



# Build Wireless Products Faster with Zephyr and MicroPython

16 April 2024

Presented by Ryan Erickson

# Intro

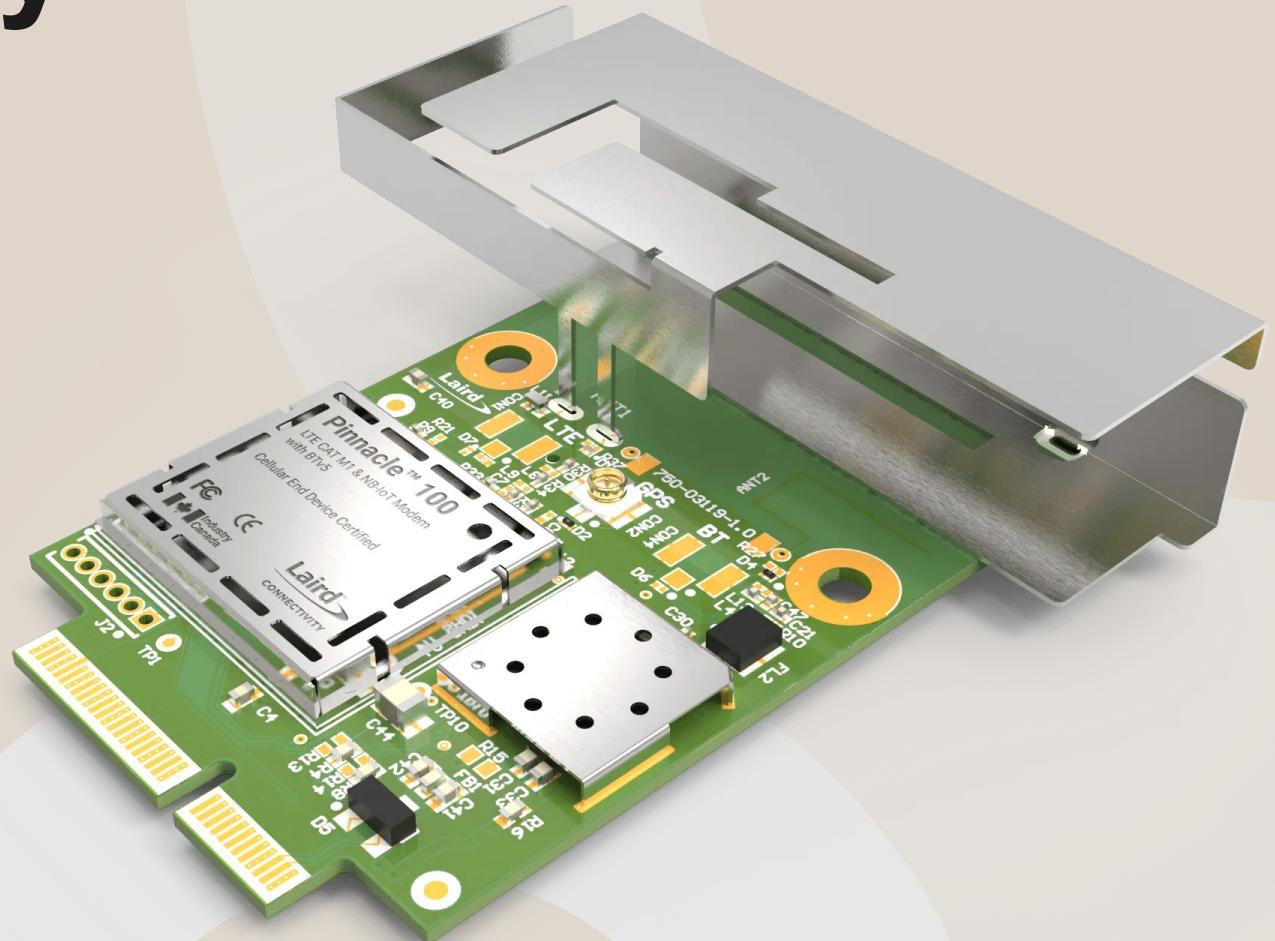
Ryan Erickson has been developing wireless products at Ezurio for 15 years. His areas of expertise include Zephyr development and maintenance, Wi-Fi, Bluetooth LE, Bluetooth Classic, 802.15.4, cellular, IoT, manufacturing automation, mobile apps, and more.



# Laird Connectivity is now **ezurio**

**Ezurio is Your  
Connectivity  
Expert**

<https://www.ezurio.com/laird-connectivity>



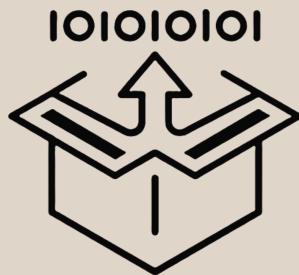
# Agenda

- Canvas Software Suite Overview
- Supported Products
- Canvas Software Suite Firmware Architecture
  - Zephyr Core
  - MicroPython Engine Integration

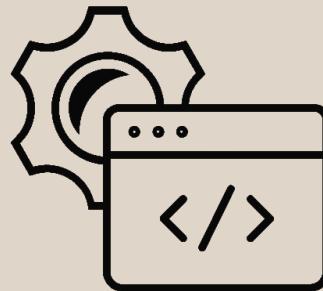
# What is Canvas Software Suite?

# CANVAS SOFTWARE SUITE

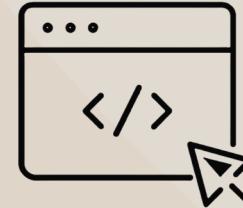
## SIMPLIFY AND ACCELERATE MODULE INTEGRATION



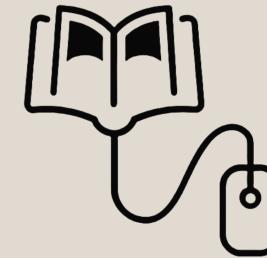
FIRMWARE



TOOLS



SAMPLES



DOCS

A **Software Platform** enabling rapid, portable development across our microcontroller-based wireless product portfolio.

# CANVAS SOFTWARE SUITE - FIRMWARE

<https://www.ezurio.com/canvas/software-suite/development-workflow#firmware>

## Install Canvas Firmware

Our Canvas Firmware contains the bootloader, RTOS and middleware required to develop your application. This firmware must be loaded in order to support on-module Python scripts, AT command set operation and interoperability with desktop and mobile tools. Firmware can be programmed using the XBIT Desktop tool.

Product	Description	Download
Sera NX040 DVK	Canvas Firmware - Zephyr + Python	 Download
Lyra 24P/S DVK	Canvas Firmware - FreeRTOS + Python	 Download
MG100	Canvas Firmware - Zephyr + Python	 Download
Pinnacle 100	Canvas Firmware - Zephyr + Python	 Download
BL5340DVK	Canvas Firmware - Zephyr + Python	 Download
BL654 USB Adapter	Canvas Firmware - Zephyr + Python	 Download
Sentrius BT510 Sensor <small>BETA</small>	Canvas Firmware - Zephyr + Python	 Download
BL654 DVK <small>BETA</small>	Canvas Firmware - Zephyr + Python	 Download
Sentrius BT610 Sensor <small>BETA</small>	Canvas Firmware - Zephyr + Python	 Download

# CANVAS SOFTWARE SUITE - TOOLS



# Xbit VS Code Extension

# Create, Load and Interact with On-Device Python Code

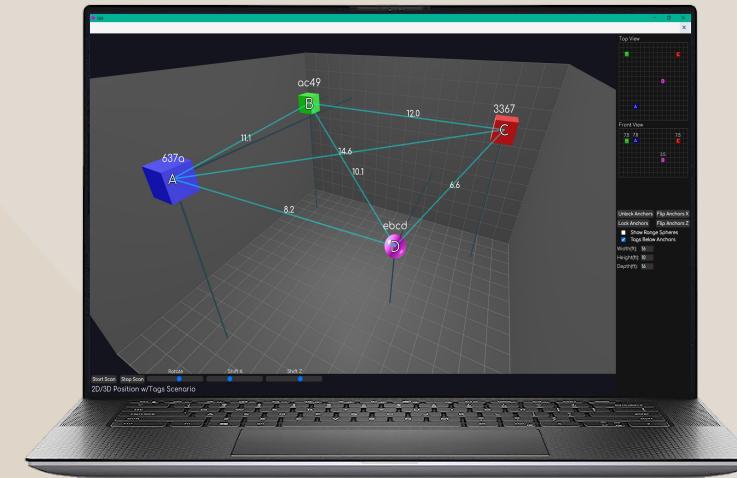
Develop Python scripts and load them onto devices directly from the industry's most popular source code editor.

