Canvas AC Current Sensor Demo

BT610

Application Note *v1.0*

1 Introduction

This guide describes the steps required to program a Sentrius BT610 sensor with the python-capable Canvas Firmware for rapid application development with sensors and demonstrates an AC Current Sensor application.

2 Overview

This document will go over all the steps necessary to program the BT610, wire the hardware and run a simple Xbit applet demonstration for Proof of Concept. In most cases end users will need to implement a custom application for collecting and displaying the data from the BT610 Bluetooth Advertisements.

1. Flashing the BT610 with Canvas Firmware
2. Loading Python script to the BT610 via the UART pins
3. Hardware setup
4. Software Setup for Xbit demo

3 Requirements

- BT610 Bluetooth IoT Sensor w/3.6V Lithium Thionyl Chloride AA Battery (included with sensor)
  - Note: Alkaline or similar 1.5V AA batteries will NOT work with this device.
- USB-SWD Programming Kit or Segger J-Link Debugger for programming HEX file
  - (Optional) Male-to-Female jumper wires to wire UART from USB-SWD to BT610

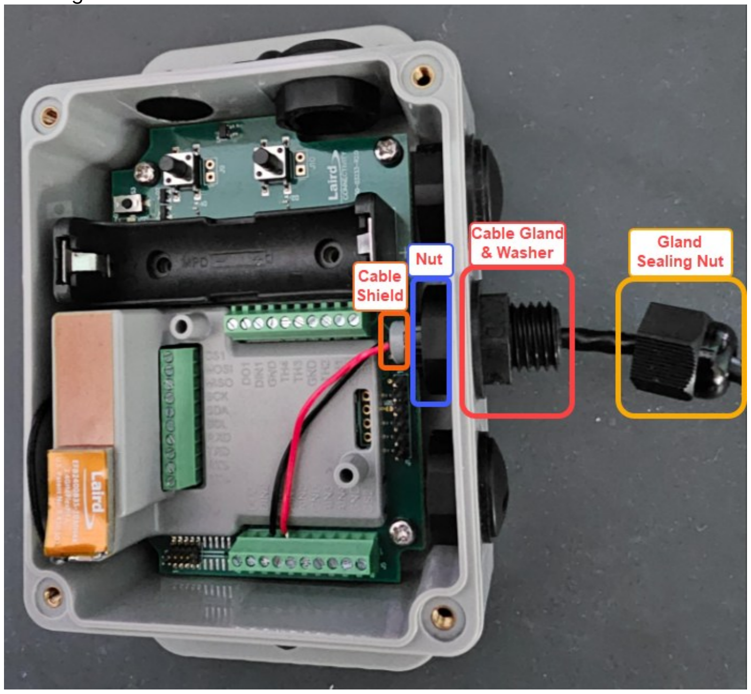
- Analog AC Current Sensor (0-20 AMP RMS)
- Canvas Firmware (BT610)
- VS Code with Xbit Extension
- Python
- AC Current Python Script
- (Optional) BL654-USB Dongle with Canvas Firmware and Xbit Scripts loaded
- (Optional) Xbit Software and Applet to display data
- (Optional) TTL-232R-3V3 Cable +3.3V USB, if preferred over USB-SWD, to connect to UART

4 Connect AC Current Sensor

4.1 Running AC Current Sensor Cable into the BT610

Once the cable gland/seal, washer, and nut are fitted, run the applicable sensor cable through the connector by doing the following:

1. Fit cable gland & washer with the nut to the housing, then pass the cable through the gland sealing nut and into the housing through the cable bladder then connect the wires to the terminal as shown in the image below. Note: Make sure the sensor outer cable shield extends through the gland and into the housing to ensure a tight seal to maintain the IP67 rating. The exposed wires must be fully located inside the housing.



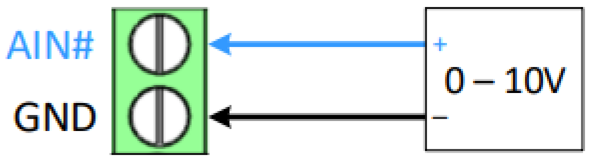
2. Tighten the gland sealing nut to hold the cable in place until the initial internal cable routing is determined. **Do not fully tighten the gland sealing nut until the wires are attached to the proper terminals**.

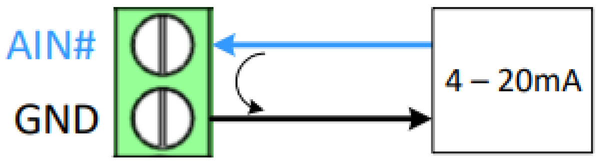
4.2 Connecting Sensor to Terminal Block of BT610

The Sentrius™ BT610 provides four analog input (AIN) ports for connecting to analog sensors/meters.

Because the BT610 cannot detect the type of analog input device connected, you must properly connect each sensor and then **configure the analog input type**, **measurement parameters**, **reporting method**, and **name** for each port **in the Python script**.

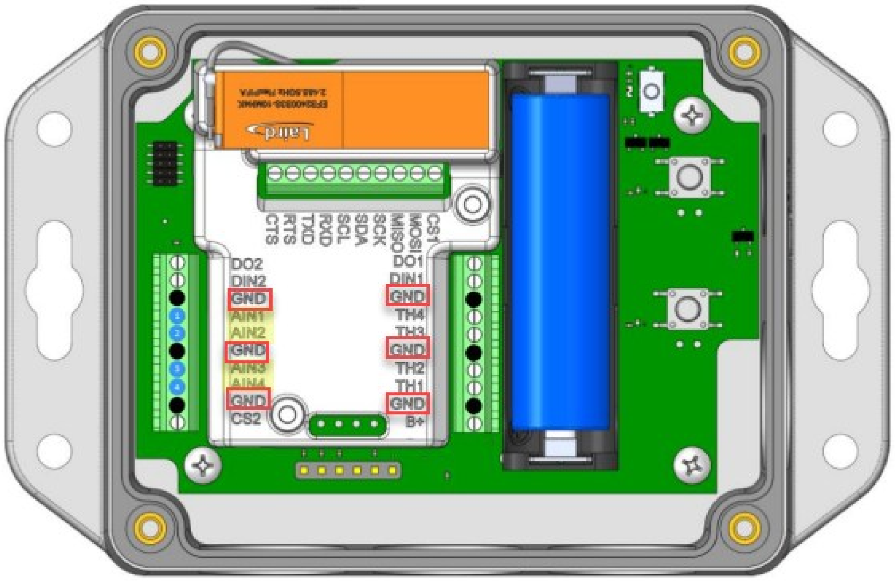
Each analog sensor consists of two wire connections to the terminal block: AIN#, and GND (ground). The analog connections must use the following voltage polarity or current flow direction to operate properly.





Note: Reverse polarity connections will be reported as a voltage of 0V or current of 0mA.
The BT610 is a passive (2-wire receiver) current sensor only. It cannot provide power for the current loop.

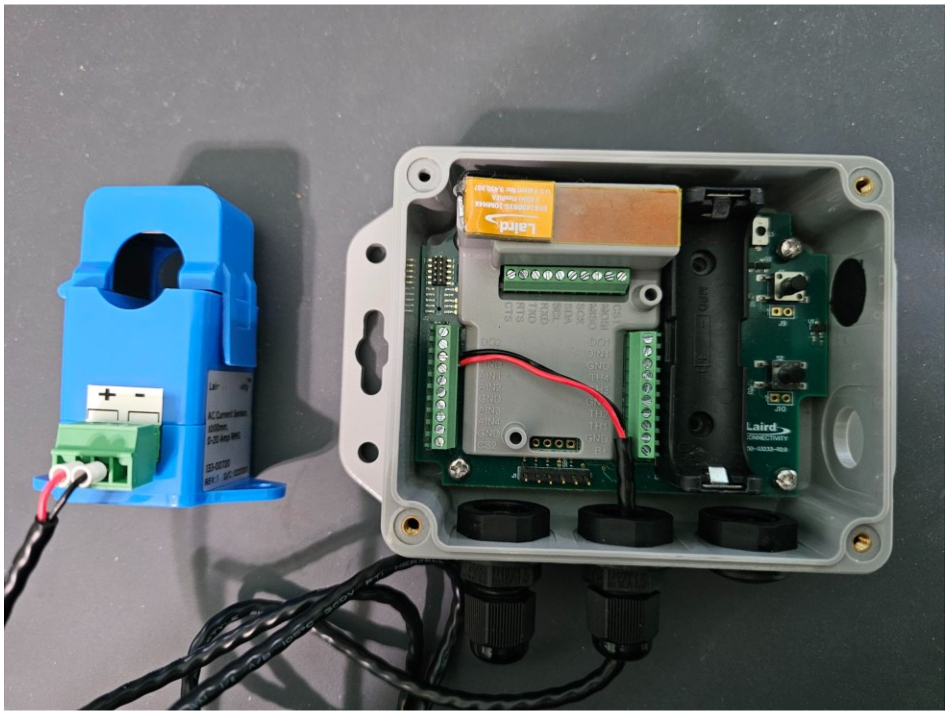
You may use any of the four analog input ports [AIN1:AIN4] and six ground ports [GND] indicated in Figure below.



The example below shows a single current sensor connected to AIN1 and GND by inserting the wire connected to + on the sensor (red) to AIN1 and the wire connected to – on the sensor (black) to GND.

To connect the wire, insert it into the terminal opening under the screw and then tighten the screw to secure the connection.

Repeat for any additional sensor connections (AIN2 – AIN4)



5 Flashing Canvas Firmware on the BT610

The BT610 can be programmed over SWD connection using the appropriate 10 pin ribbon cable and pyocd programming tool. We recommend using the Ezurio USB-SWD programmer [A] , which comes with the necessary ribbon cable to connect with and program the BT610 over the SWD header.

Note: Alternatively, the BT610 can be flashed using a Segger JLink debugger (not covered in this Application Note).

5.1 Hardware Configuration for Flashing Canvas Firmware (Hex) via SWD Interface

1. Connect from the BT610 to the USB-SWD programming kit using the 10 pin SWD ribbon cable. Ensure the red line on the cable lines up with pin 1 of the SWD port on the BT610 device as shown below.

Note: Battery should be removed when connecting the ribbon cable to the BT610 and re-inserted **before** connecting the USB cable from the USB-SWD to the USB port on the computer.

