintool
  The configuration files used by the Pin Tool, these files contain hardware pin assignments and mode configurations.

Importing *.sls projects

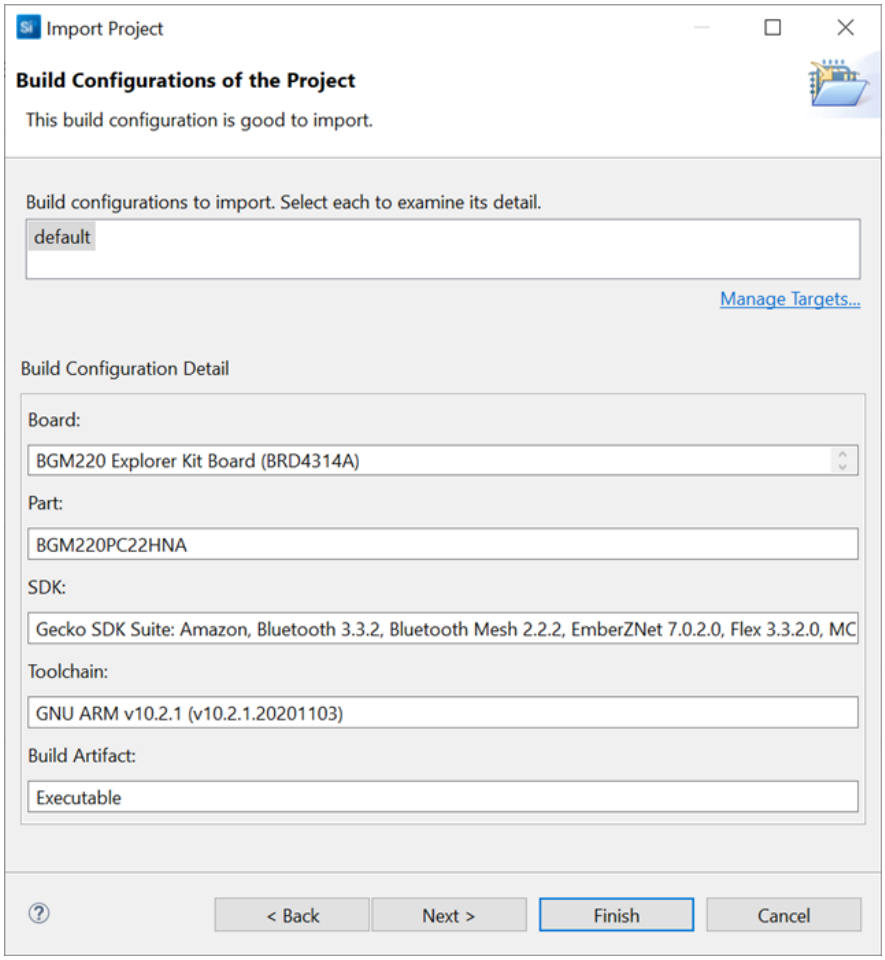
- STEP 1 [File] -> [Import]



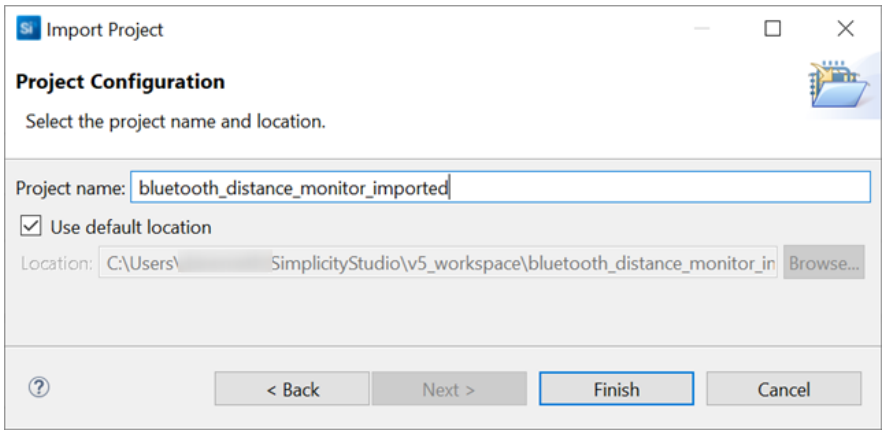
- STEP 2 Select a folder containing *.sls file(s). Select a project from the detected projects list and click on **Next**.



- STEP 3 Click on **Next**



- STEP 4 Type a name to the new project or keep the original naming. Click on **Finish**.



The selected Simplicity Studio project is imported into your workspace. You can compile the project and run the executable on a real hardware like a development kit.

Adding SDK Extensions for Hardware Drivers

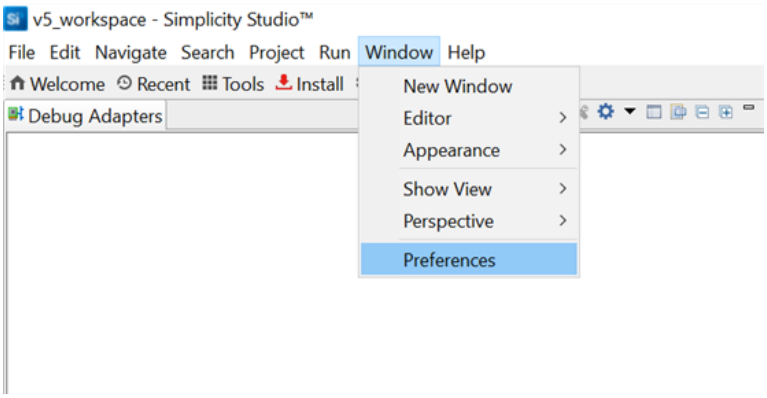
You can find the Third-Party Hardware Drivers GSDK Extension on Github; as a first step you should clone the repository to your local computer.

You can follow this step-by-step guide to install and use the extension or watch a guideline video showing the same steps.

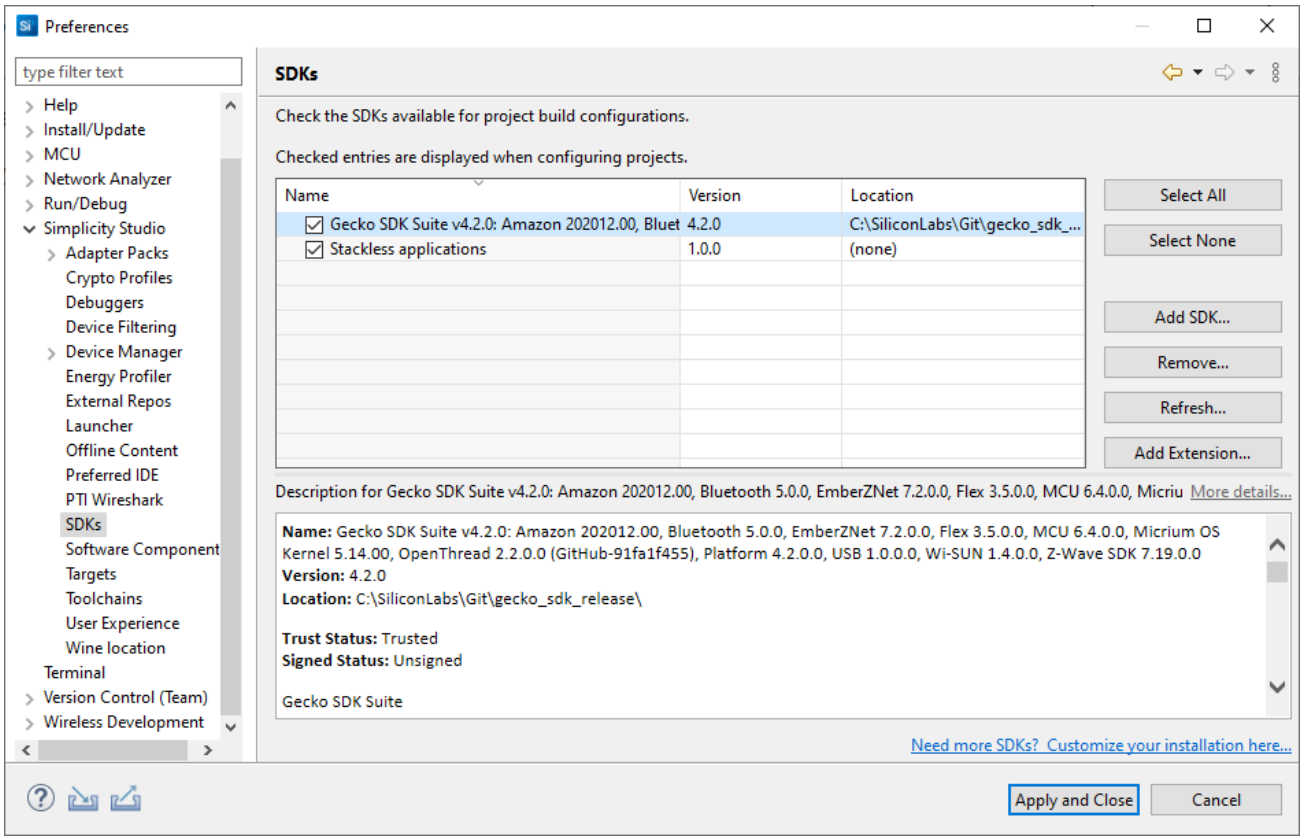
- **STEP 1** Clone Third-Party Hardware Drivers repository from Github
  It is up to you to choose a folder on your computer to clone the repository into.

  ```
  git clone https://github.com/SiliconLabs/third_party_hw_drivers_extension.git
  ```

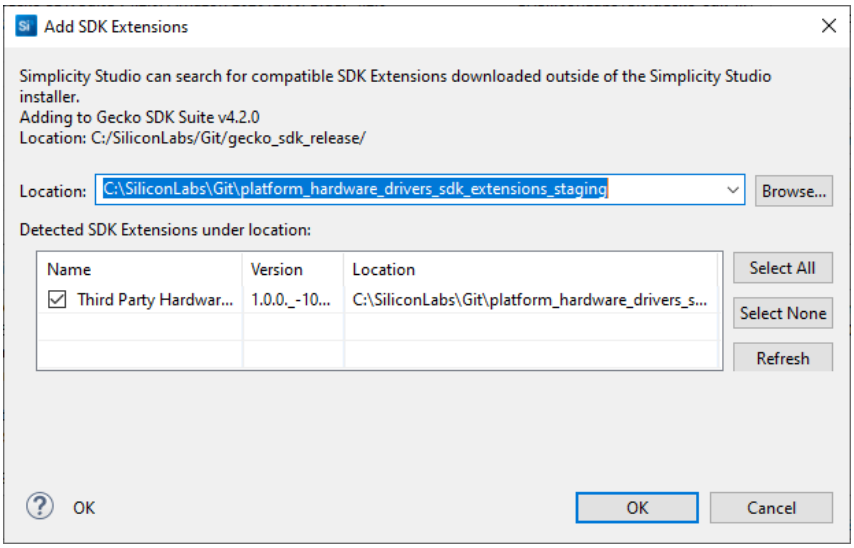
- **STEP 2** Open Simplicity Studio and go to Window/Preferences



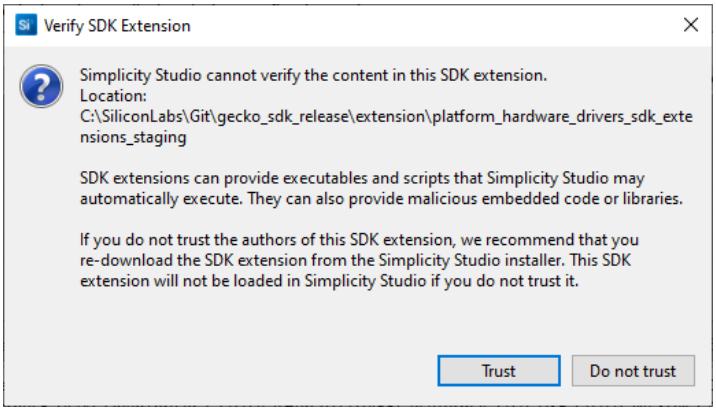
- **STEP 3** Go to Simplicity Studio/SDKs, select an installed GSDK (version >=4.2.0), and click on **Add Extension**



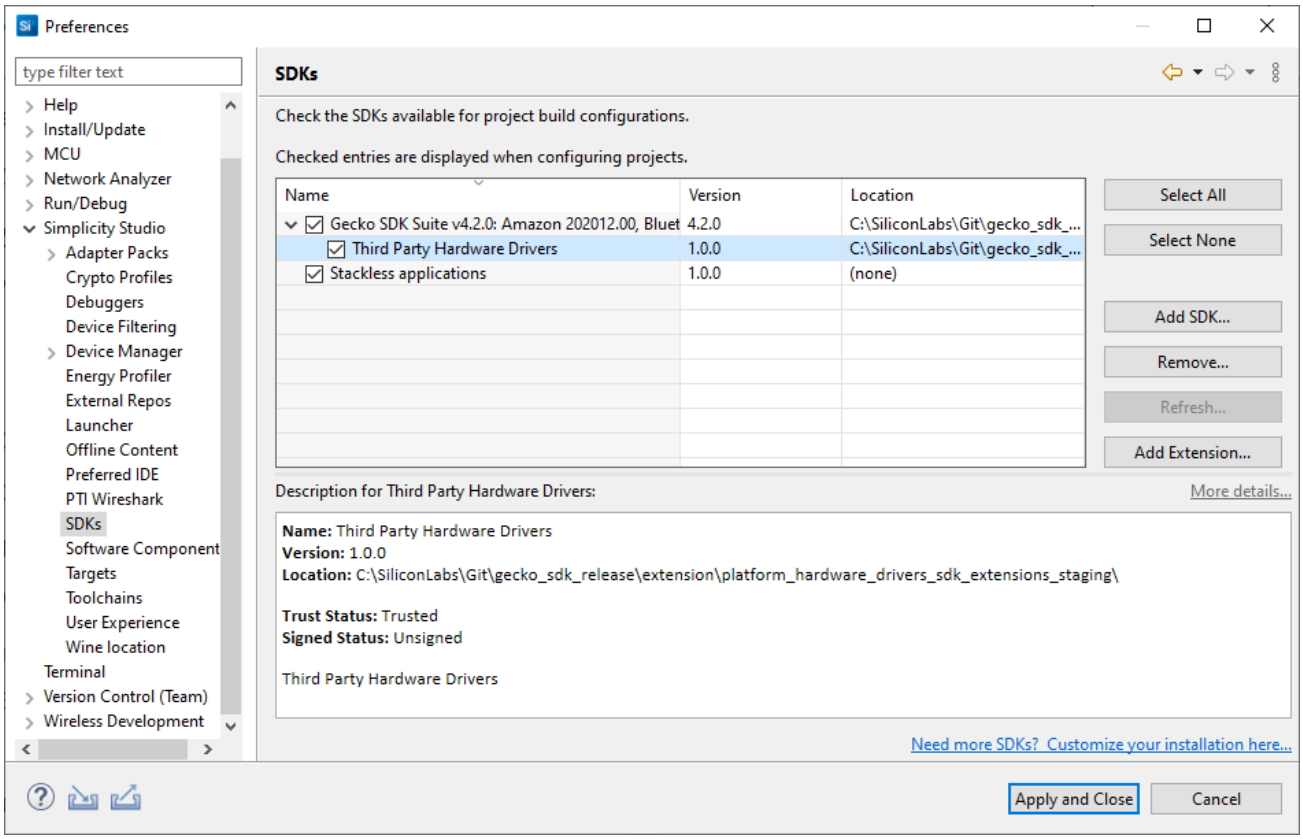
- **STEP 4** Select the SDK Extension's location, click on **OK**
  The SDK Extension will be detected in the repository folder.