Access the module's Python REPL to quickly test out APIs and prove out your application code.



# Xbit Desktop App

## Visualize Data and Interact with Canvas-enabled Devices



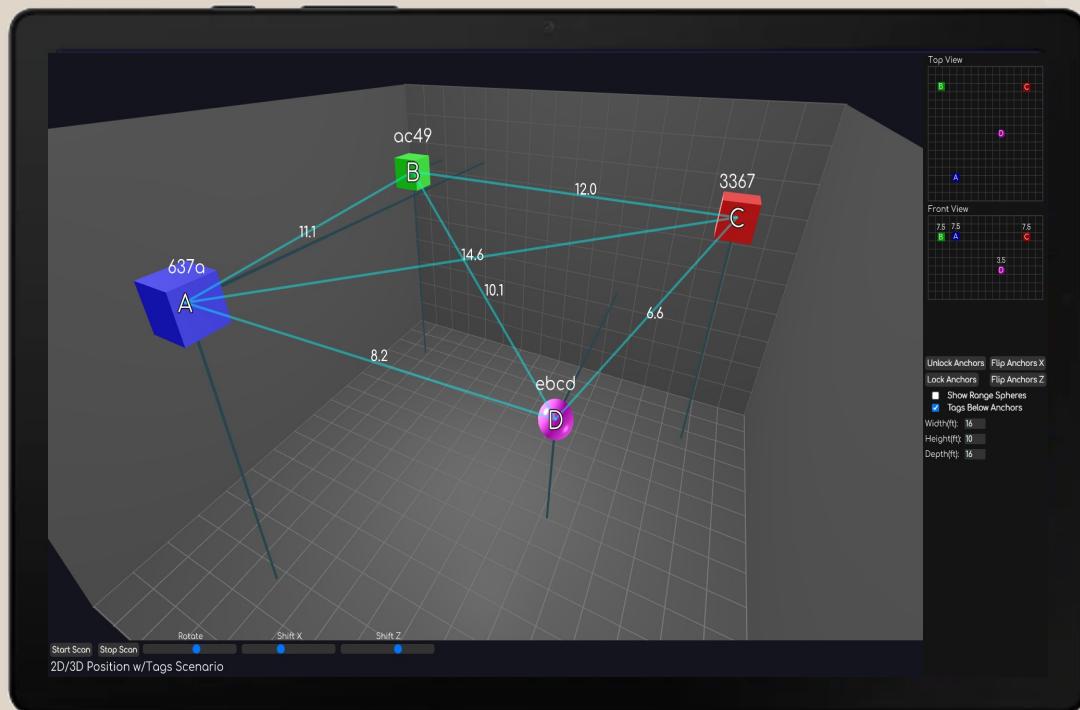
Our In-house designed Xbit Desktop application provides a cross-platform framework for advanced visualization of your device's data over Bluetooth Low Energy. Use one of our provided Applets or create your own with intuitive JavaScript APIs!

# CANVAS SOFTWARE SUITE - TOOLS



## Xbit Mobile App

Visualize Data and Interact with Canvas-enabled Devices



Xbit Mobile provides the same advanced visualization technology as Xbit Desktop. Applet code can be re-used for compatibility with both Desktop and Mobile versions of Xbit.

Target a tablet for a larger view or use a phone for simple workflows like OTA firmware update or device configuration.

# CANVAS SOFTWARE SUITE - SAMPLES

[https://github.com/LairdCP/canvas\\_python\\_samples](https://github.com/LairdCP/canvas_python_samples)

The screenshot shows the GitHub repository page for 'canvas\_python\_samples' owned by 'LairdCP'. The repository is public and contains 2 branches and 0 tags. The main branch has 37 commits from 'randy wholey' (randywholey) dated 3 weeks ago. The commits are listed below:

Commit	Message	Date
apps	Control LEDs in phone-to-tag script	3 weeks ago
basic	Clean up of sample scripts	4 months ago
bluetooth	Clean up PHY selection in samples	2 months ago
modules	UWB AT commands	3 weeks ago
networking	sample: mqts: add topic subscription	last month
peripherals	gpio: add mg100/pinnacle 100 sample	2 months ago
README.md	Clean up of sample scripts	4 months ago

The README section contains the following text:

**Canvas Python Samples**

This repository contains various sample scripts and applications for use with Laird Connectivity Canvas MicroPython firmware. See README.md inside of each subdirectory for specific information about using the scripts or applications.

# CANVAS SOFTWARE SUITE - DOCUMENTATION

CANVAS SOFTWARE SUITE WEBSITE > [ezurio.com/canvas/software-suite](https://ezurio.com/canvas/software-suite)

ezurio Products Services Support Resources Markets Partners Contact Search... Why Ezurio?

CANVAS

CANVAS SOFTWARE SUITE

Our Canvas™ software suite enables rapid embedded development across our MCU-based wireless products. Cross-chipset middleware, easy-to-use wireless APIs, on-module scripting and intuitive desktop/mobile tools are all available to dramatically ease embedded development.

Build your wireless IoT application in days, not months, with embedded Python scripts. No vendor-specific SDKs or development environments with heaps of embedded C code (although C development is available for those who require it.)

Quickly get connected and visualize your data with script-driven extensible desktop and mobile tools. Start with our feature-rich sample apps and customize to create your next big thing.

Explore Canvas Software Suite

While you're here, watch this video!

Scroll Down Here...

ezurio Products Services Support Resources Markets Partners Contact Search... Why Ezurio?

CANVAS

Canvas Documentation

We've got the answers to your questions from the high level to the most technical details. Canvas Documentation provides resources for software developers of every level. Overviews, application tutorials, and walkthroughs simplify the initial learning curve. Detailed API documentation and datasheets cover the granular details once you are ready to take a deeper dive.

