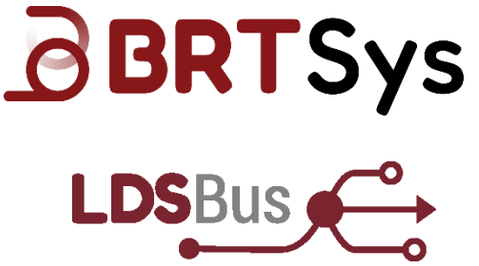


BRTSYS_API_001

LDSBus Python SDK Guide

Version 1.2

Issue Date: 22-09-2023

This document provides information about the Python SDK (V3.1.0) used in LDSBus USB Host Systems.

BRT Systems Pte Ltd (BRTSys)
1 Tai Seng Avenue, Tower A, #03-01, Singapore 536464
Tel: +65 6547 4827
Web Site: http://www.brtsys.com

Table of Contents

1 Introduction

The LDSBus Python SDK is available on Microsoft Windows 10/11, Ubuntu 20.04, Raspberry Pi 3 and Pi 4 and Raspberry Pi Pico platforms. The LDSBus Python SDK implements the LDSBus host that manages and controls LDSU devices on the LDSBus and the aforementioned platforms are referred to as LDSBus hosts.

The document has been divided into the following sections –

- **Setup –** This section provides information related to System Requirements, Hardware Pre-requisites, Hardware Setup and SDK Installation.
- **Samples & Output –** This section provides information related to the Sensor Samples (LDSBus 4in1, pH, Thermocouple, and Relay) and Sample Output
- **API –** This section describes the LDSBus Python SDK APIs

2 Setup

2.1 System Requirements

2.1.1 Hardware

- LDSBus USB-C Adapter
- USB-C to USB-A Cable
- 24V DC Adapter
- RJ11/RJ12 Cable for LDSU Port
- At least 2x RJ45 Cable for LDSBus Port
- 1 Quad T-Junction to connect to LDSBus Port
- At least 1 Supported LDSU Sensor Type

2.1.2 Operating System

- Windows 10/11 [x64]
- Ubuntu 20.04 [x64]
- Raspbian OS - (Bullseye)

2.1.3 Python Version

- Python 3.8.8 or higher

2.2 Pre-requisites

- Connect the LDSBus Sensor to LDSBus USB Adapter either to the LDSU port directly or to the LDSBus port using Quad T-Junction
- If Sensor is plugged to the LDSBus port using Quad T-Junction, please make sure that LDSBus USB Adapter is powered using a 24V PSU.
- Discover the port number of the LDSBus USB Adapter from the Device Manager or Linux Device Path based on the operating system.

2.3 Hardware Setup

2.3.1 Setup using LDSU Port

To configure LDSBus Device (Sensors / Actuators) -

1. Connect the LDSBus USB Adapter to the Windows PC with the USB-C to USB-A cable.
2. Ensure that the LDSBus Device is connected to its cable at one end.
3. Attach the other end of the cable to the LDSBus USB Adapter as shown in Figure 1.
4. Refer to the LDSBus Configuration Utility Guide for further steps on configuring the LDSBus device.

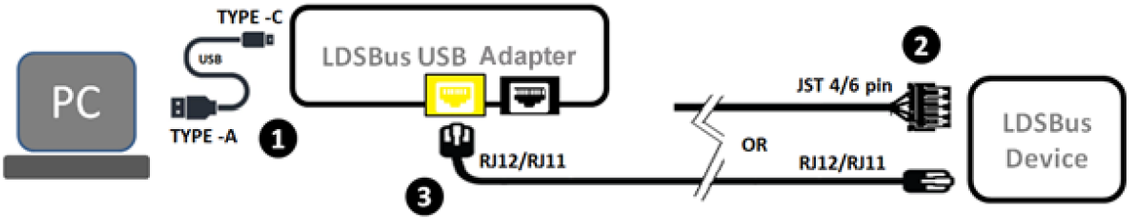


Figure 1 - LDSBus Device (Sensors / Actuators) Configuration

2.3.2 Setup using LDSBus Port

To create an LDSBus

1. Connect the LDSBus USB Adapter to the Windows PC with the USB-C to USB-A cable.
2. Connect a 24VDC/18W power Adapter to the DC jack and power on. Power to the LDSBus RJ45 connector is controlled by software.
3. Connect the first LDSBus Quad T-Junction to the LDSBus USB Adapter using a RJ45 (CAT5e). The LDSBus Devices connected to the LDSBus Quad T-Junction must be pre-configured through the LDSBus Configuration Utility tool.
4. If there is more than one LDSBus Quad T-Junction device, daisy chain them together as shown in Figure 2 using a RJ45 (CAT5e) cable. The termination on the last LDSBus device must be set to the ON state.

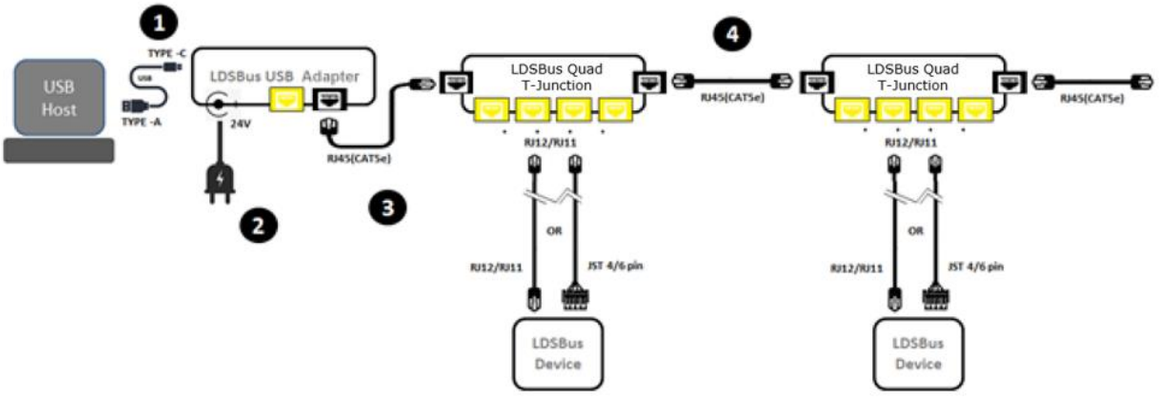


Figure 2 - LDSBus Creation

2.4 Installation

2.4.1 How to get the Python SDK

2.4.1.1 Standalone Package

- Download the standalone zip package from https://brtsys.com/resources and extract it to one directory.
- Extract the zip.

2.4.2 Software Installation

LDSBus Python SDK Package is a standard zip file. Installation is as simple as extracting the zip file to the working directory.

2.4.2.1 Pre-requisites

- Python version > 3.8.8
- pip3

2.4.2.2 LDSBus Python SDK Directory Structure

```
Python SDK Library
\--> LDSBus_Python_SDK
      \-> circuitpython_sample [Samples LDSU and LDSBus Projects for IDM2040]
            \-> sensor
            \-> actuators
      \-> drivers [required driver files for all platforms]
      \-> modules [Supported hardware modules and its binaries]
      \-> python_sample [Samples LDSU and LDSBus Projects for Windows and Linux]
            \-> sensor
            \-> actuators
      \-> 99-ftdi.rules [UDEV rules to be update on linux when running .sh script]
      \-> code.py [IDM2040 main python scripts to run circuitpython_sample]
      \-> lds_driver_install.bat [Script file to install ldsbus.dll to C:\Windows\System32]
      \-> lds_pysdk_install.bat [Script file to install sdk in windows]
      \-> lds_pysdk_install_armv7.sh [Script file to install sdk in ARM linux (Raspberry Pi)]
      \-> lds_pysdk_install_x64.sh [Script file to install sdk in x64 Linux]
      \-> pyliblds-X.X.X-py3-none-any.whl [pyliblds wheel file]
      \-> README.txt [simplify SDK guide]
      \-> Release_Note.txt [release versions and respective notes.]
      \-> requirements.txt [python dependencies]
```

Note: pyliblds-X.X.X-py3-none-any.whl – X.X.X refers to release version.

2.4.2.3 Windows Installation

Download and install FTDI driver from https://ftdichip.com/drivers/d2xx-drivers/.
Plug in LDSBus USB Adapter.
'ftd2xx.dll' should be in 'C:\Windows\System32\'
Run command prompt in Administrator mode. Go to directory to the release folder provided. Run 'lds_driver_install.bat'. 'ldsbus.dll' should be in 'C:\Windows\System32\'.

```
C:\Users\Documents\BRTSystems\LDSBus_Python_SDK>lds_driver_install.bat
```

In command prompt without Administrator mode, Run 'lds_pysdk_install.bat'.

```
C:\Users\Documents\BRTSystems\LDSBus_Python_SDK>lds_pysdk_install.bat
```

To run the samples given in the Python SDK, change directory to ".\LDSBus_Python_SDK\python_samples\sensor\" and run "python LDSBus_4in1_Sensor.py" to test the 4-in-1 sensor.

```
C:\Users\Documents\BRTSystems\LDSBus_Python_SDK\python_samples\sensor>python3
LDSBus_4in1_Sensor.py
```

2.4.2.4 Linux Installation

Run the command below to install pip

```
> sudo apt-get install pip
```

Copy the whole folder of LDSBus_Python_SDK and run the command as follows.

```
> cd LDSBus_Python_SDK
> chmod +x lds_pysdk_install_x64.sh
> sudo ./lds_pysdk_install_x64.sh
```

Unplug LDSBus USB Adapter (If LDSBus USB adapter is plugged before installation)

Plug LDSBus USB Adapter (no need command 'sudo rmmod ftdi_sio', UDEV rules are updated as such when FTDI USB device is plugged in, ftdi_sio command is automatically given.)

2.4.2.5 Raspberry Pi Installation

Run the command below to install pip

```
> sudo apt-get install pip
```

Copy the whole folder of LDSBus_Python_SDK and run the command as follows.

```
> cd LDSBus_Python_SDK
> chmod +x lds_pysdk_install_armv7.sh
> sudo ./lds_pysdk_install_armv7.sh
```

Unplug LDSBus USB Adapter (If LDSBus USB adapter is plugged before installation)

Plug LDSBus USB Adapter (no need command 'sudo rmmod ftdi_sio', UDEV rules are updated as such when FTDI USB device is plugged in, ftdi_sio command is automatically given.)

2.4.2.6 Circuit Python – IDM2040

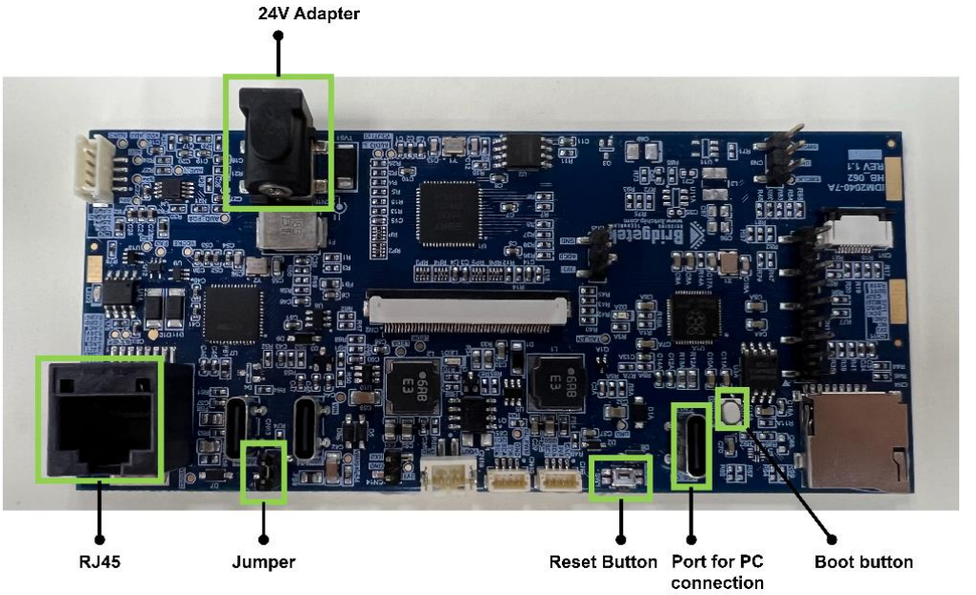
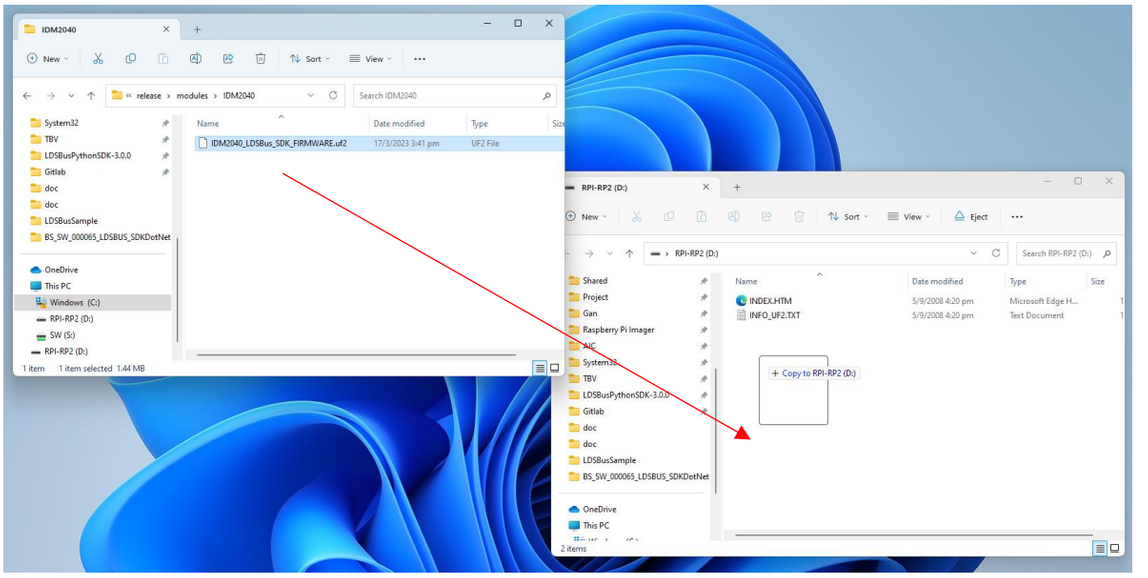