- **STEP 5** Click on the **Trust** button on the Verify SDK Extension dialog



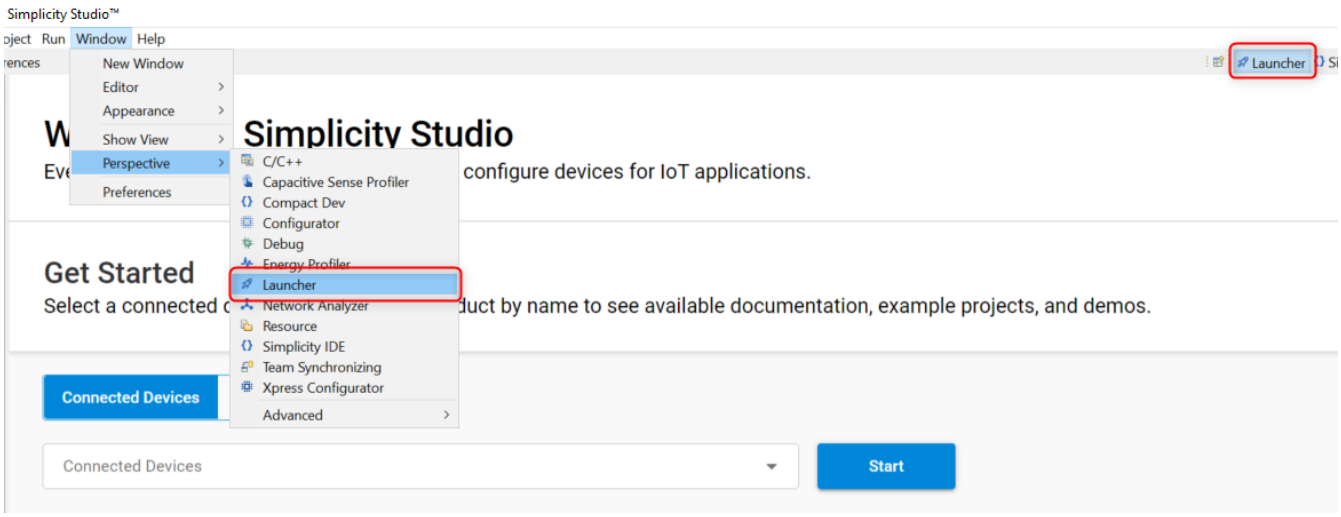
- **STEP 6** The SDK Extension successfully installed, click on **Apply and Close**



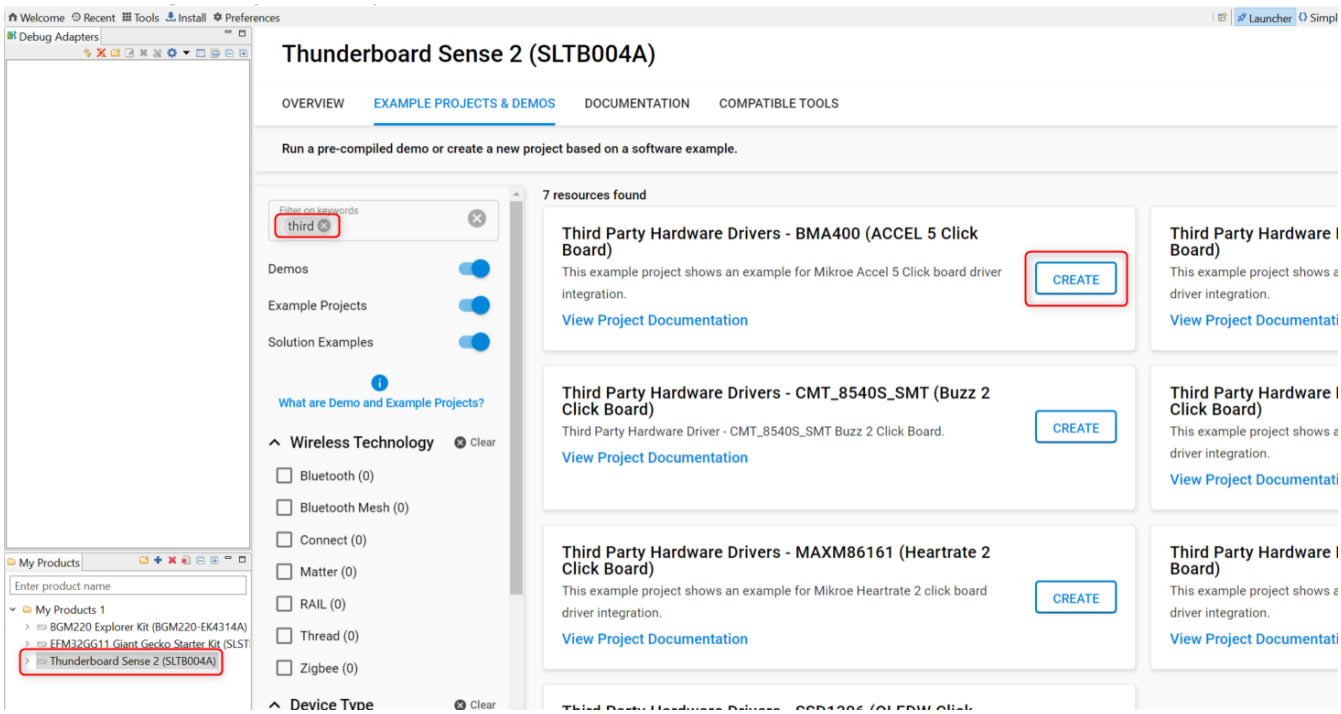
Example Project Templates

The Third-Party Hardware Drivers extension provides example project templates for each supported driver.

- **STEP 1** Open the Launcher perspective in Simplicity Studio



- **STEP 2** Select a product either in the My Products or in the Debug Adapters dialog
- **STEP 3** Filter examples by typing "third" or "Third Party" in the "filter on keywords" input
- **STEP 4** Select an example project from the resources and click on the **Create** button.



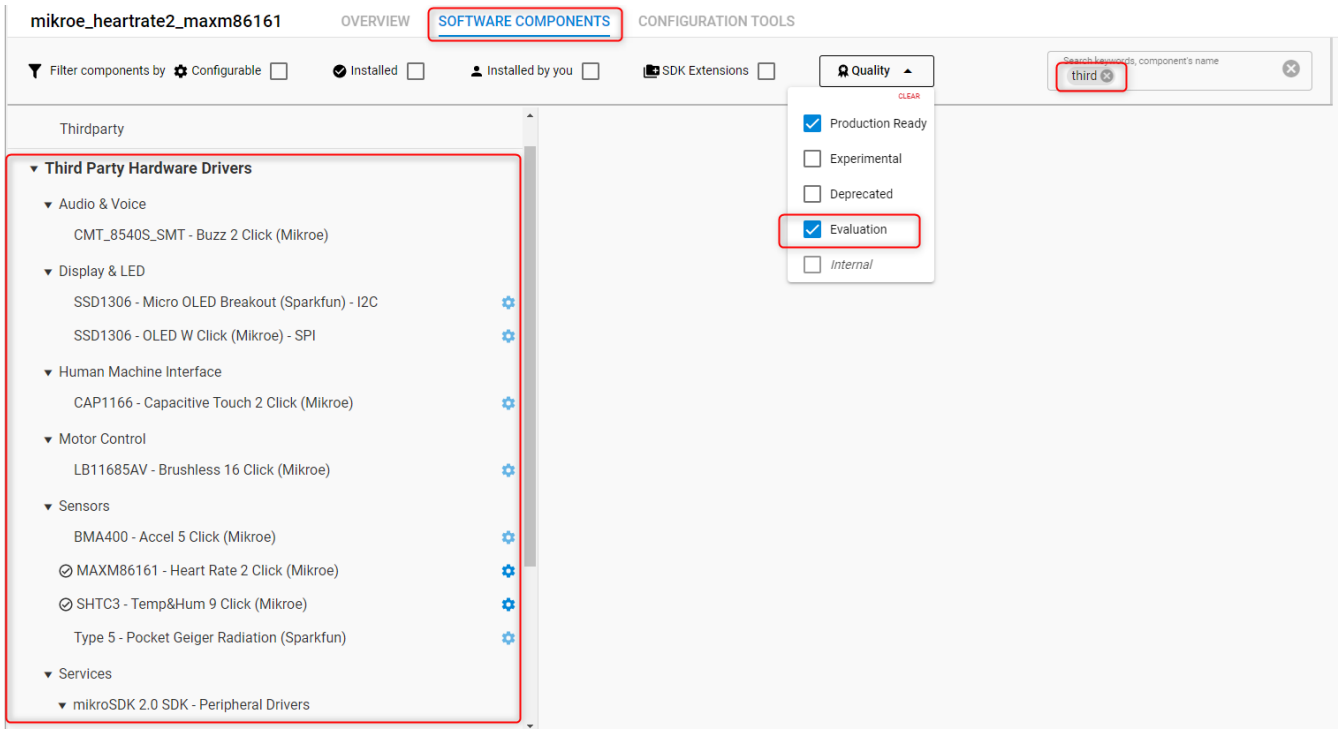
- **STEP 5** Follow the steps in the New Project Wizard dialog.

The Launcher creates a new project based on the selected template, and this project contains basic example on how your application can integrate a driver using the extension.

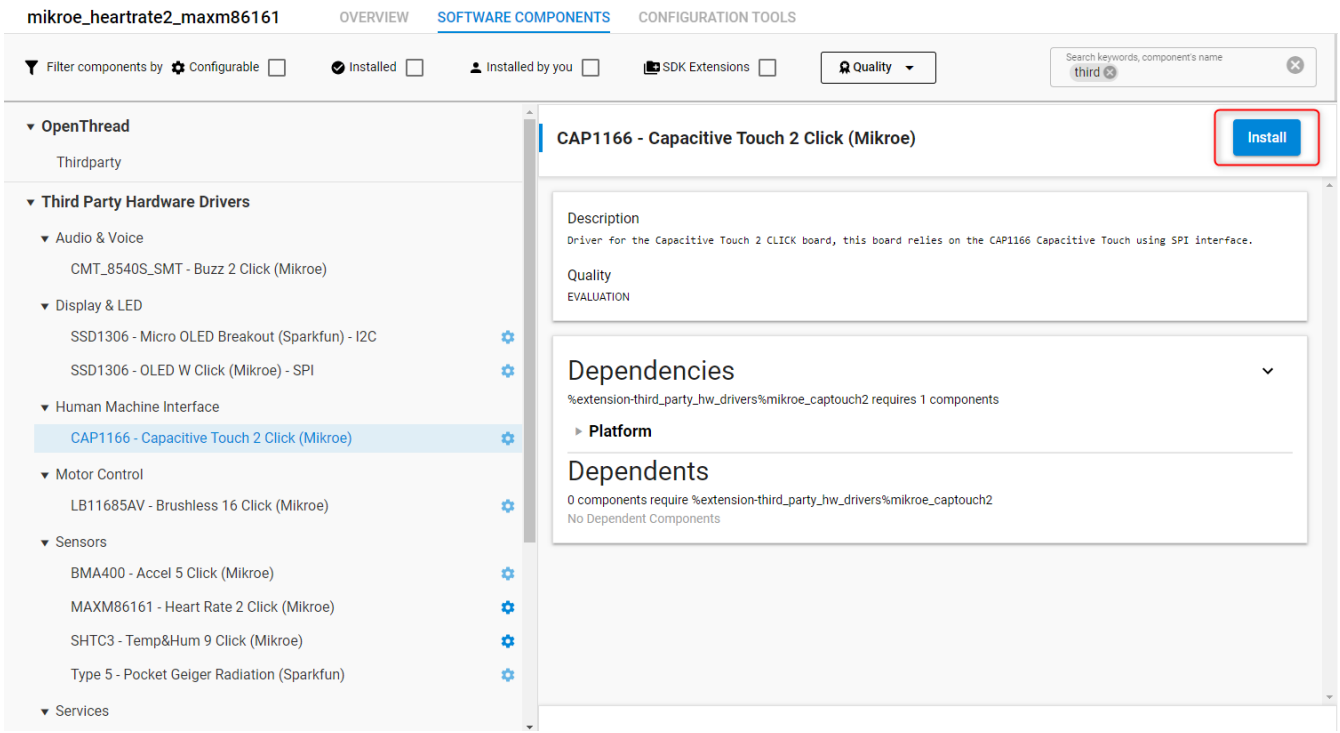
Software Components

- **STEP 1** Open a project configuration (the selected perspective should be "Simplicity IDE").
- **STEP 2** Select **SOFTWARE COMPONENTS** and search for the keyword "third" in the Search keywords, component's name input

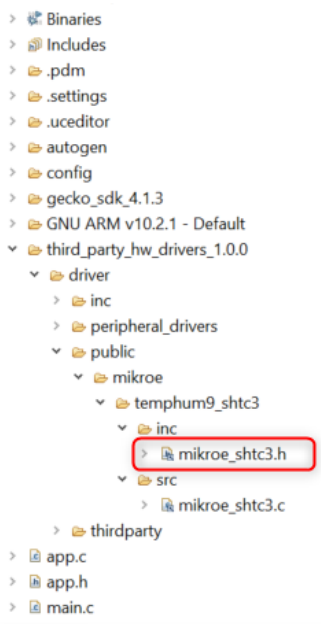
> Ensure that the components with **Evaluation** quality level are enabled in the Software Components view.



- STEP 3 Select a driver from the list by clicking on it, click on the **Install**



The selected hardware driver is installed in your project. The installed driver's API can be found in the extension's public folder.



Public header files should be included in your application for each installed driver.

For further information on how to use drivers in your project, see application example templates in the Launcher.

Application Examples

Code Examples

The appplication examples are located on Github.