3.8 Programmer Port

A 9-way Cortex-M Debug connector is available for programming and debugging of the nRF52840 MCU. This is located as shown in Figure 21. Further details of the connector and a suitable programmer can be found at Appendix A – References [A].

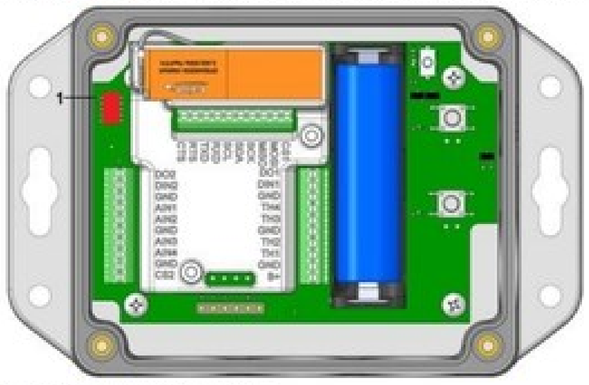
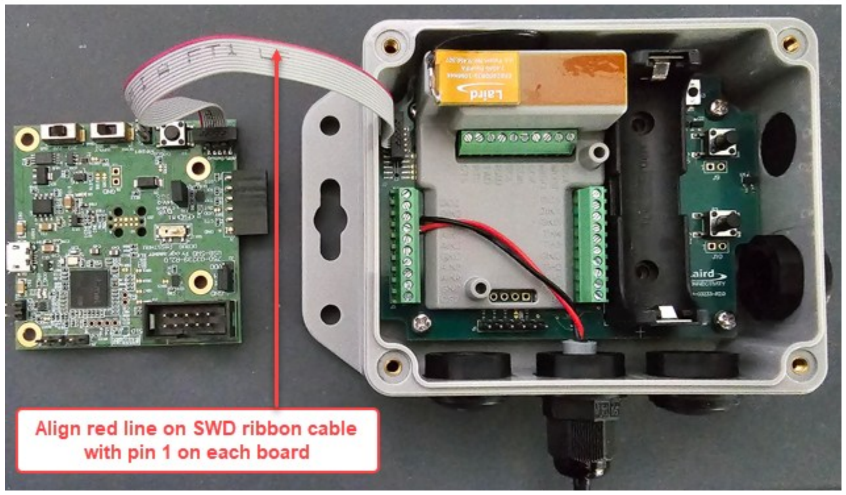


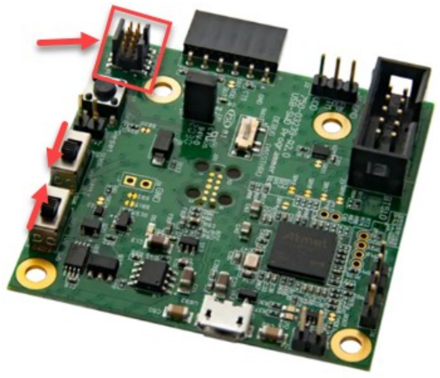
Figure 21: Programming Port connections

Important! Orientation of Programmer cable must be observed as indicated in Figure 21.



2. The SW5 switch on the USB-SWD should be in the 3V3 position.
3. The SW6 switch on the USB-SWD should be set to Supply Out position.

4. **Ensure battery is re-inserted into the BT610 to power the device properly first,** then connect the USB-SWD to a USB port on a computer with USB cable.



5.2 Flashing Firmware

IMPORTANT: Please make sure to always use the latest Canvas Firmware release

Canvas Firmware is the underlying software enabling Canvas-enabled radio modules to run Python scripts and access hardware via scripting APIs. It is important to make sure your hardware is using the latest Canvas Firmware before you start developing application scripts. See the product page for details on where to locate the latest Canvas Firmware for your specific hardware.

Steps required to program a Canvas firmware image .hex file onto the BT610:

1. Install Python 3.11.6 (or newer)
2. Install or update pip if not installed (usually pip is automatically installed with Python)
3. Install **pyocd** (using pip)
4. Open a command prompt from the folder the hex file is saved to.
5. Use pyocd from the cmd prompt to program the Canvas "**.hex"** firmware file to the BT610.

```
pyocd flash -t nrf52840 -e chip merged_x.x.xx.xxxxxxxxxx.hex
```

Note: `-t nrf52840` selects the target hardware, `-e chip` erases the chip.

6. Use pyocd to reset the BT610:

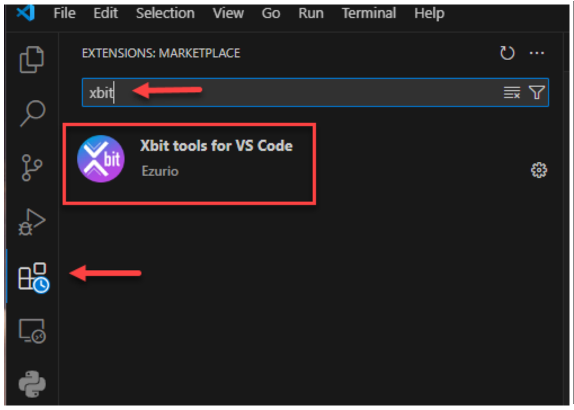
```
pyocd reset -t nrf52840
```

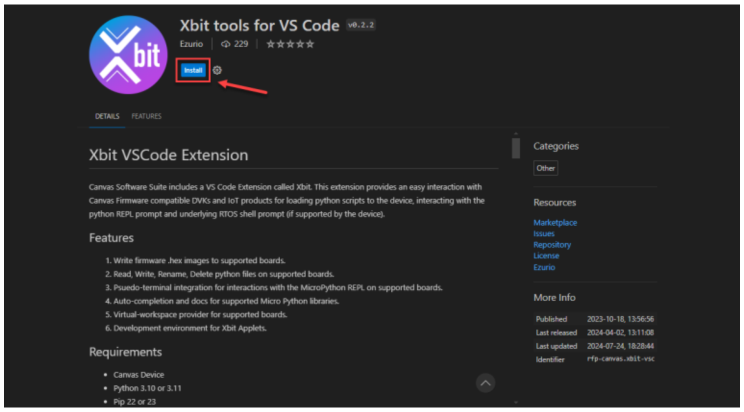
7. Disconnect the ribbon cable between the BT610 and USB-SWD programmer

6 Loading Python Script to the BT610 over the UART

6.1 Installing VS Code and Xbit Extension

1. Download and install VS Code.
2. To install the Xbit VS Code Extension, open VS Code, navigate to the Extensions tab and type “xbit” in the extension search bar. You should see Xbit tools for VS Code listed. Select it from the list and press the install button to install the extension into your VS Code environment.





Xbit VSCode Extension

Canvas Software Suite includes a VS Code Extension called Xbit. This extension provides an easy interaction with Canvas Firmware compatible DVKs and IoT products for loading python scripts to the device, interacting with the python REPL prompt and underlying RTOS shell prompt (if supported by the device).

Features

1. Write firmware .hex images to supported boards.
2. Read, Write, Rename, Delete python files on supported boards.
3. Psuedo-terminal integration for interactions with the MicroPython REPL on supported boards.
4. Auto-completion and docs for supported Micro Python libraries.
5. Virtual-workspace provider for supported boards.
6. Development environment for Xbit Applets.

Requirements

- Canvas Device
- Python 3.10 or 3.11
- Pip 22 or 23

Categories

Other

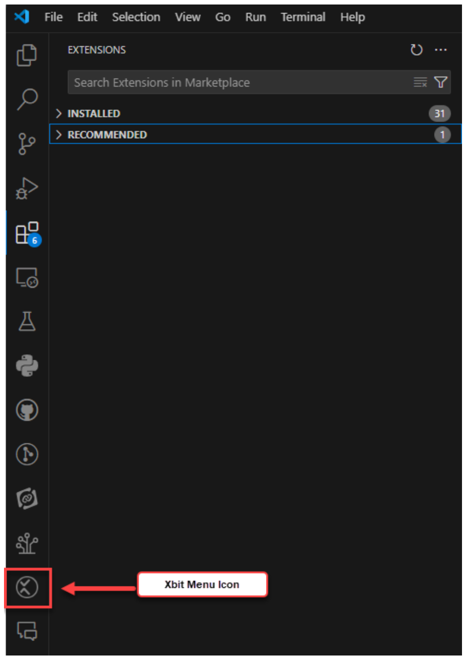
Resources

Marketplace
Issues
Repository
License
Ezurio

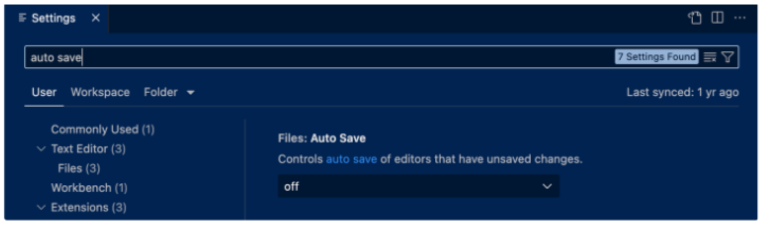
More Info

Published	2023-10-18, 13:56:56
Last released	2024-04-02, 13:11:08
Last updated	2024-07-24, 18:28:44
Identifier	rfp-canvas.xbit-vsc

After installation, you’ll see a new menu icon on the left-hand side of VS Code for the Xbit VSC extension. Click the new icon to start the Xbit Extension.



Note: Be sure to disable Auto-Save in VS-Code for working with python files on a Canvas enabled device.