Figure 3 - IDM2040 Hardware Features

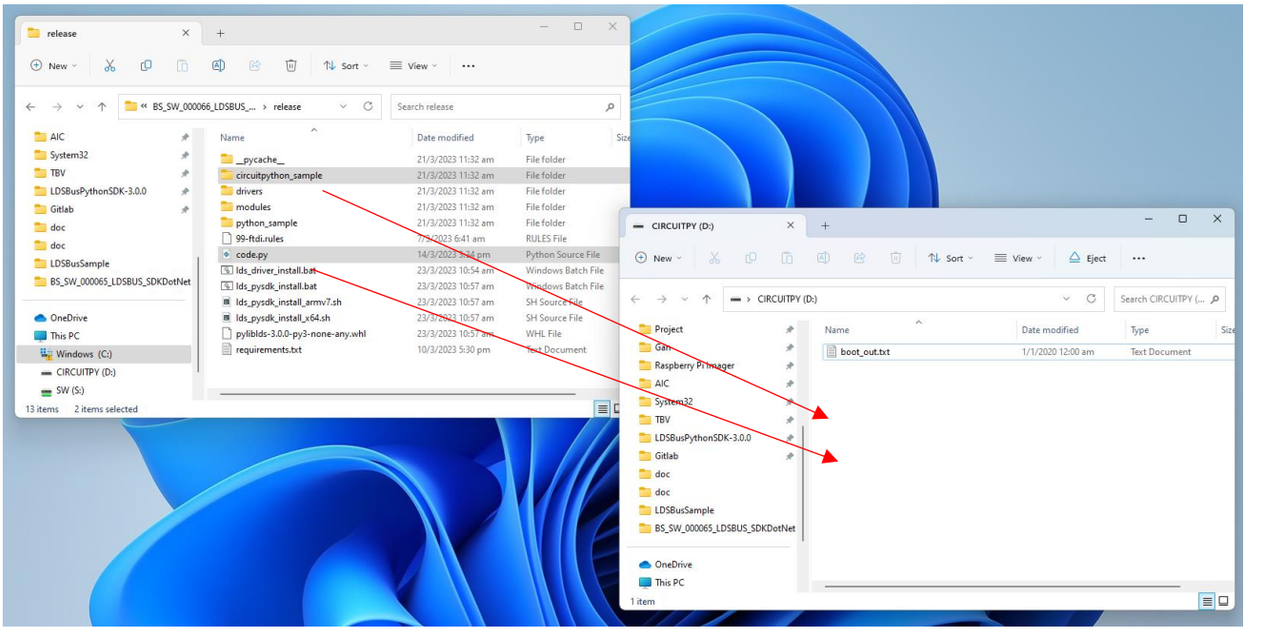
For IDM2040 installation, set the IDM2040 into bootloader mode in following sequence.

1. Remove Jumper.
2. Power up the controller by using "Port for PC connection".
3. Click and hold the "Reset Button" first.
4. Click and hold the "Boot button".
5. Release the "Reset Button".
6. "RPI-RP2" will show up.

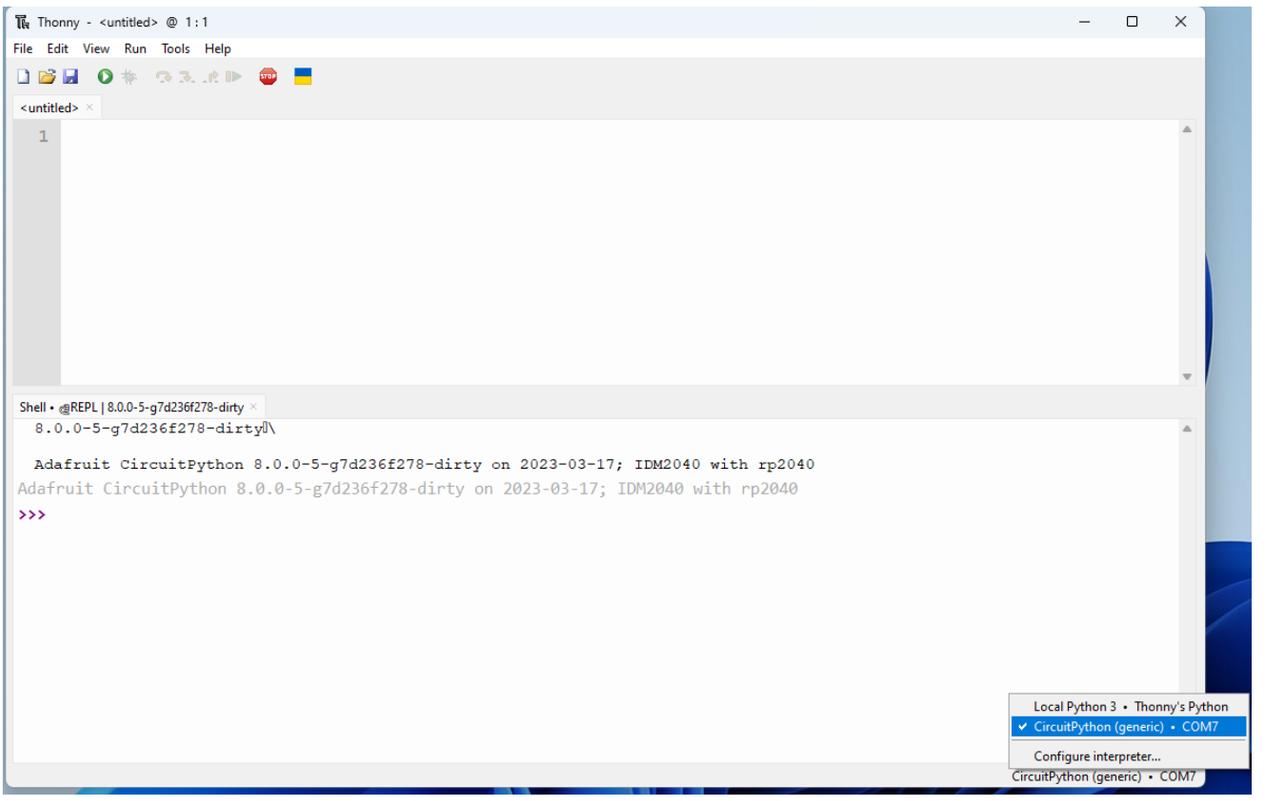


7. Copy the ".uf2" file in "modules\IDM2040\" into this drive.

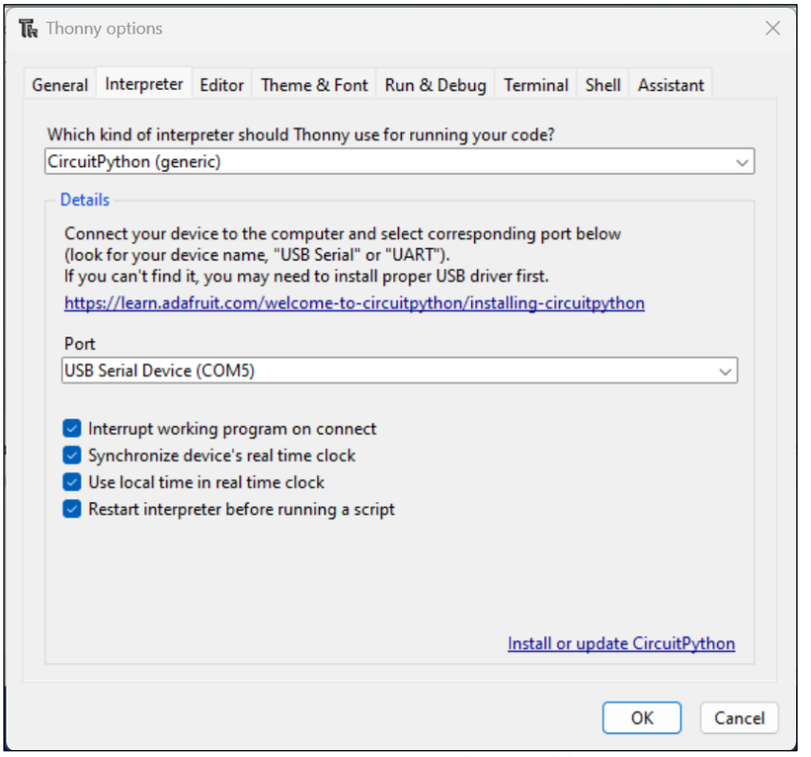
8. After it finishes copying, the device will be re-enumerated as CIRCUITPY, and a drive "CIRCUITPY" will show up.



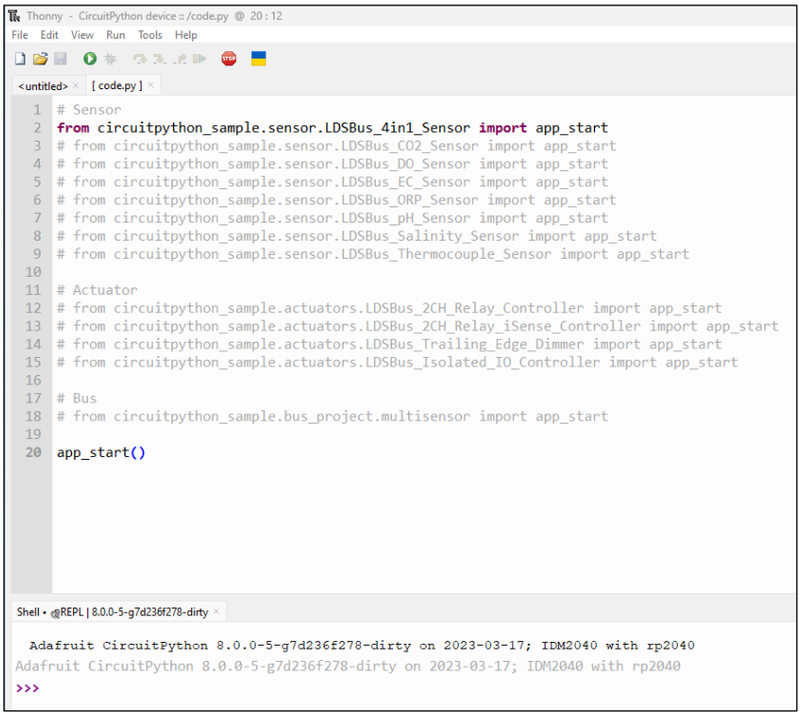
9. Copy the "circuitpython_sample" folder and "code.py" into this drive.



