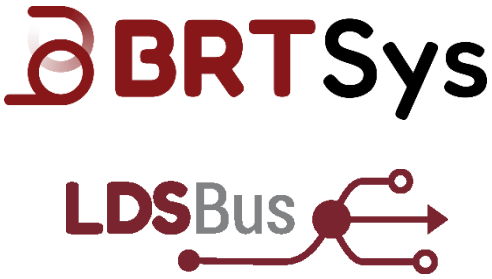

Application Note

BRTSYS_AN_002

LDSBus IR Blaster Application

Version 1.2

Issue Date: 21-09-2023

This application note explains about the LDSBus IR Blaster and LDSBus Python Library. The IR Blaster allows any remote button signal to transmit. The remote-control codes can be captured from the original remote using the Learning module. These can be sent to control the device using the IR

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

Third-Party Software & Hardware Disclaimer Notice

BRT Systems Pte Ltd (BRTSys) may recommend use of software, hardware, information, products, or websites that are owned or operated by other companies. We offer or facilitate this recommendation by hyperlinks or other methods to aid your access to the third-party resource. While BRTSys attempts to direct you to helpful, trustworthy resources, BRTSys cannot endorse, approve, or guarantee software, hardware, information, products, or services provided by or at a third-party resource or track changes in the resource. Thus, BRTSys is not responsible for the content or accuracy of any third-party resource or for any loss or damage of any sort resulting from the use of, or for any failure of, products or services provided at or from a third-party resource.

BRTSys recommends these resources on an "as is" basis. When you use a third-party resource, you will be subject to its terms and licenses and no longer be protected by our privacy policy or security practices, which may differ from the third-party policy or practices or other terms. You should familiarize yourself with any license or use terms of, and the privacy policy and security practices of, the third-party resource, which will govern your use of that resource.

1 Introduction

This application note describes the use of the LDSBus IR Blaster with the LDSBus Python SDK. 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 LDS bus and the platforms are referred to as LDSBus hosts. An LDSBus host communicates the IR sequence to the IR Blaster which then transmits the sequence.

The LDSBus IR Blaster is a "transmit only" device. IR sequences (from remote controllers or other IR transmitters) are captured using a separate 3rd party module, called the LearnIRv2 which is capable of both reception and transmission of IR sequences. Captured sequences may be stored in either the LearnIR (LIR) format or as raw pulse-pairs. This application note covers the capture of IR sequences using the LearnIRv2 module and the playback of captured sequences using the sample applications. The sample applications are built using the LDSBus Python SDK.

The following sections describe the LDSBus IR Blaster Setup, LearnIRv2 setup, IR sequence capture and playback.

2 Setup

2.1 LDSBus IR Blaster Setup

The IR Blaster may be setup in 2 ways; namely as a standalone LDS unit (LDSU mode) or as part of an LDS bus (LDSBus mode). These setups are shown in Figure 1 and Figure 2. In both setups, the common denominator is the LDSBus USB Adapter. LDSU mode requires only the IR Blaster and the LDSBus mode requires additional bus devices such as Quad T-Junction and a 24V power supply to power the bus.

Before the LDSBus IR Blaster is used in these examples, it is assumed that it has been configured using the LDSBus Configuration Utility. The LDSBus Configuration Utility is available for download from https://brtsys.com/resources. Using the LDSBus Configuration Utility, note the LDSU ID of the IR Blaster and the COM port number assigned to the LDSBus USB Adapter.