Files: Auto Save

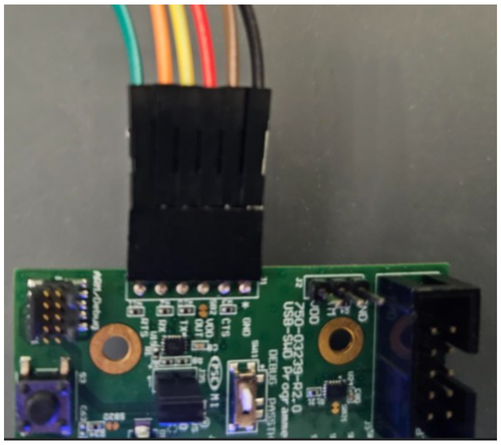
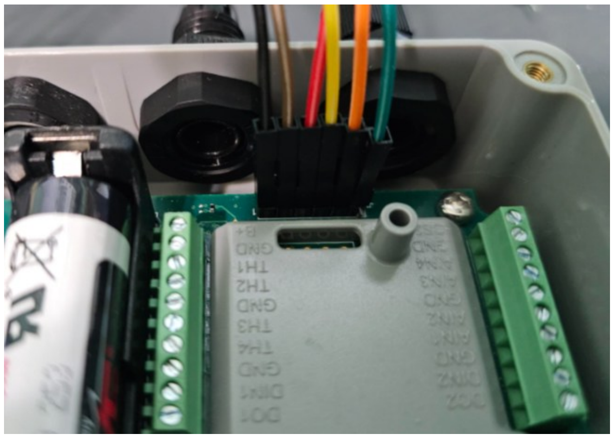
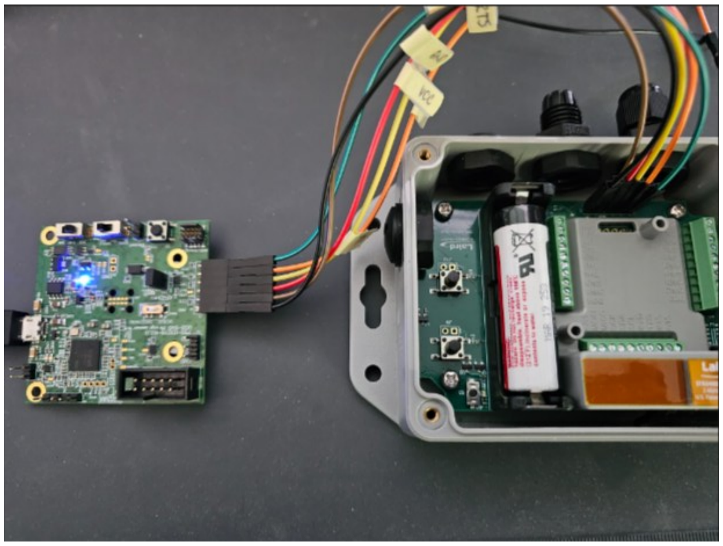
Controls auto save of editors that have unsaved changes.

off

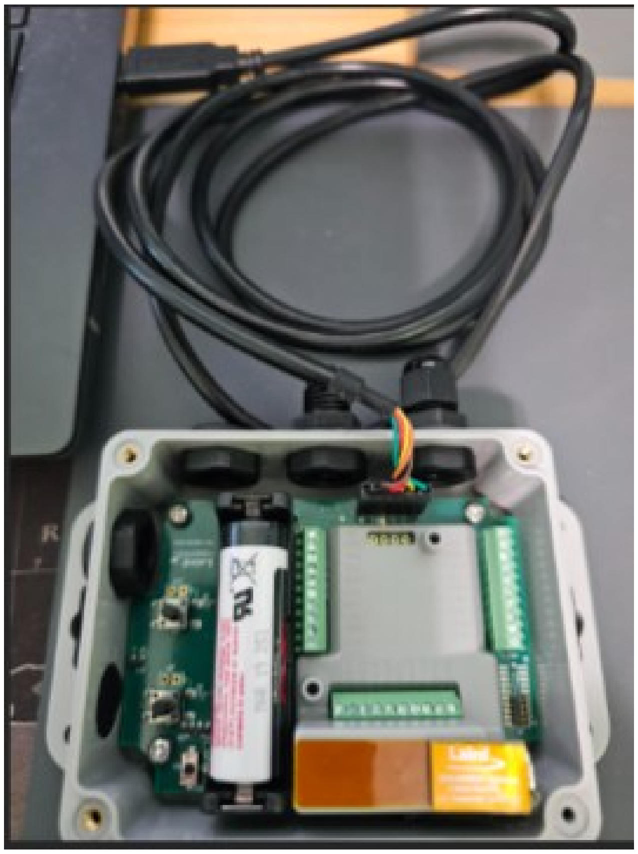
6.2 Loading a Python Script to the BT610

6.2.1 Hardware Configuration for Loading Python Scripts via UART

1. Ensure the USB-SWD cable is disconnected from the computer (no power). Battery should remain inserted in the BT610.
2. Connect USB-SWD from J1 UART header to the J1 UART header of the BT610 as shown below. (Green-CTS, Orange-RX, Yellow-TX, Red-VCC, Brown-RTS, Black-GND).
3. Connect USB-SWD via USB cable to PC to power and connect the devices.

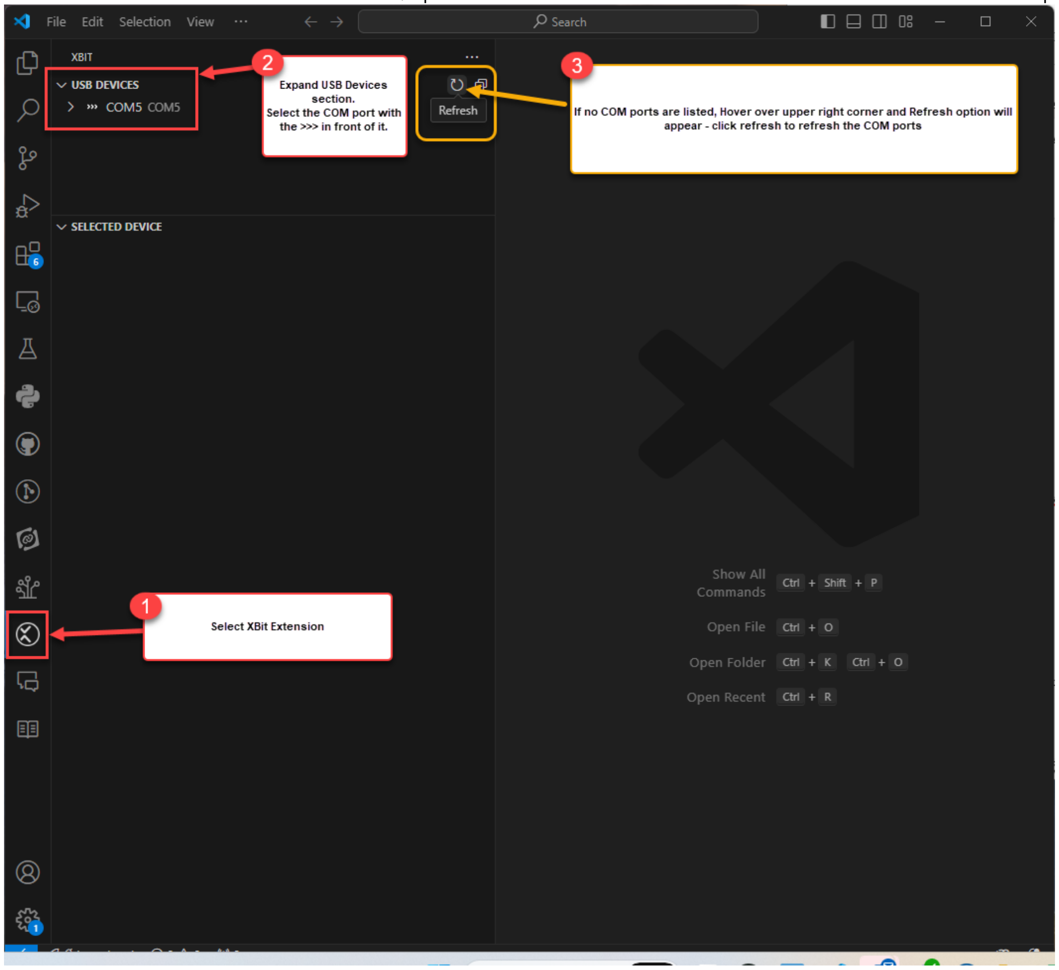




4. (**Alternative-Optional**) TTL-232R-3V3 Cable +3.3V USB cable can be used to connect directly from BT610 to USB port on the computer. (USB-SWD not required for UART connection.)