Tutorials and Walkthroughs

Product	Description	Download
All Canvas Hardware	Canvas Overview	Download
All Canvas Hardware	Canvas Python User Guide	Download
Sera NX040 DVK	Sera NX040 Software User Guide	Download
Sera NX040 DVK	Sera NX040 UWB Ranging App Note	Download
Lyra 24	Lyra 24 Software User Guide	Download
Lyra 24P DVK	Lyra 24P User Guide	Download
Lyra 24S DVK	Lyra 24S User Guide	Download
MG100	MG100 and Pinnacle 100 Software User Guide	Download
Pinnacle 100	MG100 and Pinnacle 100 Software User Guide	Download
BL5340 DVK	BL5340 DVK Software User Guide	Download

API Documents

Description	Version	Download
Canvas Python API	1.0	Download

Hardware Datasheets

Description	Version	Download
Sera NX040 Datasheet	1.0	Download
Sera NX040 DVK User Guide	1.0	Download
Lyra 24P Datasheet	1.9	Download
Lyra 24S Datasheet	1.3	Download

# What Products are Supported?

# CANVAS-ENABLED PRODUCTS



**Sera™ NX040 (BLE + UWB)**  
PN: 453-00174-K1, 453-00175-K1



**Lyra™ 24 Series (BLE)**  
PN: 453-00142-K1, 453-00145-K1,  
453-00148-K1, 453-00170-K1



**BL654 USB Adapter (BLE)**  
PN: 451-00004



**BL654 (BLE)**  
PN: 455-00001, 455-00002



**Sentrius™ BT610 Sensor**  
PN: 450-00121-K1



**Sentrius™ BT510 Sensor**  
PN: 455-00083, 450-00048B



**BL653 and BL653μ (BLE)**  
PN: 453-00039-K1, 453-00041-K1



**BL5340 (BLE)**  
PN: 453-00052-K1, 453-00053-K1



**Sentrius™ MG100 Gateway**  
PN: 450-00011-K1, 450-00038-K1,  
450-00039-K1, 450-00054-K1



**Pinnacle™ 100 Cellular  
LTE-M/NB-IoT/BLE Modem**  
PN: 453-00010-K1, 453-00011-K1



**Sentrius™ BT710 Multi-Sensor**  
PN: 450-00122-K1, 450-00122B

Support Planned for  
Future MCU-based  
Wireless Modules and  
IoT Products

# CANVAS SOFTWARE SUITE – SAMPLE APPS

Sera NX040 UWB



UWB Ranging  
Demo



Tag-to-Tag  
3D Visualization  
Applet

PYTHON

Lyra 24/BL65x BLE



Eddystone and  
iBeacon Samples

PYTHON



BLE Advertiser  
and Scanner

PYTHON



GATT Client

PYTHON

Sentrius BT510



Tilt Sensor  
Demo

PYTHON



Tilt Sensor 3D  
Visualization  
Applet

XBIT

Sentrius MG100



BLE to MQTT  
Gateway

PYTHON



LwM2M Device  
Management

PYTHON



HTTP Client  
Sample

PYTHON

BL654 USB Adapter



Xbit USB Adapter  
Script

PYTHON



Canvas Hub

XBIT



(All BLE  
Applets)

XBIT

BL654 DVK



Button and  
LED Demo

PYTHON



(All BLE Samples)

PYTHON

BL5340 DVK



BLE to MQTT  
Gateway

PYTHON



LwM2M Device  
Management

PYTHON



HTTP Client  
Sample

PYTHON

Click Board Samples  
(All DVKs w/mikroBUS)



Temp and Hum  
Click Sample

PYTHON



Weather Click

PYTHON



Accel 10 Click

PYTHON

Basic Samples  
(All Devices)



Hello World  
Sample

PYTHON



Device Sleep

PYTHON



Timer Callbacks

PYTHON

Peripheral I/O  
(All Devices)



GPIO

PYTHON



I2C

PYTHON



SPI

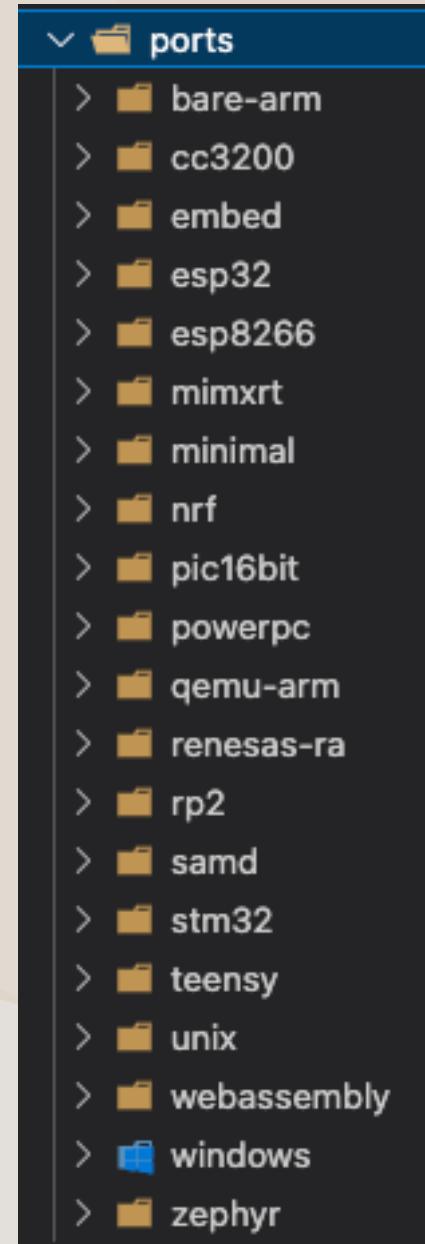
PYTHON

# Canvas Firmware Architecture

# MicroPython

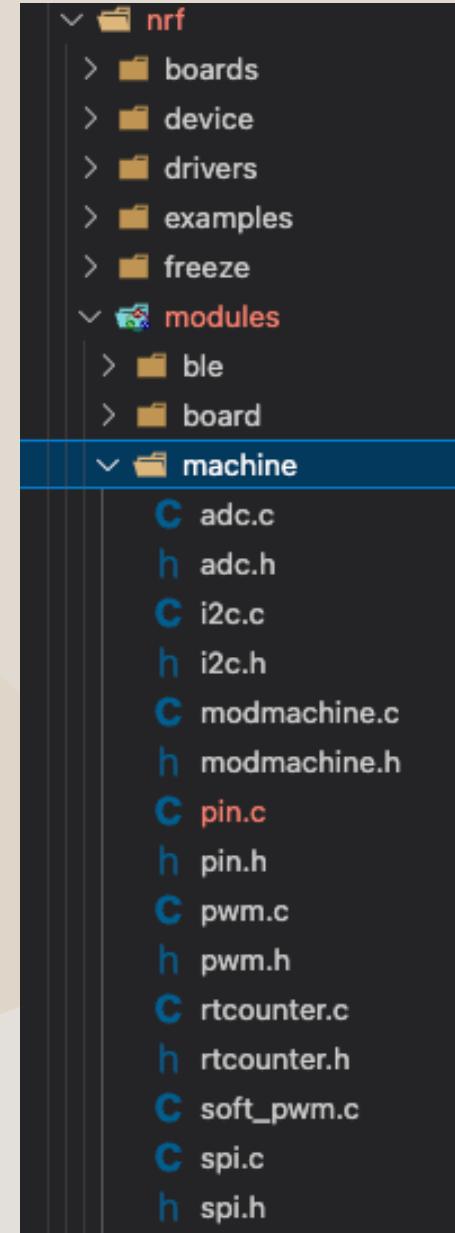
<https://github.com/micropython/micropython>

- An “implementation of Python 3.x on microcontrollers and small embedded systems.”
- Ports for different embedded systems
  - A port can be RTOS based or bare metal
- Provides source for core subsystems
  - File system, Networking, USB, etc.
  - Subsystems are based on other open-source projects
- Observations
  - Ports vary on “completeness” and maintenance
  - No firmware updates