EFM32 and EFR32 Application Examples

https://github.com/SiliconLabs/application_examples

Table of contents

TOC by Hardware Driver Class

Table of Contents by Hardware Driver Class

Hardware drivers are provided for popular third-party expansion boards such as:

1. SparkFun QWIIC
2. MikroE Click
3. Adafruit

Name	HW Driver Class	Third-party board	Application Example
Audio DAC Driver	Audio	Adafruit I2S Stereo Decoder	
Magnetic Buzzer Driver	Audio	Mikroe BUZZ 2 click	Air Quality Monitor application with BLE
Battery Fuel Gauge MAX17048	Battery Monitor	Maxim MAX17048XEVKIT	
Ir Array AMG8833 Driver	Camera	Sparkfun Grid EYE Infrared Array	
MLX90640 Far Infrared Sensor Driver	Camera	Sparkfun MLX90640 IR Array	
eInk	Display		
7-Segment LED driver	Display	Mikroe UT M 7 SEG R CLICK	Explorer Kit Bluetooth example using the I2C-bus Joystick and the SPI-bus 7-segment LED display
SparkFun Micro OLED Breakout (Qwiic) Driver	Display	Sparkfun Micro OLED Breakout	Door lock example application with BLE
OLED W Click Driver	Display	Sparkfun OLED W Click	
CAP1166 Capacitive touch driver	Human Interface	Mikroe CAP TOUCH 2 CLICK	Door lock example application with BLE
Joystick driver	Human Interface	Sparkfun Qwiic Joystick	Explorer Kit Bluetooth example using the I2C-bus Joystick and the SPI-bus 7-segment LED display
Key Pad Driver	Human Interface		BLE IR Generator Example
DC motor driver	Motor Control	Mikroe DC MOTOR 3 Click	
Stepper Motor Driver	Motor Control	Mikroe STEPPER 2 CLICK	
BG96 cellular module driver	Network	Mikroe LTE IOT 2 CLICK	Bluetooth Cellular Gateway with BG 96B
W5500 Ethernet Module	Network	Mikroe ETH WIZ Click	BLE Ethernet Gateway
GPS Driver	Other	Mikroe GPS Click	
IR Generator Driver	Other		BLE IR Generator Example
NT3H2x11 Driver	Other		Bluetooth NFC Pairing with NT3H2x11
PN71x0 NCI NFC Controller Driver	Other	Mikroe NFC CLICK	
RFID Driver	Other	Sparkfun RFID Reader	

Name	HW Driver Class	Third-party board	Application Example
ID12LA RFID Reader Driver	Other	Sparkfun RFID Reader	
External Storage - SD card driver	Other	Mikroe MicroSD Click	Environment humidity and temperature data logger with BLE
TRIACDRV	Other		
Accelerometer BMA400 driver	Sensor	Mikroe ACCEL 5 CLICK	Movement Detection application with BLE
Accelerometer MMA8452Q driver	Sensor	Sparkfun Triple Axis Accelerometer	
Barometer driver	Sensor	Adafruit DPS310 Sparkfun MS5637 Mikroe Pressure 3 Click	
Biometric Driver	Sensor	Mikroe HEART RATE 2 CLICK	Bluetooth Module Explorer Kit HRM/SpO2 Software Demo using MAXM86161 sensor
Barometer driver	Sensor	Mikroe ACCEL 5 CLICK Sparkfun Pressure Sensor	Explorer Kit Bluetooth accelerometer example using I2C bus BMA400 accelerometer
BME280 CCS811 Qwiic driver	Sensor	Sparkfun Environmental Combo Breakout	
VL53L1X Distance Sensor Driver	Sensor	Sparkfun Distance Sensor Breakout	Distance Monitor example application using VL53L1X distance sensor and BLE Mesh Room Monitor application People counting application with BLE
MLX90632 IrThermo 3 click Driver	Sensor	Mikroe IRTHERMO 3 CLICK	MLX90632 IrThermo 3 click Bluetooth example
Human Presence AK9753 Driver	Sensor	Sparkfun Human Presence AK9753	
SHTC3 Humidity Sensor Driver	Sensor	Sparkfun Humidity Sensor SHTC3	
PIR Sensor Driver	Sensor		Z-Wave Motion Sensor PIR Example
VCNL4040 Proximity Sensor Driver	Sensor	Sparkfun Proximity Sensor Breakout	
SEN17731 Soil Moisture Sensor Driver	Sensor	Sparkfun Qwiic Soil Moisture Sensor	
Triad Spectroscopy Sensor - AS7265x Driver	Sensor	Sparkfun Triad Spectroscopy Sensor	

TOC by Market Segment Application

Table of Contents by Market Segment Application

Name	Application	Market Segment	Wireless
Apple Notification Center Service	Access Notifications	Home	BLE
NT3H2111 and NT3H2211	Communication	Commercial	BLE
RFID Card Scan over BLE	Communication	Industrial	BLE
Secure Device Attestation and Application Layer Encryption	Communication	Retail	BLE
Secure SPP (Serial Port Profile) over BLE	Communication	Retail	BLE
SPP (Serial Port Profile) over BLE	Communication	Retail	BLE
BLE SPP with Windows	Communication	Retail	BLE
Distance Monitor example application using VL53L1X distance sensor and BLE	Distance Measurment	Commercial	BLE
BLE Ethernet Gateway	Embedded to gateway	Home	BLE
BLE HID Keyboard	Embedded to gateway	Commercial	BLE
Controlling LEDs from a Smartphone	Example Code	Home	BLE
Environment humidity and temperature data logger with BLE	Example Code	Home	BLE
BLE IR Generator Example	Example Code	Retail	BLE
BLE man-in-the-middle example	Example Code	Home	BLE
MIDI over BLE	Example Code	Home	BLE
Multi-Slave Multi-Master Dual-Topology Example	Example Code	Industrial	BLE
Bluetooth - RSSI based room finding	Example Code	Home	BLE
Using EM4 Energy Mode in Bluetooth Beacon App	Example Code	Commercial	BLE
Implementing Wireless Direct Test Mode (DTM)	Example Code	Home	BLE
Optimization on the EM2 Current Consumption of the DynamicMultiprotocolLightSed Example Project	Example Code	Retail	Zigbee
Log System	Example Code	Retail	BLE
NCI PN71x0 T2T Read	Example Code	Home	NFC
NCI PN71x0 T2T Write	Example Code	Home	NFC
NT3H2x11 Field Detection	Example Code	Home	NFC
NT3H2x11 Format T2T	Example Code	Home	NFC
NT3H2x11 I2C Read Tag	Example Code	Home	NFC
NT3H2x11 I2C Write Tag NDEF	Example Code	Home	NFC
Connect Multi-Region Sensor	Example Code	Industrial	Proprietary Connect

Name	Application	Market Segment	Wireless
Adding Custom CLI	Example Code	Industrial	Proprietary Rail
How to optimize the current consumption of Long Preamble Duty Cycle application on FG14	Example Code	Industrial	Proprietary Rail
How to Test Sensitivity of Long Range DSSS PHY on E4432B	Example Code	Industrial	Proprietary Rail
RAIL Using Token on SDKv3.x	Example Code	Industrial	Proprietary Rail
Thermometer Example with EFR32 Internal Temperature Sensor	Example Code	Retail	BLE
Zigbee Battery Monitor Example	Example Code	Industrial	Zigbee
Who is this boot camp for?	Example Code		Zigbee
Green Power gateway example	Example Code	Home	Zigbee
Large Network Testing guidelines	Example Code	Industrial	Zigbee
Manufacturing Library Extension	Example Code	Retail	Zigbee
Zigbee Network Testing Plugin	Example Code	Industrial	Zigbee
Optimize Boot-up Rejoin	Example Code	Retail	Zigbee
Zigbee RTC Time	Example Code	Industrial	Zigbee
Companion example for KBA - A reliable way for SED to receive asynchronous transmissions from other devices without frequent polling	Example Code	Home	Zigbee
RHT Si7021 Zigbee Sleepy End-Device and Gateway example	Example Code	Home	Zigbee
Optimization on EM2 Current Consumption of the Sleepy Z3Switch Example Project in EmberZNet 7.0	Example Code	Home	Zigbee
Zigbee Source Routing Repair Plugin	Example Code	Home	Zigbee
Switching Between Two Zigbee Applications	Example Code	Home	Zigbee
Switching Between Two Zigbee Applications Using Slot Manager	Example Code	Home	Zigbee
ZigBee Virtual UART example	Example Code	Industrial	Zigbee
Z-Wave Ambient Light Sensor Example	Example Code	Home	ZWave
Contact Sensor	Example Code	Home	ZWave
Z-Wave Gesture Sensor Wall Controller Example	Example Code	Home	ZWave

Name	Application	Market Segment	Wireless
Air Quality Monitor application with BLE	Medical Device	Home	BLE
Health Care - Blood Glucose Meters	Medical Device	Commercial	BLE
Bluetooth Cellular Gateway with BG 96B	Medical Device	Commercial	BLE
Health Care - **Continuous Glucose Monitoring**	Medical Device	Commercial	BLE
Bluetooth Module Explorer Kit HRM/SpO2 Software Demo using MAXM86161 sensor and OLED display	Medical Device	Commercial	BLE
Bluetooth Module Explorer Kit HRM/SpO2 Software Demo using MAXM86161 sensor	Medical Device	Commercial	BLE
Mesh Room Monitor application	Mesh	Home	Bluetooth Mesh
btmesh_data_log	Mesh	Home	Bluetooth Mesh
Deprecation Notice		BLE	
Implementing OTA Firmware Update in User Application	OTA	Industrial	BLE
Uploading Images to Internal/External Flash Using OTA DFU	OTA	Industrial	BLE
Door lock example application with BLE	Security	Home	BLE
Bluetooth Explorer Kit accelerometer example using BMA400 sensor with SPI bus	Sensor Integration	Retail	BLE
Explorer Kit Bluetooth accelerometer example using I2C bus BMA400 accelerometer	Sensor Integration	Retail	BLE
Explorer Kit Bluetooth barometer example using I2C bus DPS310 pressure sensor	Sensor Integration	Retail	BLE
Explorer Kit Bluetooth example using the I2C-bus Joystick and the SPI-bus 7-segment LED display	Sensor Integration	Retail	BLE
MLX90632 IrThermo 3 click Bluetooth example	Sensor Integration	Retail	BLE
Z-Wave Motion Sensor PIR Example	Sensor Integration	Home	ZWave
Movement Detection application with BLE	Smart Home	Home	BLE
People counting application with BLE	Smart Home	Home	BLE
Reporting Battery Voltage over BLE	Smart Home	Home	BLE
Tutorial Overview	Smart Home	Home	Zigbee
Zigbee Motion Sensor PIR Example	Smart Home	Home	Zigbee
Zigbee Smart Lighting with PIR and Ambient light sensor	Smart Home	Home	Zigbee

TOC by Wireless Use Case

Table of Contents by Wireless Use Case

Name	Embedded to Embedded	Embedded to Gateway	Embedded to Mobile	Wireless
Adding Custom CLI				Proprietary Rail
Air Quality Monitor application with BLE			✔	BLE
Apple Notification Center Service			✔	BLE
BLE Ethernet Gateway		✔		BLE
BLE HID Keyboard		✔		BLE
BLE IR Generator Example			✔	BLE
BLE man-in-the-middle example	✔			BLE
BLE SPP with Windows		✔		BLE
Bluetooth - RSSI based room finding		✔	✔	BLE
Bluetooth Cellular Gateway with BG 96B		✔		BLE
Bluetooth Explorer Kit accelerometer example using BMA400 sensor with SPI bus			✔	BLE
Bluetooth Module Explorer Kit HRM/SpO2 Software Demo using MAXM86161 sensor			✔	BLE
Bluetooth Module Explorer Kit HRM/SpO2 Software Demo using MAXM86161 sensor and OLED display			✔	BLE
btmesh_data_log	✔			Bluetooth Mesh
Companion example for KBA - A reliable way for SED to receive asynchronous transmissions from other devices without frequent polling	✔			Zigbee
Connect Multi-Region Sensor		✔		Proprietary Connect
Contact Sensor				ZWave
Controlling LEDs from a Smartphone			✔	BLE
Deprecation Notice				BLE
Distance Monitor example application using VL53L1X distance sensor and BLE			✔	BLE
Door lock example application with BLE			✔	BLE
Environment humidity and temperature data logger with BLE	✔		✔	BLE
Explorer Kit Bluetooth accelerometer example using I2C bus BMA400 accelerometer			✔	BLE

Name	Embedded to Embedded	Embedded to Gateway	Embedded to Mobile	Wireless
Explorer Kit Bluetooth barometer example using I2C bus DPS310 pressure sensor			✔	BLE
Explorer Kit Bluetooth example using the I2C-bus Joystick and the SPI-bus 7-segment LED display			✔	BLE
Green Power gateway example		✔		Zigbee
Health Care - Blood Glucose Meters			✔	BLE
Health Care - **Continuous Glucose Monitoring**			✔	BLE
How to optimize the current consumption of Long Preamble Duty Cycle application on FG14				Proprietary Rail
How to Test Sensitivity of Long Range DSSS PHY on E4432B				Proprietary Rail
Implementing OTA Firmware Update in User Application			✔	BLE
Implementing Wireless Direct Test Mode (DTM)	✔		✔	BLE
Large Network Testing guidelines				Zigbee
Log System	✔		✔	BLE
Manufacturing Library Extension	✔			Zigbee
Mesh Room Monitor application	✔		✔	Bluetooth Mesh
MIDI over BLE			✔	BLE
MLX90632 IrThermo 3 click Bluetooth example			✔	BLE
Movement Detection application with BLE			✔	BLE
Multi-Slave Multi-Master Dual-Topology Example	✔		✔	BLE
NCI PN71x0 T2T Read	✔			NFC
NCI PN71x0 T2T Write	✔			NFC
NT3H2111 and NT3H2211			✔	BLE
NT3H2x11 Field Detection	✔			NFC
NT3H2x11 Format T2T	✔			NFC
NT3H2x11 I2C Read Tag	✔			NFC
NT3H2x11 I2C Write Tag NDEF	✔			NFC
Optimization on EM2 Current Consumption of the Sleepy Z3Switch Example Project in EmberZNet 7.0	✔			Zigbee
Optimization on the EM2 Current Consumption of the DynamicMultiprotocolLightSed Example Project	✔			Zigbee
Optimize Boot-up Rejoin	✔			Zigbee
People counting application with BLE			✔	BLE
RAIL Using Token on SDKv3.x				Proprietary Rail
Reporting Battery Voltage over BLE			✔	BLE
RFID Card Scan over BLE			✔	BLE
RHT Si7021 Zigbee Sleepy End-Device and Gateway example		✔		Zigbee
Secure Device Attestation and Application Layer Encryption	✔			BLE

Name	Embedded to Embedded	Embedded to Gateway	Embedded to Mobile	Wireless
Secure SPP (Serial Port Profile) over BLE		✔		BLE
SPP (Serial Port Profile) over BLE		✔		BLE
Switching Between Two Zigbee Applications	✔			Zigbee
Switching Between Two Zigbee Applications Using Slot Manager	✔			Zigbee
Thermometer Example with EFR32 Internal Temperature Sensor			✔	BLE
Tutorial Overview				Zigbee
Uploading Images to Internal/External Flash Using OTA DFU			✔	BLE
Using EM4 Energy Mode in Bluetooth Beacon App				BLE
Who is this boot camp for?				Zigbee
Z-Wave Ambient Light Sensor Example				ZWave
Z-Wave Gesture Sensor Wall Controller Example				ZWave
Z-Wave Motion Sensor PIR Example				ZWave
Zigbee Battery Monitor Example	✔			Zigbee
Zigbee Motion Sensor PIR Example				Zigbee
Zigbee Network Testing Plugin	✔			Zigbee
Zigbee RTC Time	✔			Zigbee
Zigbee Smart Lighting with PIR and Ambient light sensor				Zigbee
Zigbee Source Routing Repair Plugin	✔			Zigbee
ZigBee Virtual UART example	✔			Zigbee

SDK Extensions

SDK Extensions

Third-Party Hardware Drivers

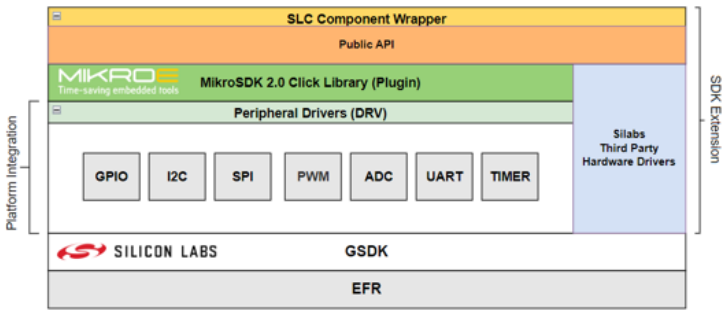
- Third-Party Hardware Drivers

Third Party Hardware Drivers

Third-Party Hardware Drivers

Third-Party Hardware Drivers are GSDK Extension to provide support for third-party external hardware.

