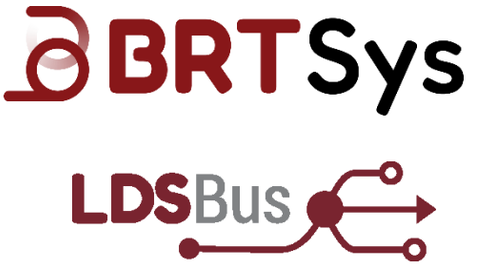


BRTSYS_API_004

LDSBus DotNet SDK Guide

Version 1.1

Issue Date: 22-09-2023

This document provides information about the DotNet SDK 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 .Net SDK is available on Microsoft Windows 10/11 [x64] platforms. The LDSBus .Net SDK implements the LDSBus host that manages and controls LDSU devices on the LDS bus 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 .Net 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 Cables 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]

2.1.3 Software

- Visual Studio 2022 Community and greater editions
- .NET Framework 4.7.2

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.

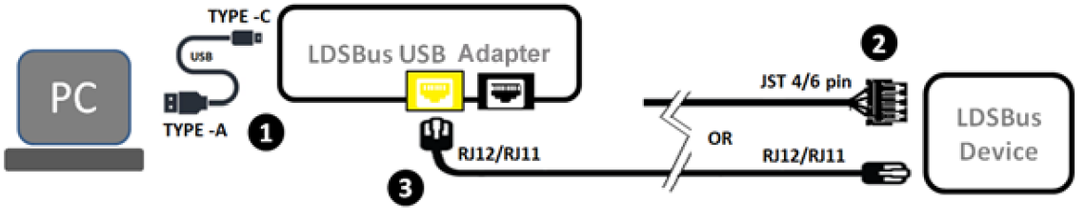


Figure 1 - LDSBus Device (Sensors / Actuators) Configuration

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.

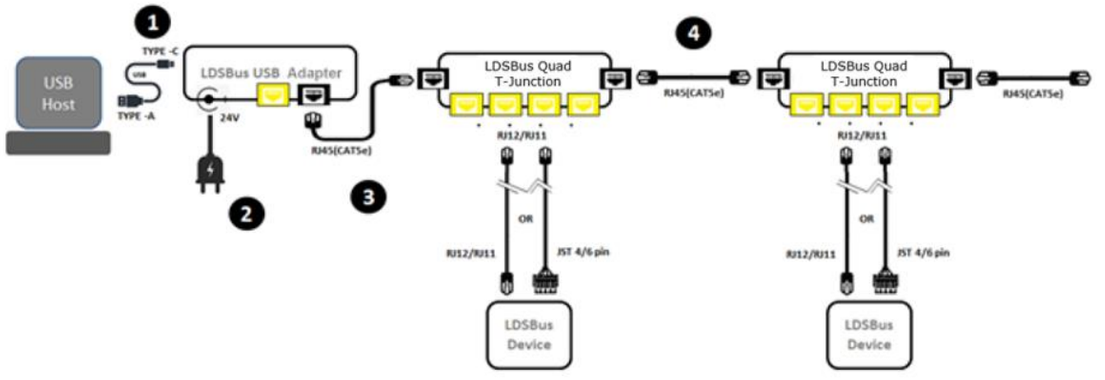


Figure 2 – LDSBus Creation

2.4 Installation

2.4.1 How to get the .Net SDK

2.4.1.1 Standalone Package

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

2.4.2 Software Installation

LDSBus .NET 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

- Visual Studio 2022 Community and greater editions
- .NET Framework 4.7.2
- Application runs in 64-bit.

2.4.2.2 LDSBus .Net SDK Directory Structure

```
DotNet SDK Library
\--> LDSBus_Csharp_Sample [Test scripts and program start location]
    \--> Library [DLL for LDSBus]
        \--> lds_driver_install.bat [Batch script to copy ldsbus.dll to C:\Windows\System32]
        \--> ldsbus.dll [Library to be used by LDSBusCSharpSDK.dll]
        \--> ftd2xx.dll [Library to access FTDI device]
        \--> LDSBusCSharpSDK.dll [.NET Framework 4.7.2 library for development]
    \--> Actuators [Sample scripts for Actuators]
    \--> Sensors [Sample scripts for Sensors]
    \--> LDSBusSample.csproj [Visual Studio C# project file]
    \--> LDSBusSample.sln [Visual Studio Solution file]
    \--> Program.cs [the main program file to run one of the sample scripts.]
```