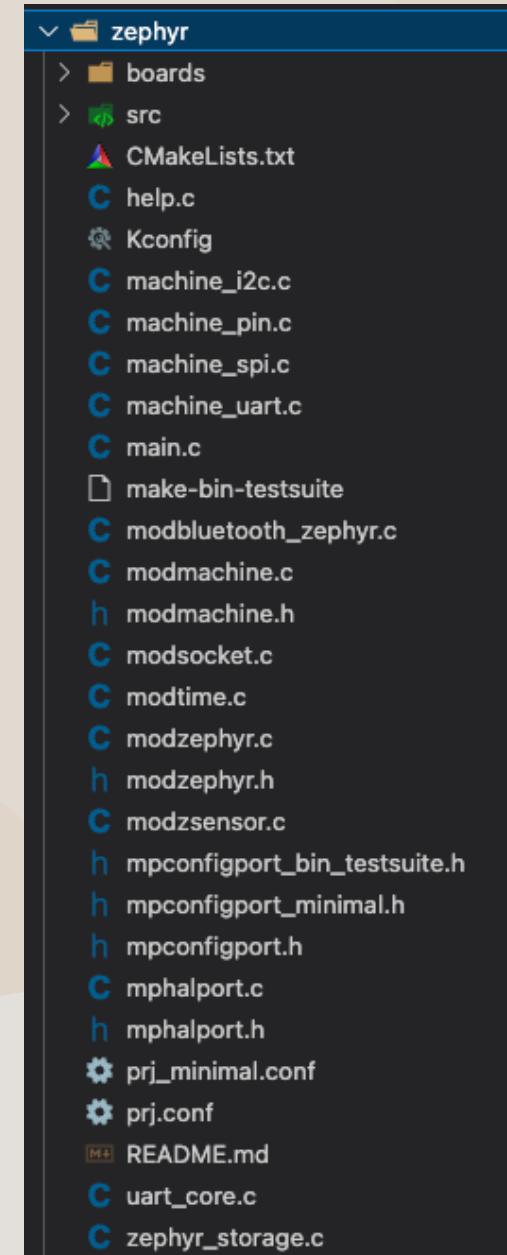
# MicroPython nRF port

- Outdated
  - Bluetooth LE based on nRF5 SDK softdevice
- Not supported by vendor
  - No Bluetooth qualifications
- No RTOS benefits
- Manually add features yourself



# MicroPython Zephyr Port

- Not up to date with Zephyr
- Does not leverage all Zephyr subsystems
  - Uses its own VFS
  - Limited Bluetooth LE features (lack of python glue)
    - No extended advertising/scanning
    - All PHYs not supported
- No bootloader/firmware update



# CANVAS PLATFORM FIRMWARE STACK

## COMMAND SETS

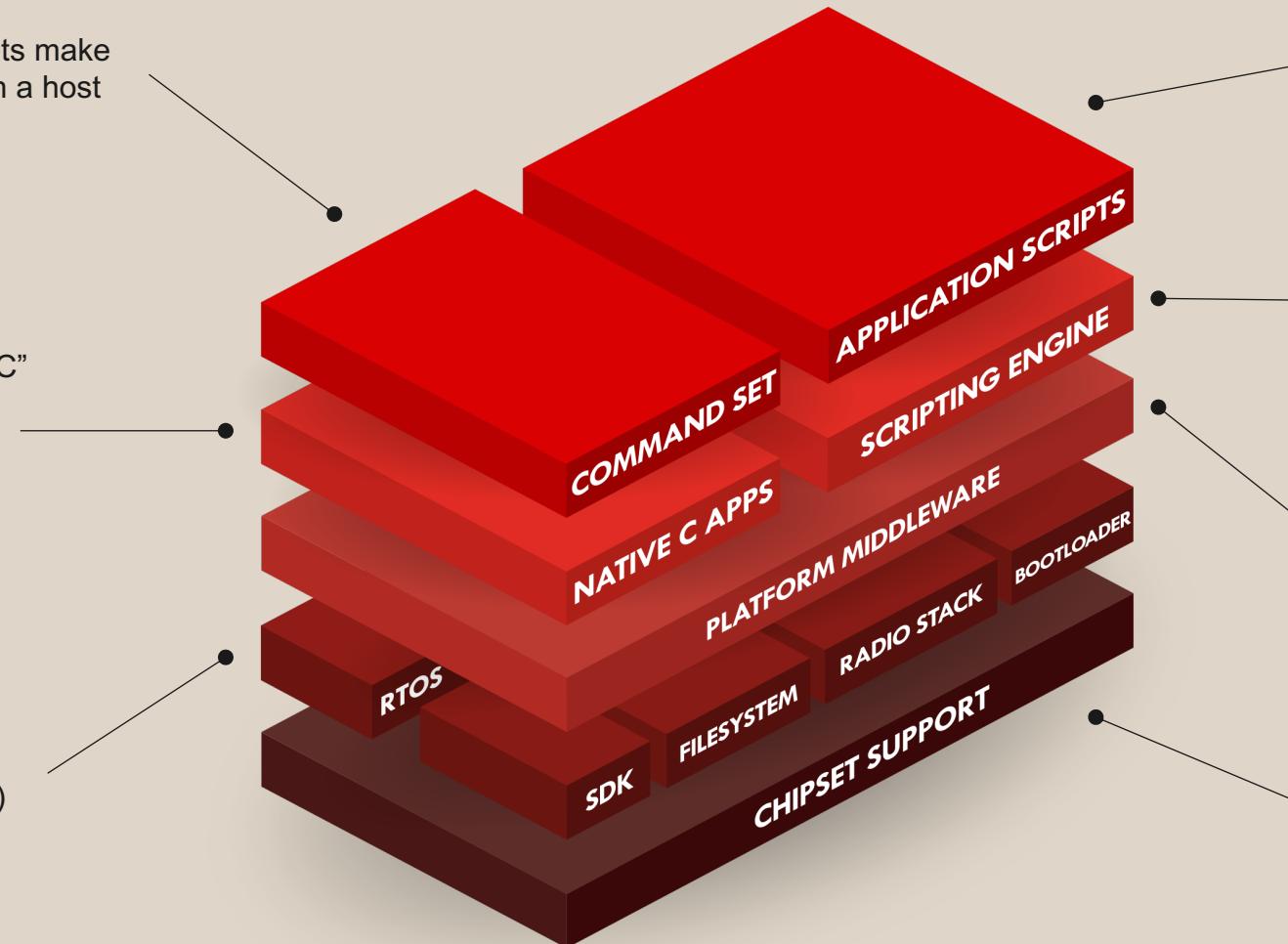
AT and/or binary command sets make modules easy to integrate with a host

## C-BASED APPLICATIONS

Applications requiring “native C” development can build upon middleware API to speed development cycles

## STANDARD RTOS

Builds upon standard RTOS platforms (e.g., Zephyr RTOS)



## APPLICATION-SPECIFIC SCRIPTS

Quickly enable application-specific functionality with scripting to orchestrate underlying firmware operations

## SCRIPTING ENGINE

Language integration layer for common scripting engines

## PLATFORM MIDDLEWARE

Ezurio designed abstraction layers enable application portability across MCU-based products, RTOS, scripting language and peripherals

## VENDOR SDK & LIBRARIES

Builds upon Hardware-specific, SDKs, Drivers, Bootloaders, Filesystem and other low-level firmware supported by chipset vendors