- **Scaling GSDK functionality with SDK Extension**
  - One-click solution for tested third-party sensor boards
  - Developed wrapper can be used to add untested boards easily
- **Accelerate Design Phase**
  - Quick and easy integration of 1,100+ devices manufactured by different board providers
  - 10x faster than developing individual HW drivers from scratch
- **Customer Self-Serve Support**
  - Easy to start, fast to learn, time-saving
  - Based on third-party boards, diverse applications can be created



This extension consumes the mikroSDK Click Plugin for the mikroSDK developed by Mikroe.

See the instructions of the Getting Started section for more information.

Audio & Voice

- CMT-8540S-SMT - Buzz 2 Click (Mikroe)
- MIC 2 Click (Mikroe)

Display & LED

- E-Paper display 1.54" 200x200 dots (Mikroe)
- ILI9341 - TFT LCD Display (Adafruit) - SPI
- ILI9341 - TFT LCD Display (Adafruit) - SPI with DMA
- MAX6969 - UT-M 7-SEG R Click (Mikroe)
- SSD1306 - Micro OLED Breakout (Sparkfun) - I2C
- SSD1306 - OLED W Click (Mikroe) - SPI
- SSD1351 - OLED C Click (Mikroe)

Human Machine Interface

- A-172-MRQ - Fingerprint 2 Click (Mikroe)
- CAP1166 - Capacitive Touch 2 Click (Mikroe)
- Qwiic Joystick (Sparkfun)
- Touch Screen (Analog)

Interface

- STN1110 - OBDII Click (Mikroe)
- W5500 - ETH WIZ Click (Mikroe)

Miscellaneous

- IR Generator (Silabs)
- LCA717 - Relay 2 Click (Mikroe)
- RNG Click (Mikroe)
- Triac Driver (Silabs)

Motor Control

- A3967 - Stepper Click (Mikroe)
- A4988 - Stepper 2 Click (Mikroe)
- LB11685AV - Brushless 16 Click (Mikroe)
- PCA9685 - Servo Click (Mikroe)
- Si8711CC - PWM Driver Click (Mikroe)
- TB6549FG - DC Motor 3 Click (Mikroe)

Power Management

- MAX17048 - MAX17048EVKIT Evaluation Kits (Maxim)

Sensors

- AD8318 - RF Meter Click (Mikroe)
- AK9753 - Human Presence Sensor (Sparkfun) - I2C
- AMG88XX - Grid-EYE Infrared Array Breakout (Sparkfun)
- AS3935 - Thunder Click (Mikroe)
- AS7265x - Triad Spectroscopy Sensor (Sparkfun) - I2C
- BMA400 - Accel 5 Click (Mikroe) - I2C
- BMA400 - Accel 5 Click (Mikroe) - SPI
- BME280 - Atmospheric Sensor (Sparkfun)
- BME688 - Environment 3 Click (Mikroe) - I2C
- BME688 - Environment 3 Click (Mikroe) - SPI
- CCS811 - Air Quality Sensor (Sparkfun)
- DPS310 - Pressure 3 Click (Mikroe) - I2C
- DPS310 - Pressure 3 Click (Mikroe) - SPI
- EM3080-W - Barcode 2 Click (Mikroe)
- EMG Click (Mikroe)
- FSR400 - Force 3 Click (Mikroe)
- IRA-S210ST01 - PIR Sensor (Silabs)
- MAX30101 & MAX32664 - Pulse Oximeter and Heart Rate Sensor (Sparkfun)
- MAX30101 - Heart Rate 4 Click (Mikroe)
- MAXM86161 - Heart Rate 2 Click (Mikroe)
- MCP606 - Water Detect Click (Mikroe)
- ML8511A - UV Click (Mikroe)
- MLX90632 - IrThermo 3 Click (Mikroe)
- MLX90640 - IR Array Breakout (Sparkfun)
- MM5D91-00 - Radar Click (Mikroe)
- MMA8452Q - Triple Axis Accelerometer Breakout (Sparkfun)
- MQ131 - Ozone 2 Click (Mikroe)
- MQ3 - Alcohol Click (Mikroe)
- MQ7 - CO Click (Mikroe)
- Pocket Geiger Radiation - Type 5 (Sparkfun)
- Qwiic Soil Moisture Sensor (Sparkfun) - I2C
- SGP40 - Air Quality Sensor (Sparkfun)
- SHT40 & SGP40 - Environment 2 Click (Mikroe)

- SHT40 - Temp&Hum 15 Click (Mikroe)
- SHTC3 - Temp&Hum 9 Click (Mikroe)
- TSD-10 - Turbidity Click (Mikroe)
- VCNL4040 - Proximity Sensor (Sparkfun)
- VL53L1X - Distance Sensor Breakout (Sparkfun)

Services

- BTHome v2
- BTHome v2 - Server
- EnOcean Switch Proxy Server
- FatFS - Generic FAT Filesystem
- GLIB - OLED Graphics Library
- LIN bus slave
- LVGL - Graphics Library
- NFC
  - NFC - Common
  - NFC - NCI
  - NFC - NDEF
  - NFC - Tag
- mikroSDK 2.0 SDK - Peripheral Drivers
  - ADC
  - Digital I/O
  - I2C
  - PWM
  - SPI
  - UART

Storage

- microSD - microSD Click (Mikroe)

Wireless Connectivity

- BG96 - LTE IoT 2 Click (Mikroe)
- ID-12LA - RFID Reader (Sparkfun) - I2C
- LEA-6S - GPS Click (Mikroe)
- NT3H2111 - NFC Tag 2 Click (Mikroe) - I2C
- PN7150 - NFC 2 Click (Mikroe) - I2C

The above drivers are tested and integrated into the extension.

Besides the integrated drivers, it is possible to add additional drivers from the mikroSDK Click Plugin Repository by using the [Services] -> mikroSDK 2.0 SDK - Peripheral Drivers software components.

Software components in the mikroSDK 2.0 SDK - Peripheral Drivers are implemented as the required peripheral driver interfaces for the MikroSDK Click plugin.

In general, the software components are named in accordance with the following naming convention:

<IC_NAME> - <BOARD_NAME> (<BOARD_VENDOR>) - <INTERFACE>

This includes:

- **IC_NAME** - The name of the integrated circuit on the external board. (e.g.,: SSD1306)
- **BOARD_NAME** - The name of the external board. (OLED W Click)
- **BOARD_VENDOR** - External board vendor. (e.g.,: Mikroe, Sparkfun, Adafruit, etc.)
- **INTERFACE** - Optional parameter to indicate the communication interface in case the SDK extension implements multiple drivers for the same device with different interfaces. (e.g.,: SPI, I2C)

Although, the drivers were mainly developed and tested with specific external boards, in most cases they should work with other boards using the same IC as well.

Example

SSD1306 - Micro OLED Breakout (Sparkfun) - I2C driver was developed and tested with Sparkfun Micro OLED Breakout board. However, it may be compatible with most OLED displays available on the market, which are controlled by the SSD1306 display controller. To achieve compatibility, changes to the I2C address or display resolution in the configuration by the display board may be required.

Integrate New mikroSDK 2.0 Click Drivers

The Third-Party Hardware Drivers extension provides one-click solution for tested hardware drivers allowing you to integrate 30+ hardware drivers into your project with ease.

In addition to the tested hardware drivers, the extension also provides a peripheral driver wrapper to easily connect the mikroSDK 2.0 Click drivers with the Silicon Labs GSDK.

If you are not afraid to do some extra development, thanks to the developed wrapper, over 1,100+ hardware drivers can be added to your project from the mikroSDK 2.0 Click library. This will accelerate the design phase and provide you with a greater level of customer self-serve support.

This chapter is aimed to guide you in integrating a hardware driver from the mikroSDK 2.0 Click library using the wrappers from the *mikroSDK 2.0 SDK - Peripheral Drivers* components.

Currently, the following peripherals are supported:

- ADC
- Digital I/O
- I2C
- PWM
- SPI
- UART

In general, drivers in the mikroSDK 2.0 Click library provide interfaces for initializing and configuring the drivers:

- **<driver_name>_cfg_setup** function
- **<driver_name>_init** function

Configuration

Configuration interfaces are used to configure the peripheral-related configuration parameters, such as pin assignments, speed, and address values for the communication interface.

The required configuration parameters are defined in the configuration structure of the drivers. (**<driver_name>_cfg_t** structure)

Pin configuration and other interface specific settings are provided by the GSDK via the component instances.

In general, you should invoke the <driver_name>_cfg_setup function and there is no need to configure the parameters available in the configuration structure.

Initialization

Interfaces for initializing the drivers require a click context object and a click driver configuration object to perform the correct initialization of the driver.

Click context objects typically contain an interface for the peripheral driver.

The peripheral interface objects provide a handler for the platform-dependent peripheral driver and also provide variables to store the device-specific parameters, such as the address of the device for I2C interfaced hardware.

This handler must be configured before calling the <driver_name>_init function.

The approach and steps to integrate drivers for devices using different interfaces are similar. You can find examples in the table below.

Examples of integrating drivers using different interfaces

Interface	Circuit	Click Board	Example
ADC	MQ-7	CO Click	Link
I2C	SHTC3	Temp-Hum 9 Click	Link
PWM	CMT-8540S-SMT	Buzz 2 Click	Link
SPI	W5500	ETH WIZ Click	Link
UART	EM3080W	Barcode 2 Click	Link

In the following section, you can find a detailed guideline for integrating a driver from the mikroSDK 2.0 Click library using the I2C interface.

Basic Integration Steps

- **STEP 0** Select and download a driver from the mikroSDK 2.0 Click library.
  Make sure that the selected driver is using a supported communication interface. See the supported interfaces above.
- **STEP 1.1 - Optional** Add the Third-Party Hardware Drivers extension. See the instructions in detail here.
- **STEP 1.2 - Optional** Connect your board to the PC via a USB cable.
- **STEP 1.3** Open the **Launcher** perspective in Simplicity Studio, and select the target board.
- **STEP 2** Create a new empty project using an empty project template. (e.g., **Empty C Project**)
- **STEP 3** Install the required mikroSDK 2.0 Peripheral Driver components from the Third-Party Hardware Drivers extension. (Note: If the selected board uses I2C to communicate with the host controller, then install the [Third Party Hardware Drivers] -> [Services] -> [mikroSDK 2.0 SDK - Peripheral Drivers] -> I2C component.)
- **STEP 4** Install additional components your project requires. (e.g., Log, Assert, etc.)
  Default I2CSPM instance is "mikroe", make sure that your I2CSPM instance is configured properly for the target board.
- **STEP 5** Copy and paste the new driver into the projects folder.
- **STEP 6** Add the <driver_folder>/lib/include folder to the list of include directories in the project configuration.
- **STEP 7** Exclude the <driver_folder>/example/ *.c files from the build.

At this point the new project is ready to integrate the new driver.

- **STEP 8** Add objects for the click context and click configuration as global variables.
- **STEP 8.1** Create a custom init function for your new driver.
- **STEP 8.2** Set/configure the peripheral driver handler in the click context object. This handler should point to an existing peripheral instance.
- **STEP 8.3** Invoke the <driver_name>_cfg_setup function from your custom init function.
- **STEP 8.4** Invoke the <driver_name>_init function from your custom init function.
- **STEP 9** Invoke your custom init function from app_init.
- **STEP 10** Implement your application logic. (Integrate the driver APIs.)
- **STEP 11** Build the project.
- **STEP 12** Test the application on a target device.

Example - Integrate the SHTC3 Temperature and Humidity sensor driver

Select and download a driver from the mikroSDK 2.0 Click Library

- **STEP 0** Select and download the temphum9 driver.

Create a new project

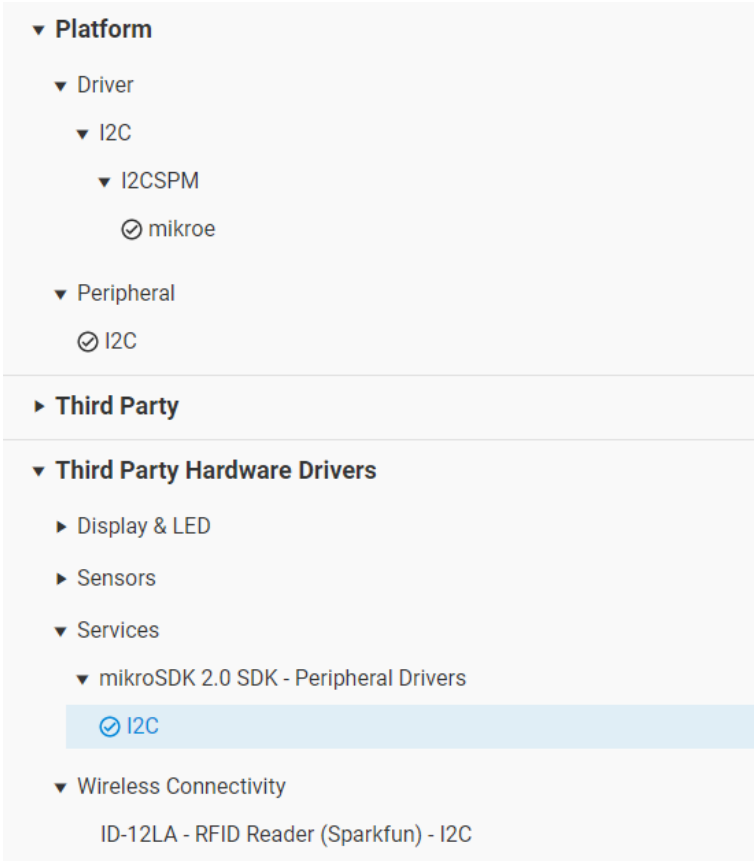
- **STEP 1** Open the **Simplicity Studio**
- **STEP 2** Add the **Third-Party Hardware Drivers extension**, see details here.
- **STEP 3** Connect your board to the PC via a USB cable
- **STEP 4** Open the **Launcher** perspective in Simplicity Studio, select the target board
- **STEP 5** Select the **EXAMPLE PROJECTS & DEMOS** tab in the launcher view
- **STEP 6** Select an empty template project (e.g., **Empty C Project**), click on the **Create** button
- **STEP 7** Give a name for the new project and click on the **Finish** button

Add required peripheral drivers from the TPHD extension

The SHTC3 Sensor has an I2C interface to communicate with the host microcontroller. You should check the required interface(s) needed by the external hardware you want to integrate the driver for.