6.2.2 Loading Python Script

1. With the Device connected via the UART, open VS Code and select the Xbit extension and select the COM port.

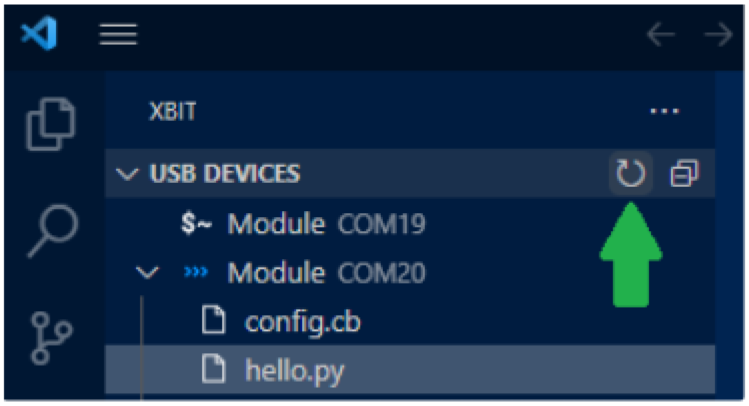


1 Select XBit Extension

2 Expand USB Devices section. Select the COM port with the >>> in front of it.

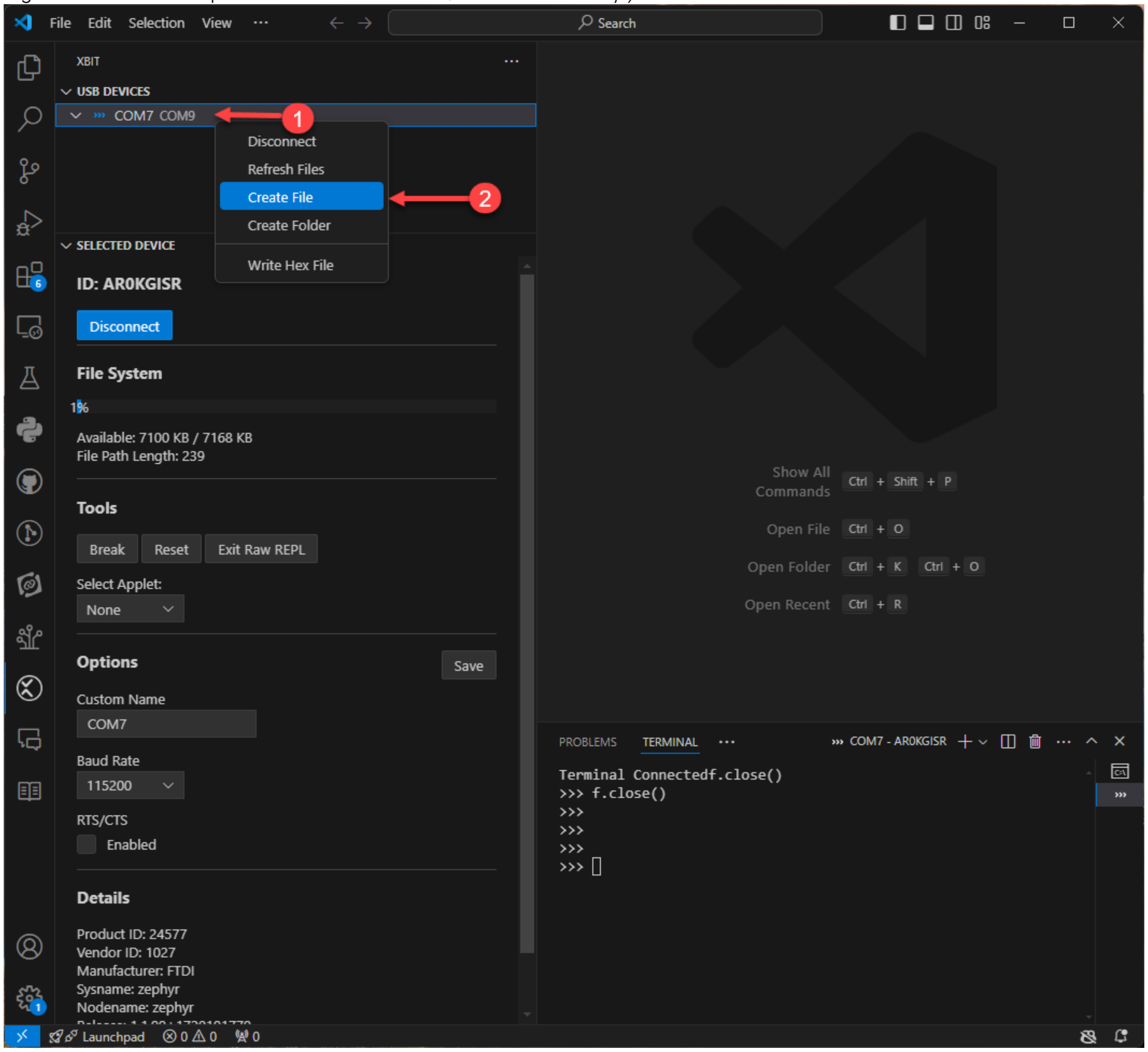
3 If no COM ports are listed, Hover over upper right corner and Refresh option will appear - click refresh to refresh the COM ports

2. The >>> symbol indicates this is the REPL, which is where you will add Python application scripts. If a USB serial port is detected but the identity of the port cannot be determined, you'll see a generic device listing identified by a default USB icon. At any time you can refresh the list of devices by clicking the refresh button in the USB DEVICES header bar.



3. The ***bt610_ac_current.py*** script provides a proof-of-concept approach to reading a 20A rated AC current sensor on channel AIN1 of a BT610 running Canvas Firmware. There are two options in VS Code for loading the script to the BT610.

A. Right click on the COM port and select Create File, and name it *main.py*.



ID: AR0KGISR

File System

1%

Available: 7100 KB / 7168 KB
File Path Length: 239

Select Applet: None

Options

Custom Name: COM7

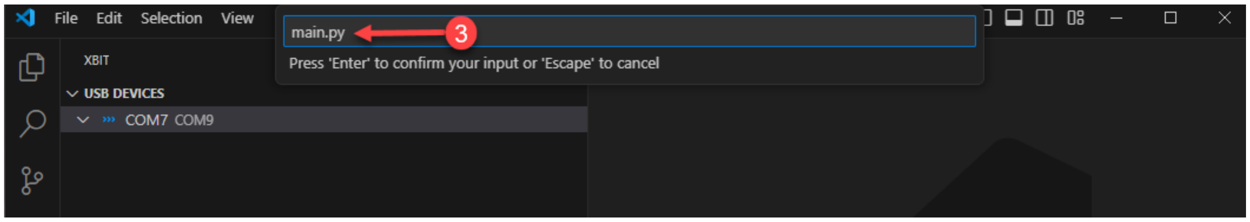
Baud Rate: 115200

RTS/CTS: Enabled

Details

Product ID: 24577
Vendor ID: 1027
Manufacturer: FTDI
Sysname: zephyr
Nodename: zephyr

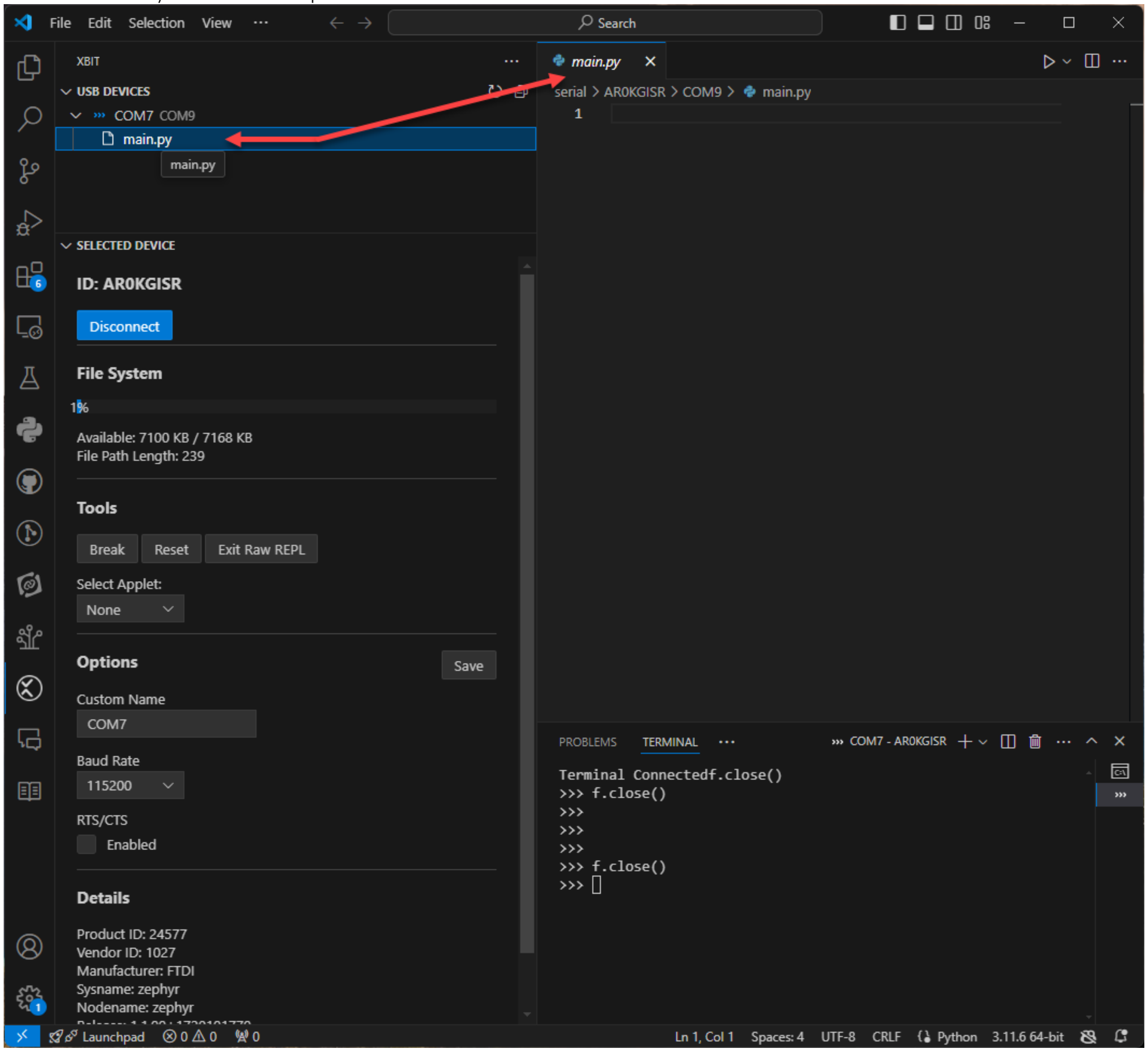
```
Terminal Connectedf.close()
>>> f.close()
>>>
>>>
>>>
>>>
```



main.py

Press 'Enter' to confirm your input or 'Escape' to cancel

Click on the newly created file to open a window in VS Code where the file can be edited.