# Canvas MicroPython

## Zephyr is the core!

- Zephyr is the core with MicroPython added as a module
- Benefit from all Zephyr features and the community behind it
- MicroPython in a thread

```
- name: micropython
  path: modules/lib/laird_connect/micropython
  revision: v1.21.0
  remote: micropython
- name: micropython_embed
  path: modules/lib/laird_connect/micropython_embed
- name: micropython_embed_tools
  path: modules/lib/laird_connect/micropython_embed_tools
```

```
SYS_INIT(zephyr_micropython_init, APPLICATION, CONFIG_ZEPHYR_MICROPYTHON_INIT_PRIORITY);

static int zephyr_micropython_init(void)
{
    /* Store and reset the reset cause */
    hwinfo_get_reset_cause(&mp_reset_cause);
    hwinfo_clear_reset_cause();

    /* Initialize the UART */
    (void)mp_platform_uart_init();

    /* Start a thread */
    micropython_main_thread_id =
        k_thread_create(&micropython_thread, micropython_thread_stack,
                       K_THREAD_STACK_SIZEOF(micropython_thread_stack),
                       micropython_thread_fn, NULL, NULL, NULL,
                       CONFIG_ZEPHYR_MICROPYTHON_THREAD_PRIORITY, 0, K_NO_WAIT);
    k_thread_name_set(micropython_main_thread_id, MICROPYTHON_THREAD_NAME);

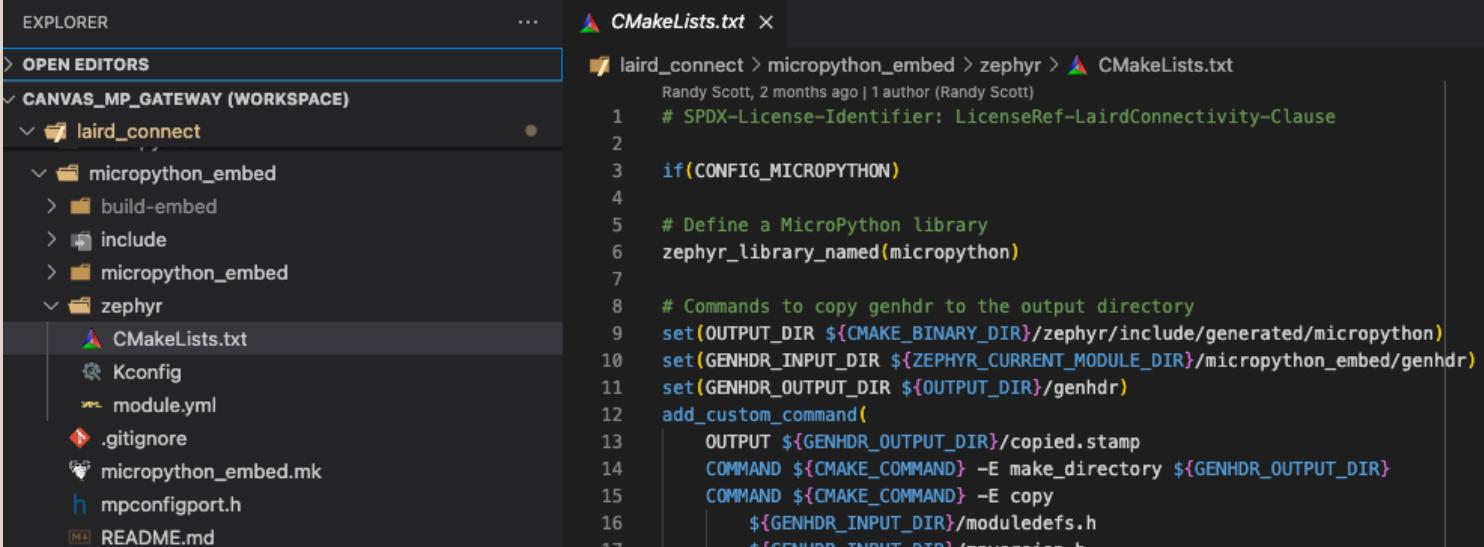
    return 0;
}
```

# Zephyr with MicroPython Embed Port

<https://github.com/micropython/micropython/tree/master/ports/embed>

The MicroPython embed port is designed to embed MicroPython into an existing C application.

- The embed port was turned into a Zephyr module
  - Can also be used in Non-Zephyr code bases
- MicroPython engine can be started in a thread



The screenshot shows a code editor interface with two main panes. The left pane is the 'EXPLORER' view, which displays the project structure under 'CANVAS\_MP\_GATEWAY (WORKSPACE)'. It includes a folder for 'laird\_connect' containing 'micropython\_embed' and 'zephyr' subfolders. The 'zephyr' folder contains files like 'CMakeLists.txt', 'Kconfig', 'module.yml', '.gitignore', 'micropython\_embed.mk', 'mpconfigport.h', and 'README.md'. The right pane is the 'CMakeLists.txt' editor, showing the following code:

```
laird_connect > micropython_embed > zephyr > CMakeLists.txt
Randy Scott, 2 months ago | 1 author (Randy Scott)
# SPDX-License-Identifier: LicenseRef-LairdConnectivity-Clause
if(CONFIG_MICROPYTHON)
# Define a MicroPython library
zephyr_library_named(micropython)
# Commands to copy genhdr to the output directory
set(OUTPUT_DIR ${CMAKE_BINARY_DIR}/zephyr/include/generated/micropython)
set(GENHDR_INPUT_DIR ${ZEPHYR_CURRENT_MODULE_DIR}/micropython_embed/genhdr)
set(GENHDR_OUTPUT_DIR ${OUTPUT_DIR}/genhdr)
add_custom_command(
    OUTPUT ${GENHDR_OUTPUT_DIR}/copied.stamp
    COMMAND ${CMAKE_COMMAND} -E make_directory ${GENHDR_OUTPUT_DIR}
    COMMAND ${CMAKE_COMMAND} -E copy
        ${GENHDR_INPUT_DIR}/moduledefs.h
        ${GENHDR_INPUT_DIR}/mpversion.h
)
```

# Zephyr File System and Device Tree

A file system is a core requirement for a MicroPython device

**With Zephyr kconfig and device tree no C code needs to be written to enable a filesystem on a device.**

Enable the correct subsystems with kconfig.

```
# Enable the LittleFS file system.  
CONFIG_FILE_SYSTEM=y  
CONFIG_FILE_SYSTEM_LITTLEFS=y
```

Setup the filesystem partition and automount it on boot with device tree.

```
/ {  
    fstab {  
        compatible = "zephyr,fstab";  
        lfs1: lfs1 {  
            compatible = "zephyr,fstab,littlefs";  
            mount-point = "/lfs1";  
            partition = <&littlefs_storage>;  
            automount;  
            read-size = <16>;  
            prog-size = <16>;  
            cache-size = <64>;  
            lookahead-size = <32>;  
            block-cycles = <512>;  
        };  
    };  
    &mx25r64 {  
        partitions {  
            compatible = "fixed-partitions";  
            #address-cells = <1>;  
            #size-cells = <1>;  
  
            littlefs_storage: partition@200000 {  
                label = "littlefs_storage";  
                reg = <0x00200000 0x00600000>;  
            };  
        };  
    };  
};
```

# Connect the Filesystem to MicroPython

```
zephyr_micropython
> include
< zephyr
> src
  C gchelper_generic.c
  C micropython_glue.c
  C modtime.c
  C mp_deferred_exception.c
  C mp_platform_fs.c
  C mp_platform_machine.c
  C mp_platform_network.c
  C mp_platform_time.c
  C mp_platform_uart_async.c
  C mp_platform_uart_interrupt.c
  C mphalport.c
> zephyr
> .clang-format
> CMakeLists.txt
> Kconfig
```

Implement a new module  
to “glue” MP to Zephyr.

Implement file stream  
operations.