Notes	Pin	mikroBUS				Pin	Notes
	NC	1	AN	PWM	16	NC	
	NC	2	RST	INT	15	NC	
	NC	3	CS	RX	14	NC	
	NC	4	SCK	TX	13	NC	
	NC	5	MISO	SCL	12	**SCL**	I2C Clock
	NC	6	MOSI	SDA	11	**SDA**	I2C Data
Power Supply	**+3V3**	7	3.3V	5V	10	NC	
Ground	**GND**	8	GND	GND	9	**GND**	Ground

- **STEP 1** Open the project configuration by double clicking on the *.slcp file in the project's folder.**
- **STEP 2** Select the software components tab in the project configuration view.
- **STEP 3** Enable the extension and clear the quality filters.
- **STEP 4** Install the I2C wrapper from [Third Party Hardware Drivers] -> [Services] -> [mikroSDK 2.0 SDK - Peripheral Drivers] -> I2C.

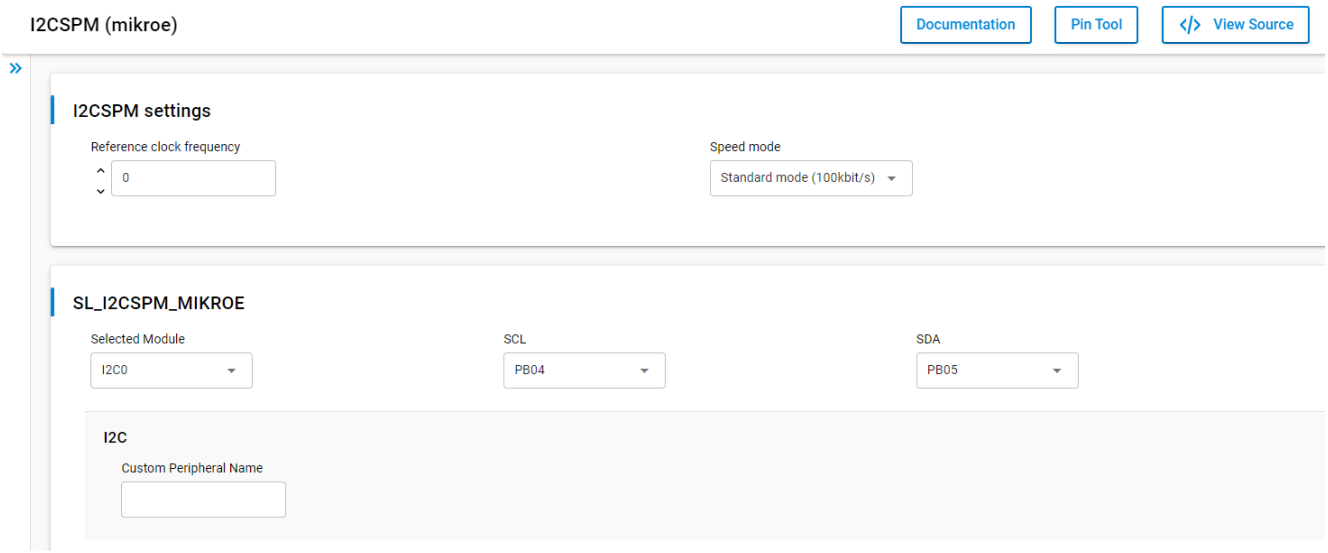


- **STEP 5** Install Log, and Sleep Timer components
  - [Application] -> [Utility] -> Log

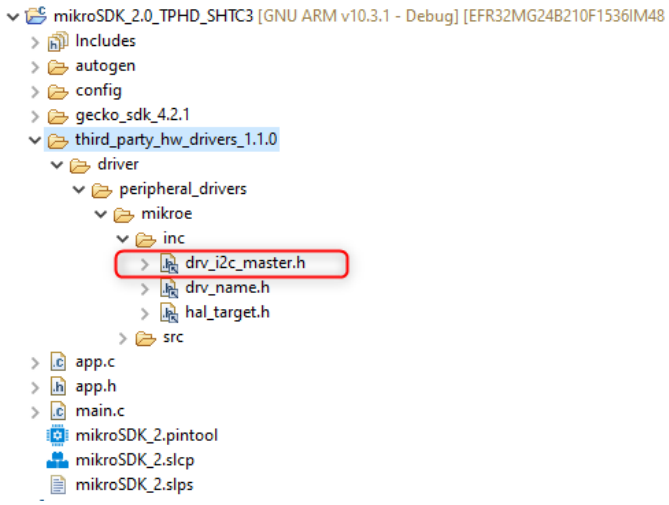
- [Services] -> [Timers] -> Sleep Timer

The default I2CSPM instance is "mikroe". Make sure that your I2CSPM instance is configured properly for the target board.

See an example configuration for the EFR32xG24 Explorer Kit below.



Once the I2C software component is installed, the header and source files provided by this component will be available in the project's file structure.

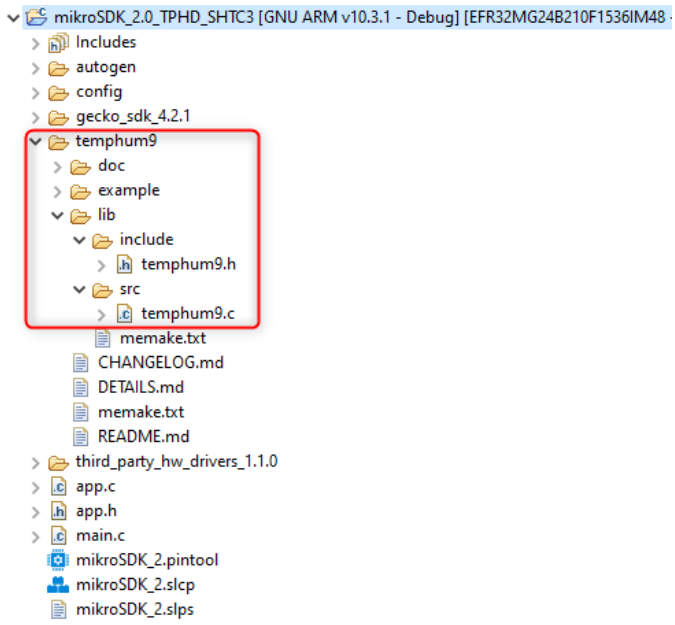


The driver will use the I2C peripheral interfaces provided by the **drv_i2c_master.h** header file in the background.

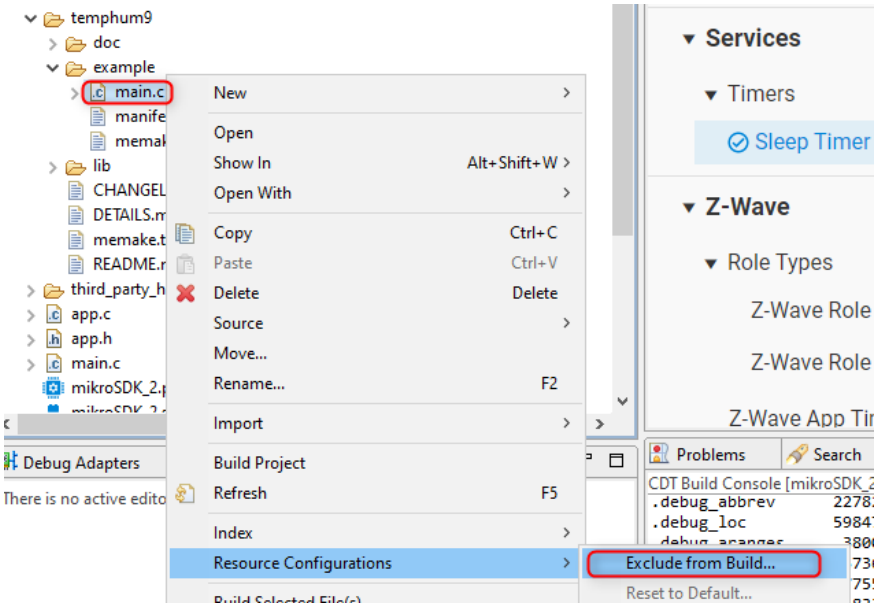
Add driver source files to the project

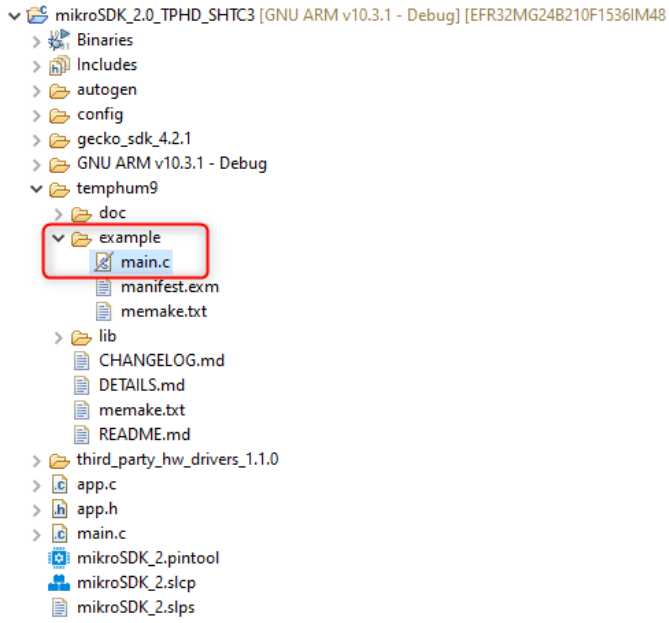
Download the driver source files from the mikroSDK 2.0 Click library.

- **STEP 1** Copy and paste the driver's folder containing the source files for the selected driver

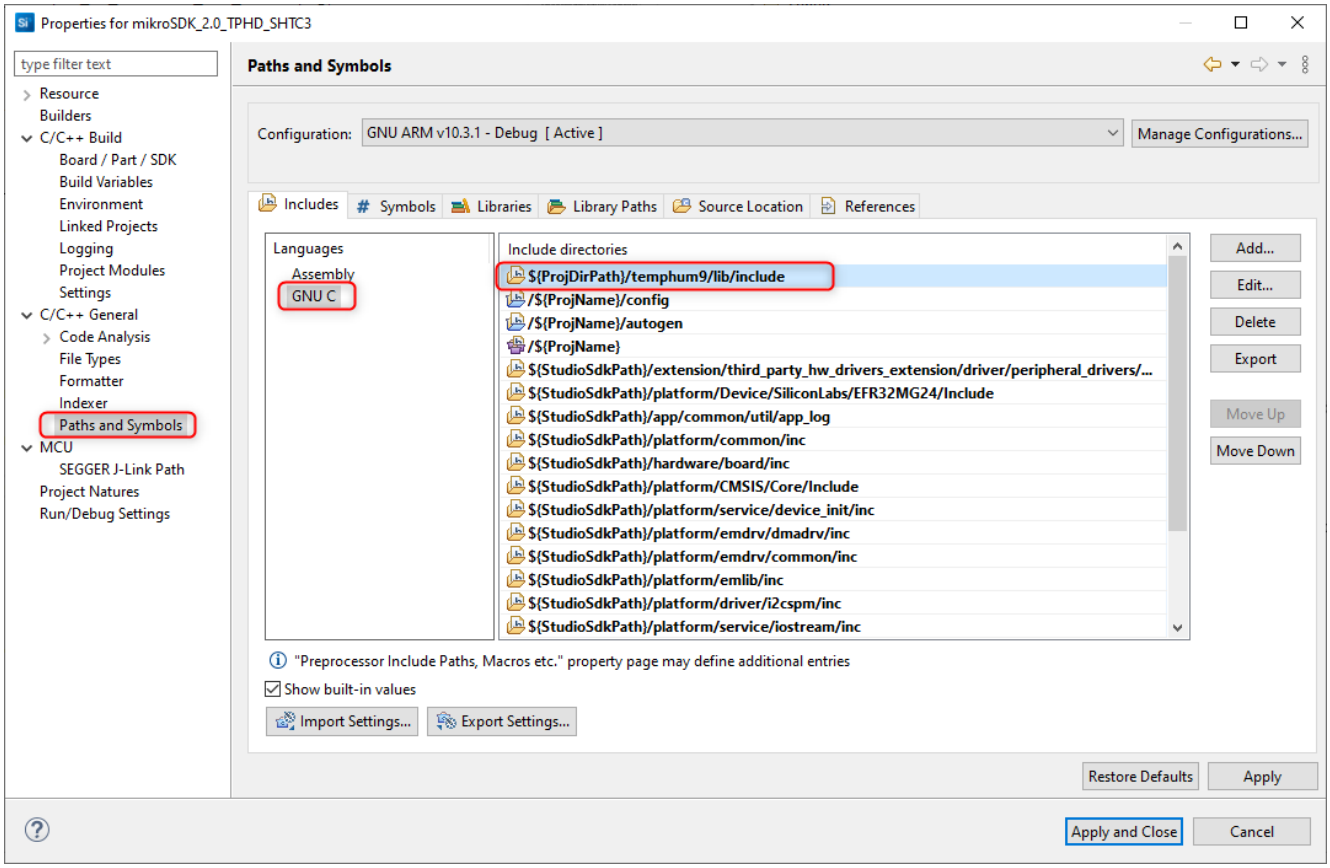


- STEP 2 Exclude the main.c and other *.c files from the temphum9/lib/example folder