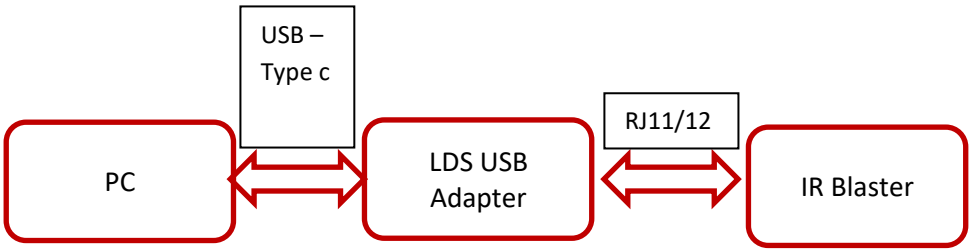

Figure 1 - LDSBus Mode Connection

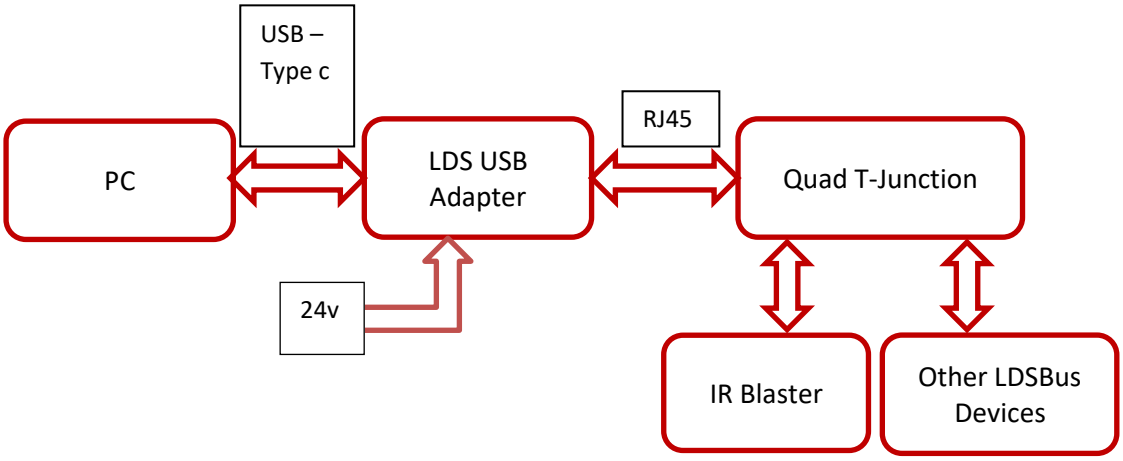

Figure 2 - LDSBus Mode Connection using Quad T-Junction

2.2 Remote Controller

Select an available remote-controlled appliance (e.g., Television, Air Conditioner or Fan) that is within line of sight of the IR Blaster setup. Though the IR Blaster has a maximum range of up to 7m, choosing a device within a 3m to 5m range ensures range issues do not affect the setup. The remote controller of the selected appliance shall be used in the following examples.

2.3 LearnIRv2 Setup

The LearnIRv2 is a module designed for capturing (learning) IR signals. It is a USB dongle in the shape of a thumb-drive and comes in 2 varieties (with and without an enclosure). It comes bundled with a companion LearnIRv2 Microsoft Windows application which works with official LearnIRv2 devices, only. More details may be obtained at the company's website.

Follow these steps for setup –

1. Attach the LearnIRv2 dongle to an available USB port on the PC.

Figure 3 – LearnIRv2 USB Dongle

2. Launch the Device Manager, locate the LearnIRv2 dongle and note the COM port number. This COM Port number will be used in the LearnIR application. Note that the COM port number may change whenever it is detached and attached again. In the below figure, the USB-SERIAL CH340 (COM23) represents the LearnIRv2 dongle.