10. Open "Thonny" application. (Download in https://thonny.org/) Select COM port in the right bottom corner.



11. Select "CircuitPython (generic)" for interpreter. Select COM port for the connected IDM2040.



12. By default, "code.py" runs sample scripts of "LDSBus_4in1_Sensor.py". Change this script by commenting out unwanted and uncomment the "app_start" matches the sensor connected. Press "Ctrl + D" in the Thonny shell to re-run "code.py".

2.4.2.7 Running Python Sample from Command Line

All the below commands have to be run at the Python SDK Root Directory containing the Python samples.

For Windows:

```
> python3 LDSBus_4in1Sensor.py
```

Note 1: sudo requires Linux administrative user privileges which will prompt for logged in user password to be entered when invoking the above command.

For Ubuntu 20.04 x64:

```
# sudo python3 LDSBus_4in1Sensor.py
```

For Raspbian Bullseye for RPI4:

```
# sudo python3 LDSBus_4in1Sensor.py
```

2.4.2.8 Sensor / Actuator Sample Scripts

All supported sensor / actuator sample scripts have been provided as an example for users to start to integrate with their own API.

The following is the list of sensor / actuator samples provided -

- Sensors
  - LDSBus_4in1_Sensor.py
  - LDSBus_CO2_Sensor.py
  - LDSBus_Thermocouple_Sensor.py
  - LDSBus_pH_Sensor.py
  - LDSBus_EC_Sensor.py
  - LDSBus_DO_Sensor.py
  - LDSBus_Salinity_Sensor.py
  - LDSBus_ORP_Sensor.py
- Actuators
  - LDSBus_Relay_Controller.py
  - LDSBus_Relay_iSense_Controller.py
  - LDSBus_Trailing_Edge_Dimmer.py
  - LDSBus_Isolated_IO_Controller.py
  - LDSBus_IR_Blaster.py
  - LDSBus_RFID_Reader.py

3 Samples & Output

3.1 LDSBus Sample Script Pseudocode

All the Sensor Sample scripts follow the pseudocode as mentioned below with minor modifications to Sensor & Actuator Scripts,

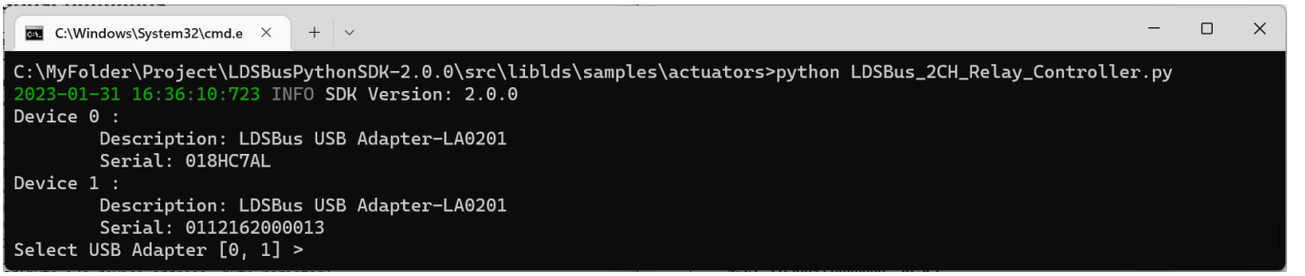
1. Select the USB Adapter (If there is more than one LDSBus USB Adapter)
2. Turn on the bus power (Depends on the mode)
3. Perform a scan (entire bus scan or specific device scan)
4. Number of devices found
5. Get info of the device (device found and related info)
6. Enter into a loop
7. Every 5 seconds display the sensor data

```
import time
from liblds.ldsbus import LDSBus
from liblds.ldsdevice import LDSBus4in1Sensor
from liblds.ldslogger import LDSLogging as log
import liblds
log.info("SDK Version: %s", liblds.__version__)

def app_start():
    try:
        ldsbus = LDSBus()
        if len(ldsbus.usb_adapter_list) == 0:
            log.error("LDSBus USB Adapter not connected.")
            return

        for adapter in ldsbus.usb_adapter_list:
            print(f"Device {adapter['index']}")
            print(f"\tDescription: {adapter['description'].decode('utf8')}")
            print(f"\tSerial: {adapter['serial'].decode('utf8')}")

        if len(ldsbus.usb_adapter_list) > 1:
            # Select USB Adapter (if there is more than one LDSBus USB Adapter)
            index_list = [x['index'] for x in ldsbus.usb_adapter_list]
            port = input(f"Select USB Adapter {index_list} > ")
            ldsbus.open_comm_port(int(port))
        else:
            # if there is only one LDSBus USB Adapter
            ldsbus.open_comm_port(0)
        log.info("Successfully opened the port…")

        # Turn on the bus power
        ldsbus.ldsu_port_power(1)
        # ldsbus.ldsbus_port_power(1)

        # Perform a scan
        log.info("Scanning…")
        device_list = ldsbus.scan_ldsu()
        if len(device_list) == 0:
            log.error("LDSBus device not found.")
            return
```

```
        # Number of devices found
        log.info(f"Number of devices found: {len(device_list)}, ID(s) is/are:
{device_list}")
        device_id = device_list[0]

        # Based on the known device connected, initialise the device
        ldsu = LDSBus4in1Sensor(ldsbus, device_id)
        # Get info of the device
        log.info(f"UUID: {ldsu.ldsu.get_uuid()}")
        log.info(f"Last command : 0x{ldsu.ldsu.get_lastcommand():2X}")
        log.ldsu_info(ldsu.ldsu.get_info())
        log.ldsu_descriptor(ldsu.descriptor)
        log.sensor_actuator_list(ldsu.sa_list)

        # Enter into a loop
        while True:
            # Every 5 seconds display the sensor data
            print(chr(27) + "[2J"]
            log.ldsu_descriptor(ldsu.descriptor)
            log.sensor_actuator_list(ldsu.sa_list)

            log.info("SENSOR DATA")
            log.info(len('SENSOR DATA')*'*')
            vals = ldsu.read()
            log.info(f"TEMPERATURE : {vals[0]:.2f} °C")
            log.info(f"HUMIDITY : {vals[1]:.2f} %")
            log.info(f"ALS : {vals[2]:.0f} LUX")
            log.info(f"MOTION : {vals[3]:d}")
            time.sleep(5)

    except (KeyboardInterrupt, Exception) as err:
        log.critical("Keyboard interrupt or Something went wrong!")
        ldsbus.close_comm_port()

if __name__ == "__main__":
    app_start()
```

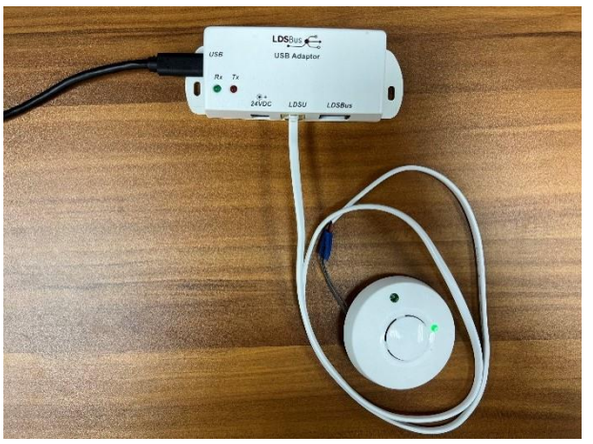
Sample code output references:

Step 1, Select USB Adapter – Key in 0 or 1, then press Enter, to select one of the LDSBus USB Adapter.



Step 2, Depending on how the LDSU device is connected,

If device is connected to the LDSU port, use `ldsbus.ldsu_port_power(1)`
If device is connected to the LDSBus port with Quad T-Junction, use `ldsbus.ldsbus_port_power(1)`



Using command ldsbus.ldsu_port_power(1)

Using command ldsbus.ldsbus_port_power(1)

Step 3, Throughout the LDSBus, perform a scan of available devices.

```
C:\Windows\System32\cmd.e
C:\MyFolder\Project\LDSBusPythonSDK-2.0.0\src\liblds\samples\actuators>python LDSBus_2CH_Relay_Controller.py
2023-01-31 16:36:10:723 INFO SDK Version: 2.0.0
Device 0 :
        Description: LDSBus USB Adapter-LA0201
        Serial: 018HC7AL
Device 1 :
        Description: LDSBus USB Adapter-LA0201
        Serial: 0112162000013
Select USB Adapter [0, 1] > 0
2023-01-31 16:39:53:966 INFO Successfully opened the port...
2023-01-31 16:39:54:980 INFO Scanning...
```

Upon completion of the scan, the number of devices found with their ID will be displayed

```
C:\Windows\System32\cmd.e
2023-01-31 16:42:53:853 INFO Successfully opened the port...
2023-01-31 16:42:54:853 INFO Scanning...
2023-01-31 16:43:46:234 INFO Number of devices found : 1, ID(s) is/are :  (126,)
```

Step 4 to Step 7, as the output depends on the connected devices, it is described in the later sections from Section 3.2 to Section 3.14.

3.2 LDSBus 4in1 Sensor

The LDSBus 4in1 Sensor consists of 4 sensors in a compact low-profile design. Temperature, humidity, Passive Infra-Red (PIR) based motion detection and ambient light measurement sensors are incorporated in this multi-sensor device. The device can be flush mounted on ceilings or swivel mounted on walls. The multi-sensor works with the BRTSys's PanL Smart Living, IoTPortal Gateway and LDSBus Python SDK products.

Sample File Reference: **LDSBus_4in1_Sensor.py**

```
2022-12-14 15:51:22:181 INFO LDSU Descriptor
2022-12-14 15:51:22:181 INFO ===============
2022-12-14 15:51:22:181 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 15:51:22:182 INFO     Product Name : LDSBus 4in1 Sensor
2022-12-14 15:51:22:182 INFO     Product Version : 0.1
2022-12-14 15:51:22:182 INFO     UUID : LS010101210722000046
2022-12-14 15:51:22:182 INFO Sensors and actuators
2022-12-14 15:51:22:183 INFO ======================
2022-12-14 15:51:22:183 INFO ID: 0
2022-12-14 15:51:22:183 INFO     Manufacturer : Bridgetek Pte Ltd.
2022-12-14 15:51:22:184 INFO     Part Number : BRT-VDEV
2022-12-14 15:51:22:184 INFO     I2C Address : 0
2022-12-14 15:51:22:184 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 15:51:22:184 INFO ID: 1
2022-12-14 15:51:22:185 INFO     Manufacturer : LITE-ON Technology
2022-12-14 15:51:22:185 INFO     Part Number : LTR-303ALS-01
2022-12-14 15:51:22:185 INFO     I2C Address : 41
2022-12-14 15:51:22:185 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 15:51:22:186 INFO ID: 2
2022-12-14 15:51:22:186 INFO     Manufacturer : Texas Instrument
2022-12-14 15:51:22:186 INFO     Part Number : HDC1080
2022-12-14 15:51:22:186 INFO     I2C Address : 64
2022-12-14 15:51:22:186 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 15:51:22:187 INFO ID: 3
2022-12-14 15:51:22:187 INFO     Manufacturer : Texas Instrument
2022-12-14 15:51:22:187 INFO     Part Number : HDC1080
2022-12-14 15:51:22:187 INFO     I2C Address : 64
2022-12-14 15:51:22:187 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 15:51:22:187 INFO ID: 4
2022-12-14 15:51:22:188 INFO     Manufacturer : Analog Devices
2022-12-14 15:51:22:188 INFO     Part Number : AD5242
2022-12-14 15:51:22:188 INFO     I2C Address : 44
2022-12-14 15:51:22:188 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 15:51:22:188 INFO ============SENSOR DATA============
2022-12-14 15:51:22:212 INFO TEMPERATURE : 25.19 °C
2022-12-14 15:51:22:212 INFO HUMIDITY : 39.51 %
2022-12-14 15:51:22:258 INFO ALS : 111 LUX
2022-12-14 15:51:22:274 INFO Motion Sensor : 1
```

Figure 4 - LDSBus 4in1 Sensor Sample Output

3.3 LDSBus CO2 Sensor

LDSBus CO2 Sensor is a true CO2 sensor that features four sensors in a compact, low-profile design. It includes sensors to measure CO2, temperature, humidity and ambient light. The device can be flush mounted on the ceiling or swivel mounted on the wall. LDSBus CO2 Sensors are compatible with the BRTSys's IoTPortal, PanL Smart Living and LDSBus Python SDKs. The sensor is available in 2 versions, namely Basic and Pro.