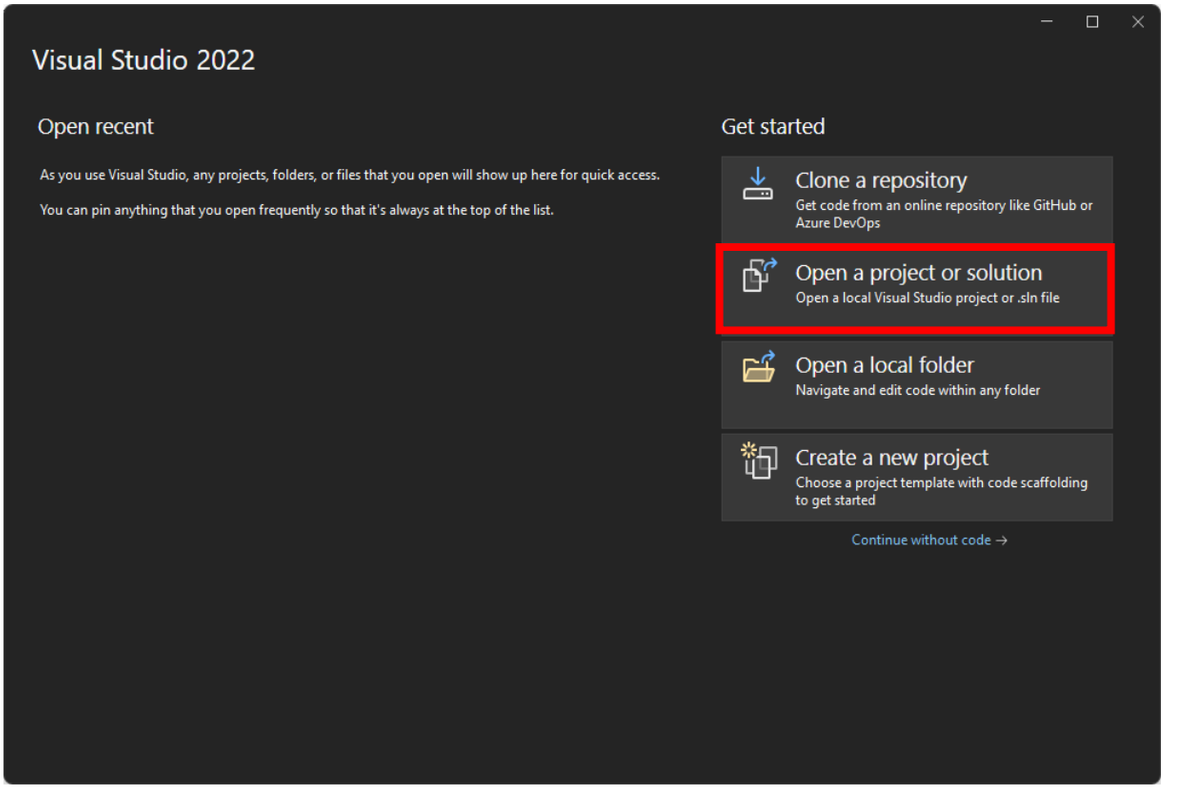
2.4.2.3 Windows Installation

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

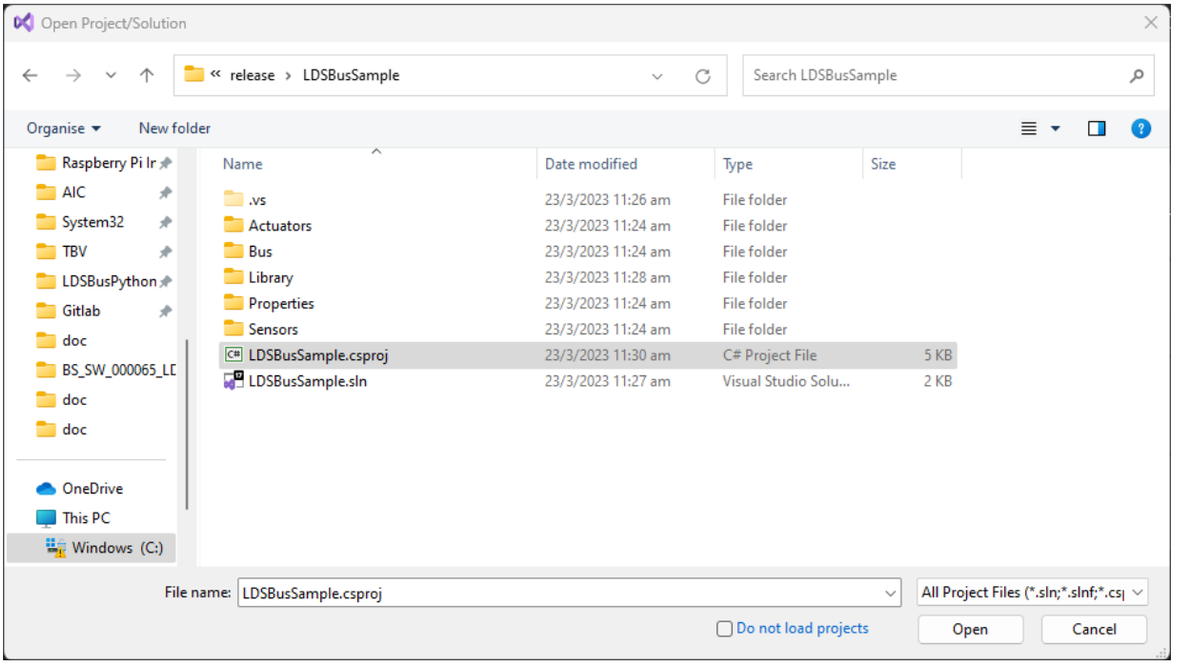
```
C:\Users\Documents\BRTSystems\LDSBus_Csharp_Sample\Library>lds_driver_install.bat
```

2.4.2.4 How to open the project file

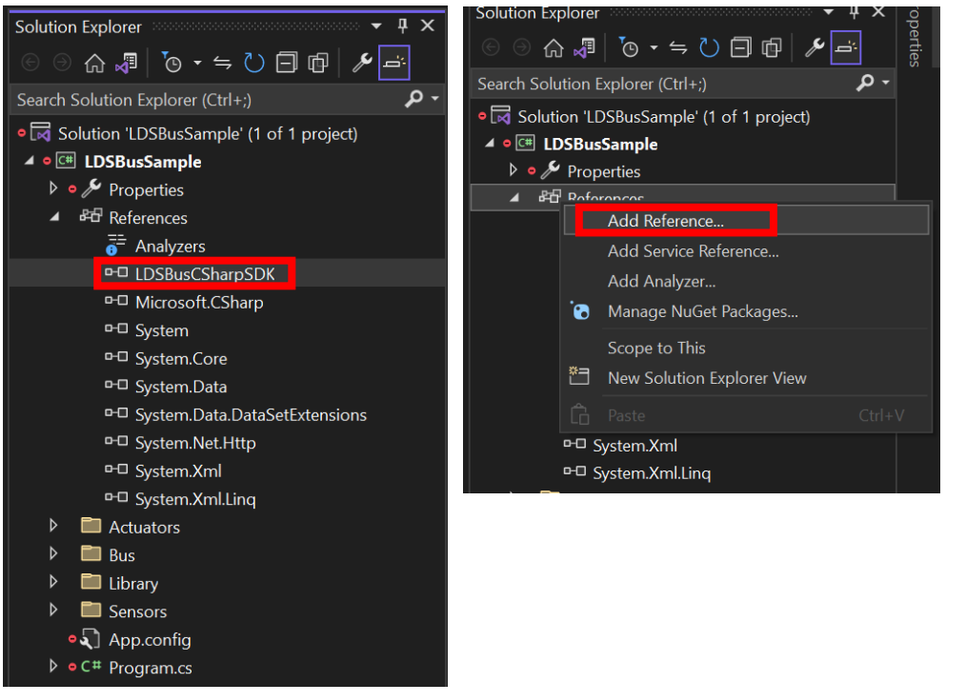
- Open Visual Studio; Click **"Open a project or solution"**.