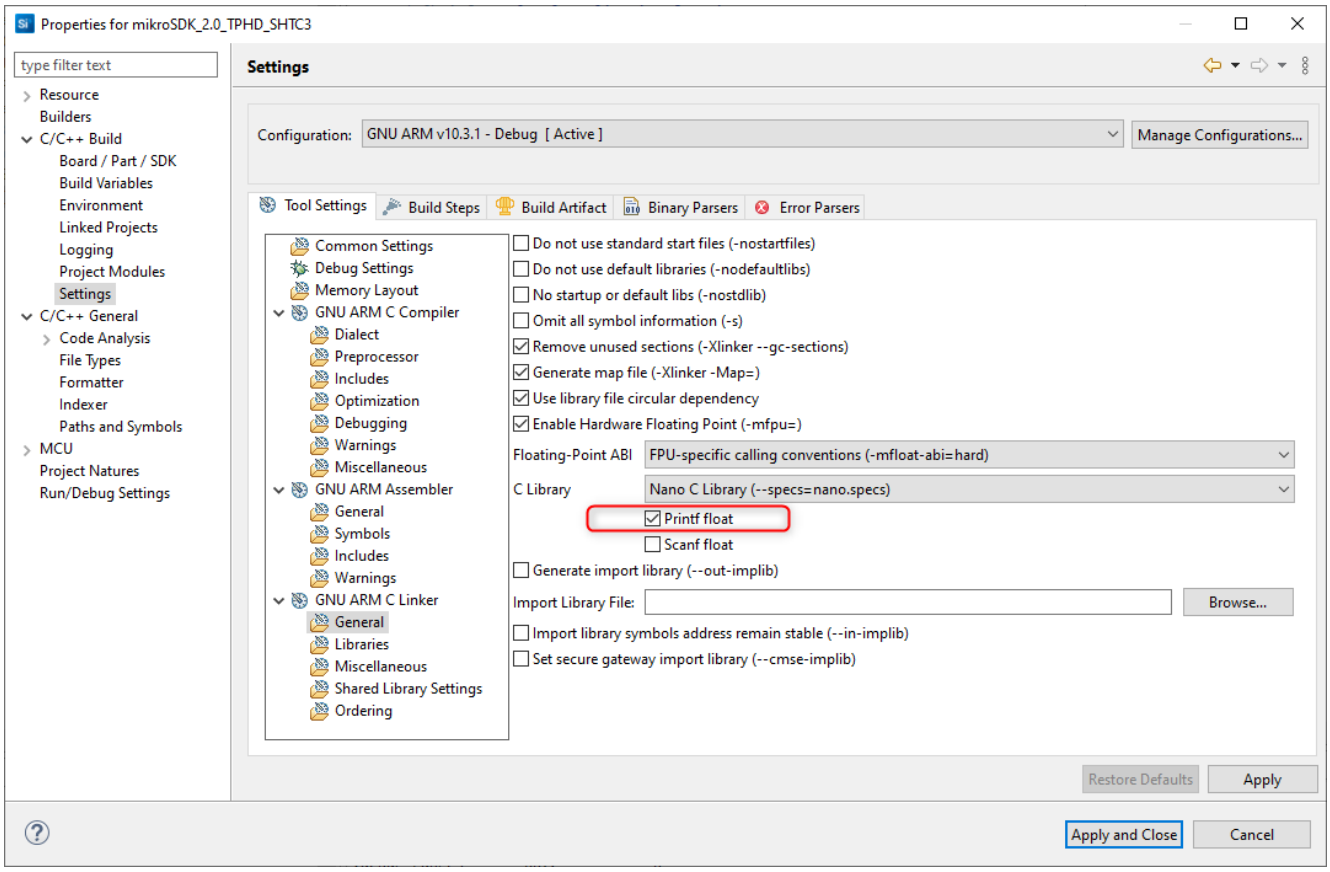




- STEP 3 Append the temphum9/lib/include folder to the list of the include directories



- STEP 4 Enable printf for floating point numbers



Integrate the driver

- STEP 1 Open the app.c file
- STEP 2 Create a custom init function for the driver and add the required driver and driver config instances to the project.

```
#include "app_log.h"
#include "sl_status.h"
#include "sl_i2cspm_instances.h"
#include "sl_sleeptimer.h"

#include "temphum9.h"

static temphum9_t temphum9;
static temphum9_cfg_t temphum9_cfg;
static sl_sleeptimer_timer_handle_t handle_periodic;

sl_status_t mikroe_custom_shtc3_init(sl_i2cspm_t *i2cspm_instance);
void measure_periodic(sl_sleeptimer_timer_handle_t *handle, void *data);

sl_status_t mikroe_custom_shtc3_init(sl_i2cspm_t *i2cspm_instance)
{
  if (NULL == i2cspm_instance) {
    return SL_STATUS_INVALID_PARAMETER;
  }
  // Configure default i2csmp instance
  temphum9.i2c.handle = i2cspm_instance;

  // Call basic setup functions
  temphum9_cfg_setup(&temphum9_cfg);

  return temphum9_init(&temphum9, &temphum9_cfg) ? SL_STATUS_FAIL : SL_STATUS_OK;
}
```

The mikroSDK driver provides the temphum9_t and temphum9_cfg_t types to configure the driver.

```
typedef struct
{
    // Modules

    i2c_master_t i2c;

    // ctx variable

    uint8_t slave_address;

} temphum9_t;

typedef struct
{
    // Communication gpio pins

    pin_name_t scl;
    pin_name_t sda;

    // static variable

    uint32_t i2c_speed;
    uint8_t i2c_address;

} temphum9_cfg_t;
```

Silicon Labs wrapper provides the high level configuration for the I2CSPM instance, therefore it is not required to configure the speed, pin, or any other parameters except the i2c parameter in the temphum9_t type.

Only the i2c.handle pointer should be configured to point to the configured I2CSPM instance.

Please check the provided drivers as examples for other peripheral (SPI, UART, etc.) integration.

Initialization

```
void app_init(void)
{

  if (SL_STATUS_OK != mikroe_custom_shtc3_init(sl_i2cspm_mikroe)) {
    app_log("TempHum9 initialization failed.");
  } else {
    app_log("TempHum9 initialization succeed.");

    sl_sleeptimer_start_periodic_timer_ms(&handle_periodic, 1000,
        measure_periodic, NULL, 0, 0);
  }
}
```

Reading and printing the measured values

```
void measure_periodic(sl_sleeptimer_timer_handle_t *handle, void *data)
{
  (void) handle;
  (void) data;

  float _measurement_data[2];
  temhum9_get_temperature_and_humidity(&temphum9, TEMPHUM9_NORMAL_MODE,
    _measurement_data);
  app_log(">> Temp: %.2f °C RH: %.2f %%\n", _measurement_data[0],
    _measurement_data[1]);
}
```

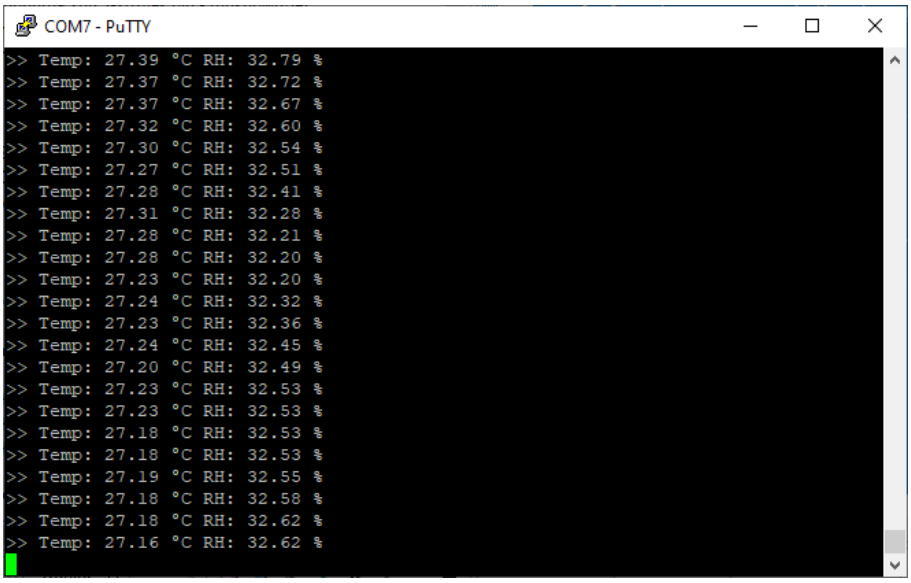
The whole example app.c

```
#include "app_log.h"
#include "sl_status.h"
#include "sl_i2cspm_instances.h"
#include "sl_sleeptimer.h"

#include "temphum9.h"

static temphum9_t temphum9;
static temphum9_cfg_t temphum9_cfg;
static sl_sleeptimer_timer_handle_t handle_periodic;

sl_status_t mikroe_custom_shtc3_init(sl_i2cspm_t *i2cspm_instance);
void measure_periodic(sl_sleeptimer_timer_handle_t *handle, void *data);

sl_status_t mikroe_custom_shtc3_init(sl_i2cspm_t *i2cspm_instance)
{
  if (NULL == i2cspm_instance) {
    return SL_STATUS_INVALID_PARAMETER;
  }
  // Configure default i2csmp instance
  temphum9.i2c.handle = i2cspm_instance;

  // Call basic setup functions
  temphum9_cfg_setup(&temphum9_cfg);

  return temphum9_init(&temphum9, &temphum9_cfg) ? SL_STATUS_FAIL : SL_STATUS_OK;
}

/***************************************************************************//**
 * Initialize application.
 ******************************************************************************/
void app_init(void)
{

  if (SL_STATUS_OK != mikroe_custom_shtc3_init(sl_i2cspm_mikroe)) {
    app_log("TempHum9 initialization failed.");
  } else {
    app_log("TempHum9 initialization succeed.");

    sl_sleeptimer_start_periodic_timer_ms(&handle_periodic, 1000,
        measure_periodic, NULL, 0, 0);
  }
}

void measure_periodic(sl_sleeptimer_timer_handle_t *handle, void *data)
{
  (void) handle;
  (void) data;

  float _measurement_data[2];
  temhum9_get_temperature_and_humidity(&temphum9, TEMPHUM9_NORMAL_MODE,
      _measurement_data);
  app_log(">> Temp: %.2f °C RH: %.2f %%\n", _measurement_data[0],
      _measurement_data[1]);
}

/***************************************************************************//**
 * App ticking function.
 ******************************************************************************/
void app_process_action(void)
{

}
```

- Build and flash the application

If you connect the Temphum9 board to the Explorer Kit, the driver and the demo application should operate properly and you should be able to read the temperature and humidity measurements.

Output



CircuitPython

CircuitPython Getting Started Guide

Introduction

Introduction

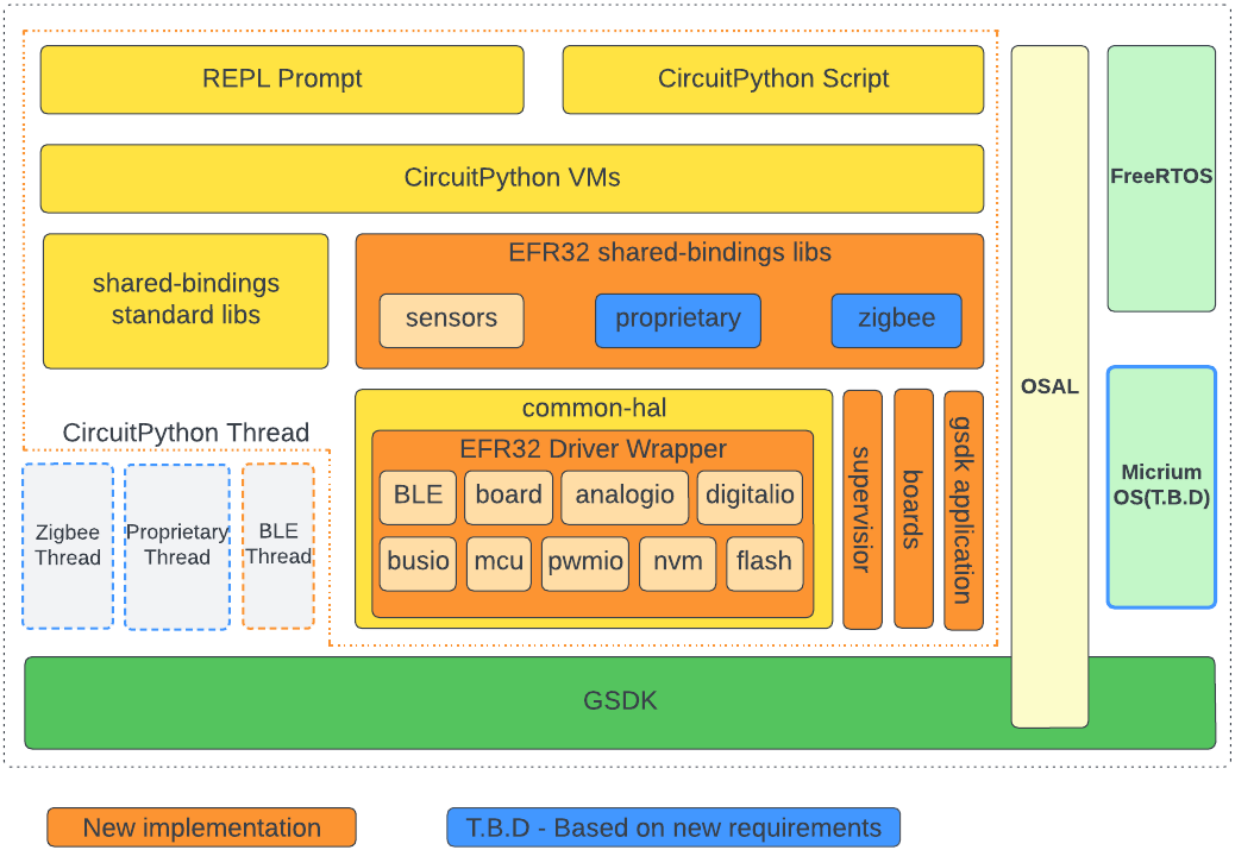
CircuitPython is a programming language designed to simplify experimenting and learning to code on low-cost microcontroller boards. Unlike traditional programming languages, CircuitPython is easy to learn and use, making it a great choice for beginners or anyone who wants to quickly create projects without having to spend a lot of time learning complicated programming concepts.

CircuitPython supports for a wide range of microcontrollers, including those made by popular manufacturers. This allows users to choose the microcontroller that best fits their needs and use CircuitPython to control it.

We have already successfully ported CircuitPython to the Silabs xG24, making this powerful programming language available to developers who want to create innovative and exciting projects using our hardware.