Sample File Reference: **LDSBus_CO2_Sensor.py**

```
2022-12-14 15:55:44:633 INFO LDSU Descriptor
2022-12-14 15:55:44:633 INFO ===============
2022-12-14 15:55:44:634 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 15:55:44:634 INFO     Product Name : LDSBus CO2 Sensor Pro
2022-12-14 15:55:44:634 INFO     Product Version : 0.1
2022-12-14 15:55:44:634 INFO     UUID : LS01010159072200012
2022-12-14 15:55:44:635 INFO Sensors and actuators
2022-12-14 15:55:44:635 INFO ====================
2022-12-14 15:55:44:635 INFO ID: 0
2022-12-14 15:55:44:636 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 15:55:44:636 INFO     Part Number : BRT-VDEV
2022-12-14 15:55:44:637 INFO     I2C Address : 0
2022-12-14 15:55:44:637 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 15:55:44:637 INFO ID: 1
2022-12-14 15:55:44:637 INFO     Manufacturer : ROHM Semiconductor
2022-12-14 15:55:44:638 INFO     Part Number : BH1730FVC
2022-12-14 15:55:44:638 INFO     I2C Address : 41
2022-12-14 15:55:44:638 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 15:55:44:638 INFO ID: 2
2022-12-14 15:55:44:639 INFO     Manufacturer : Sensirion
2022-12-14 15:55:44:639 INFO     Part Number : SCD41
2022-12-14 15:55:44:639 INFO     I2C Address : 98
2022-12-14 15:55:44:640 INFO     Maximum report rate (millsecs) : 5000
2022-12-14 15:55:44:640 INFO ID: 3
2022-12-14 15:55:44:640 INFO     Manufacturer : Sensirion
2022-12-14 15:55:44:640 INFO     Part Number : SCD41
2022-12-14 15:55:44:641 INFO     I2C Address : 98
2022-12-14 15:55:44:641 INFO     Maximum report rate (millsecs) : 5000
2022-12-14 15:55:44:641 INFO ID: 4
2022-12-14 15:55:44:641 INFO     Manufacturer : Sensirion
2022-12-14 15:55:44:641 INFO     Part Number : SCD41
2022-12-14 15:55:44:642 INFO     I2C Address : 98
2022-12-14 15:55:44:642 INFO     Maximum report rate (millsecs) : 5000
2022-12-14 15:55:44:642 INFO ============SENSOR DATA============
2022-12-14 15:55:44:662 INFO ALS : 271 LUX
2022-12-14 15:55:44:694 INFO CO2 : 1732 ppm
2022-12-14 15:55:44:695 INFO Temperature : 25.97 °C
2022-12-14 15:55:44:696 INFO Humidity : 74.40 %
```

Figure 5 - LDSBus CO2 Sensor Sample Output

3.4 LDSBus Thermocouple Sensor Adapter

The LDSBus Thermocouple Sensor Adapter is designed to operate with any K-type thermocouple probe and provides temperature measurements ranging between -200°C to 1372°C with an accuracy of ±0.5°C. The adapter automatically handles all the necessary signal conditioning and analog to digital conversions to produce linearized temperature readings and can sustain high report rates. The LDSBus Thermocouple Sensor Adapter may be used in applications such as food production, metal extruders, furnaces, cryogenic baths, and freezers to name a few.

Sample File Reference: **LDSBus_Thermocouple_Sensor.py**

```
2022-12-14 16:04:39:706 INFO LDSU Descriptor
2022-12-14 16:04:39:706 INFO ===============
2022-12-14 16:04:39:706 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 16:04:39:707 INFO     Product Name : LDSBus Thermocouple Sensor
2022-12-14 16:04:39:707 INFO     Product Version : 0.1
2022-12-14 16:04:39:707 INFO     UUID : LS01010150072200101
2022-12-14 16:04:39:707 INFO Sensors and actuators
2022-12-14 16:04:39:707 INFO ====================
2022-12-14 16:04:39:707 INFO ID: 0
2022-12-14 16:04:39:708 INFO     Manufacturer : Bridgetek Pte Ltd.
2022-12-14 16:04:39:708 INFO     Part Number : BRT-VDEV
2022-12-14 16:04:39:708 INFO     I2C Address : 0
2022-12-14 16:04:39:708 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 16:04:39:708 INFO ID: 1
2022-12-14 16:04:39:708 INFO     Manufacturer : MICROCHIP
2022-12-14 16:04:39:708 INFO     Part Number : MCP9600
2022-12-14 16:04:39:709 INFO     I2C Address : 96
2022-12-14 16:04:39:709 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 16:04:39:709 INFO ============SENSOR DATA============
2022-12-14 16:04:39:710 INFO Temperature: 24.52 °C
```

Figure 6 - LDSBus Thermocouple Sensor Sample Output

3.5 LDSBus pH Sensor Adapter

The LDSBus pH Sensor Adapter is designed to work with pH probes to form a complete pH sensor. The adapter consists of a built-in BNC connector used to attach pH probes. The adapter and probe are calibrated using a 2-point calibration algorithm and supports measurement of pH ranging from 0 to 14 pH with a 0.01 pH resolution. The adapter and probe combination are suitable for use in applications such as nutrient tanks, water treatment plants, sewage treatment plants, swimming pools. Real time monitoring, alert notifications and control automation can be achieved.

Sample File Reference: **LDSBus_PH_Sensor.py**

```
2022-12-14 16:08:44:003 INFO LDSU Descriptor
2022-12-14 16:08:44:004 INFO ===============
2022-12-14 16:08:44:004 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 16:08:44:004 INFO     Product Name : LDSBus pH Sensor
2022-12-14 16:08:44:005 INFO     Product Version : 0.1
2022-12-14 16:08:44:005 INFO     UUID : LS040101042121000l9
2022-12-14 16:08:44:005 INFO Sensors and actuators
2022-12-14 16:08:44:006 INFO ====================
2022-12-14 16:08:44:006 INFO ID: 0
2022-12-14 16:08:44:006 INFO     Manufacturer : Bridgetek Pte. Ltd.
2022-12-14 16:08:44:006 INFO     Part Number : BRT-VDEV
2022-12-14 16:08:44:006 INFO     I2C Address : 0
2022-12-14 16:08:44:007 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 16:08:44:007 INFO ============SENSOR DATA============
2022-12-14 16:08:44:007 INFO pH: 4.598 pH
```

Figure 7 - LDSBus pH Sensor Sample Output

3.6 LDSBus EC Sensor

The LDSBus Electrical Conductivity (EC) Sensor Adapter is designed to work with EC probes to form a complete EC sensor. The adapter consists of built-in BNC connector used to attach EC probes. The adapter and probe are calibrated using a two-point calibration procedure and the resulting sensor supports EC measurements ranging from 0.001mS/cm to 150mS/cm with a 0.001 mS/cm resolution. The sensor is suitable for use in measuring salts, nutrients, and impurities in water in hydroponics, aquaponics and aquaculture and freshwater systems. Monitoring, alerting, and controlling the system can be done in real-time.

Sample File Reference: **LDSBus_EC_Sensor.py**

```
2022-12-15 11:39:48:527 INFO LDSU Descriptor
2022-12-15 11:39:48:528 INFO ===============
2022-12-15 11:39:48:528 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-15 11:39:48:528 INFO     Product Name : LDSBus EC Sensor
2022-12-15 11:39:48:529 INFO     Product Version : 0.1
2022-12-15 11:39:48:529 INFO     UUID : LS050101381522000004
2022-12-15 11:39:48:529 INFO Sensors and actuators
2022-12-15 11:39:48:530 INFO ====================
2022-12-15 11:39:48:530 INFO ID: 0
2022-12-15 11:39:48:530 INFO     Manufacturer : Bridgetek Pte Ltd.
2022-12-15 11:39:48:531 INFO     Part Number : BRT-VDEV
2022-12-15 11:39:48:531 INFO     I2C Address : 0
2022-12-15 11:39:48:531 INFO     Maximum report rate (millsecs) : 1000
2022-12-15 11:39:48:531 INFO ============SENSOR DATA============
2022-12-15 11:39:48:532 INFO EC: 0.950 mS/cm
```

Figure 8 - LDSBus EC Sensor Sample Output

3.7 LDSBus DO Sensor

The LDSBus Dissolved Oxygen (DO) Sensor Adapter is designed to work with an analog galvanic probe to form a complete DO sensor. A BNC connector is built into the adapter for attaching such a probe. The adapter and probe are calibrated using a single-point calibration procedure and the resulting sensor supports DO measurements ranging from 0 to 20 mg/L with a resolution of 0.01mg/L. The sensor is suitable for use in water quality measurement applications such as nutrient tanks, fisheries and hatcheries, water treatment and sewage treatment plants, swimming pools, aquariums, and many other applications. Monitoring, alerting, and controlling the system can be done in real-time.

Sample File Reference: **LDSBus_DO_Sensor.py**

```
2022-12-14 16:14:47:745 INFO LDSU Descriptor
2022-12-14 16:14:47:745 INFO ===============
2022-12-14 16:14:47:745 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 16:14:47:746 INFO     Product Name : LDSBus DO Sensor
2022-12-14 16:14:47:746 INFO     Product Version : 0.1
2022-12-14 16:14:47:746 INFO     UUID : LS010101571422000003
2022-12-14 16:14:47:746 INFO Sensors and actuators
2022-12-14 16:14:47:747 INFO ====================
2022-12-14 16:14:47:747 INFO ID: 0
2022-12-14 16:14:47:747 INFO     Manufacturer : Bridgetek Pte Ltd.
2022-12-14 16:14:47:747 INFO     Part Number : BRT-VDEV
2022-12-14 16:14:47:747 INFO     I2C Address : 0
2022-12-14 16:14:47:748 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 16:14:47:748 INFO ============SENSOR DATA============
2022-12-14 16:14:47:748 INFO DO: 6.54 mg/L
```

Figure 9 - LDSBus DO Sensor Sample Output

3.8 LDSBus Salinity Sensor

The LDSBus Salinity Sensor Adapter is designed to work with relevant probe to form a complete Salinity sensor. This adapter has a BNC connector for attaching the Salinity probe. A 2-point calibration method is used to calibrate the adapter and probe and allows measurements of Salinity between 1 ppt and 120ppt with a precision of 1ppt. These adapters and probes are suitable for use in applications such as Aquaculture fish pond, shrimp pond, sea water applications, nutrient tanks, water treatment plants, sewage treatment plants, swimming pools. Monitoring, alert notifications and control automation are possible in real time.

Sample File Reference: **LDSBus_Salinity_Sensor.py**

```
2022-12-14 17:10:30:809 INFO LDSU Descriptor
2022-12-14 17:10:30:809 INFO ===============
2022-12-14 17:10:30:809 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 17:10:30:810 INFO     Product Name : LDSBus Salinity Sensor
2022-12-14 17:10:30:810 INFO     Product Version : 0.1
2022-12-14 17:10:30:810 INFO     UUID : LS13010112142200001
2022-12-14 17:10:30:810 INFO Sensors and actuators
2022-12-14 17:10:30:811 INFO ====================
2022-12-14 17:10:30:811 INFO ID: 0
2022-12-14 17:10:30:811 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 17:10:30:811 INFO     Part Number : BRT-VDEV
2022-12-14 17:10:30:811 INFO     I2C Address : 0
2022-12-14 17:10:30:812 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 17:10:30:812 INFO ============SENSOR DATA============
2022-12-14 17:10:30:812 INFO Salinity: 1.20 ppt
```

Figure 10 - LDSBus Salinity Sensor Sample Output

3.9 LDSBus ORP Sensor

The LDSBus Oxidation Reduction Potential (ORP) Sensor Adapter is designed to work with relevant probe to form a complete ORP sensor. This adapter has a BNC connector for attaching the ORP probe. A 1-point calibration methodology is used to calibrate the adapter and probe, and ORP measurements can be undertaken with a resolution of 1 mV between -2000mV and +2000mV. These adapters and probes are suitable for use in applications such as aquaculture, nutrient tanks, water treatment plants and swimming pools.