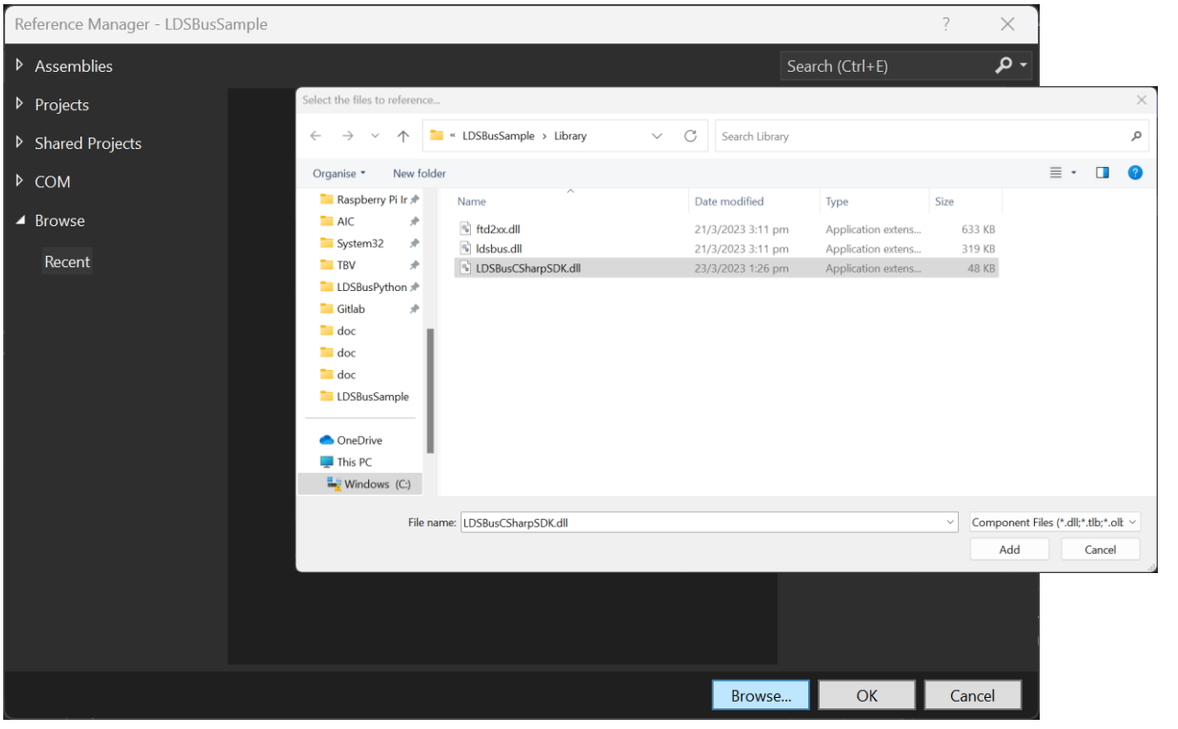
- Select **"LDSBusSample.csproj"** to open the project file.



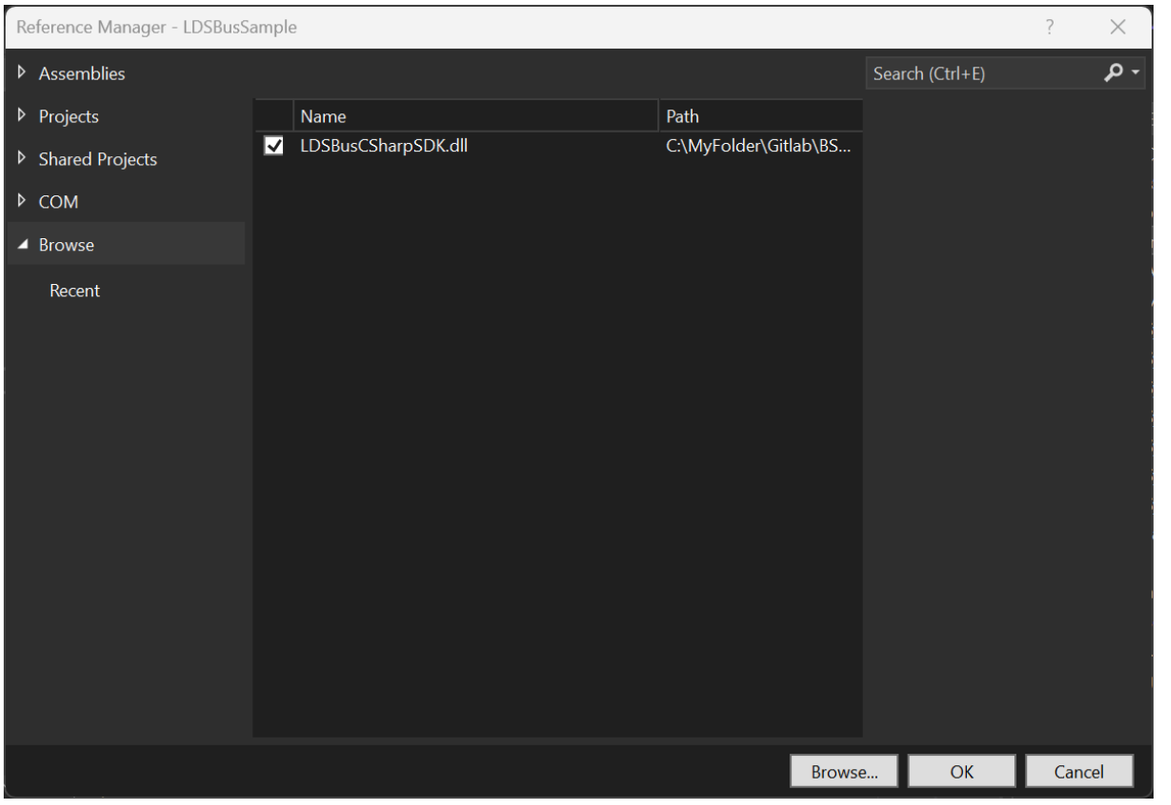
- Select the **"LDSBusCSharpSDK.dll"** checkbox under **"References"**; If **"LDSBusCSharpSDK.dll"** is not found, then, right click **"References > Add Reference..."**.