ID: AR0KGISR

File System

1%

Available: 7100 KB / 7168 KB
File Path Length: 239

Tools

Select Applet: None

Options

Custom Name: COM7

Baud Rate: 115200

RTS/CTS: Enabled

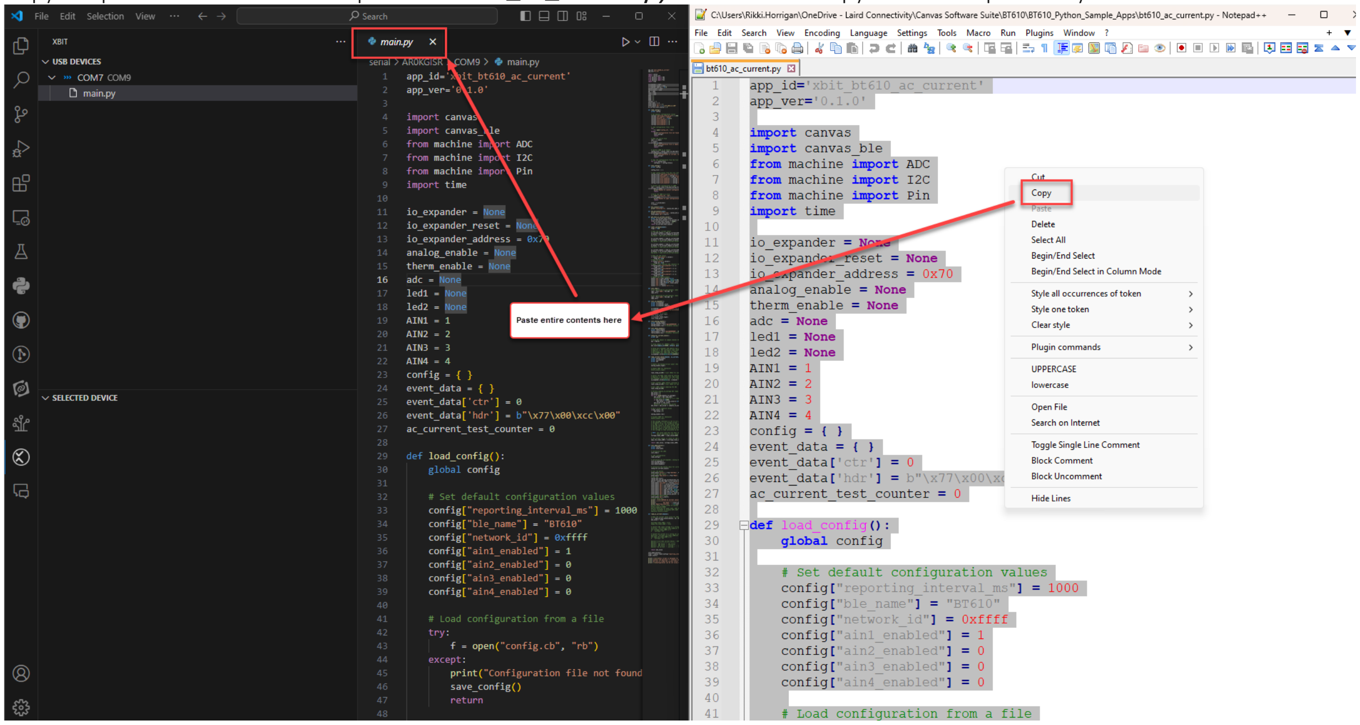
Details

Product ID: 24577
Vendor ID: 1027
Manufacturer: FTDI
Sysname: zephyr
Nodename: zephyr

```
Terminal Connectedf.close()
>>> f.close()
>>>
>>>
>>>
>>> f.close()
>>>
```

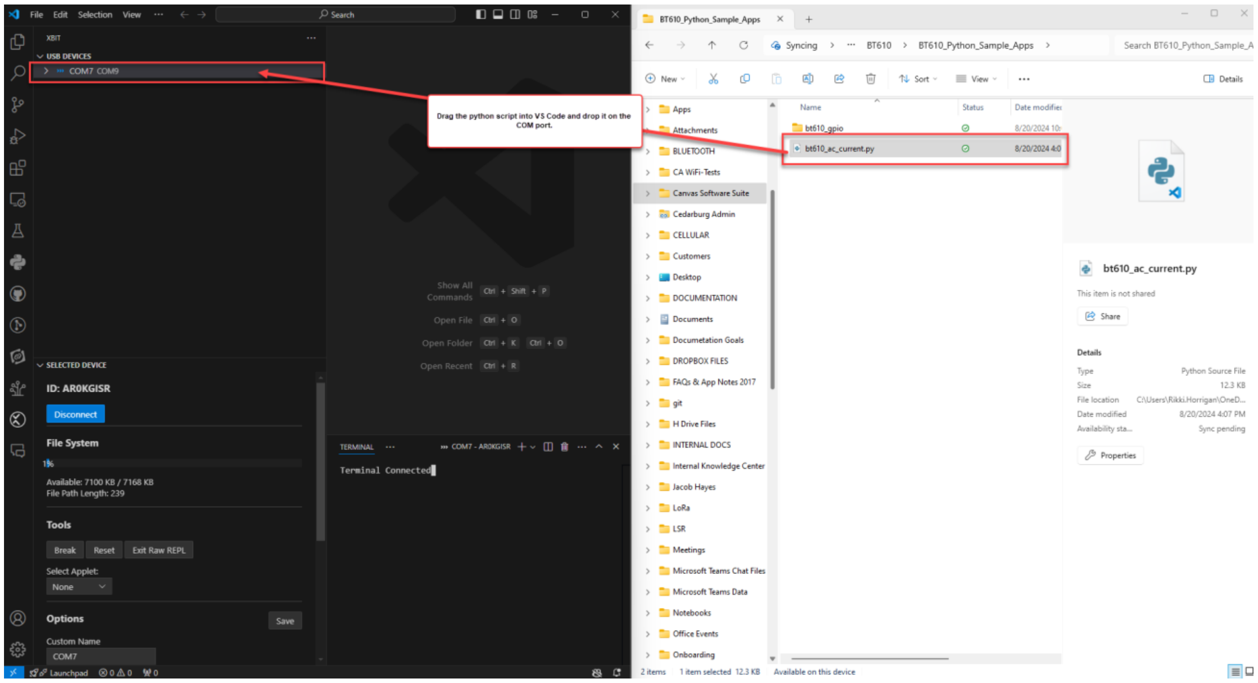
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.6 64-bit

Copy and paste the contents of the provided ***bt610_ac_current.py*** file into the main.py file created previously.



```python
app_id='xbit_bt610_ac_current'
app_ver='0.1.0'

import canvas
import canvas_ble
from machine import ADC
from machine import I2C
from machine import Pin
import time

io_expander = None
io_expander_reset = None
io_expander_address = 0x70
analog_enable = None
therm_enable = None
adc = None
led1 = None
led2 = None
AIN1 = 1
AIN2 = 2
AIN3 = 3
AIN4 = 4
config = { }
event_data = { }
event_data['ctr'] = 0
event_data['hdr'] = b"\x77\x00\xcc\x00"
ac_current_test_counter = 0

def load_config():
    global config

    # Set default configuration values
    config["reporting_interval_ms"] = 1000
    config["ble_name"] = "BT610"
    config["network_id"] = 0xffff
    config["ain1_enabled"] = 1
    config["ain2_enabled"] = 0
    config["ain3_enabled"] = 0
    config["ain4_enabled"] = 0

    # Load configuration from a file
    try:
        f = open("config.cb", "rb")
    except:
        print("Configuration file not found
        save_config()
        return

```

B. (**Alternative method**) In VS Code, drag and drop the ***bt610_ac_current.py*** file onto the COM Port in VS Code and then rename the file main.py.

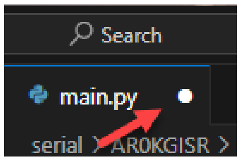


```
Terminal Connected
```

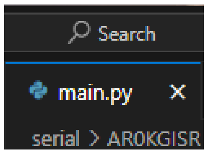
Then rename the file main.py so that it will run on startup/reset.

4. Save the file by typing **CTRL+s**

Note: The file name will have a small white dot next to it when the file has NOT been saved and an X when it has been saved:

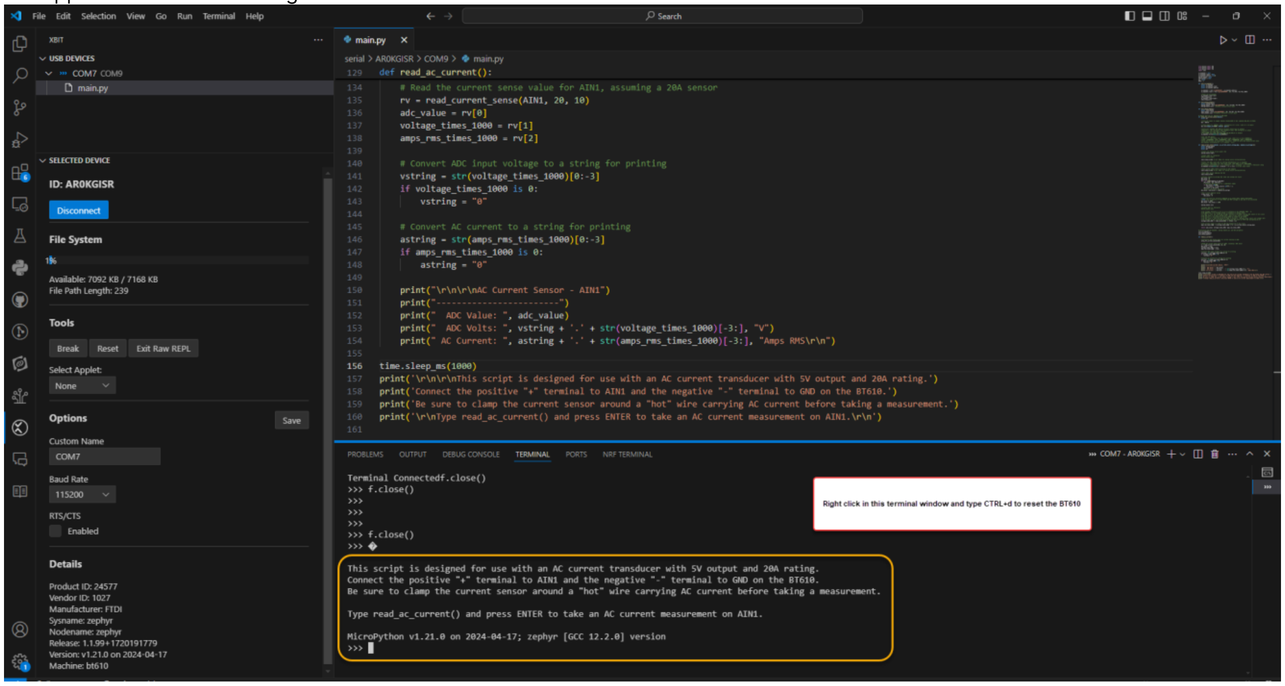


FILE NOT SAVED



FILE SAVED

5. Reset the BT610 to start the application, which will now run by default on power-up/reset because the file is named main.py. To reset, click in the terminal window below the script window and type **CTRL+d**. You will see an explanation of the demo output over the UART, indicating the application is now running.