Systems Overview

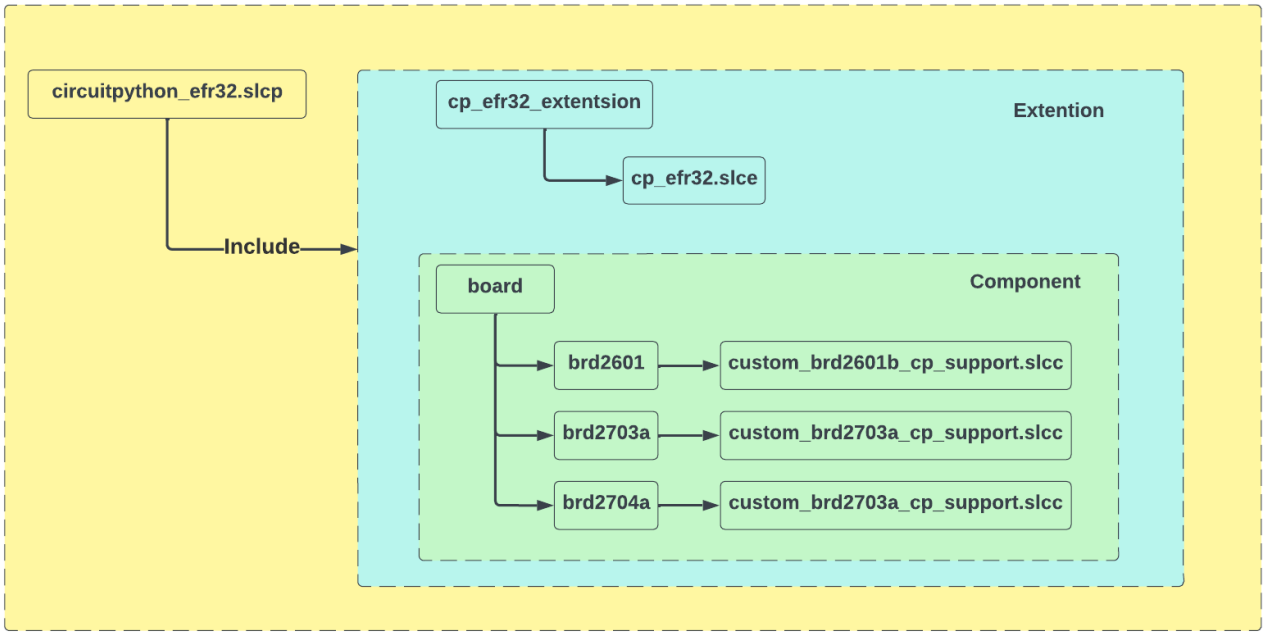
This is an implementation of CircuitPython for the xG24.



On the xG24 chips, CircuitPython is run on a thread in the Dynamic Multiprotocol system. The RTOS can be selected between FreeRTOS or Micrium OS. There are also other threads for Zigbee, BLE, and Proprietary corresponding with protocols. Currently, only BLE is supported.

Supporting protocols requires writing additional custom modules in shared-bindings.

Since the system uses the SLC tool to create the project, configuring the components of the Gecko SDK is quite easy.



The system utilizes the SLC tool to generate the make file. A separate slcc file is required for each board.

Supported Kits

This port provides support for three xG24 boards, which are listed below:

- EFR32xG24 Dev Kit
- EFR32xG24 Explorer Kit
- SparkFun Thing Plus Matter MGM240P

Module Support Matrix

The following lists the available built-in modules for xG24 board, as well as each frozen module included on.

Built-in modules available: _asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, analogio, array, atexit, binascii, bitmaptools, board, builtins, busio, collections, digitalio, displayio, errno, fontio, framebufferio, getpass, gifio, json, math, microcontroller, msgpack, nvm, onewireio, os, pwmio, rainbowio, random, re, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, sys, terminalio, time, traceback, ulab, vectorio, watchdog, zlib

Included frozen modules: adafruit_ble, adafruit_register

Building Firmware

Building CircuitPython Firmware

The CircuitPython port for xG24 is readily accessible through Adafruit's CircuitPython repository.

How this Port is Organized

- **boards/** contains the configuration files for each development board and breakout available on the port, as well as system files and both shared and SoC-specific linker files. Board configuration includes a pin mapping of the board, oscillator information, board-specific build flags, and setup for some other peripheral where applicable.
- **common-hal/** contains the port-specific module implementations, used by shared-module and shared-bindings.
- **cp_efr32_extension/** sdk extension contains a list of paths to search for components
- **gecko_sdk/** Silicon Labs Gecko SDK as submodule
- **peripherals/** contains peripheral setup files and peripheral mapping information, sorted by family and sub-variant. Most files in this directory can be generated with the python scripts in **tools/**.
- **supervisor/** contains port-specific implementations of internal flash and serial, as well as the **port.c** file, which initializes the port at startup.
- **tools/** contains Silicon Labs configurator (SLC) tool, python scripts for generating peripheral and pin mapping files in **peripherals/** and **board/**.

At the root level, refer to **mpconfigboard.h** and **mpconfigport.mk** for port specific settings and a list of enabled modules.

Prerequisites

Please ensure you set up your build environment appropriately, as per the guide. You will need:

- Linux: https://learn.adafruit.com/building-circuitpython/linux
- Windows Subsystem for Linux (WSL): https://learn.adafruit.com/building-circuitpython/windows-subsystem-for-linux
- Windows: Not supported yet
- MacOS: Not supported yet

Install necessary packages

```
$ sudo apt install default-jre gcc-arm-none-eabi python3 python3-pip git git-lfs gettext uncrustify
$ sudo python -m pip install --upgrade pip
```

Build Instructions

Ensure your clone of Circuitpython is ready to build by following the guide on the Adafruit Website. This includes installing the toolchain, synchronizing submodules, and running `mpy-cross`.

Clone the source code of CircuitPython from Github:

```
$ git clone https://github.com/adafruit/circuitpython.git
$ cd circuitpython
$ make fetch-submodules
```

Checkout the branch or tag you want to build. For example:

```
$ git checkout main
```

Follow the guideline below to install required packages for SLC tool: https://www.silabs.com/documents/public/user-guides/ug520-software-project-generation-configuration-with-slc-cli.pdf

Once the one-time build tasks are complete, you can build at any time by navigating to the port directory:

```
$ make BOARD=explorerkit_xg24_brd2703a
```

You may also build with certain flags available in the makefile, depending on your board and development goals:

```
$ make BOARD=explorerkit_xg24_brd2703a DEBUG=1
```

Clean the project by using:

```
$ make BOARD=explorerkit_xg24_brd2703a clean
```

You can use the following command build for each xG24 board:

Board	Build CMD
xG24 Dev Kit	devkit_xg24_brd2601b
xG24 Explorer Kit	explorerkit_xg24_brd2703a
Sparkfun Thing Plus MGM240P	sparkfun_thingplus_matter_mgm240p_brd2704a

Once the build process is complete, navigate to the build folder for the corresponding board, such as build-sparkfun_thingplus_matter_mgm240p_brd2704a, and verify that the **firmware.bin** file is present. This file contains the compiled binary firmware and is the file that should be uploaded to the microcontroller to run the application. By confirming the presence of the firmware.bin file, you can ensure that the build completed successfully and that the firmware is ready to be loaded onto the board.

Troubleshooting

If you encounter issues with the libbluetooth.a file, it may be due to an incomplete or corrupted clone of the Gecko SDK submodule. To prevent this issue, make sure to install **git-lfs** before cloning the submodule.

```
./circuitpython/ports/silabs/gecko_sdk/protocol/bluetooth/lib/EFR32MG24/GCC/libbluetooth.a: file format not recognized
./circuitpython/ports/silabs/gecko_sdk/protocol/bluetooth/lib/EFR32MG24/GCC/libbluetooth.a:1: syntax error
collect2: error: ld returned 1 exit status
make[1]: *** [Makefile:150: build-devkit_xg24_brd2601b/firmware.out] Error 1
make: *** [Makefile:141: build-devkit_xg24_brd2601b/firmware.bin] Error 2
```

Running Applications

Running CircuitPython Applications

Dowload CircuitPython Firmware

Official binaries for all supported boards are available through circuitpython.org/downloads. The site includes stable, unstable, and continuous builds.

If making changes, clone and re-build the source. The firmware.bin file can be found in the build folder corresponding to the appropriate board, such as build-sparkfun_thingplus_matter_mgm240p_brd2704a.

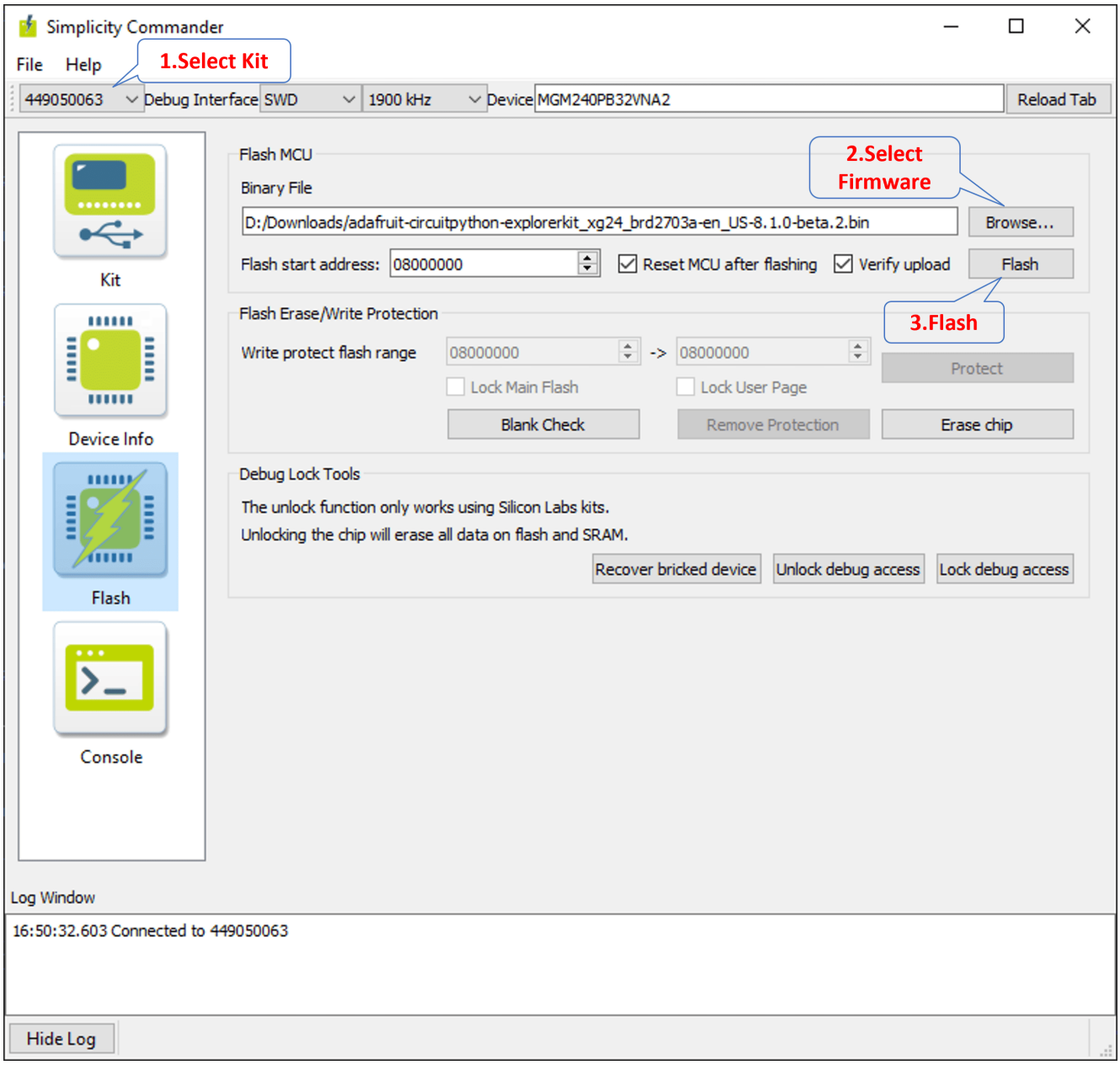
> *NOTE:* The examples in this repository require CircuitPython v8.2.0 or higher.

Flash Firmware

To flash the firmware file into the board, you need to use Simplicity Commander. You can install Simplicity Commander using Simplicity Studio or download a standalone version by following this Knowledge Article.

To flash the firmware into the xG24 kit using Simplicity Commander, follow these simple steps:

1. Connect your xG24 kit to your computer and ensure that it is recognized by your programming tool.
2. Browse and select the firmware file that you wish to upload to the board.
3. Initiate the flashing process to upload the firmware to the xG24 kit.



Getting a REPL Prompt

Connect the devkit to the PC via the USB cable. The board uses serial for REPL access and debugging because the EFR32 chips has no USB support.

Windows

On Windows, we need to install a serial console, e.g., PuTTY, MobaXterm. The JLink CDC UART Port can be found in the Device Manager.

Linux

Open a terminal and issue the following command:

```
$ ls /dev/ttyACM*
```

Then note down the correct name and substitute com-port-name in the following command with it:

```
$ screen /dev/'com-port-name'
```

Using the REPL Prompt

After flashing the firmware to the board, at your first connecting to the board, you might see a blank screen. Press enter and you should be presented with a Circuitpython prompt, >>>. If not, try to reset the board (see instructions below).

You can now type in simple commands such as:

```
>>> print("Hello world!")

Hello world
```

If something goes wrong with the board, you can reset it. Pressing CTRL+D when the prompt is open performs a soft reset.

Recommended Editors

Thonny is a simple code editor that works with the Adafruit CircuitPython boards.

Running CircuitPython Scripts

At the boot stage, two scripts will be run (if not booting in safe mode). First, the file boot.py will be executed. The file **boot.py** can be used to perform the initial setup. Then, after boot.py has been completed, the file **code.py** will be executed.

After code.py has finished executing, a REPL prompt will be presented on the serial port. Other files can also be executed by using **Thonny** editors or using the **Ampy** tool.