Sample File Reference: **LDSBus_ORP_Sensor.py**

```
2022-12-14 16:31:36:554 INFO LDSU Descriptor
2022-12-14 16:31:36:554 INFO ===============
2022-12-14 16:31:36:554 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 16:31:36:555 INFO     Product Name : LDSBus ORP Sensor
2022-12-14 16:31:36:555 INFO     Product Version : 0.1
2022-12-14 16:31:36:555 INFO     UUID : LS120101120222000001
2022-12-14 16:31:36:556 INFO Sensors and actuators
2022-12-14 16:31:36:556 INFO ======================
2022-12-14 16:31:36:556 INFO ID: 0
2022-12-14 16:31:36:556 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 16:31:36:556 INFO     Part Number : BRT-VDEV
2022-12-14 16:31:36:557 INFO     I2C Address : 0
2022-12-14 16:31:36:557 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 16:31:36:557 INFO ============SENSOR DATA============
2022-12-14 16:31:36:557 INFO ORP: 320 mV
```

Figure 11 - LDSBus ORP Sensor Sample Output

3.10 LDSBus 2CH Relay / iSense Controller

LDSBus Relays can be integrated with actuators and used to control ON/OFF and motor movement forward/ reversal. Users can connect their integrated devices to normal open/normal closed. LDSBus Relay has two types of models: LDSBus 2CH Relay and LDSBus 2CH Relay + iSense. With LDSBus 2CH Relay + iSense, current sense is included. The current sense monitoring can be performed standalone or in conjunction with relays to monitor actuator status.

Sample File Reference: **LDSBus_2CH_Relay_Controller.py**

```
2022-12-14 16:45:10:027 INFO LDSU Descriptor
2022-12-14 16:45:10:027 INFO ===============
2022-12-14 16:45:10:028 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 16:45:10:028 INFO     Product Name : LDSBus 2CH Relay
2022-12-14 16:45:10:028 INFO     Product Version : 0.1
2022-12-14 16:45:10:028 INFO     UUID : LC01110101042200002
2022-12-14 16:45:10:029 INFO Sensors and actuators
2022-12-14 16:45:10:029 INFO ====================
2022-12-14 16:45:10:029 INFO ID: 0
2022-12-14 16:45:10:029 INFO     Manufacturer : Bridgetek Pte Ltd.
2022-12-14 16:45:10:029 INFO     Part Number : BRT-VDEV
2022-12-14 16:45:10:029 INFO     I2C Address : 0
2022-12-14 16:45:10:030 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 16:45:10:030 INFO ID: 1
2022-12-14 16:45:10:030 INFO     Manufacturer : Bridgetek Pte Ltd.
2022-12-14 16:45:10:030 INFO     Part Number : 2CH Relay
2022-12-14 16:45:10:030 INFO     I2C Address : 88
2022-12-14 16:45:10:030 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 16:45:10:031 INFO ============RELAY STATES============
2022-12-14 16:45:10:031 INFO Relay - CH 1: ON
2022-12-14 16:45:10:031 INFO Relay - CH 2: OFF
```

Figure 12 - LDSBus 2CH Relay Controller Sample Output

Sample File Reference: **LDSBus_2CH_Relay_iSense_Controller.py**

```
2022-12-14 17:00:18:749 INFO LDSU Descriptor
2022-12-14 17:00:18:749 INFO ===============
2022-12-14 17:00:18:749 INFO     Manufacturer : BRT Systems Pte Ltd.
2022-12-14 17:00:18:750 INFO     Product Name : LDSBus 2CH Relay + iSENSE
2022-12-14 17:00:18:750 INFO     Product Version : 0.1
2022-12-14 17:00:18:750 INFO     UUID : LC01010101042200092
2022-12-14 17:00:18:750 INFO Sensors and actuators
2022-12-14 17:00:18:751 INFO ====================
2022-12-14 17:00:18:751 INFO ID: 0
2022-12-14 17:00:18:751 INFO     Manufacturer : Bridgetek Pte Ltd.
2022-12-14 17:00:18:751 INFO     Part Number : BRT-VDEV
2022-12-14 17:00:18:752 INFO     I2C Address : 0
2022-12-14 17:00:18:752 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 17:00:18:752 INFO ID: 1
2022-12-14 17:00:18:752 INFO     Manufacturer : Bridgetek Pte Ltd.
2022-12-14 17:00:18:752 INFO     Part Number : 2CH Relay + iSENSE
2022-12-14 17:00:18:753 INFO     I2C Address : 88
2022-12-14 17:00:18:753 INFO     Maximum report rate (millsecs) : 1000
2022-12-14 17:00:18:753 INFO ============SENSOR DATA============
2022-12-14 17:00:18:753 INFO Current - CH 1: 1103.000 mA
2022-12-14 17:00:18:753 INFO Current - CH 2: 0.000 mA
2022-12-14 17:00:18:754 INFO ============RELAYS STATES============
2022-12-14 17:00:18:754 INFO Relay - CH 1: ON
2022-12-14 17:00:18:754 INFO Relay - CH 2: OFF
```

Figure 13 - LDSBus 2CH Relay iSense Controller Sample Output

3.11 LDSBus Trailing Edge Light Dimmer

LDSBus Trailing Edge Light Dimmer can be integrated with dimmable LED lamps for adjusting the percentage of light dimming. Our trailing edge technology uses a current that is turned off when the AC waveform ends. The operation is smoother, soft starting and silent. It can control up to 550W@240VAC or 230W@100VAC for single-change loading. The LDSBus Trailing Edge Light Dimmer has a 2-digit display to show the percentage of dimming. Zero crossing detection determines whether the AC input frequency is 50Hz or 60Hz before enabling dimming. Additionally, an external dimmer controller can be used to control light dimming.

```
2023-02-14 14:25:05:469 INFO LDSU Descriptor
2023-02-14 14:25:05:469 INFO ===============
2023-02-14 14:25:05:469 INFO     MANUFACTURER : BRT Systems Pte Ltd.
2023-02-14 14:25:05:469 INFO     PRODUCT_NAME : LDSBus Trailing Edge Dimmer
2023-02-14 14:25:05:469 INFO     PRODUCT_VERSION : 1.0
2023-02-14 14:25:05:469 INFO     UUID : LC030101051022000003
2023-02-14 14:25:05:469 INFO     NUMBER_OF_SENSORS_ACTUATORS : 2
2023-02-14 14:25:05:469 INFO Sensors and Actuators
2023-02-14 14:25:05:469 INFO ====================
2023-02-14 14:25:05:469 INFO     ID : 0
2023-02-14 14:25:05:469 INFO     MANUFACTURER : Bridgetek Pte Ltd.
2023-02-14 14:25:05:469 INFO     PART_NUMBER : BRT-VDEV
2023-02-14 14:25:05:469 INFO     I2C_ADDRESS : 0x0
2023-02-14 14:25:05:469 INFO     REPORT_RATE_MS : 1000
2023-02-14 14:25:05:470 INFO     ID : 1
2023-02-14 14:25:05:470 INFO     MANUFACTURER : Bridgetek Pte Ltd.
2023-02-14 14:25:05:470 INFO     PART_NUMBER : Light Dimmer
2023-02-14 14:25:05:470 INFO     I2C_ADDRESS : 0x58
2023-02-14 14:25:05:470 INFO     REPORT_RATE_MS : 1000
2023-02-14 14:25:05:470 INFO DIMMER
2023-02-14 14:25:05:470 INFO ******
2023-02-14 14:25:05:499 INFO Set brightness: 81
2023-02-14 14:25:05:530 INFO Get brightness: 81
```

Figure 14 - LDSBus Trailing Edge Light Dimmer Sample Output

3.12 LDSBus IO Controller

LDSBus IO Controller can be used for controlling Digital Output, Analog Output and reading Digital Input and Analog Input.

```
2023-02-15 10:48:10:718 INFO LDSU Descriptor
2023-02-15 10:48:10:719 INFO ===============
2023-02-15 10:48:10:719 INFO     MANUFACTURER : BRT Systems Pte Ltd.
2023-02-15 10:48:10:721 INFO     PRODUCT_NAME : LDSBus Isolated IO Controller
2023-02-15 10:48:10:721 INFO     PRODUCT_VERSION : 1.0
2023-02-15 10:48:10:721 INFO     UUID : LC060101051022000003
2023-02-15 10:48:10:721 INFO     NUMBER_OF_SENSORS_ACTUATORS : 1
2023-02-15 10:48:10:721 INFO Sensors and Actuators
2023-02-15 10:48:10:721 INFO ====================
2023-02-15 10:48:10:721 INFO     ID : 0
2023-02-15 10:48:10:721 INFO     MANUFACTURER : Bridgetek Pte Ltd.
2023-02-15 10:48:10:723 INFO     PART_NUMBER : BRT-VDEV
2023-02-15 10:48:10:723 INFO     I2C_ADDRESS : 0x0
2023-02-15 10:48:10:723 INFO     REPORT_RATE_MS : 1000
2023-02-15 10:48:10:724 INFO ===========SET DIGITAL OUTPUT==========
2023-02-15 10:48:10:724 INFO CH 1: ON
2023-02-15 10:48:10:725 INFO CH 2: OFF
2023-02-15 10:48:10:725 INFO ===========SET ANALOG OUTPUT===========
2023-02-15 10:48:10:725 INFO CH 1: 1.0 V
2023-02-15 10:48:10:726 INFO CH 2: 1.0 V
2023-02-15 10:48:10:726 INFO ===========GET DIGITAL INPUT===========
2023-02-15 10:48:10:726 INFO CH 1: OFF
2023-02-15 10:48:10:726 INFO CH 2: ON
2023-02-15 10:48:10:726 INFO ============GET ANALOG INPUT===========
2023-02-15 10:48:10:727 INFO CH 1: 2.0 V
2023-02-15 10:48:10:727 INFO CH 2: 2.0 V
```

Figure 15 - LDSBus Isolated IO Controller Sample Output

3.13 LDSBus IR Blaster

The LDSBus IR Blaster replaces traditional single device IR remote controllers to function as a universal remote controller.

It is equipped with 7 high power IR LEDs that provide 360° of coverage ranging up to 7 meters. Suitable for controlling Air-conditioners, TVs, Set-Top Boxes, curtains and blinds, fans and many more appliances.

```
2023-07-17 14:07:45:485 INFO LDSU Info
2023-07-17 14:07:45:485 INFO =========
2023-07-17 14:07:45:485 INFO     RESET_SOURCE : NONE
2023-07-17 14:07:45:485 INFO     FIRMWARE_VERSION : 1.4
2023-07-17 14:07:45:485 INFO     LAST_COMMAND_STATUS : 1
2023-07-17 14:07:45:485 INFO     BOARD_VOLTAGE : 5002
2023-07-17 14:07:45:485 INFO     NUMBER_OF_RECEIVED_QUERIES_COMMANDS : 47
2023-07-17 14:07:45:485 INFO     NUMBER_OF_RESPONSES : 51
2023-07-17 14:07:45:485 INFO     NUMBER_OF_ERROR_PACKETS_RECEIVED : 0
2023-07-17 14:07:45:485 INFO LDSU Descriptor
2023-07-17 14:07:45:485 INFO ===============
2023-07-17 14:07:45:485 INFO     MANUFACTURER : BRT Systems Pte Ltd.
2023-07-17 14:07:45:485 INFO     PRODUCT_NAME : LDSBus IR Blaster
2023-07-17 14:07:45:485 INFO     UUID : LC02010102032100084
2023-07-17 14:07:45:485 INFO     PRODUCT_VERSION : 1.0
2023-07-17 14:07:45:485 INFO     NUMBER_OF_SENSORS_ACTUATORS : 2
2023-07-17 14:07:45:485 INFO Sensors and Actuators
2023-07-17 14:07:45:485 INFO ====================
2023-07-17 14:07:45:485 INFO     ID : 0
2023-07-17 14:07:45:485 INFO     MANUFACTURER : Bridgetek Pte. Ltd.
2023-07-17 14:07:45:485 INFO     PART_NUMBER : BRT-VDEV
2023-07-17 14:07:45:485 INFO     I2C_ADDRESS : 0
2023-07-17 14:07:45:485 INFO     REPORT_RATE_MS : 1000
2023-07-17 14:07:45:485 INFO     ID : 1
2023-07-17 14:07:45:485 INFO     MANUFACTURER : Bridgetek Pte. Ltd.
2023-07-17 14:07:45:485 INFO     PART_NUMBER : IR Transmitter
2023-07-17 14:07:45:485 INFO     I2C_ADDRESS : 88
2023-07-17 14:07:45:485 INFO     REPORT_RATE_MS : 1000
2023-07-17 14:07:45:485 INFO Enter LIR Command:
26 0DD6 06EA 01B6 01B4 01AE 050A 019C 267C 01 2F 1B 2F 17 22 21 12 F1 EF 1E 11 22 F1 53 01 2
F 1B 2F 17 22 21 12 F1 D2 11 2F 15 21 11 2F 19 F2 51 21 22 12 2F 1D 22 2F 19 22 2F 1E 11 2F
16 2F 1E F1 42 22 12 21
2023-07-17 14:15:42:059 INFO Device State: 0
2023-07-17 14:15:42:059 INFO Enter LIR Command:
```