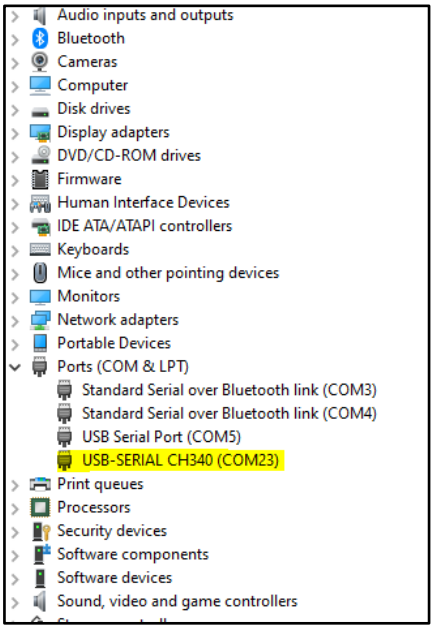

Figure 4 – Device Manager View

3. Launch the LearnIR application.

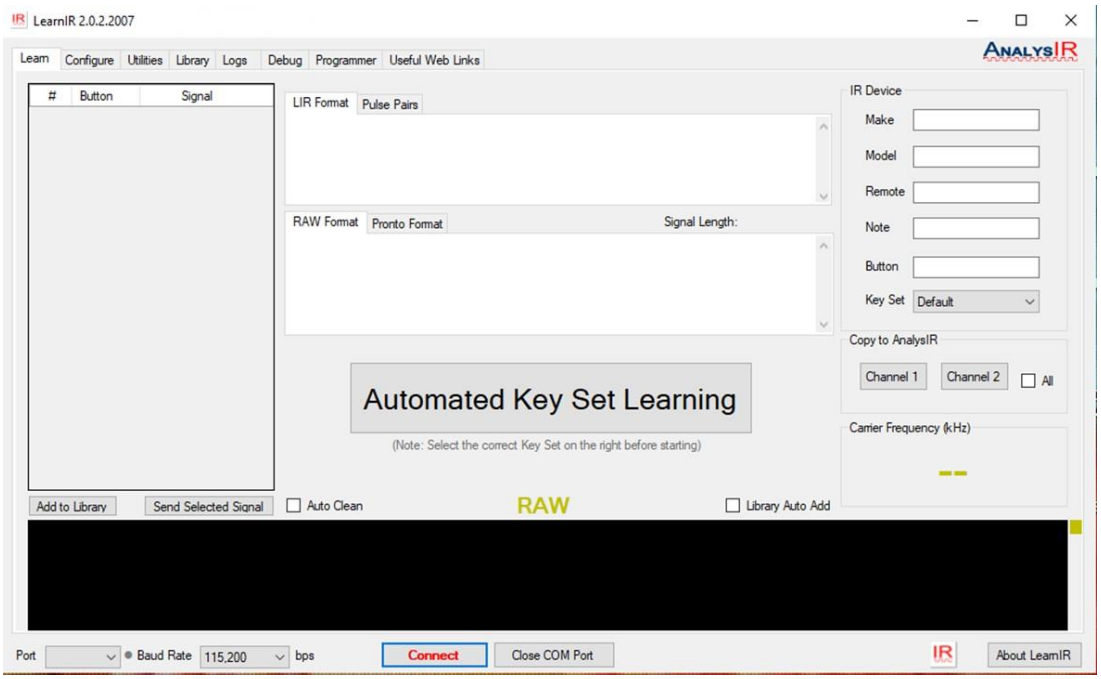

Figure 5 – LearnIR Application Window upon entry

4. Observe the bottom row of buttons. Some buttons are not shown on this initial screen and to make them appear, click on the **Configure** tab once and then click on the **Learn** tab. Additional buttons **"Open Device File"** and **"Save Device File"** will now be displayed. See Figure 6.