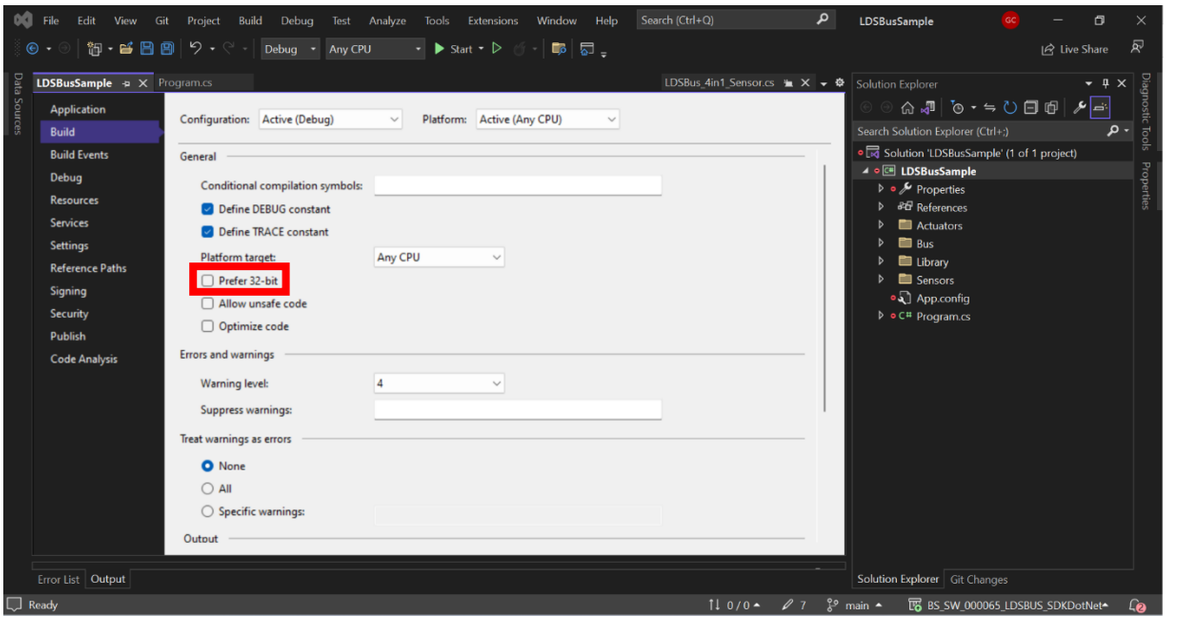
- Click **"Browse"**; Look for the **"LDSBusCSharpSDK.dll"** in **".\LDSBus_Csharp_Sample\Library\"** folder; Click **[Add]**;



- Upon adding the file, click **[OK]**.



- Ensure **"Prefer 32-bit"** is unticked to run the project in 64-bit since **"LDSBusCSharpSDK.dll"** and **"ldsbus.dll"** are built for 64-bit platforms.



2.4.2.5 Sensor Sample Scripts

All supported sensor 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 samples provided –

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

3 Samples & Output

3.1 LDSBus Sample Script Pseudocode

All the sample scripts follow the pseudocode as mentioned below with minor modifications to sensors and 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

```
using System;
using System.Collections.Generic;
using System.Threading;
using LDSBusCSharpSDK;

namespace LDSBusSample.Sensors
{
    using log = LDSLogger;

    internal class LDSBus_4in1_Sensor
    {
        static LDSBus ldsbus;

        private static int handle_selection()
        {
            string input = string.Empty;
            int _check_input = 0;
            List<uint> indices = new List<uint>();
            foreach (var adapter in ldsbus.usb_adapter_list)
            {
                indices.Add(adapter.index);
            }
            string index_list = string.Join(“, “, indices.ToArray());

            while (string.IsNullOrEmpty(input))
            {
                log.info($”Select USB Adapter [{index_list}] > “);
                input = Console.ReadLine();
                if (string.IsNullOrEmpty(input)) continue;
                if (!Int32.TryParse(input, out _check_input))
                {
                    input = string.Empty;
                    continue;
                }
            }
            return _check_input;
        }

        public static void app_start()
        {
```

```
            try
            {
                ldsbus = new LDSBus();
                if (ldsbus.usb_adapter_list.Count == 0)
                {
                    log.error(“LDSBus USB Adapter not connected.”);
                    log.info(“Press ‘Enter’ to exit”…");
                    Console.ReadLine();
                    return;
                }

                Console.CancelKeyPress += delegate
                {
                    ldsbus.close_comm_port();
                    log.in“o("Ctrl + C is pressed. Application stopp”d.");
                };

                foreach (var adapter in ldsbus.usb_adapter_list)
                {
                    log.inf”($"Device {adapter.ind”x}");
                    log.inf”($"\t Description: {adapter.descripti”n}");
                    log.inf”($"\t Serial: {adapter.seri”l}");
                }

                if (ldsbus.usb_adapter_list.Count > 1)
                {
                    // Select USB Adapter (if there is more than one LDSBus USB Adapter)
                    ldsbus.open_comm_port(handle_selection());
                }
                else
                {
                    // if there is only one LDSBus USB Adapter
                    ldsbus.open_comm_port();
                }
                log.in“o("Successfully opened the por”..");

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

                // Perform a scan
                log.in“o("Scanning”..");
                var device_list = ldsbus.scan_ldsu();
                if (device_list.Length == 0)
                {
                    log.err“r("LDSBus device not fou”d.");
                    log.in“o("Pre‘s 'En’er' to exit”..");
                    Console.ReadLine();
                    return;
                }

                // Number of devices found
                log.inf”($"Number of devices found : {device_list.Length}, ID(s) is/are :
{string.Jo“n(“, ", device_lis”)}");
                var device_id = device_list[0];

                // Based on the known device connected, initialise the device
                var ldsu = new LDSBus4in1Sensor(ldsbus, device_id);
                // Get info of the device
                log.inf”($"UUID: {ldsu.ldsu.get_uuid”)}");
```

```
                log.inf”($"Last command : 0x{ldsu.ldsu.get_lastcommand():”X}");
                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)
                {
                    log.clear();
                    log.ldsu_descriptor(ldsu.descriptor);
                    log.sensor_actuator_list(ldsu.sa_list);
                    log.in“o("SENSOR D”TA");
                    log.info(new string('*', "SENSOR_DATA".Length));

                    ldsu.read();

                    log.info($"TEMPERATURE : {ldsu.temperature:F2} °C");
                    log.info($"HUMIDITY : {ldsu.humidity:F2} %");
                    log.info($"ALS : {ldsu.als:F0} LUX");
                    log.info($"MOTION : {ldsu.motion}");

                    Thread.Sleep(5000);
                }

            }
            catch (Exception ex)
            {
                log.critical("Something went wrong! " + ex.ToString());
                ldsbus.close_comm_port();
                log.info("Press 'Enter' to exit...");
                Console.ReadLine();
            }
        }
    }
}
```

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.

```
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] >
```

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)`

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

```
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

```
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 3.12.