Figure 16 - LDSBus IR Blaster Sample Output

3.14 LDSBus RFID Reader

The LDSBus RFID Reader is designed to read access cards. This RFID Reader can be programmed for either high frequency (13.56MHz) or low frequency (125Hz) or dual frequency for both RFID cards. The RGB LED Indicator strips can be programmed with different colors and the reader features a built-in buzzer for alert.

```
2023-07-19 11:50:54:414 INFO LDSU Info
2023-07-19 11:50:54:414 INFO =========
2023-07-19 11:50:54:414 INFO     RESET_SOURCE : NONE
2023-07-19 11:50:54:414 INFO     FIRMWARE_VERSION : 2.7
2023-07-19 11:50:54:414 INFO     LAST_COMMAND_STATUS : 1
2023-07-19 11:50:54:414 INFO     BOARD_VOLTAGE : 4759
2023-07-19 11:50:54:414 INFO     NUMBER_OF_RECEIVED_QUERIES_COMMANDS : 237
2023-07-19 11:50:54:414 INFO     NUMBER_OF_RESPONSES : 287
2023-07-19 11:50:54:414 INFO     NUMBER_OF_ERROR_PACKETS_RECEIVED : 0
2023-07-19 11:50:54:414 INFO LDSU Descriptor
2023-07-19 11:50:54:414 INFO ===============
2023-07-19 11:50:54:414 INFO     MANUFACTURER : BRT Systems Pte Ltd.
2023-07-19 11:50:54:414 INFO     PRODUCT_NAME : LDSBus RFID Reader
2023-07-19 11:50:54:414 INFO     UUID : LC05010107142200005
2023-07-19 11:50:54:414 INFO     PRODUCT_VERSION : 1.0
2023-07-19 11:50:54:414 INFO     NUMBER_OF_SENSORS_ACTUATORS : 2
2023-07-19 11:50:54:414 INFO Sensors and Actuators
2023-07-19 11:50:54:414 INFO ====================
2023-07-19 11:50:54:414 INFO     ID : 0
2023-07-19 11:50:54:414 INFO     MANUFACTURER : BRT Systems Pte Ltd.
2023-07-19 11:50:54:414 INFO     PART_NUMBER : AIOT
2023-07-19 11:50:54:414 INFO     I2C_ADDRESS : 0
2023-07-19 11:50:54:414 INFO     REPORT_RATE_MS : 1000
2023-07-19 11:50:54:414 INFO     ID : 1
2023-07-19 11:50:54:414 INFO     MANUFACTURER : Kinetic Technologies
2023-07-19 11:50:54:414 INFO     PART_NUMBER : KTD2061
2023-07-19 11:50:54:414 INFO     I2C_ADDRESS : 0
2023-07-19 11:50:54:414 INFO     REPORT_RATE_MS : 1000
2023-07-19 11:50:56:517 INFO Card type : Single Frequency
2023-07-19 11:50:56:533 INFO RFID : 02 09 10 C2 55 D7 9A C3 03
```

Figure 17 - LDSBus RFID Reader Sample Output

4 API

4.1 LDSBus

This class handles all the communication of the LDSBus to the communication port Interface. User should instantiate this class and call the APIs to perform the required functionalities to read and write from the LDS Sensors.

```
class LDSBus:
    1. def usb_adapter_list (self)
    2. def open_comm_port (self, device_index : int = 0)
    3. def close_comm_port (self)
    4. def port_is_opened (self)
    5. def port_power_status (self)
    6. def ldsu_port_power (self, on_off=0)
    7. def ldsbus_port_power (self, on_off=0)
    8. def address_ldsu (self, ldsu_id : int)
    9. def scan_ldsu (self)
    10. def terminate_ldsu (self)
```

4.2 Member Functions

4.2.1 LDSBus List USB Adapters

The property returns the list of LDSBus USB Adapters connected to the machine.

4.2.1.1 API

```
def usb_adapter_list(self) -> List[AdapterInfo]:
```

4.2.1.2 Parameters

None	

4.2.1.3 Return

List	Returns the list of LDSBus USB adapters

4.2.2 LDSBus Open USB Adapter

This function initializes the COM (or) ttyUSB Port Interface of the Operating System. Windows users can find out the COM Port number from the Device Manager, while the ttyUSB Port number can be obtained from the Linux OS (/dev/ttyUSB*).

4.2.2.1 API

```
def open_comm_port(self, device_index: int) -> object:
```

4.2.2.2 Parameters

index	USB Adapter index from the usb list

4.2.2.3 Return

object	Return communication port handle if successfully open or None if not opened

4.2.3 LDSBus Close USB Adapter

This function will de-initialize the COM (or) ttyUSB Port Interface of the Operating System and turn off the LDSU Port or LDSBus power.

4.2.3.1 API

```
def close_comm_port(self):
```

4.2.3.2 Parameters

None	

4.2.3.3 Return

None	

4.2.4 LDSBus Port Status

This property returns the status whether the port is opened.

4.2.4.1 API

```
def port_is_opened(self) -> bool:
```

4.2.4.2 Parameters

None	

4.2.4.3 Return

bool	Return True if opened otherwise False

4.2.5 LDSU Port Power Control

With this function, then can discover only certain types of sensor devices during SCAN.

4.2.5.1 API

```
def ldsu_port_power(self, on_off=0) -> None:
```

4.2.5.2 Parameters

on_off	Power control value 0- Off (Default) 1- On

4.2.5.3 Return

None	

4.2.6 LDSBus Port Power Control

This function is used to control the power of the LDSBus Port.

4.2.6.1 API

```
def ldsbus_port_power(self, on_off=0) -> None:
```

4.2.6.2 Parameters

on_off	Power control value 0- Off (Default) 1- On

4.2.6.3 Return

None	

4.2.7 LDSBus Power Status

This property returns current power status.

4.2.7.1 API

```
def port_power_status(self) -> int:
```

4.2.7.2 Parameters

None	

4.2.7.3 Return

Int	Returns the current port status 0x00 - LDSBus and LDSU port power is turned OFF 0x01 – LDSBus Port power turned ON and LDSU port power is turned OFF 0x02 – LDSBus Port power turned OFF and LDSU port power is turned ON 0x03 – 0xFF – Reserved

4.2.8 LDSBus address the LDSU Device

This function is used to initiate communication with the LDSU device.

4.2.8.1 API

```
def address_ldsu(self, ldsu_id : int) -> bool:
```

4.2.8.2 Parameters

ldsu_id	**1-126** – Integer to search only specific LDSUID

4.2.8.3 Return

Int	returns True if device found in the bus, otherwise False

4.2.9 LDSBus scan LDSU devices

This function is used to scan the number of LDSU(s) connected to the bus.

4.2.9.1 API

```
def scan_ldsu(self) -> list:
```

4.2.9.2 Parameters

None	

4.2.9.3 Return

list	returns a list of numbers of LDSU(s) in the bus

4.2.10 LDSBus terminate the LDSU

This function is used to terminate the last addressed LDSU from the communication chain.

4.2.10.1 API

```
def terminate_ldsu(self) -> None:
```

4.2.10.2 Parameters

None	

4.2.10.3 Return

None	

4.3 LDSU

```
class LDSU:
    1. __init__(self, ldsbus : LDSBus)
    2. def get_uuid(self)
    3. def get_info(self)
    4. def get_descriptors(self)
    5. def get_sensors_actuators(self)
```

4.3.1 Constructor of LDSU

The instance of LDSBus is required to communicate with LDSU while instantiate LDSU.

4.3.1.1 API

```
def __init__ (self, ldsbus : LDSBus)
```

4.3.1.2 Parameters

LDSBus	Instance of LDSBus

4.3.1.3 Return

None	

4.3.2 Get UUID

This function is used to get the LDSU UUID.

4.3.2.1 API

```
def get_uuid(self) -> str:
```

4.3.2.2 Parameters

None	

4.3.2.3 Return

str	Returns the UUID String, length of 20 chars

4.3.3 Get Information

This function is used to get the last addressed LDSU information.

4.3.3.1 API

```
def get_info(self) -> LDSUInformation:
```

4.3.3.2 Parameters

None	

4.3.3.3 Return

LDSUInformation	Returns the device current status information
	{
	'firmware_version': 'x.x',
	'last_command_status': xx,
	'reset_source': 'EXTERNAL RESET',
	'board_voltage': xxxx,
	'number_of_received_queries_commands': xxxxx,
	'number_of_responses': xxxxx,
	'number_of_error_packets_received': xxxxx
	}

4.3.4 Get Descriptors

This function is used to get LDSU descriptions (*manufacturer, product name, version, uuid, number of sensors/actuators*).

4.3.4.1 API

```
def get_descriptors(self) -> LDSUDescription:
```

4.3.4.2 Parameters

None	

4.3.4.3 Return

LDSUDescription	Returns the ldsu description
	{
	'manufacturer': 'xxxxxxxx',
	'product_name': 'xxxxxxxx',
	'product_version': 'x.x',
	'uuid': 'Lxxxxxxxxxxxxxxxxxxx',
	'product_name': 'xxxxxxxxxxxx'
	}

4.3.5 Get Sensor and Actuator list

This function is used to get sensors actuators (i2c device id, manufacturer, part number, i2c address of component, report rate in milliseconds).

4.3.5.1 API

```
def get_sensors_actuators(self) -> List[LDSU_Sensor_Actuator]:
```

4.3.5.2 Parameters

None	

4.3.5.3 Return

List[LDSU_Sensor_Actuator]	Returns the list of sensors and actuators and it informations
	{
	'id': 'x',
	'manufacturer': 'xxxxxxxx',
	'part_number': 'xxxx',
	'i2c_7bit_address': xx,
	'report_rate_ms': xxxx
	}

4.4 LDSBus 4in1 Sensor

LDSBus4in1Sensor is a subclass of LDSUDevice class that specifically designed to work with 4-in-1 sensor. It provides the methods for reading temperature in °C, humidity in percentage, ambient light in lux and motion detection.

```
class LDSBus4in1Sensor (LDSUDevice):
    1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
    2. init(self)
    3. read(self)
```

4.4.1 Constructor

The constructor for the LDSBus 4in1 Sensor class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.4.1.1 API

```
def __init__ (self, ldsbus: LDSBus, ldsu_id: int)
```

4.4.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.4.1.3 Return

None	

4.4.2 Initialise Device

This function initializes the 4in1 sensor with

1. I2C speed setting.
2. Enable Virtual Device setting.

4.4.2.1 API

```
def init (self)
```

4.4.2.2 Parameters

None	

4.4.2.3 Return

None	

4.4.3 Read Sensor

This function reads the sensor and returns temperature, humidity, ambient light and motion detection.

4.4.3.1 API

```
def read (self)
```

4.4.3.2 Parameters

None	

4.4.3.3 Return

tuple	Returns the tuple of readings of 1. Temperature in °C 2. Humidity in % 3. Ambient light in lux 4. Motion (1 = movement detected, 0 = movement not detected)

4.5 LDSBus CO2 Sensor

LDSBusCO2Sensor is a subclass of LDSUDevice class that is specifically designed to work with the CO2 sensor. It provides the methods for reading ambient light in lux, CO2 in ppm, temperature in °C and humidity in percentage.

```
class LDSBusCO2Sensor (LDSUDevice):
   1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
   2. init(self)
   3. read(self)
```

4.5.1 Constructor

The constructor for the LDSBusCO2Sensor class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.5.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.5.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
int	The ID of the LDSU device

4.5.1.3 Return

None	

4.5.2 Initialise Device

This function initializes the CO2 sensor with

1. I2C speed setting.
2. Controller setting.
3. Sensor specific setting

4.5.2.1 API

```
def init (self)
```

4.5.2.2 Parameters

None	

4.5.2.3 Return

None	

4.5.3 Read Sensor

This function reads the sensor and returns ambient light in lux, CO2 in ppm, temperature in °C and humidity in percentage.