Available: 7092 KB / 7168 KB
File Path Length: 239

Select Applet: None

Custom Name: COM7
Baud Rate: 115200
RTS/CTS: Enabled

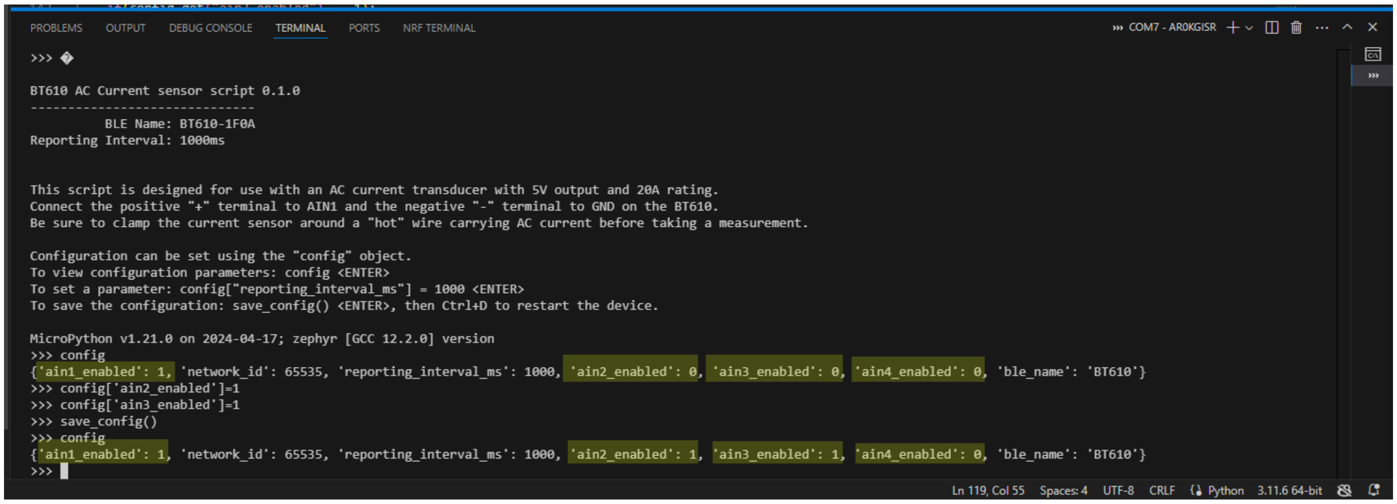
Product ID: 24577
Vendor ID: 1027
Manufacturer: FTDI
Sysname: zephyr
Nodename: zephyr
Release: 1.1.99+1720191779
Version: v1.21.0 on 2024-04-17
Machine: bt610

```python
def read_ac_current():
    # Read the current sense value for AIN1, assuming a 20A sensor
    rv = read_current_sense(AIN1, 20, 10)
    adc_value = rv[0]
    voltage_times_1000 = rv[1]
    amps_rms_times_1000 = rv[2]

    # Convert ADC input voltage to a string for printing
    vstring = str(voltage_times_1000)[0:-3]
    if voltage_times_1000 is 0:
        vstring = "0"

    # Convert AC current to a string for printing
    astring = str(amps_rms_times_1000)[0:-3]
    if amps_rms_times_1000 is 0:
        astring = "0"

    print("\r\n\r\nAC Current Sensor - AIN1")
    print("------------------------")
    print("  ADC Value: ", adc_value)
    print("  ADC Volts: ", vstring + '.' + str(voltage_times_1000)[-3:], "V")
    print(" AC Current: ", astring + '.' + str(amps_rms_times_1000)[-3:], "Amps RMS\r\n")

time.sleep_ms(1000)
print('\r\n\r\nThis script is designed for use with an AC current transducer with 5V output and 20A rating.')
print('Connect the positive "+" terminal to AIN1 and the negative "-" terminal to GND on the BT610.')
print('Be sure to clamp the current sensor around a "hot" wire carrying AC current before taking a measurement.')
print('\r\nType read_ac_current() and press ENTER to take an AC current measurement on AIN1.\r\n')
```

```
Terminal Connectedf.close()
>>> f.close()
>>>
>>>
>>>
>>> f.close()
>>> �
This script is designed for use with an AC current transducer with 5V output and 20A rating.
Connect the positive "+" terminal to AIN1 and the negative "-" terminal to GND on the BT610.
Be sure to clamp the current sensor around a "hot" wire carrying AC current before taking a measurement.

Type read_ac_current() and press ENTER to take an AC current measurement on AIN1.

MicroPython v1.21.0 on 2024-04-17; zephyr [GCC 12.2.0] version
>>> █
```

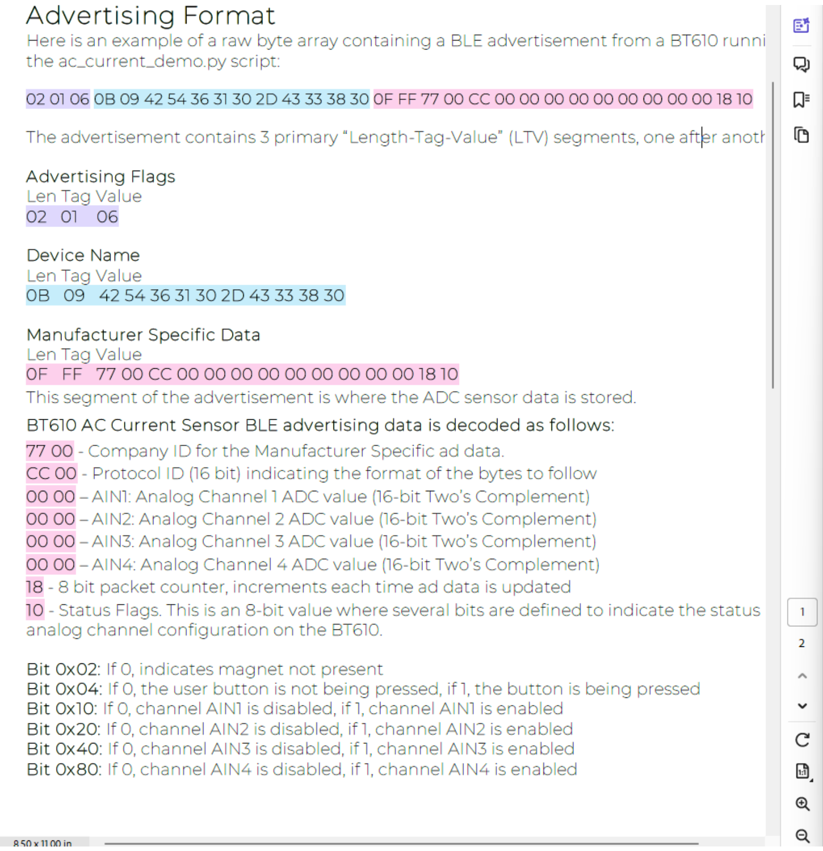
6. By default, only AIN1 is enabled (set to 1). To enable additional analog sensors, you will need to use the following commands:
   - Enter `config` to see the current configurations.
   - Enter `config['ain2_enabled']=1`
   - Enter `config['ain3_enabled']=1`
   - Enter `save_config()`
   - Enter `config` to see the new configurations with AIN2 and AIN 3 now enabled.

```
>>> config
{'ain1_enabled': 1, 'network_id': 65535, 'reporting_interval_ms': 1000, 'ain2_enabled': 0,
'ain3_enabled': 0, 'ain4_enabled': 0, 'ble_name': 'BT610'}
>>> config['ain2_enabled']=1
>>> config['ain3_enabled']=1
>>> save_config()
>>> config
{'ain1_enabled': 1, 'network_id': 65535, 'reporting_interval_ms': 1000, 'ain2_enabled': 1,
'ain3_enabled': 1, 'ain4_enabled': 0, 'ble_name': 'BT610'}
>>>
```



```
>>> �

BT610 AC Current sensor script 0.1.0
----------------------------
          BLE Name: BT610-1F0A
Reporting Interval: 1000ms


This script is designed for use with an AC current transducer with 5V output and 20A rating.
Connect the positive "+" terminal to AIN1 and the negative "-" terminal to GND on the BT610.
Be sure to clamp the current sensor around a "hot" wire carrying AC current before taking a measurement.

Configuration can be set using the "config" object.
To view configuration parameters: config <ENTER>
To set a parameter: config["reporting_interval_ms"] = 1000 <ENTER>
To save the configuration: save_config() <ENTER>, then Ctrl+D to restart the device.

MicroPython v1.21.0 on 2024-04-17; zephyr [GCC 12.2.0] version
>>> config
{'ain1_enabled': 1, 'network_id': 65535, 'reporting_interval_ms': 1000, 'ain2_enabled': 0, 'ain3_enabled': 0, 'ain4_enabled': 0, 'ble_name': 'BT610'}
>>> config['ain2_enabled']=1
>>> config['ain3_enabled']=1
>>> save_config()
>>> config
{'ain1_enabled': 1, 'network_id': 65535, 'reporting_interval_ms': 1000, 'ain2_enabled': 1, 'ain3_enabled': 1, 'ain4_enabled': 0, 'ble_name': 'BT610'}
>>>
```

7. View advert using any central role application such as nRF Connect on a mobile device.



Advertising Format

Here is an example of a raw byte array containing a BLE advertisement from a BT610 runni
the ac_current_demo.py script:

02 01 06 0B 09 42 54 36 31 30 2D 43 33 38 30 0F FF 77 00 CC 00 00 00 00 00 00 00 00 18 10

The advertisement contains 3 primary "Length-Tag-Value" (LTV) segments, one after anoth

Advertising Flags

Len	Tag	Value
02	01	06

Device Name

Len	Tag	Value
0B	09	42 54 36 31 30 2D 43 33 38 30

Manufacturer Specific Data

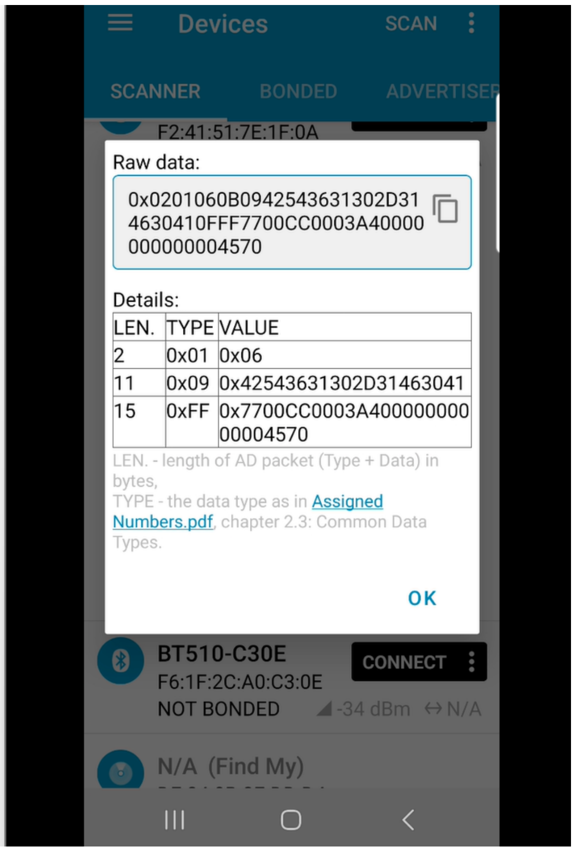
Len	Tag	Value
0F	FF	77 00 CC 00 00 00 00 00 00 00 00 00 18 10

This segment of the advertisement is where the ADC sensor data is stored.