3.2 LDSBus 4in1 Sensor

The LDSBus Multi 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. Available in 4in1 and 3in1 (without motion detector) combinations, the device can be flush mounted on ceilings or swivel mounted on walls. The multi-sensor works with the BRTSys PanL Smart Living, IoTPortal Gateway and LDSBus Python SDK products.

Sample File Reference: **LDSBus_4in1_Sensor.cs**

```
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 3 - LDSBus 4in1 Sensor Sample Output

3.3 LDSBus CO2 Sensor

LDSBus CO2 Sensor features four sensors in a compact, low-profile design. The CO2 Sensor device includes sensors to measure CO2, temperature, humidity and ambient light. The devices can be flush mounted on the ceiling or swivel mounted on the wall. CO2 Sensors are compatible with the BRTSYS IoTPortal, PanL Smart Living and LDSBus Python SDKs.

Sample File Reference: **LDSBus_CO2_Sensor.cs**

```
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 : LS010101590722000l2
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 4 - 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.cs**

```
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 5 - 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.cs**

```
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 6 - 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.cs**

```
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 7 - 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.cs**

```
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 8 - 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.cs**

```
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 9 - 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.cs**

```
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 10 - 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 closer. 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.cs**

```
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 11 - LDSBus 2CH Relay Controller Sample Output

Sample File Reference: **LDSBus_2CH_Relay_iSense_Controller.cs**

```
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 12 - 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.

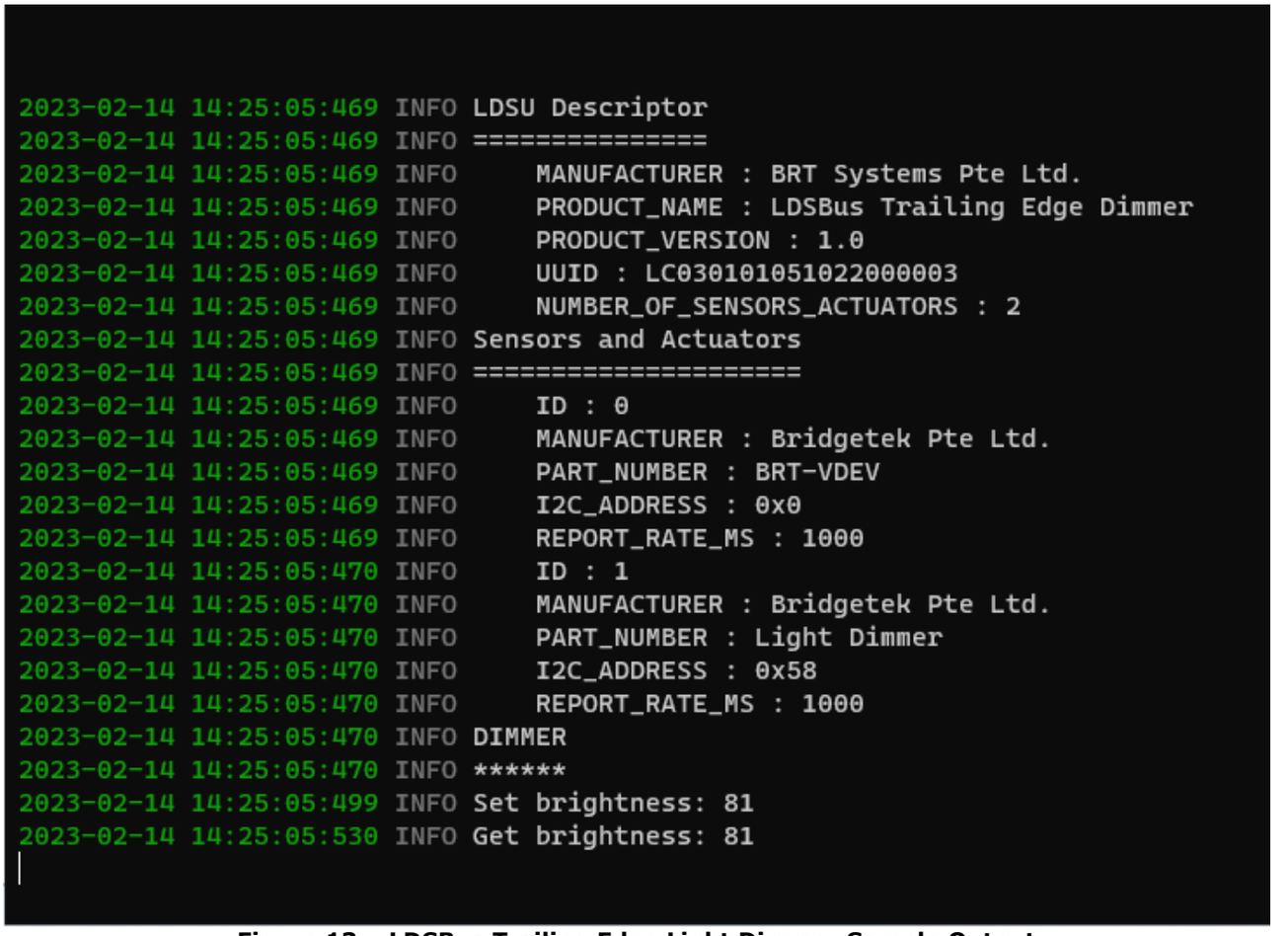


Figure 13 – LDSBus Trailing Edge Light Dimmer Sample Output

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 14 – LDSBus IO Controller 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.

```
public class LDSBus
```

```
1. List<AdapterInfo> usb_adapter_list
2. bool open_comm_port (int device_index = 0)
3. void close_comm_port ()
4. bool port_is_opened
5. uint port_power_status
6. void ldsu_port_power (int on_off = 0)
7. void ldsbus_port_power (int on_off = 0)
8. bool address_ldsu (byte ldsu_id)
9. byte[] scan_ldsu ()
10. void terminate_ldsu ()
```

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

```
List<AdapterInfo> usb_adapter_list
```

4.2.1.2 Return

List<AdapterInfo>	Returns the list of LDSBus USB adapters

4.2.2 LDSBus Open USB Adapter

This function initializes the COM Port Interface of the Operating System. Windows users can find out the COM Port number from the Device Manager.