4.5.3.1 API

```
def read (self)
```

4.5.3.2 Parameters

None	

4.5.3.3 Return

tuple	Returns the tuple of readings of 1. Ambient light in lux 2. CO2 in ppm 3. Temperature in °C 4. Humidity in %

4.6 LDSBus DO Sensor

LDSBusDOSensor is a subclass of LDSUDevice class that specifically designed to work with DO sensor. It provides the methods for reading DO saturation in percentage and DO value in mg/L. DO represents Dissolved Oxygen.

```
class LDSBusDOSensor (LDSUDevice):
```

1. `__init__(self, ldsbus : LDSBus, ldsu_id : int)`
2. `init(self)`
3. `read(self)`

4.6.1 Constructor

The constructor for the LDSBusDOSensor class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.6.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.6.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
int	The ID of the LDSU device

4.6.1.3 Return

None	

4.6.2 Initialise Device

This function initializes the DO sensor with

1. I2C speed setting.
2. Get Calibrations of Sensor.
3. Controller setting.

4.6.2.1 API

```
def init (self)
```

4.6.2.2 Parameters

None	

4.6.2.3 Return

None	

4.6.3 Read Sensor

This function reads the sensor and returns temperature, humidity, ambient light and motion detection.

4.6.3.1 API

```
def read (self)
```

4.6.3.2 Parameters

None	

4.6.3.3 Return

tuple	Returns the tuple of readings of 1. Saturation in % 2. Value in mg/L

4.7 LDSBus Salinity Sensor

LDSBusSalinitySensor is a subclass of LDSUDevice class that specifically designed to work with Salinity sensor. It provides the methods for reading salinity of sensor in ppt.

```
class LDSBusSalinitySensor(LDSUDevice):
   1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
   2. init(self)
   3. read(self)
```

4.7.1 Constructor

The constructor for the LDSBusSalinitySensor class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.7.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.7.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.7.1.3 Return

None	

4.7.2 Initialise Device

This function initializes the 4 in 1 sensor with

1. I2C speed setting.
2. Get Calibrations of Sensor
3. Controller setting.

4.7.2.1 API

```
def init (self)
```

4.7.2.2 Parameters

None	

4.7.2.3 Return

None	

4.7.3 Read Sensor

This function reads the sensor and returns salinity in ppt.

4.7.3.1 API

```
def read (self)
```

4.7.3.2 Parameters

None	

4.7.3.3 Return

Float	Returns the tuple of readings of 1. Salinity in ppt

4.8 LDSBus EC Sensor

LDSBusECSensor is a subclass of LDSUDevice class that is specifically designed to work with EC sensor. It provides the methods for reading EC in mS/cm. EC represents Electrical Conductivity.

```
class LDSBusECSensor(LDSUDevice):
    1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
    2. init(self)
    3. read(self)
```

4.8.1 Constructor

The constructor for the LDSBusECSensor class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.8.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.8.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.8.1.3 Return

None	

4.8.2 Initialise Device

This function initializes the EC sensor with

1. I2C speed setting.
2. Get Calibration of Sensor.
3. Enable Virtual Device setting.

4.8.2.1 API

```
def init (self)
```

4.8.2.2 Parameters

None	

4.8.2.3 Return

None	

4.8.3 Read Sensor

This function reads the sensor and returns EC value in mS/cm.

4.8.3.1 API

```
def read (self)
```

4.8.3.2 Parameters

None	

4.8.3.3 Return

float	Returns the tuple of readings of 1. EC in mS/cm.

4.9 LDSBus ORP Sensor

LDSBusORPSensor is a subclass of LDSUDevice class that is specifically designed to work with the ORP sensor. It provides the methods for reading ORP value in mV. ORP represents Oxidation-Reduction Potential of a solution.

```
class LDSBusORPSensor(LDSUDevice):
   1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
   2. init(self)
   3. read(self)
```

4.9.1 Constructor

The constructor for the LDSBus ORP Sensor class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.9.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.9.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
int	The ID of the LDSU device

4.9.1.3 Return

None	

4.9.2 Initialise Device

This function initializes the ORP sensor with

1. I2C speed setting.
2. Get Calibration of Sensor.
3. Enable Virtual Device setting.

4.9.2.1 API

```
def init (self)
```

4.9.2.2 Parameters

None	

4.9.2.3 Return

None	

4.9.3 Read Sensor

This function reads the sensor and returns ORP value in mV (millivolt).

4.9.3.1 API

```
def read (self)
```

4.9.3.2 Parameters

None	

4.9.3.3 Return

float	Returns the tuple of readings of 1. ORP value in mV.

4.10 LDSBus pH Sensor

LDSBuspHSensor is a subclass of LDSUDevice class that is specifically designed to work with the pH sensor. It provides the methods for reading pH value.

```
class LDSBusPHSensor(LDSUDevice):
    1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
    2. init(self)
    3. read(self)
```

4.10.1 Constructor

The constructor for the LDSBusPHSensor class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.10.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.10.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.10.1.3 Return

None	

4.10.2 Initialise Device

This function initializes the 4 in 1 sensor with

1. I2C speed setting.
2. Get Calibration of Sensor.
3. Enable Virtual Device setting.

4.10.2.1 API

```
def init (self)
```

4.10.2.2 Parameters

None	

4.10.2.3 Return

None	

4.10.3 Read Sensor

This function reads the sensor and returns pH value.

4.10.3.1 API

```
def read (self)
```

4.10.3.2 Parameters

None	

4.10.3.3 Return

tuple	Returns the tuple of readings of 1. pH value.

4.11 LDSBus Thermocouple Sensor

LDSBusThermocoupleSensor is a subclass of LDSUDevice class that is specifically designed to work with a thermocouple. It provides the methods for reading hot-junction temperature, delta-junction temperature, cold-junction temperature in °C.

```
class LDSBusThermocoupleSensor(LDSUDevice):
   1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
   2. init(self)
   3. read(self)
```

4.11.1 Constructor

The constructor for the LDSBus Thermocouple Sensor class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.11.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.11.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.11.1.3 Return

None	

4.11.2 Initialise Device

This function initializes the 4 in 1 sensor with

1. I2C speed setting.
2. Enable Virtual Device setting.
3. I2C configuration setting

4.11.2.1 API

```
def init (self)
```

4.11.2.2 Parameters

None	

4.11.2.3 Return

None	

4.11.3 Read Sensor

This function reads the sensor and returns hot-junction temperature, delta-junction temperature and cold junction temperature in °C.

4.11.3.1 API

```
def read (self)
```

4.11.3.2 Parameters

None	

4.11.3.3 Return

Tuple	Returns the tuple of readings of 1. Hot Junction Temperature in °C 2. Delta Junction Temperature in °C 3. Cold Junction Temperature in °C

4.12 LDSBus 2CH Relay Controller

LDSBus2CHRelayController is a subclass of LDSUDevice class that is specifically designed to work with the 2CH Relay Controller. It provides the methods for setting of the relays and reading the states of relays.

```
class LDSBus2CHRelayController(LDSUDevice):
    1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
    2. init(self)
    3. write(self, channel, state, mode=0, polar=1, time1=1, time2=1, cycles=1)
    4. read(self)
```

4.12.1 Constructor

The constructor for the LDSBus2CHRelayController class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.12.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.12.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.12.1.3 Return

None	

4.12.2 Initialise Device

This function initializes the 4in1 sensor with

1. I2C speed setting.
2. Enable Virtual Device setting.

4.12.2.1 API

```
def init (self)
```

4.12.2.2 Parameters

None	

4.12.2.3 Return

None	

4.12.3 Write Controller

This function writes to controller to control relays.

4.12.3.1 API

```
def write (self, channel, state, mode=0, polar=1, time1=1, time2=1, cycles=1)
```

4.12.3.2 Parameters

channel	Relay channel 1 or 2
State	State 1 to set COM to NO, Set 0 to set COM to NC
Mode	Mode 0 to set Logic mode. Relay only response to what state given. Mode 1 to set Pulse mode. Relay at one state for time1 duration in seconds, then the other state for time2 duration in seconds.
Polar	Polarity 1 for Positive: 1. When State is set COM to NO, the physical relay set COM to NO 2. When State is set COM to NC, the physical relay set COM to NC Polarity 0 for Negative: 1. When State is set COM to NO, the physical relay set COM to NC 2. When State is set COM to NC, the physical relay set Com to NO
time1	The first portion of pulse duration in seconds.
time2	The second portion of pulse duration in seconds.
cycles	Number of cycles to repeat pulse pattern.

4.12.3.3 Return

None	

4.12.4 Read Controller

This function reads the states of relays.

4.12.4.1 API

```
def read (self)
```

4.12.4.2 Parameters

None	

4.12.4.3 Return

tuple	Returns the tuple of readings of 1. The state of relay1 2. The busy of relay1 (only response in pulse mode) 3. The state of relay2 4. The busy of relay2 (only response in pulse mode)

4.13 LDSBus 2CH Relay ISense Controller

LDSBus2CHRelayISenseController is a subclass of LDSUDevice class that is specifically designed to work with the 2CH Relay ISense Controller. It provides the methods for setting of the relays and reading the states of relays and current pass through.

```
class LDSBus2CHRelayISenseController(LDSUDevice):
   1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
   2. init(self)
   3. write(self, channel, state, mode=0, polar=1, time1=1, time2=1, cycles=1)
   4. read(self)
```

4.13.1 Constructor

The constructor for the LDSBus2CHRelayISenseController class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.13.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.13.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.13.1.3 Return

None	

4.13.2 Initialise Device

This function initializes the 4 in 1 sensor with

1. I2C speed setting.
2. Enable Virtual Device setting.

4.13.2.1 API

```
def init (self)
```

4.13.2.2 Parameters

None	

4.13.2.3 Return

None	

4.13.3 Write Controller

This function writes to controller to control relays.

4.13.3.1 API

```
def write (self, channel, state, mode=0, polar=1, time1=1, time2=1, cycles=1)
```

4.13.3.2 Parameters

channel	Relay channel 1 or 2
State	State 1 to set COM to NO, Set 0 to set COM to NC
Mode	Mode 0 to set Logic mode. Relay only response to what state given. Mode 1 to set Pulse mode. Relay at one state for time1 duration in seconds, then the other state for time2 duration in seconds.
Polar	Polarity 1 for Positive: 1. When State is set COM to NO, the physical relay set COM to NO 2. When State is set COM to NC, the physical relay set COM to NC Polarity 0 for Negative: 3. When State is set COM to NO, the physical relay set COM to NC 4. When State is set COM to NC, the physical relay set Com to NO
time1	The first portion of pulse duration in seconds.
time2	The second portion of pulse duration in seconds.
cycles	Number of cycles to repeat pulse pattern.

4.13.3.3 Return

None	

4.13.4 Read Controller

This function reads the states of relays and current pass through.

4.13.4.1 API

```
def read (self)
```

4.13.4.2 Parameters

None	

4.13.4.3 Return

tuple	Returns the tuple of readings of 1. The state of relay1 2. The busy of relay1 (only response in pulse mode) 3. The current pass-through relay 1 (in mA)

	4. The state of relay2 5. The busy of relay2 (only response in pulse mode) 6. The current pass-through relay 2 (in mA)

4.14 LDSBus Trailing Edge Light Dimmer

LDSBusDimmer is a subclass of LDSUDevice class that is specifically designed to work with the Trailing Edge Dimmer. It provides the methods for setting of the relays and reading the states of relays and current pass through.

```
class LDSBusDimmer(LDSUDevice):
    1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
    2. init(self)
    3. write(self, brightness)
    4. read(self)
```

4.14.1 Constructor

The constructor for the LDSBus Trailing Edge Light Dimmer class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.14.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.14.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.14.1.3 Return

None	

4.14.2 Initialise Device

This function initializes the 4 in 1 sensor with

1. I2C speed setting.
2. Enable Virtual Device setting.

4.14.2.1 API

```
def init (self)
```

4.14.2.2 Parameters

None	

4.14.2.3 Return

None	

4.14.3 Write Controller

This function set the brightness of the Trailing Edge Light Dimmer.

4.14.3.1 API

```
def write (self, brightness : int = 0)
```

4.14.3.2 Parameters

brightness	Set 0 to 100 for the brightness percentage

4.14.3.3 Return

None	

4.14.4 Read Sensor

This function reads the brightness setting of dimmer.

4.14.4.1 API

```
def read (self)
```

4.14.4.2 Parameters

None	

4.14.4.3 Return

Int	Returns the setting of 1. Brightness setting of the Trailing Edge Dimmer in percentage.

4.15 LDSBus Isolated IO Controller

LDSBusIOController is a subclass of LDSUDevice class that is specifically designed to work with the Isolated IO Controller. It provides the methods for writing Digital Output and Analog Output, reading Digital Input and Analog Input.

```
class LDSBusIOController(LDSUDevice):
    1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
    2. init(self)
    3. write(self, channel: str, value: int | float)
    4. read(self, channel: str = None)
```

4.15.1 Constructor

The constructor for the LDSBusIOController class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.15.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.15.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
Int	The ID of the LDSU device

4.15.1.3 Return

None	

4.15.2 Initialise Device

This function initializes the 4in1 sensor with

1. I2C speed setting.
2. Enable Virtual Device setting.

4.15.2.1 API

```
def init (self)
```

4.15.2.2 Parameters

None	

4.15.2.3 Return

None	

4.15.3 Write Controller

This function writes values to specific channel given.