Implement open to get a file object.

```
mp_obj_t mp_builtin_open(size_t n_args, const mp_obj_t *args, mp_map_t *kwargs)
{
    zephyr_mp_file_descriptor_obj_t *self = NULL;

    if (n_args == 1) {
        /* Default mode to "r" if not provided */
        self = zephyr_file_open(mp_obj_str_get_str(args[0]), "r");
    } else if (n_args >= 2) {
        self = zephyr_file_open(mp_obj_str_get_str(args[0]), mp_obj_str_get_str(args[1]));
    }

    if (self == NULL) {
        mp_raise_OSError(MP_ENOENT);
    }

    return MP_OBJ_FROM_PTR(self);
}

MP_DEFINE_CONST_FUN_OBJ_KW(mp_builtin_open_obj, 1, mp_builtin_open);
```

```
STATIC const mp_map_elem_t zephyr_file_locals_dict_table[] = {
    /* Stream operations */
    { MP_ROM_QSTR(MP_QSTR_read), (mp_obj_t)&mp_stream_read_obj },
    { MP_ROM_QSTR(MP_QSTR_readinto), (mp_obj_t)&mp_stream_readinto_obj },
    { MP_ROM_QSTR(MP_QSTR_readline), (mp_obj_t)&mp_stream_unbuffered_readline_obj },
    { MP_ROM_QSTR(MP_QSTR_readlines), (mp_obj_t)&mp_stream_unbuffered_readlines_obj },
    { MP_ROM_QSTR(MP_QSTR_write), (mp_obj_t)&mp_stream_write_obj },
    { MP_ROM_QSTR(MP_QSTR_flush), (mp_obj_t)&mp_stream_flush_obj },
    { MP_ROM_QSTR(MP_QSTR_close), (mp_obj_t)&mp_stream_close_obj },
    { MP_ROM_QSTR(MP_QSTR_seek), (mp_obj_t)&mp_stream_seek_obj },
    { MP_ROM_QSTR(MP_QSTR_tell), (mp_obj_t)&mp_stream_tell_obj },
    { MP_ROM_QSTR(MP_QSTR_del_), (mp_obj_t)&mp_stream_close_obj },
    { MP_ROM_QSTR(MP_QSTR_enter_), (mp_obj_t)&mp_identity_obj },
    { MP_ROM_QSTR(MP_QSTR_exit_), (mp_obj_t)&mp_stream_exit_obj },
};

STATIC MP_DEFINE_CONST_DICT(zephyr_file_locals_dict, zephyr_file_locals_dict_table);

/* clang-format off */
STATIC const mp_stream_p_t zephyr_text_stream = {
    .read = zephyr_file_read,
    .write = zephyr_file_write,
    .ioctl = zephyr_file_ioctl,
    .is_text = true,
};

MP_DEFINE_CONST_OBJ_TYPE(
    zephyr_text_stream_type,
    MP_QSTR_TextFileIO,
    MP_TYPE_FLAG_NONE,
    protocol, &zephyr_text_stream,
    locals_dict, &zephyr_file_locals_dict);

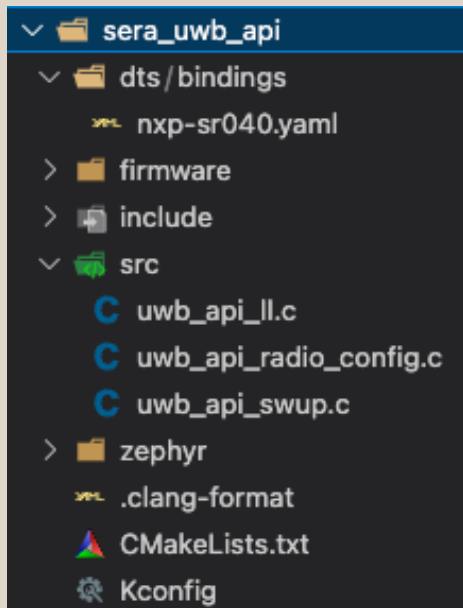
STATIC const mp_stream_p_t zephyr_binary_stream = {
    .read = zephyr_file_read,
    .write = zephyr_file_write,
    .ioctl = zephyr_file_ioctl
};

MP_DEFINE_CONST_OBJ_TYPE(
    zephyr_binary_stream_type,
    MP_QSTR_BinaryFileIO,
    MP_TYPE_FLAG_NONE,
    protocol, &zephyr_binary_stream,
    locals_dict, &zephyr_file_locals_dict);
```

# Zephyr Driver Architecture

To add new hardware, re-use Zephyr's driver architecture.

We implemented an out-of-tree driver for the NXP SR040 UWB radio.



```
/* SPI bus to UWB radio */
&spi1 {
    compatible = "nordic,nrf-spim";
    status = "okay";
    cs-gpios = <&gpio0 14 GPIO_ACTIVE_LOW>;
    pinctrl-0 = <&spi1_default>;
    pinctrl-1 = <&spi1_sleep>;
    pinctrl-names = "default", "sleep";
    uwb_radio: sr040@0 {
        compatible = "nxp,sr040";
        reg = <0>;
        spi-max-frequency = <8000000>;
        int-gpios = <&gpio0 15 GPIO_ACTIVE_LOW>;
        ready-gpios = <&gpio0 12 GPIO_ACTIVE_LOW>;
        reset-gpios = <&gpio1 7 GPIO_ACTIVE_LOW>;
    };
};
```



# Expose C APIs to MicroPython

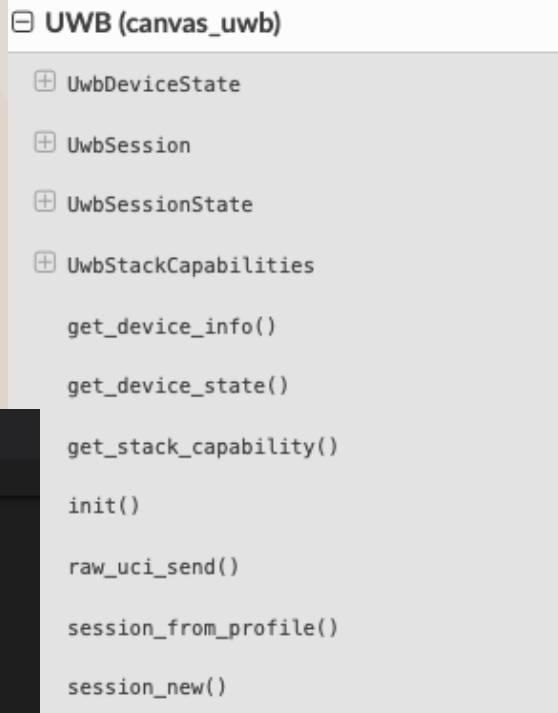
Python API is defined with a pyi file.

```
canvas_uwb.pyi ×
laird_connect > li_mp_uwb > canvas_uwb.pyi > ...
372     def init():
373         """
374             Initialize the UWB stack. This method must be called before any other
375             UWB stack methods are called.
376         """
377         pass
378
379     def shutdown():
380         """
381             Shutdown the UWB stack. This method must be called when the UWB stack
382             is no longer needed. To use the UWB stack again, the `init()` method.
383         """
384         pass
385
386     def get_stack_capability() -> UwbStackCapabilities:
387         """
388             Get the capabilities of the UWB stack.
389
390             :return: The capabilities of the UWB stack as a
391             | :py:class:`UwbStackCapabilities` object.
392         """
393         pass
394
395     def get_device_state() -> UwbDeviceState:
396         """
397             Get the state of the UWB device.
398
399             :return: The state of the UWB device as a :py:class:`UwbDeviceState`
400             | object.
401         """
402         pass
```

# C APIs and MicroPython Cont.

- Python scripts to generate C from the pyi file.
- User Python API is generated from pyi file.
- C developer implements C to python API.

```
C canvas_uwb.c x
laird_connect > li_mp_uwb > src > C canvas_uwb.c > canvas_uwb_raw_uci_send(mp_obj_t)
198
199 mp_obj_t canvas_uwb_session_new(int session_id, int role)
200 {
201     mp_obj_t params[1];
202     mp_obj_t obj;
203     PI_HAPI_ERROR_T err;
204
205     /* Create a session */
206     err = pi_hapi_uwb_session_init(session_id, role);
207     if (err != PI_HAPI_ERROR_OKAY) {
208         mp_raise_msg_varg(&mp_type_RuntimeError, MP_ERROR_TEXT("session_new: failed %d"),
209                         err);
210         return mp_const_none;
211     }
212
213     /* Create a new session object */
214     params[0] = mp_obj_new_int(session_id);
215     obj = MP_OBJ_TYPE_GET_SLOT(&canvas_uwb_type_UwbSession,
216                               make_new)(&canvas_uwb_type_UwbSession, 1, 0, params);
217
218     return obj;
219 }
```



Home / UWB (canvas\_uwb)

## UWB (canvas\_uwb)

The modules described in this chapter provide the interface

`class canvas_uwb.UwbDeviceState(state: int)`

Helper class to represent the state of the UWB device.