BT610 AC Current Sensor BLE advertising data is decoded as follows:

77 00 - Company ID for the Manufacturer Specific ad data.
CC 00 - Protocol ID (16 bit) indicating the format of the bytes to follow
00 00 – AIN1: Analog Channel 1 ADC value (16-bit Two's Complement)
00 00 – AIN2: Analog Channel 2 ADC value (16-bit Two's Complement)
00 00 – AIN3: Analog Channel 3 ADC value (16-bit Two's Complement)
00 00 – AIN4: Analog Channel 4 ADC value (16-bit Two's Complement)
18 - 8 bit packet counter, increments each time ad data is updated
10 - Status Flags. This is an 8-bit value where several bits are defined to indicate the status
analog channel configuration on the BT610.

Bit 0x02: If 0, indicates magnet not present
Bit 0x04: If 0, the user button is not being pressed, if 1, the button is being pressed
Bit 0x10: If 0, channel AIN1 is disabled, if 1, channel AIN1 is enabled
Bit 0x20: If 0, channel AIN2 is disabled, if 1, channel AIN2 is enabled
Bit 0x40: If 0, channel AIN3 is disabled, if 1, channel AIN3 is enabled
Bit 0x80: If 0, channel AIN4 is disabled, if 1, channel AIN4 is enabled



Raw data:

0x0201060B0942543631302D31
4630410FFF7700CC0003A40000
000000004570

Details:

LEN.	TYPE	VALUE
2	0x01	0x06
11	0x09	0x42543631302D31463041
15	0xFF	0x7700CC0003A40000000000004570

LEN. - length of AD packet (Type + Data) in bytes,
TYPE - the data type as in Assigned Numbers.pdf, chapter 2.3: Common Data Types.

Note: The example Python scripts we provide support Advertising Extensions and 2M PHY as secondary PHY option. If this is not supported on devices scanning for adverts some modifications to the script may be required. Contact Ezurio Support for assistance with this.

7 (Optional) XBit Demonstration

7.1.1 Download and install Xbit for Desktop application

The Xbit standalone desktop application provides a framework developers can use to quickly interact and visualize data from Bluetooth Low Energy devices. 'Xbit' is short for cross(**X**) platform (**B**)luetooth (**I**)nterface (**T**)ool. This application is designed to run on Mac/Windows/Linux desktop platforms and provides a framework for development of "Applets", or small purpose built "mini-applications" that can be bundled with the tool to enable various software development workflows with a friendly graphical user interface for proof-of-concept presentations. The AC Current Demo application was tested for use on the Windows platform.

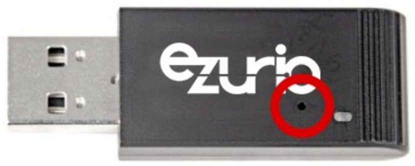
Select the appropriate platform version provided on the GitHub repository and follow the installer instructions to install the application.

7.1.2 Program BL654-USB Dongle (451-00004)

The Xbit Desktop tool depends on local USB-serial access to a BL654 USB Adapter (P/N 451-00004) programmed with Canvas firmware allowing control of the BL654 radio on the USB adapter via Python scripts running on the module. With the USB adapter installed in the workstation, users can select the corresponding USB serial port from the tool and provide BLE functionality to Xbit applets.

To program the BL654 USB Adapter (P/N 451-00004) do the following:

1. Insert the Dongle into a USB port on your PC and ensure it is in bootloader mode. (the LED should be going from bright to dim when in bootloader mode). If not, enter bootloader mode using a thin object to press the button on the dongle through the hole in the case (whilst keeping the USB dongle plugged into the computer).



2. Download the most current BL654-USB Canvas Firmware release. The ".hex" file can be found under the bl654/usb/firmware/<version> subdirectory within the GitHub repository.
3. Navigate to the folder the firmware .hex file was downloaded to and open a cmd line window.
4. Flash the firmware to the dongle using the nRFConnect Programmer utility from Nordic Semiconductor. The Programmer tool will identify the BL654 USB dongle in "Nordic DFU" mode. More details on using nRFConnect Programmer can be found in the Quick Start Guide BL654 USB Dongle – Nordic/Zephyr (451-00004).
5. Download the xbit_lib.py and xbit_usb.py files from the Canvas Python Sample repository. These scripts are located in the demos/bl654_usb/xbit_usb subdirectory.
6. Open VS Code – Xbit Extension and refresh the COM Ports.
7. The BL654-USB should present as a repl COM port (>>>). Follow the same steps as were followed in section 6.2.2 to load both the xbit_lib.py and xbit_usb.py files to the COM port associated with the BL654 dongle.
8. Rename xbit_usb.py → main.py
9. Be sure to save the files to the dongle by pressing **Ctrl+s .**
10. Reset the dongle by clicking in the terminal window of VS Code and pressing **Ctrl+d**.

7.1.3 Download the Xbit Desktop application

1. Download the appropriate version of Xbit Desktop for your operating system from our Canvas Xbit Desktop GitHub repository.
2. Follow the prompts to install the software.

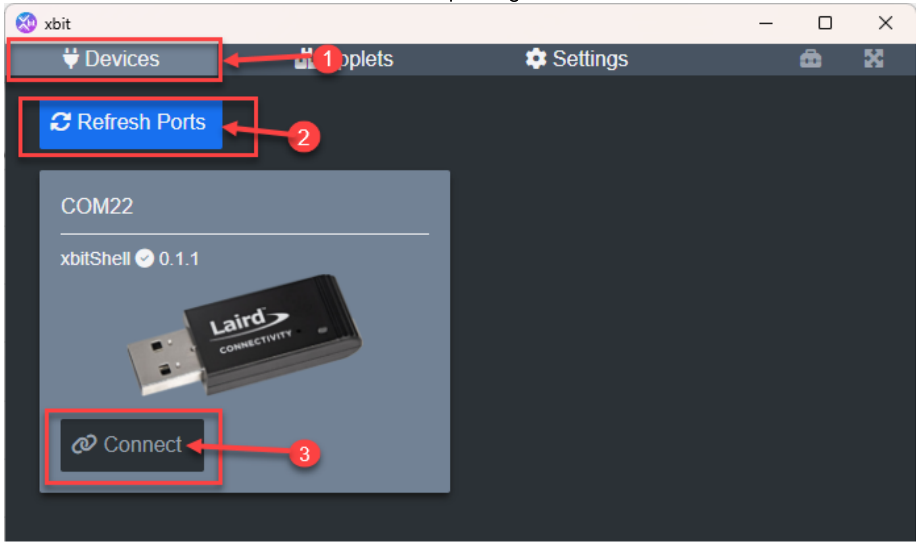
7.1.4 Install the bt610-ac-sense Applet

The bt610-ac-sense applet is available to install directly from within the Xbit Desktop and Xbit Mobile applications. To install the applet, do the following:
Xbit Desktop: Navigate to the **Applets** tab, then scroll down to the **BT610 Current Sense Applet** and click **Install**.
Xbit Mobile: Press the "Refresh" button (circle with arrow in it) in the upper right corner of the main Xbit app screen.

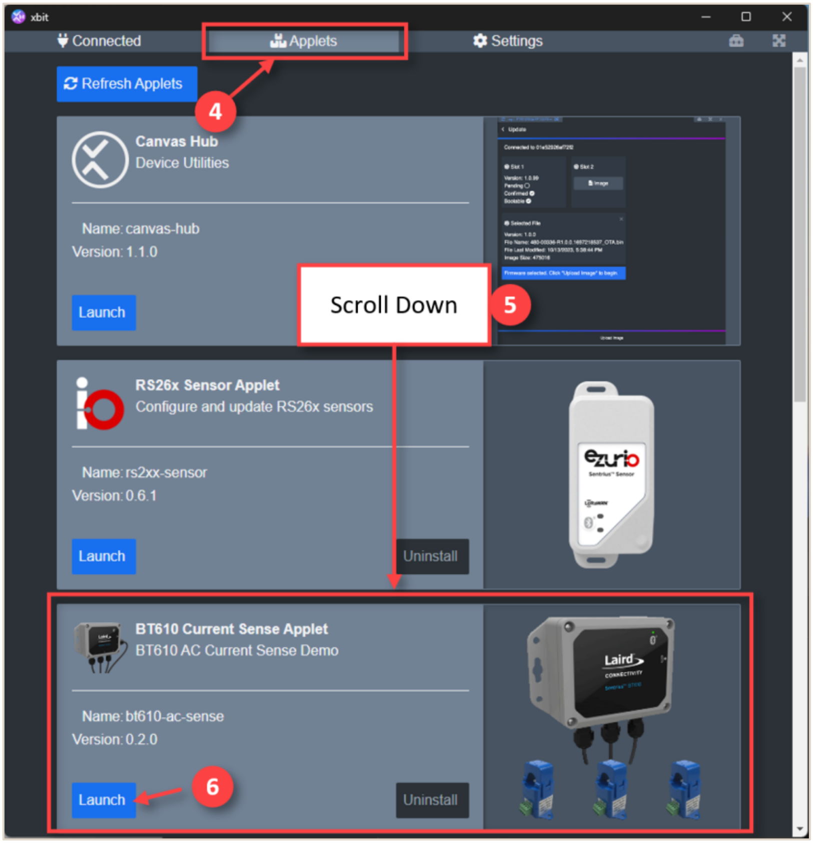
7.1.5 Running the BT610 Current Sense Applet with Xbit Desktop

1. With BL654 Xbit Dongle inserted in USB port, click on ***Devices.***
2. If the BL654 xbitShell is not displayed – click refresh ports.