4.15.3.1 API

```
def write (self, channel: str, value : int | float)
```

4.15.3.2 Parameters

channel: str	1. “OUT1” and “OUT2” to set the Digital Output 2. “VOUT1” and “VOUT2” to set the Analog Output
value: int | float	1. When channel is “OUT1” and “OUT2”, value = 1 to set the Digital Output “ON” and value = 0 to set the Digital Output “OFF”. 2. When channel is “VOUT1” and “VOUT2”, value from 0.0 to 10.0 to set the Analog Output voltage.

4.15.3.3 Return

None	

4.15.4 Read Sensor

This function reads the digital inputs and analog inputs.

4.15.4.1 API

```
def read (self, channel: str=None)
```

4.15.4.2 Parameters

Channel: str	1. "IN1" to read Digital Input channel 1 2. "IN2" to read Digital Input channel 2 3. "VIN1" to read Analog Input channel 1 4. "VIN2" to read Analog Input channel 2 5. None to read all the above.

4.15.4.3 Return

Tuple | int | float	Returns the tuple of readings of 1. "IN1": 1 for "ON" and 0 for "OFF" 2. "IN2": 1 for "ON" and 0 for "OFF" 3. "VIN1": 0.0 to 10.0 volt 4. "VIN2": 0.0 to 10.0 volt.

4.16 LDSBus IR Blaster

LDSBusIRBlaster is a subclass of LDSUDevice class that specifically designed to work with LDSBus IR Blaster. It provides the methods for writing IR Command.

```
class LDSBusIRBlaster(LDSUDevice):
    1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
    2. init(self)
    3. write(self, command:str, user_cfreq:int=38, dutycycle:int=0, format:str='lir')
```

4.16.1 Constructor

The constructor for the LDSBus IR Blaster class takes in the instance of LDSBus class and ID of the LDSU device.

The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.16.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.16.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
int	The ID of the LDSU device

4.16.1.3 Return

None	

4.16.2 Write Command

This function write and blast the IR command.

4.16.2.1 API

```
def write(self, command:str, user_cfreq:int=38, dutycycle:int=0, format:str='lir')
```

4.16.2.2 Parameters

command	The IR command to be blasted.
user_cfreq	Carrier frequency. Default = 38 kHz.
dutycycle	Duty cycle of the carrier in percentage. Default = 0, firmware will set to default 36 percent.
format	Format of the IR blaster command. 1. 'lir' : based on LIR format. Learn more from LDSBus IR Blaster Application Note 2. 'base64': Encoded LIR format in base64.

4.16.2.3 Return

dict	Returns a dictionary of readings:
	{
	"Device State": int,
	"Error": bool,
	"Busy": bool
	}
	
	1. "Device State": LDSBus IR Blaster device state that includes both Error and Busy state. Zero should be expected after sending the correct IR command format.
	2. "Error": True = IR command is invalid.
	3. "Busy": True = IR Blaster is busy sending out command.

4.17 LDSBus RFID Reader

LDSBusRFIDReader is a subclass of LDSUDevice class that specifically designed to work with LDSBus RFID Reader. It provides the methods for reading RFID card info and writing LED strips states.

```
class LDSBusRFIDReader(LDSUDevice):
```

1. __init__(self, ldsbus : LDSBus, ldsu_id : int)
2. init(self)
3. read(self)
4. read_rfid_mode(self)
5. read_rfid_enable(self)
6. set_rfid_enable(self, enable: int=0x11)
7. set_rfid_mode(self, ledEn: bool=False, buzzerEn: bool=False, lowFreqEn: bool=False, highFreqEn: bool=False)
8. set_led_onoff(self, onOff: bool=False)
9. set_led_onColor(self, onColor: int)
10. set_led_offColor(self, offColor: int)
11. set_led_strip(self, strip: int)
12. set_led_current(self, redCurrent: int, greenCurrent: int, blueCurrent: int)

4.17.1 Constructor

The constructor for the LDSBus RFID Reader class takes in the instance of LDSBus class and ID of the LDSU device. The constructor will call the parent constructor to address the LDSU device, getting information, then call the init function for device specific initializations.

4.17.1.1 API

```
def __init__ (self, ldsbus : LDSBus, ldsu_id : int)
```

4.17.1.2 Parameters

LDSBus	Instance of LDSBus class that represents the bus with the device is connected
int	The ID of the LDSU device

4.17.1.3 Return

None	

4.17.2 Read Command

This function reads the detected RFID Card info. After successfully read the RFID Card info, RFID Reader will clears the RFID Card info and waits for the next RFID Card to be tapped.

4.17.2.1 API

```
def read(self)
```

4.17.2.2 Parameters

None	

4.17.2.3 Return

Dict	Returns a dictionary of readings:
	{
	"CardDetected": bool,
	"CardType": str,
	"RFID": bytes
	}
	
	1. "CardDetected": True if new card was detected.
	2. "CardType": "Single Frequency" or "Dual Frequency". None if no card was detected.
	3. "RFID": return in bytes. None if no card was detected.

4.17.3 Read RFID Reader Current Mode

This function reads the RFID Reader Mode.

4.17.3.1 API

```
def read_rfid_mode(self)
```

4.17.3.2 Parameters

None	

4.17.3.3 Return

Dict	Returns a dictionary of readings:

	{
	"Mode": int,
	"LedEnable": bool,
	"BuzzerEnable": bool,
	"LowFreqEnable": bool,
	"HighFreqEnable": bool
	}
	
	1. "Mode": RFID Reader mode.
	2. "LedEnable": True if Internal RFID module LED enable.
	3. "BuzzerEnable": True if Internal RFID module Buzzer enable.
	4. "LowFreqEnable": True if Internal RFID module enables Low Freq Card detection.
	5. "HighFreqEnable": True if Internal RFID module enables High Freq Card detection.

4.17.4 Read RFID Enable Status

This function reads the RFID Reader enable status.

4.17.4.1 API

```
def read_rfid_enable(self)
```

4.17.4.2 Parameters

None	

4.17.4.3 Return

Int	Returns int with the following definition 1. 0b1xxx xxxx: Software RFID Read Enabled. 2. 0b0xxx xxxx: Software RFID Read Disabled due to Card detected. 3. 0bxxxx xxx1: Internal RFID module Power Enable.

4.17.5 Set RFID Enable

This function sets the RFID Reader enable.

4.17.5.1 API

```
def set_rfid_enable(self, enable: int=0x11)
```

4.17.5.2 Parameters

enable	Sets the RFID Reader enable with following definition 1. 0b0001 0000: Enable software read RFID module 2. 0b0000 0001: Power Enable RFID module 3. 0b0001 0001: Enable both stated above.

4.17.5.3 Return

int	Returns int with the following definition 1. 0b1xxx xxxx: Software RFID Read Enabled. 2. 0b0xxx xxxx: Software RFID Read Disabled due to Card detected. 3. 0bxxxx xxx1: Internal RFID module Power Enable.

4.17.6 Set RFID Mode

This function sets the RFID Reader mode.

4.17.6.1 API

```
def set_rfid_mode(self, ledEn:bool=False, buzzerEn:bool=False,
lowFreqEn:bool=False, highFreqEn:bool=False)
```

4.17.6.2 Parameters

ledEn	True to enable internal RFID Module LED
buzzerEn	True to enable RFID Module Buzzer.
lowFreqEn	True to enable Low Frequency RFID Card detection.
highFreqEn	True to enable High Frequency RFID Card detection.

4.17.6.3 Return

None	

4.17.7 Set LED Strip ON/OFF

This function sets the RFID Reader LED Strip ON or OFF.

4.17.7.1 API

```
def set_led_onoff(self, onOff: bool=False)
```

4.17.7.2 Parameters

onOff	True to turn on LED Strip. False to turn off LED Strip. *Note: LED Current setting will affect the LED Strip color or brightness.

4.17.7.3 Return

None	

4.17.8 Set LED Strip ON Color

This function sets the RFID Reader LED Strip ON Color. Default firmware setting is Red Color.

4.17.8.1 API

```
def set_led_onColor(self, onColor: int)
```

4.17.8.2 Parameters

onColor	ON Color setting as described below 0 = White 1 = Yellow 2 = Pink 3 = Red 4 = Turquoise 5 = Green 6 = Blue 7 = OFF

4.17.8.3 Return

None	

4.17.9 Set LED Strip OFF Color

This function sets the RFID Reader LED Strip OFF Color. Default firmware setting is OFF.

4.17.9.1 API

```
def set_led_offColor(self, offColor: int)
```

4.17.9.2 Parameters

offColor	OFF Color setting as described below 0 = White 1 = Yellow 2 = Pink 3 = Red 4 = Turquoise 5 = Green 6 = Blue 7 = OFF

4.17.9.3 Return

None	

4.17.10 Set LED Strip Mode

This function sets the RFID Reader LED Strip mode either left strip, right strip or both strips light up.

4.17.10.1 API

```
def set_led_strip(self, strip: int)
```

4.17.10.2 Parameters

Strip	0 = Left strip light up. (Connector port pointing down.) 1 = Right strip light up. (Connector port pointing down.) 2 = Both strips light up.

4.17.10.3 Return

None	

4.17.11 Set LED Strip Current

This function sets the RFID Reader LED Strip current to control RGB channel brightness.

4.17.11.1 API

```
def set_led_current(self, redCurrent: int, greenCurrent:int, blueCurrent:int)
```

4.17.11.2 Parameters

redCurrent	Red channel of LED current output. (Limited to 0x20 or 4.0mA, any number above will be set to 0x20 or 4.0mA)
greenCurrent	Green channel of LED current output. (Limited to 0x20 or 4.0mA, any number above will be set to 0x20 or 4.0mA)
blueCurrent	Blue channel of LED current output.

	(Limited to 0x20 or 4.0mA, any number above will be set to 0x20 or 4.0mA)

4.17.11.3 Return

None	

4.18 Logging APIs

```
class LDSLogging:
    1. def info(cls, msg, *args, **kwargs)
    2. def warning(cls, msg, *args, **kwargs)
    3. def error(cls, msg, *args, **kwargs)
    4. def critical(cls, msg, *args, **kwargs)
    5. def debug(cls, msg, *args, **kwargs)
    6. def ldsu_info(cls, info)
    7. def sensor_actuator_list(cls, list_of_sensors)
    8. def ldsu_descriptor(cls, descriptor)
```

Logging format : %(asctime)s:%(msecs)03d %(levelname)s %(message)s'

5 Contact Information

Refer to https://brtsys.com/contact-us/ for contact information.

Appendix A – References

Document References

BRTSYS_AN_001_LDSBus_Configuration_Utility_Guide

BRTSYS_AN_002_LDSU IR Blaster Application Note

BRTSYS_AN_003_LDSBus Python SDK on IDM2040 User Guide

Acronyms and Abbreviations

Terms	Description
API	Application Programming Interface
IR	Infra-Red
LDSBus	Long Distance Sensor Bus
LDSU	Long Distance Sensor Unit
SDK	Software Development Kit
USB	Universal Serial Bus

Appendix B – List of Figures & Tables

List of Figures

List of Tables

NA

Appendix C – Revision History

Document Title: BRTSYS_API_001 LDSBus Python SDK Guide
Document Reference No.: BRTSYS_000013
Clearance No.: BRTSYS#034
Product Page: https://brtsys.com/ldsbus/
Document Feedback: Send Feedback

Revision	Changes	Date
Version 1.0	Initial Release	29-11-2021
Version 1.1	Updated release under BRT Systems	05-04-2023
Version 1.2	Document updated as per SDK V3.1.0-1.0.3 Section 1 errata: Windows 10/11 Section 2.1.2 Ubuntu should be 20.02 Section 2.4.2.6 IDM2040 diagram updated Section 3.1 Added LDSBus port power diagram to highlight the functions of LDSBus USB Adapter. Added Section 3.13 for LDSBus IR Blaster Added Section 3.14 for LDSBus RFID Reader Added Section 4.16 for LDSBus IR Blaster Added Section 4.17 for LDSBus RFID Reader Updated HVT references to Quad T-Junction; Updated Singapore Address	22-09-2023