This class should not be created directly. Instead,

`get_device_state()`.

`is_active() → bool`

## Canvas Python API:

[https://lairdcp.github.io/canvas\\_python\\_docs/](https://lairdcp.github.io/canvas_python_docs/)

# Connectivity First

Zephyr provides the connectivity we need!

- Zephyr Bluetooth LE stack is full featured and SIG qualified
- Networking: TCP, UDP, CoAP, LwM2M, MQTT, HTTP
- Security: TLS, secure boot
- Drivers: Modem, Ethernet, UWB, USB
- LoRa and LoRaWAN



# Production Ready with Firmware Updates!

- mcuboot bootloader
- Wired/wireless updates with mcumgr
  - USB/UART, Bluetooth LE
- Cloud
  - LwM2M/HTTP(s)

# Firmware Updates With Bluetooth LE

- SMP over Bluetooth LE
  - Image management – core firmware
  - File system management – update python apps
  - SMP GATT service is always present in addition to user's python app
  - Default script runs if no user app

```
# Enable the Bluetooth mcumgr transport (unauthenticated).
CONFIG_MCUMGR_TRANSPORT_BT=y
CONFIG_MCUMGR_TRANSPORT_BT_AUTHEN=n
CONFIG_MCUMGR_TRANSPORT_BT_CONN_PARAM_CONTROL=y
```

```
#define MP_DEFAULT_SCRIPT
    "import canvas_ble\n\n"
    "def ble_connect(conn):\n"
    "    print('Connected to', canvas_ble.addr_to_str(conn.get_addr()))\n\n"
    "def ble_disconnect(conn):\n"
    "    print('Disconnected from', canvas_ble.addr_to_str(conn.get_addr()))\n"
    "    adv.start()\n\n"
    "canvas_ble.init()\n"
    "canvas_ble.set_periph_callbacks(ble_connect, ble_disconnect)\n"
    "adv = canvas_ble.Advertiser()\n"
    "adv.set_interval(250, 300)\n"
    "adv.add_ltv(canvas_ble.AD_TYPE_FLAGS, bytes([0x06]), False)\n"
    "addr = canvas_ble.addr_to_str(canvas_ble.my_addr())[12:14]\n"
    "adv.add_tag_string(canvas_ble.AD_TYPE_NAME_COMPLETE, \"Canvas\" + addr, False)\n"
    "adv.add_smp_uuid(False)\n"
    "adv.add_canvas_data(0, 0, True)\n"
    "print('Advertising with SMP UUID')\n"
    "adv.start()\n"
```

# Firmware Updates From the Cloud

Watch my ZDS 2023 presentation on device management! [https://youtu.be/KB8TJAK\\_F7Y?si=IPkr6fpsNT8sroy2](https://youtu.be/KB8TJAK_F7Y?si=IPkr6fpsNT8sroy2)

- Use Zephyr LwM2M client
  - Firmware Update (Object 5)
  - Update core firmware
  - Software Management (Object 9)
  - HL7800 modem firmware
  - Python app



Zero configuration sample:  
[https://github.com/LairdCP/canvas\\_python\\_samples/tree/main/networking/lwm2m](https://github.com/LairdCP/canvas_python_samples/tree/main/networking/lwm2m)

Connects to public Leshan server:  
<https://leshan.eclipseprojects.io/#/clients>

# LwM2M Firmware Update Cont.

- User can control LwM2M from python
- Start LwM2M and firmware handles update functions
  - User doesn't need to worry about implementing complicated update logic

```
52  def event_cb(evt):
53      print("LwM2M event: {}".format(LWM2M_EVENTS[evt]))
54
55  # Configure the LWM2M client
56  lwm2m = Lwm2m(event_cb)
57  lwm2m.set_endpoint_name(endpoint_name)
58  lwm2m.set((lwm2m.OBJ_SECURITY, 0, 0), server_url)
59  lwm2m.set((lwm2m.OBJ_SECURITY, 0, 1), bootstrap)
60  lwm2m.set((lwm2m.OBJ_SECURITY, 0, 2), security_mode)
61
62  if security_mode == lwm2m.SECURITY_PSK:
63      lwm2m.set((lwm2m.OBJ_SECURITY, 0, 3), psk_id)
64      lwm2m.set((lwm2m.OBJ_SECURITY, 0, 5), psk)
65
66  if bootstrap == 0:
67      lwm2m.set((lwm2m.OBJ_SECURITY, 0, 10), 101)
68      lwm2m.create((lwm2m.OBJ_SERVER, 0))
69      lwm2m.set((lwm2m.OBJ_SERVER, 0, 0), 101)
70      lwm2m.set((lwm2m.OBJ_SERVER, 0, 1), 60)
71
72  # Configure the device object
73  lwm2m.create((lwm2m.OBJ_DEVICE, 0, 0), 32)
74  lwm2m.set((lwm2m.OBJ_DEVICE, 0, 0), "Laird Connectivity")
75  lwm2m.create((lwm2m.OBJ_DEVICE, 0, 3), 32)
76  lwm2m.set((lwm2m.OBJ_DEVICE, 0, 3), os.uname().release)
77  lwm2m.create((lwm2m.OBJ_DEVICE, 0, 17), 32)
78  lwm2m.set((lwm2m.OBJ_DEVICE, 0, 17), os.uname().machine)
79  lwm2m.set_exec_handler((lwm2m.OBJ_DEVICE, 0, 4), reboot_exec_cb)
80
81  # Wait for network to come up
82  print("Waiting for network")
83  net = NetHelper(None)
84  net.wait_for_ready()
85
86  print("Starting LwM2M with endpoint name: {}".format(endpoint_name))
87  # Start the LwM2M client
88  lwm2m.start(bootstrap != 0)
```

# Update Python App with LwM2M

Package your python app with our tool:

[https://github.com/LairdCP/canvas\\_software\\_packager/blob/main/canvas\\_packager.py](https://github.com/LairdCP/canvas_software_packager/blob/main/canvas_packager.py)

- Python tool to package a python app and version it
- The package contains a manifest of all files with SHA256 for validation
- Use LwM2M object 9 to download and install the package

```
1  [{"name": "lwm2m",
2   "version": "1.0.0",
3   "verify": "sha256",
4   "files": {
5     "main.py": "6e067f07a28c4aff65e12c21fce0735b5ea763f1c9c5eb6ada282d833d2cad64"
6   }
7 }
8 ]
9 ]
```

# Device Management

- LwM2M is a great protocol for device management
  - Our partner, [EdgelQ](#), offers a platform for managing your fleet of devices
  - The open-source [Eclipse Leshan project](#) can be used for testing or starting your own server