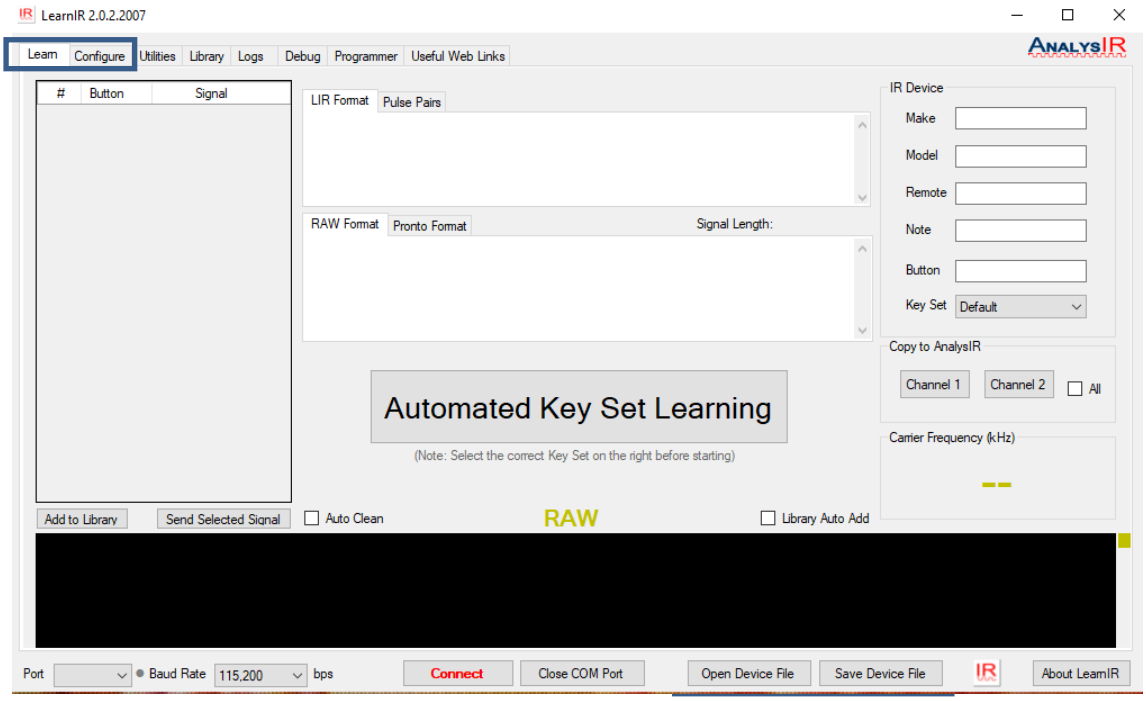

Figure 6 – Additional Function Buttons

5. Set the Port number – At the bottom row, from the **"Port"** drop-down box, select the *COM* Port number assigned to the LearnIRv2 dongle (from Device Manager). In the example screen (Figure 7), the selected COM Port is COM23.

6. Set the Baud Rate - From the **"Baud Rate"** drop-down box, select *115,200 bps* to set the baud rate.

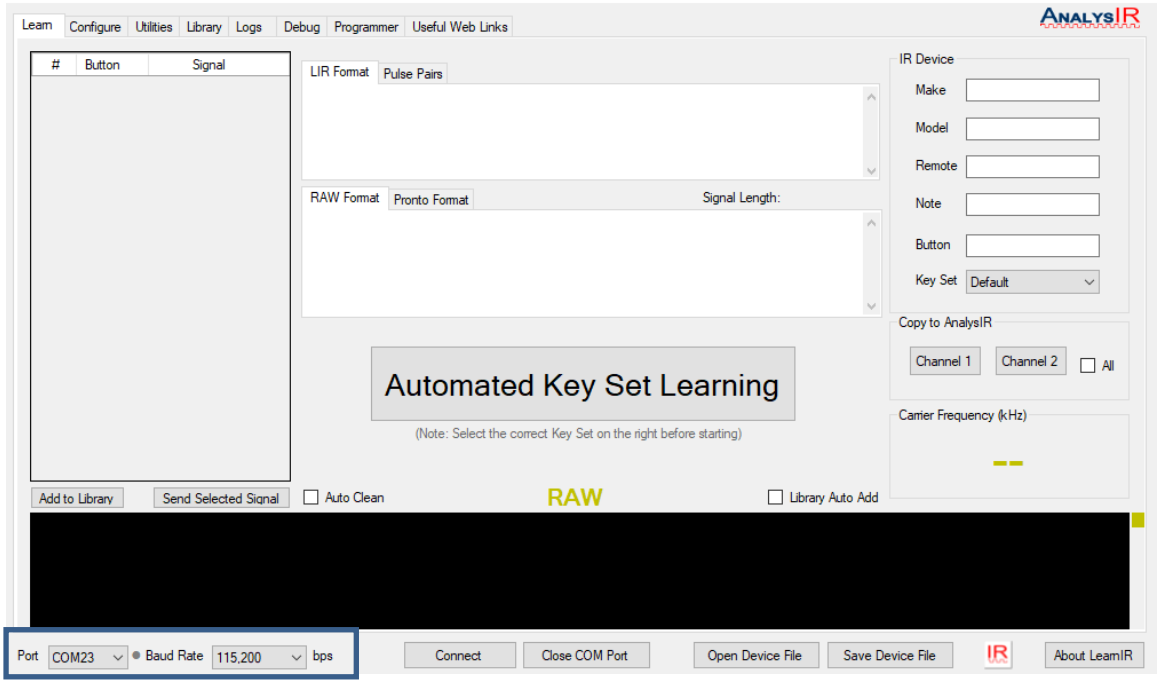

Figure 7 – LearnIR ready to Capture

LearnIR module is now ready to capture IR signals from the selected remote controller. In this application note, a Panasonic AC remote controller is used as the remote controller.