4.2.2.1 API

```
bool open_comm_port(int device_index = 0)
```

4.2.2.2 Parameters

index	USB Adapter index from the usb adapter list

4.2.2.3 Return

bool	Return true if successfully open or false if not opened

4.2.3 LDSBus Close USB Adapter

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

4.2.3.1 API

```
void close_comm_port()
```

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

```
bool port_is_opened
```

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

By using this function, only certain sensor types can be discovered during scan.

4.2.5.1 API

```
void ldsu_port_power(int on_off = 0)
```

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

```
void ldsbus_port_power(int on_off = 0)
```

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

```
uint port_power_status
```

4.2.7.2 Parameters

None	

4.2.7.3 Return

uint	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 – LDSBus Port power turned ON and LDSU port power is turned ON

4.2.8 LDSBus address the LDSU Device

The purpose of this function is to initiate communication with the LDSU device.

4.2.8.1 API

```
bool address_ldsu(byte ldsu_id)
```

4.2.8.2 Parameters

ldsu_id	**1-126** – Integer to search only specific LDSU ID

4.2.8.3 Return

bool	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

```
byte[] scan_ldsu()
```

4.2.9.2 Parameters

None	

4.2.9.3 Return

byte[]	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

```
void terminate_ldsu()
```

4.2.10.2 Parameters

None	

4.2.10.3 Return

None	

4.3 LDSU

```
public class LDSU
```

1. LDSU(LDSBus ldsbus)
2. string get_uuid()
3. LDSUInformation get_info()
4. LDSU_Descriptor get_descriptors()
5. List<LDSU_Sensor_Actuator> get_sensors_actuators()

4.3.1 Constructor of LDSU

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

4.3.1.1 API

```
LDSU (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 last addressed LDSU UUID.

4.3.2.1 API

```
string get_uuid()
```

4.3.2.2 Parameters

None	

4.3.2.3 Return

string	Returns the UUID String, length of 19 chars

4.3.3 Get Information

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

4.3.3.1 API

```
LDSUInformation get_info()
```

4.3.3.2 Parameters

None	

4.3.3.3 Return

LDSUInformation	Returns the device current status information string firmware_version = "x.x" uint last_command_status = xx string reset_source = "EXTERNAL RESET" uint board_voltage = xxxx uint number_of_received_queries_commands = xxxxx uint number_of_responses = xxxxx uint 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

```
LDSU_Descriptor get_descriptors()
```

4.3.4.2 Parameters

None	

4.3.4.3 Return

LDSU_Descriptor	Returns the ldsu description string manufacturer = "xxxxxxxx" string product_name = "xxxxxxxx" string product_version = "x.x" string uuid = "Lxxxxxxxxxxxxxxxxxxx" uint number_of_sensors_actuators = xx

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

```
List<LDSU_Sensor_Actuator> get_sensors_actuators()
```

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 uint id = xx string manufacturer = "xxxxxxxx" string part_number = "xxxx" uint i2c_7bit_address = xx uint report_rate_ms = xxxx

4.4 LDSU Devices

This class is responsible for initializing the LDSU and reading its information. This class works as an abstract class for 4 in 1 sensor, CO2 sensor, DO sensor, Salinity sensor, EC sensor, ORP sensor, pH sensor, thermocouple sensor, Relay Controller, Trailing Edge light dimmer and Isolated IO Controller.

```
abstract class LDSUDevice
```

1. LDSUDevice(LDSBus ldsbus, byte ldsu_id)
2. int status
3. virtual object read()

4.4.1 Constructor

4.4.1.1 API

```
LDSUDevice (LDSBus ldsbus, byte ldsu_id)
```

4.4.1.2 Parameters

ldsbus	Instance of LDSBus that holding the LDSU device
ldsu_id	LDSU index number.

4.4.1.3 Return

None	

4.4.2 LDSU Status

This property returns the object status after instantiated.

4.4.2.1 API

```
int status
```

4.4.2.2 Parameters

None	

4.4.2.3 Return

int	Returns the LDSU status after instantiated 0x00 - OK 0x01 - Initialize error 0x02 - Calibration error

4.4.3 Read from LDSU Device

This virtual function is a placeholder for class extends from this abstract class.

4.4.3.1 API

```
virtual object read()
```

4.4.3.2 Parameters

None	

4.4.3.3 Return

object	None

4.5 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. LDSBus4in1Sensor(LDSBus ldsbus, byte ldsu_id)
2. void init()
3. object read()

4.5.1 Constructor

The constructor for the LDSBus4in1Sensor 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

```
LDSBus4in1Sensor (LDSBus ldsbus, byte ldsu_id)
```

4.5.1.2 Parameters

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

4.5.1.3 Return

None	

4.5.2 Initialise Device

This function initializes the 4 in 1 sensor with

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

4.5.2.1 API

```
void init ()
```

4.5.2.2 Parameters

None	

4.5.2.3 Return

None	

4.5.3 Read Sensor

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

4.5.3.1 API

```
object read ()
```

4.5.3.2 Parameters

None	

4.5.3.3 Return

object	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)

4.6 LDSBus CO2 Sensor

LDSBusCO2Sensor is a subclass of LDSUDevice class that specifically designed to work with 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. LDSBusCO2Sensor (LDSBus ldsbus, byte ldsu_id)
2. void init()
3. object read()

4.6.1 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.

4.6.1.1 API