# Reliability – Automated Test Platform

- Automated testing with Robot Framework
- MicroPython REPL gives a standard way to interact with all devices

```
Connect Central
[Arguments]  ${primary_board}  ${board_adv_name}  ${phy}

${resp}=  User REPL Send  ${primary_board}  required_filter_name = "${board_adv_name}"
${resp}=  User REPL Send  ${primary_board}  required_phy = ${phy}
${resp}=  Run Script on Board  ${primary_board}  ${CENTRAL_SCAN_SCRIPT}
${resp}=  Convert To String  ${resp}
Should Contain  ${resp}  ${CENTRAL_SCAN_SCRIPT_START_RESP}

${total_time}=  Set Variable  ${20}
${result}=  Set Variable  ${False}
WHILE  ${total_time} > ${0}
    ${resp1}=  User REPL Send  ${primary_board}  print(found)
    ${resp1}=  Convert To String  ${resp1}
    ${resp1}=  Replace String  ${resp1}  \r\n  ${EMPTY}
    IF  ${resp1} == True
        ${result}=  Set Variable  ${True}
        BREAK
    ELSE
        Sleep  1s
        ${total_time}=  Evaluate  ${total_time} - 1
    END
IF  ${result} == False  Fail  Failed to connect
```

```

apps > bl654_usb > BleAd > BleAd.py > ...
scottlederer-lcl, 2 months ago | 1 author (scottlederer-lcl)
1 import canvas_ble as ble
2
3 ble_name = "Canvas BLE"
4 service_uuid = "11223344-5566-7788-99aa-bbccddeeef00"
5 char_uuid = "00000001-5566-7788-99aa-bbccddeeef00"
6
7 def cb_con(conn): # Connect callback
8     print("Connected")
9
10 def cb_discon(conn): # Disconnect callback
11     global advert
12     print("Disconnected")
13     advert.start()
14
15 def cb_write(event_object):
16     print('Received message: ' + event_object.data.decode())
17
18 def uuidToBytes(uuid): # Convert the 128 bit UUID string to bytes
19     uuid_bytes = list(bytes.fromhex(uuid.replace("-", "")))
20     uuid_bytes.reverse()
21     return bytes(uuid_bytes)
22
23 # GATT table definition for a service with a writable "message" characteristic
24 gatt_table = {
25     "Service 1":{
26         "Name": "My BLE Service",
27         "UUID": service_uuid,
28         "Characteristic 1":{
29             "Name": "message",
30             "UUID" : char_uuid,
31             "Length" : 20,
32             "Read Encryption" : "None",
33             "Write Encryption" : "None",
34             "Capability" : "Write",
35             "Callback" : cb_write
36         }
37     }
38 }
```

# A Look at Writing Python BLE Peripheral

```

39
40     # Initialize the BLE stack and set the connection and disconnection callbacks
41     ble.init()
42     ble.set_periph_callbacks(cb_con, cb_discon)
43
44     # Start advertising our primary service by the 128 bit UUID
45     advert = bleAdvertiser()
46     advert.stop()
47     advert.clear_buffer(False)
48     advert.add_ltv(1, bytes([6]), False)
49     advert.add_ltv(7, uuidToBytes(service_uuid), False)
50     advert.add_tag_string(9, ble_name, False)
51     advert.set_phys(ble.PHY_1M, ble.PHY_2M)
52     advert.set_properties(True, False, True)
53     advert.set_interval(240, 250)
54     advert.start()
55
56     # Define the gatt server and callbacks and start the GATT server
57     my_gattserver = bleGattServer()
58     my_gattserver.build_from_dict(gatt_table)
59     my_gattserver.start()
```

BleScan.py 1 ×

```
apps > bl654_usb > BleScan > BleScan.py > ...
scottlederer-lcl, 2 months ago | 1 author (scottlederer-lcl)
1 import canvas_ble as ble
2 import binascii
3 import time
4
5 service_uuid = "11223344-5566-7788-99aa-bbccddeeff00"
6 char_uuid = "00000001-5566-7788-99aa-bbccddeeff00"
7 peripheral_device_address = None
8 connection = None
9 gatt_client = None
10
11 def uuidToBytes(uuid): # Convert the 128 bit UUID string to bytes
12     uuid_bytes = list(bytes.fromhex(uuid.replace("-", "")))
13     uuid_bytes.reverse()
14     return bytes(uuid_bytes)
15
16 def scan_init():
17     global scanner
18     scanner = ble.Scanner(scan_cb)
19     # Filter ads that contain manufacturer ID 0x0077
20     scanner.filter_add(ble.Scanner.FILTER_UUID, uuidToBytes(service_uuid))
21     scanner.set_timing(100, 100) # Set scan to 100% duty cycle
22
23 def scan_cb(sr):
24     global peripheral_device_address
25     print("DID={},RS={},AD={}".format(
26         binascii.hexlify(sr.addr).decode(),
27         sr.rssi, binascii.hexlify(sr.data).decode()))
28     peripheral_device_address = sr.addr
29
30 def send(message):
31     global gatt_client
32     global char_uuid
33     gatt_client.write(char_uuid, message)
34
```

# Writing Python

## BLE Central

```
35 def connection_cb(conn):
36     global connection
37     global gatt_client
38     print("Connection established with {}".format(
39         ble.addr_to_str(conn.get_addr())))
40     connection = conn
41     gatt_client = ble.GattClient(conn)
42     gatt_client.discover()
43     print('Discovering services...')
44     send(b'Hello, BLE!')
45     print('Writing to characteristic...')
46
47 def disconnection_cb(conn):
48     print("Connection closed with {}".format(
49         ble.addr_to_str(conn.get_addr())))
50
51 ble.init()
52 scan_init()
53 scanner.start(1)
54 print('Scanning for peripheral devices for 2 seconds...')
55 time.sleep_ms(2000)
56 scanner.stop()
57
58 if peripheral_device_address is not None:
59     print("Client found, attempting connection to", ble.addr_to_str(peripheral_device_address))
60     ble.connect(peripheral_device_address, ble.PHY_2M, connection_cb, disconnection_cb)
```

```

44
45     def scan_cb(evt):
46         global BLE_LED, num_tracked_devices
47
48         # Get manufacturer-specific data from the advertisement
49         msg = canvas_ble.find_ltv(0xff, evt.data)
50         if msg is None:
51             return
52         msg = msg[4:].decode()
53
54         name = canvas_ble.find_ltv(9, evt.data)
55         if name is None:
56             return
57         name = name.decode()
58
59         addr = binascii.hexlify(evt.addr).decode()
60         # Check if we've seen this device before
61         if addr not in devices:
62             try:
63                 devices[addr] = {'deviceId': addr}
64                 devices[addr]['timestamp'] = time.time()
65                 devices[addr]['message'] = ''
66                 devices[addr]['name'] = name
67                 num_tracked_devices += 1
68                 print("Tracking device: {} ({})".format(addr, num_tracked_devices))
69             except:
70                 print("Error tracking {}. {} tracked.".format(
71                     addr, num_tracked_devices))
72             try:
73                 del devices[addr]
74             except:
75                 pass
76             return
77
78         now = time.time()
79         if now - devices[addr]['timestamp'] < DEVICE_MSG_LIMIT_TIME_SEC:
80             return
81
82         last_message = devices[addr]['message']
83         if last_message != msg and len(publish_list) < (MAX_MSG_PUBLISH - 1):
84             BLE_LED.flash()
85             devices[addr]['name'] = name
86             devices[addr]['timestamp'] = time.time()
87             devices[addr]['message'] = msg
88             print("Received: {}".format(devices[addr]))
89             publish_list.append(devices[addr].copy())

```

# Writing Python

## BLE to the Cloud!

```

92     def mqtt_publish_timer(event):
93         publish = False
94         msg = {
95             'message': {
96                 'messages': []
97             }
98         }
99         while len(publish_list) > 0:
100             publish = True
101             device = publish_list.pop(0)
102             msg['message']['messages'].append(device)
103
104             if publish:
105                 msg_str = json.dumps(msg)
106                 print("Publishing: {}".format(msg_str))
107                 client.publish(msg_str)
108

```

# Future Features

- MicroPython threading with Zephyr threads
- More flexible firmware updates
  - User can download images via a transport of their choice
- Encrypted file system
- Bluetooth LE secure connections
- Memfault – Crash analytics and device metrics

# Summary

- Development is faster with virtually no setup time.
- Production level code without writing in C.
- Putting all the firmware pieces together is hard. Customers no longer need to worry.

# Questions?

---

Ryan Erickson

[Ryan.Erickson@ezurio.com](mailto:Ryan.Erickson@ezurio.com)

[Ezurio Discord Channel!](#)