3. Click ***Connect*** to connect to the BL654 for capturing BLE advertisement data.

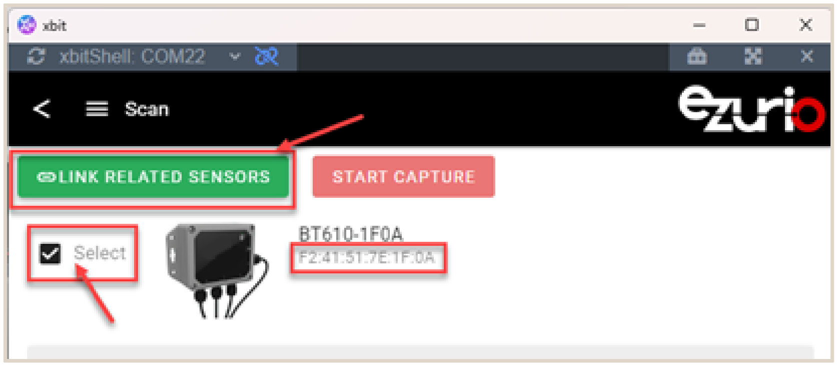


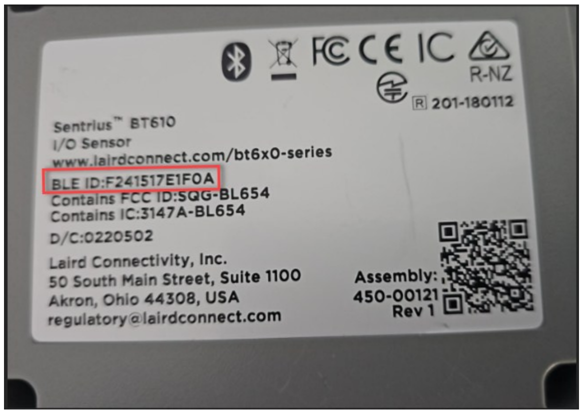
4. Click the ***Applets*** tab.
5. Scroll Down to locate the BT610 Current Sense applet.
6. Click launch to open the applet.



7. In the Applet click **Scan for Sensors** to start scanning.
8. Check the box to select the sensor you want to associate with a capture and the click **Link Related Sensors** to create your first simulated "Oven.". The "Oven" represents the appliance you are measuring AC current of for purposes of the demonstration application.

Note: If you do not see your sensor, it may be necessary to restart the Xbit application and/or power-cycle the BT610. The BLE ID displayed should match the BLE ID on the BT610 label.

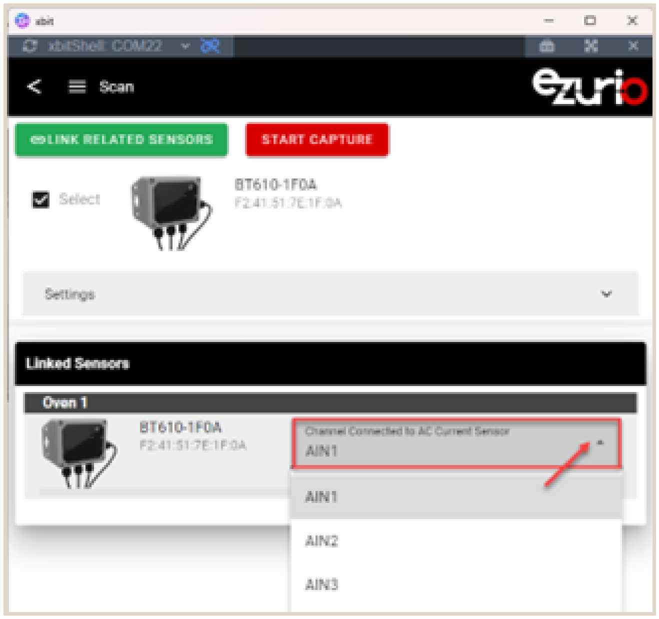




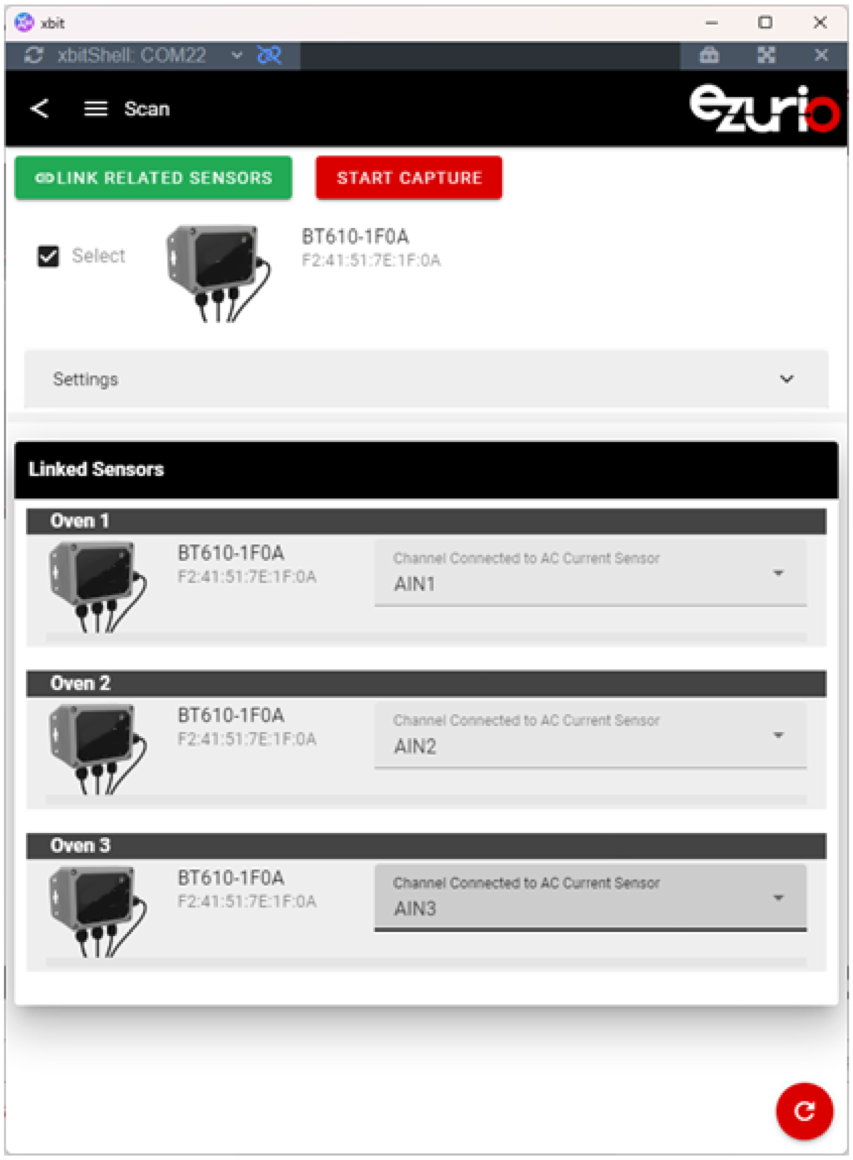
Sentrius™ BT610
I/O Sensor
www.lairdconnect.com/bt6x0-series
BLE ID:F241517E1F0A
Contains FCC ID:SQG-BL654
Contains IC:3147A-BL654
D/C:0220502
Laird Connectivity, Inc.
50 South Main Street, Suite 1100
Akron, Ohio 44308, USA
regulatory@lairdconnect.com

R-NZ
R 201-180112
Assembly:
450-00121
Rev 1

9. Once the Oven is created, click the drop-down arrow to select the Analog Input (AIN1, AIN2 OR AIN3) the oven is connected to.



10. Repeat steps 8 and 9, selecting the same Sensor, and selecting a different AIN# for each of the current sensors until you have created 3 ovens, all from the same sensor, but with different AIN#.



BT610-1F0A
F2:41:51:7E:1F:0A

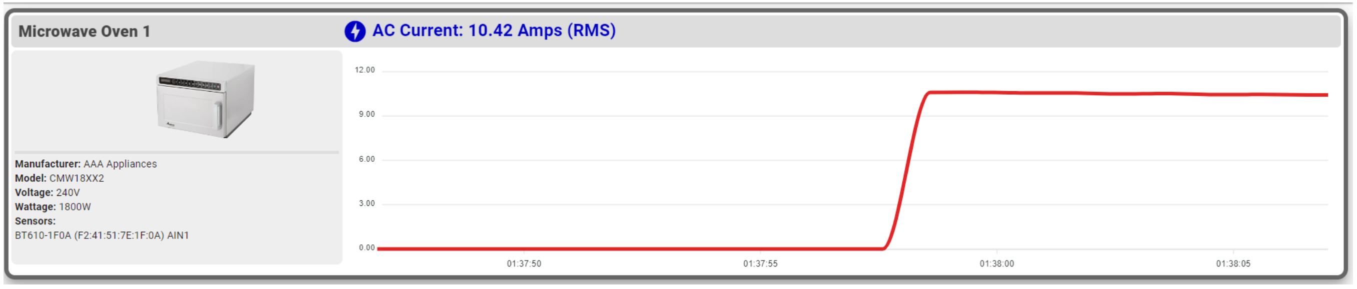
Linked Sensors

Oven 1
BT610-1F0A
F2:41:51:7E:1F:0A
Channel Connected to AC Current Sensor: AIN1

Oven 2
BT610-1F0A
F2:41:51:7E:1F:0A
Channel Connected to AC Current Sensor: AIN2

Oven 3
BT610-1F0A
F2:41:51:7E:1F:0A
Channel Connected to AC Current Sensor: AIN3

11. After the AC current sensor is correctly placed around a hot line from a source, press the Start Capture button to begin capturing and displaying the data.



Microwave Oven 1

AC Current: 10.42 Amps (RMS)

Manufacturer: AAA Appliances
Model: CMW18XX2
Voltage: 240V
Wattage: 1800W
Sensors:
BT610-1F0A (F2:41:51:7E:1F:0A) AIN1

Note: The Xbit Applet is for Proof of Concept only. It is not designed for deployment, only to illustrate how data can be captured and displayed.

8 Revision History

Version	Date	Notes	Contributor(s)	Approver
0.1	23 Aug 2024	Preliminary	Rikki Horrigan	Scott Lederer
1.0	22 July 2025	Initial Release	Dave Drogowski	Scott Lederer

Ezurio's products are subject to standard Terms & Conditions.