```
LDSBusCO2Sensor (LDSBus ldsbus, byte ldsu_id)
```

4.6.1.2 Parameters

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

4.6.1.3 Return

None	

4.6.2 Initialise Device

This function initializes the CO2 sensor with

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

4.6.2.1 API

```
void init ()
```

4.6.2.2 Parameters

None	

4.6.2.3 Return

None	

4.6.3 Read Sensor

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

4.6.3.1 API

```
object read ()
```

4.6.3.2 Parameters

None	

4.6.3.3 Return

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

4.7 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. LDSBusDOSensor (LDSBus ldsbus, byte ldsu_id)
2. void init()
3. object read()

4.7.1 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.

4.7.1.1 API

```
LDSBusDOSensor (LDSBus ldsbus, byte ldsu_id)
```

4.7.1.2 Parameters

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

4.7.1.3 Return

None	

4.7.2 Initialise Device

This function initializes the DO sensor with

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

4.7.2.1 API

```
void init ()
```

4.7.2.2 Parameters

None	

4.7.2.3 Return

None	

4.7.3 Read Sensor

This function reads the sensor and returns DO saturation in percentage and DO value in mg/L.

4.7.3.1 API

```
object read ()
```

4.7.3.2 Parameters

None	

4.7.3.3 Return

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

4.8 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. LDSBusSalinity(LDSBus ldsbus, byte ldsu_id)
2. void init()
3. object read()

4.8.1 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.

4.8.1.1 API

```
LDSBusSalinitySensor (LDSBus ldsbus, byte ldsu_id)
```

4.8.1.2 Parameters

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

4.8.1.3 Return

None	

4.8.2 Initialise Device

This function initializes the Salinity sensor with

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

4.8.2.1 API

```
void init ()
```

4.8.2.2 Parameters

None	

4.8.2.3 Return

None	

4.8.3 Read Sensor

This function reads the sensor and returns salinity in ppt.

4.8.3.1 API

```
object read ()
```

4.8.3.2 Parameters

None	

4.8.3.3 Return

object	Returns the readings of 1. Salinity in ppt

4.9 LDSBus EC Sensor

LDSBusECSensor is a subclass of LDSUDevice class that 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. LDSBusECSensor(LDSBus ldsbus, byte ldsu_id)
   2. void init()
   3. object read()
```

4.9.1 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.

4.9.1.1 API

```
LDSBusECSensor(LDSBus ldsbus, byte ldsu_id)
```

4.9.1.2 Parameters

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

4.9.1.3 Return

None	

4.9.2 Initialise Device

This function initializes the EC sensor with

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

4.9.2.1 API

```
void init ()
```

4.9.2.2 Parameters

None	

4.9.2.3 Return

None	

4.9.3 Read Sensor

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

4.9.3.1 API

```
object read ()
```

4.9.3.2 Parameters

None	

4.9.3.3 Return

object	Returns the readings of 1. EC in mS/cm.

4.10 LDSBus ORP Sensor

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

```
class LDSBusORPSensor : LDSUDevice

   1. LDSBusORPSensor(LDSBus ldsbus, int ldsu_id)
   2. void init()
   3. object read()
```

4.10.1 Constructor

The constructor for the LDSBusORPSensor 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

```
LDSBusORPSensor (LDSBus ldsbus, byte ldsu_id)
```

4.10.1.2 Parameters

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

4.10.1.3 Return

None	

4.10.2 Initialise Device

This function initializes the ORP sensor with

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

4.10.2.1 API

```
void init ()
```

4.10.2.2 Parameters

None	

4.10.2.3 Return

None	

4.10.3 Read Sensor

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

4.10.3.1 API

```
object read ()
```

4.10.3.2 Parameters

None	

4.10.3.3 Return

object	Returns the readings of 1. ORP value in mV.

4.11 LDSBus pH Sensor

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

```
class LDSBusPHSensor : LDSUDevice
```

1. LDSBusPHSensor(LDSBus ldsbus, byte ldsu_id)
2. void init()
3. object read()

4.11.1 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.

4.11.1.1 API

```
LDSBusPHSensor (LDSBus ldsbus, byte ldsu_id)
```

4.11.1.2 Parameters

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

4.11.1.3 Return

None	

4.11.2 Initialise Device

This function initializes the pH sensor with

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

4.11.2.1 API

```
void init ()
```

4.11.2.2 Parameters

None	

4.11.2.3 Return

None	

4.11.3 Read Sensor

This function reads the sensor and returns pH value.

4.11.3.1 API

```
object read ()
```

4.11.3.2 Parameters

None	

4.11.3.3 Return

object	Returns the readings of 1. pH value.

4.12 LDSBus Thermocouple Sensor

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

```
class LDSBusThermocoupleSensor : LDSUDevice
```

1. LDSBusThermocoupleSensor(LDSBus ldsbus, byte ldsu_id)
2. void init()
3. object read()

4.12.1 Constructor

The constructor for the LDSBusThermocoupleSensor 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

```
LDSBusThermocoupleSensor (LDSBus ldsbus, byte ldsu_id)
```

4.12.1.2 Parameters

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

4.12.1.3 Return

None	

4.12.2 Initialise Device

This function initializes the thermocouple sensor with

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

4.12.2.1 API

```
void init ()
```

4.12.2.2 Parameters

None	

4.12.2.3 Return

None	

4.12.3 Read Sensor

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

4.12.3.1 API

object read ()

4.12.3.2 Parameters

None	

4.12.3.3 Return

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

4.13 LDSBus 2CH Relay Controller

LDSBus2CHRelaySensor is a subclass of LDSUDevice class that specifically designed to work with 2CH Relay Controller. It provides the methods for setting of the relays and reading the states of relays.

```
class LDSBus2CHRelayController : LDSUDevice
```

1. LDSBus2CHRelayController(LDSBus ldsbus, byte ldsu_id)
2. void init()
3. void write(uint channel, uint state, uint mode=0, uint polar=1, uint time1=1, uint time2=1, uint cycles=1)
4. object read()

4.13.1 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.