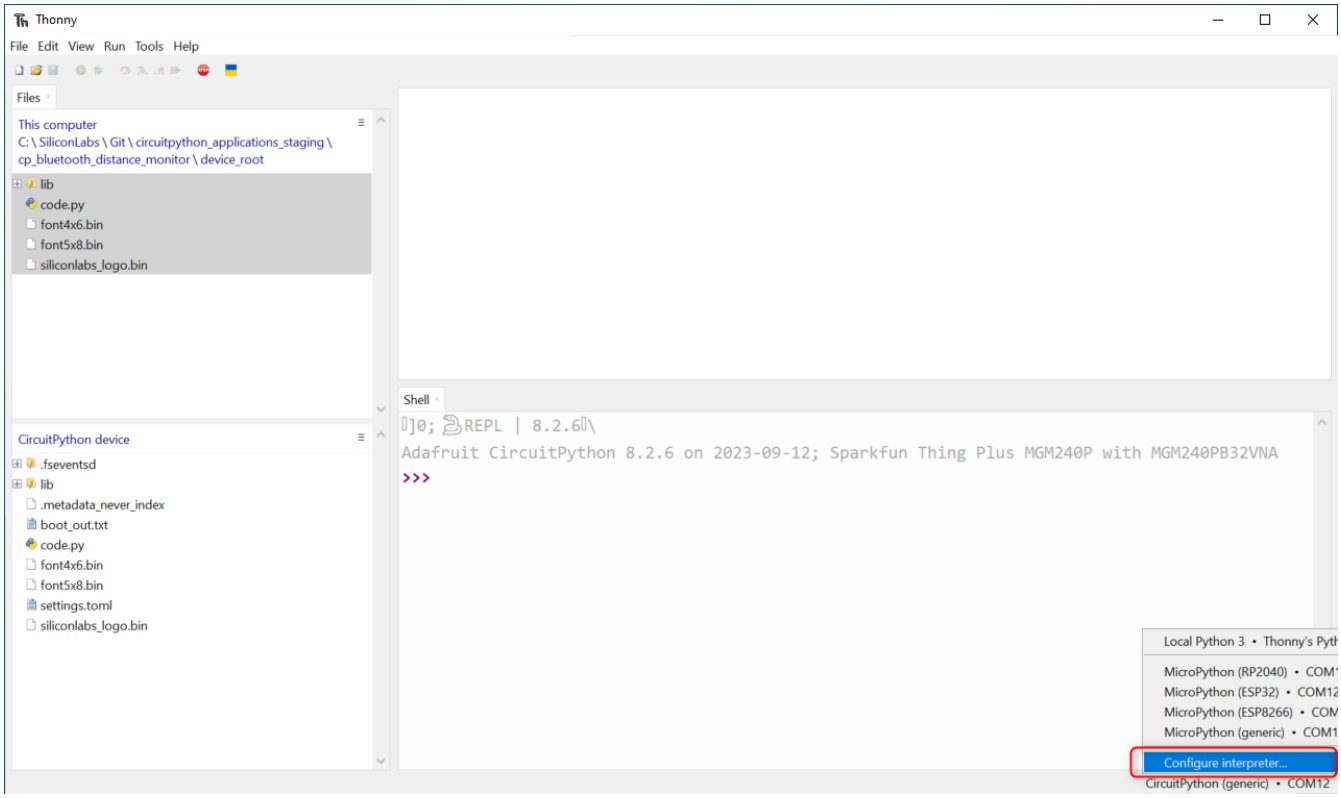
Thonny

Download and Install Thonny

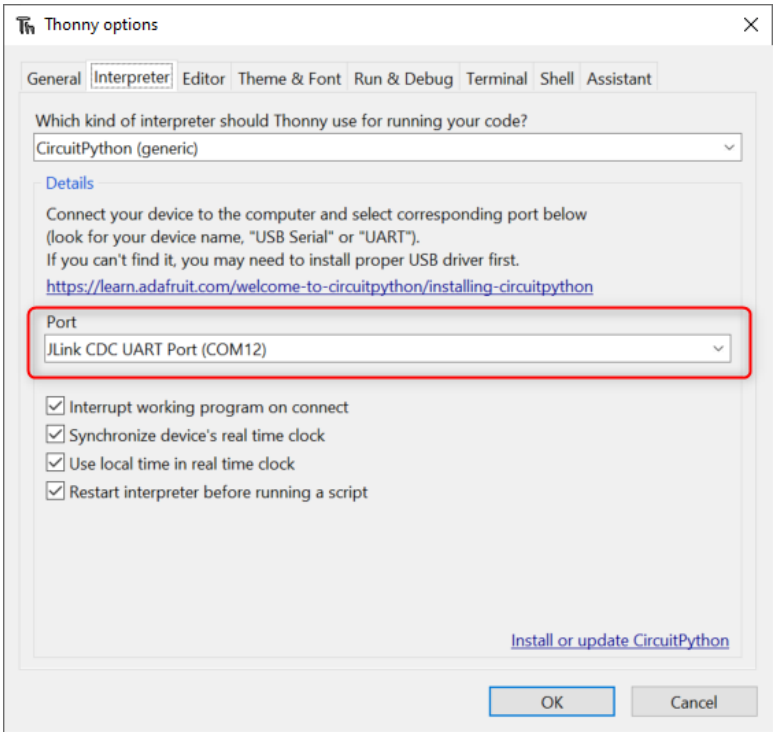
- Download Thonny

Connect to the Serial Prompt of the Target Board

- Open the interpreter configuration

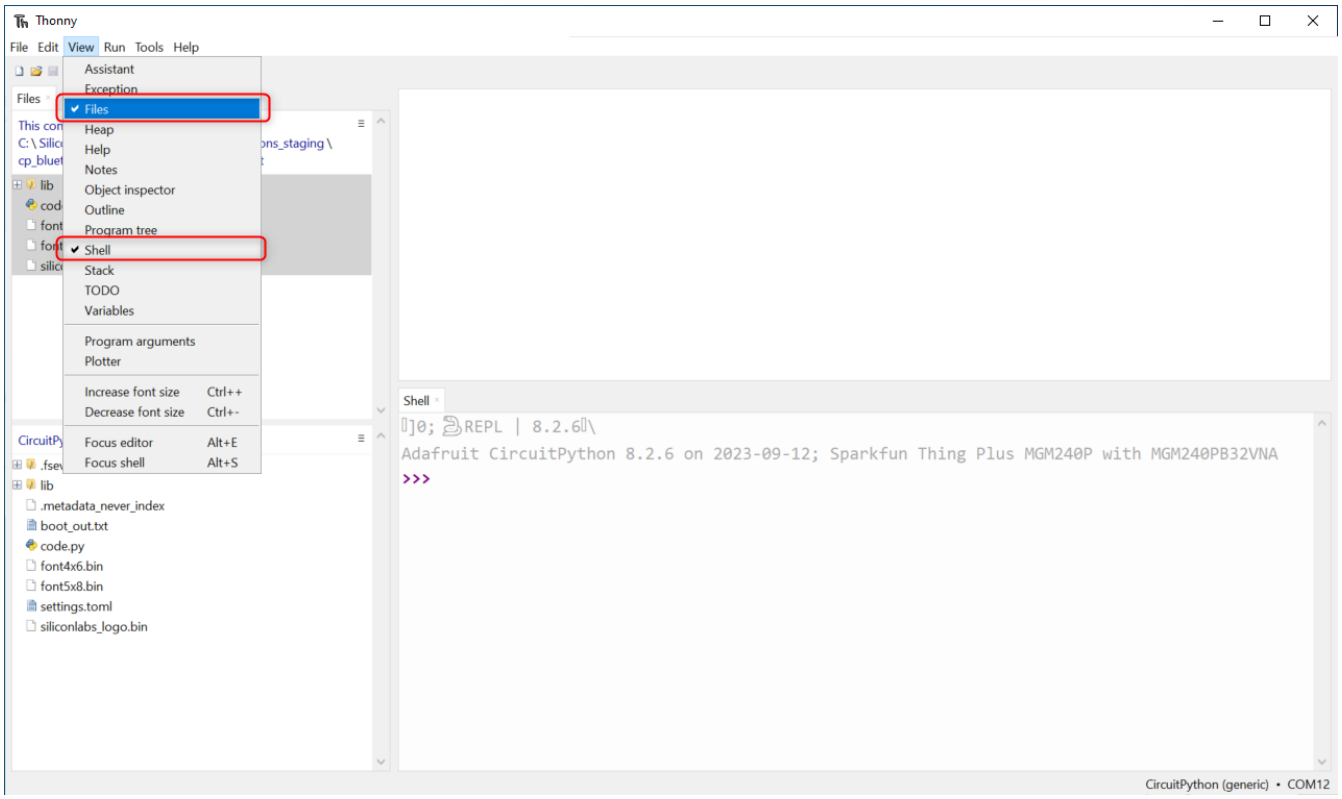


- Select Port Jlink CDC UART Port



Upload the Project Files

- Show Files and Shell views

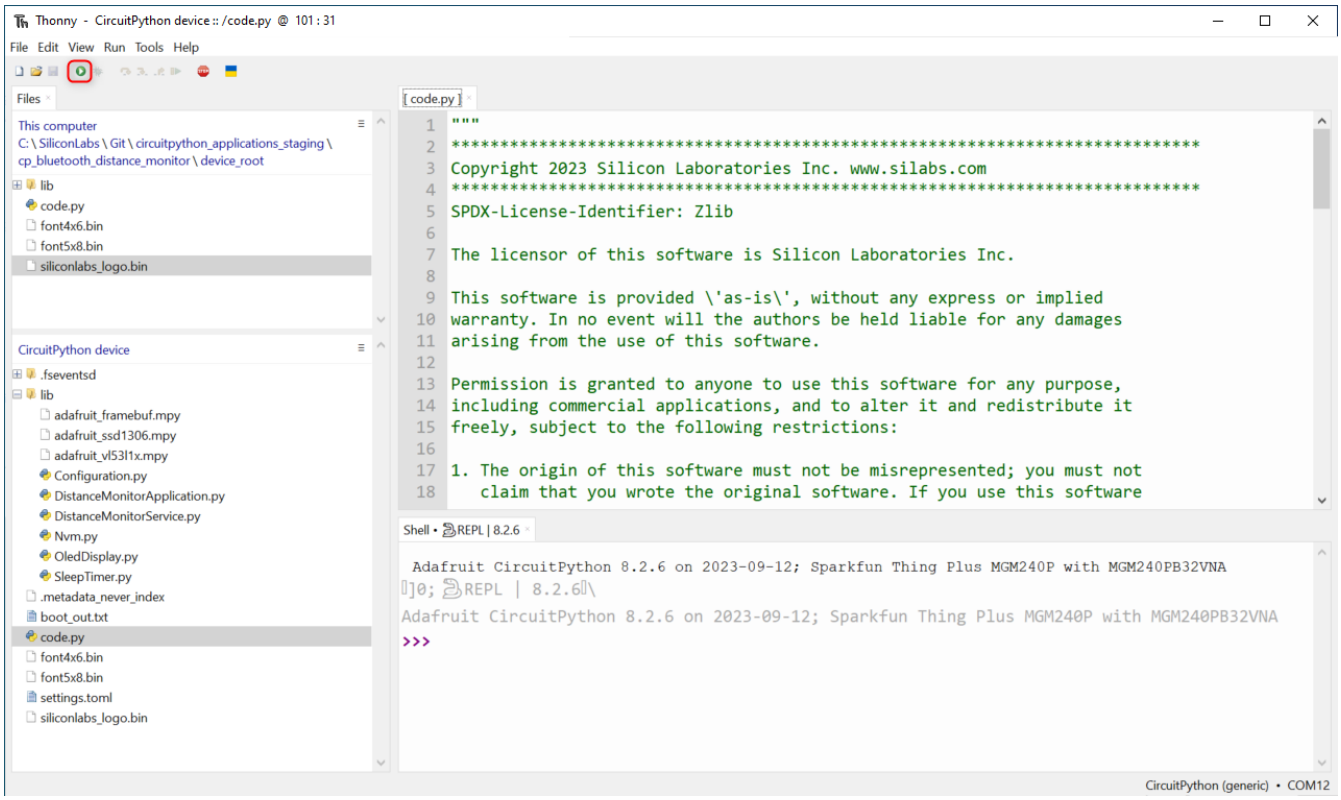


- Select and upload application files from the device_root folder

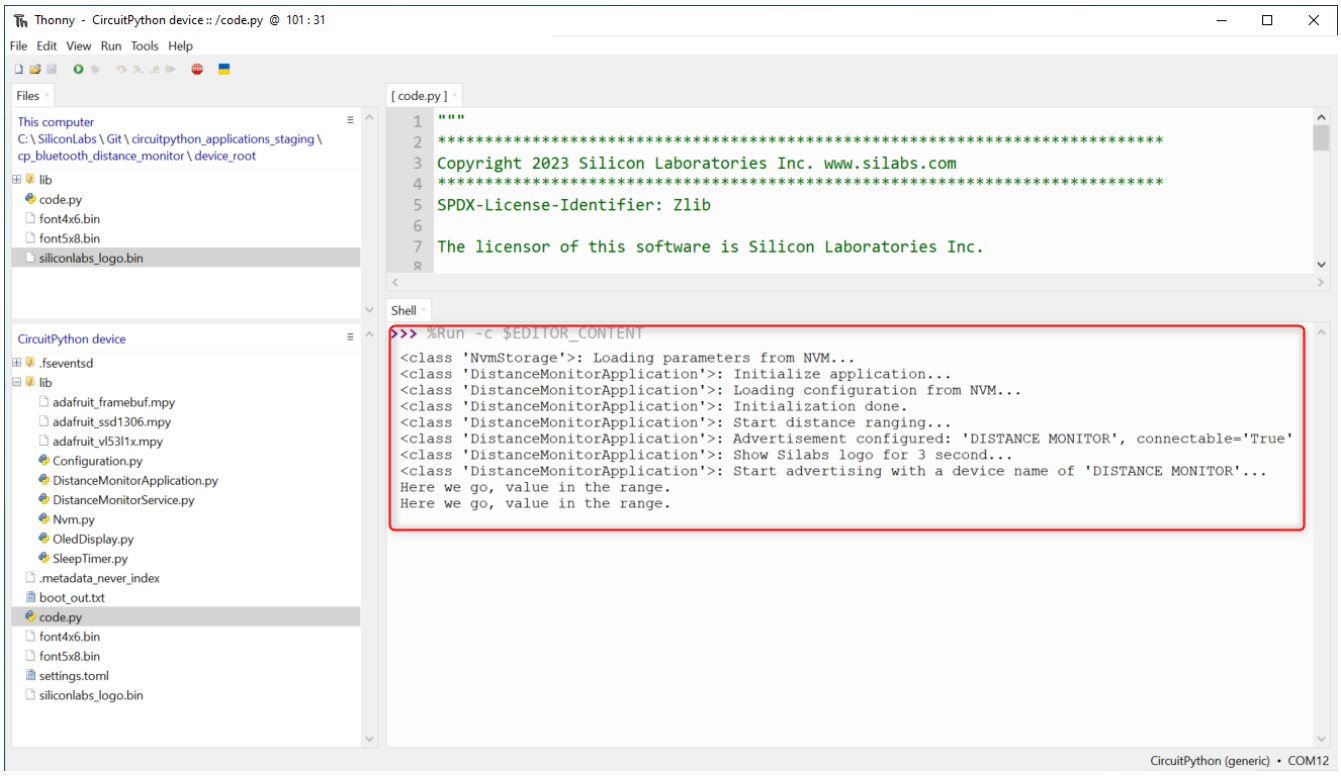
- The application files are uploaded to the target device

Run the Application

- Open the code.py from the target device and push Run current script (F5) button



The application should be running on the target device, you can check the log output in the Shell window



> *NOTE:* The application files are permanently stored on the target device, so the uploaded application should run automatically if the target device is reset or powered up while the serial prompt is not connected via Thonny or other tools like Ampy.

Ampy

With the boards which support USB mass storage, we can drag the files to the board file system. However, because the EFR32 boards don't support USB mass storage, we need to use a tool like **Ampy** to copy the file to the board. You can use the latest version of **Ampy** and its command to copy the module directories to the board.

Refer to the guideline below for installing the **Ampy** tool:

https://learn.adafruit.com/micropython-basics-load-files-and-run-code/install-ampy