4.13.1.1 API

LDSBus2CHRelayController (LDSBus ldsbus, byte ldsu_id)

4.13.1.2 Parameters

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

4.13.1.3 Return

None	

4.13.2 Initialise Device

This function initializes the controller with

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

4.13.2.1 API

```
void init ()
```

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

```
void write (uint channel, uint state, uint mode=0, uint polar=1, uint time1=1,
uint time2=1, uint cycles=1)
```

4.13.3.2 Parameters

channel	Relay channel 1 or 2
state	Set 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. 1 to 65535
time2	The second portion of pulse duration in seconds. 1 to 65535
cycles	Number of cycles to repeat pulse pattern. 1 to 65535

4.13.3.3 Return

None	

4.13.4 Read Controller

This function reads the states of relays.

4.13.4.1 API

```
object read()
```

4.13.4.2 Parameters

None	

4.13.4.3 Return

object	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)

4.14 LDSBus 2CH Relay iSense Controller

LDSBus2CHRelayISenseSensor is a subclass of LDSUDevice class that specifically designed to work with 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. LDSBus2CHRelayISenseController(LDSBus ldsbus, byte ldsu_id)
2. void init()
3. void write(uint channel, uint state, uint mode=0, uint polar=1, uint time1=1, uint time2=1, uint cycles=1)
4. object read()

4.14.1 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.

4.14.1.1 API

```
LDSBus2CHRelayISenseController (LDSBus ldsbus, byte ldsu_id)
```

4.14.1.2 Parameters

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

4.14.1.3 Return

None	

4.14.2 Initialise Device

This function initializes the controller with

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

4.14.2.1 API

```
void init ()
```

4.14.2.2 Parameters

None	

4.14.2.3 Return

None	

4.14.3 Write Controller

This function writes to controller to control relays.

4.14.3.1 API

```
void write (uint channel, uint state, uint mode=0, uint polar=1, uint time1=1,
uint time2=1, uint cycles=1)
```

4.14.3.2 Parameters

channel	Relay channel 1 or 2
state	Set 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 3. Polarity 0 for Negative: 4. When State is set COM to NO, the physical relay set COM to NC 5. 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.

4.14.3.3 Return

None	

4.14.4 Read Controller

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

4.14.4.1 API

```
object read ()
```

4.14.4.2 Parameters

None	

4.14.4.3 Return

object	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)

4.15 LDSBus Dimmer

LDSBusDimmer is a subclass of LDSUDevice class that specifically designed to work with Trailing Edge Dimmer. It provides the methods for setting of the brightness and read back the setting of the brightness.

```
class LDSBusDimmer : LDSUDevice
```

1. LDSBusDimmer (LDSBus ldsbus, byte ldsu_id)
2. void init()
3. void write(uint brightness = 0)
4. object read()

4.15.1 Constructor

The constructor for the LDSBusDimmer 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

```
LDSBusDimmer (LDSBus ldsbus, byte ldsu_id)
```

4.15.1.2 Parameters

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

4.15.1.3 Return

None	

4.15.2 Initialise Device

This function initializes the dimmer with

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

4.15.2.1 API

```
void init ()
```

4.15.2.2 Parameters

None	

4.15.2.3 Return

None	

4.15.3 Write Controller

This function set the brightness of the Trailing Edge Dimmer.

4.15.3.1 API

```
void write (uint brightness = 0)
```

4.15.3.2 Parameters

brightness	Set 0 to 100 for the brightness percentage

4.15.3.3 Return

None	

4.15.4 Read Sensor

This function reads the controller and returns the brightness setting.

4.15.4.1 API

```
object read ()
```

4.15.4.2 Parameters

None	

4.15.4.3 Return

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

4.16 LDSBus IO Controller

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

```
class LDSBusIOController : LDSUDevice

   1. LDSBusIOController (LDSBus ldsbus, byte ldsu_id)
   2. void init()
   3. void write(string channel, uint value)
   4. void write(string channel, float value)
   5. object read()
```

4.16.1 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.

4.16.1.1 API

```
LDSBusIOController (LDSBus ldsbus, byte ldsu_id)
```

4.16.1.2 Parameters

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

4.16.1.3 Return

None	

4.16.2 Initialise Device

This function initializes the 4 in 1 sensor with

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

4.16.2.1 API

```
void init ()
```

4.16.2.2 Parameters

None	

4.16.2.3 Return

None	

4.16.3 Write Controller Digital Output

This function writes value to digital output channel given.

4.16.3.1 API

```
void write (string channel, uint value)
```

4.16.3.2 Parameters

channel: str	"OUT1" and "OUT2" to set the Digital Output
value: uint	When channel is "OUT1" and "OUT2", value = 1 to set the Digital Output "ON" and value = 0 to set the Digital Output "OFF".

4.16.3.3 Return

None	

4.16.4 Write Controller Analog Output

This function writes value to analog output channel given.

4.16.4.1 API

```
void write (string channel, float value)
```

4.16.4.2 Parameters

channel: str	"VOUT1" and "VOUT2" to set the Analog Output
value: float	When channel is "VOUT1" and "VOUT2", value from 0.0 to 10.0 to set the Analog Output voltage.

4.16.4.3 Return

None	

4.16.5 Read Sensor

This function reads the controller and returns the digital input and analog input value.

4.16.5.1 API

```
object read ()
```

4.16.5.2 Parameters

None	

4.16.5.3 Return

object	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.

4.17 Logging APIs

```
static class LDSLogger
```

```
1. static void info(string msg, params object[] args)
2. static void warning(string msg, params object[] args)
3. static void error(string msg, params object[] args)
4. static void critical(string msg, params object[] args)
5. static void debug(string msg, params object[] args)
6. static void ldsu_info(LDSUInformation information)
7. static void sensor_actuator_list(List<LDSU_Sensor_Actuator> list_of_sensors)
8. static void ldsu_descriptor(LDSU_Descriptor descriptor)
```

5 Contact Information

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

Appendix A – References

Document References

LDSBus_Configuration_Utility_User_Guide

LDSU IR Blaster Application Note

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_004 LDSBus_DotNet_SDK_Guide
Document Reference No.: BRTSYS_000046
Clearance No.: BRTSYS#040
Product Page: https://brtsys.com/ldsbus/
Document Feedback: Send Feedback

Revision	Changes	Date
Version 1.0	Initial Release	18-05-2023
Version 1.1	Updated HVT references to Quad T-Junction; Updated Singapore Address	22-09-2023