1. Point the remote controller away from the LearnIRv2 dongle and at the selected device. Turn it off by pressing the Power OFF button.

2. Now, point the remote controller at the LearnIR dongle. Press the Power ON button on the remote controller. The dongle will capture the emission from the remote controller.

3. The application will display the captured pulses.

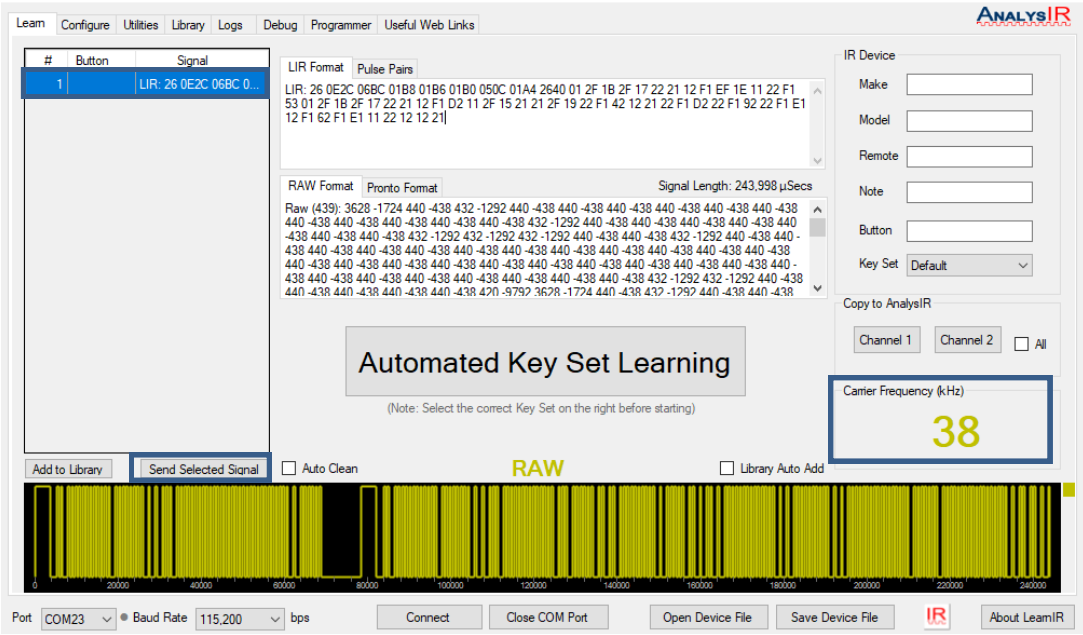

Figure 8 – Automatic Protocol Decode

4. Repeat step 1.

5. Select the LIR sequence (shown in Blue).

6. Click **[Send Selected Signal]** button to playback the captured signal.

7. The target device should be Powered On.

8. Select "**Save Device File"** button to save the sequence. The file will be used in the following examples and so, save it under "lir_input_file.txt". Do not modify the contents of this file.

Figure 9 – LearnIR – Saved Device File

9. Note the Carrier Frequency displayed on the bottom right side of the Figure 8.

2.4 LDSBus Python SDK Setup

The LDSBus Python SDK uses Python 3.8.8 or later. If Python3 has not been installed, then please do so before proceeding to the next step.

2.4.1 LDSBus Python SDK

The LDSBus Python SDK is a Python3 compatible library named pyliblds. It is usually installed together with the LDSBus Configuration Utility. If the library was not installed earlier, then visit https://brtsys.com/resources to download and install the SDK. The samples in this application note were developed using pyliblds v3.1.0 and the package may contain a more updated version of the SDK.

3 Sample Applications

The sample application implements a command line that takes input parameters which are described below. The first 3 parameters describe the LDSBus setup to the pyliblds.

3.1 Source Code Description

The sample application processes the LIR sequence for the IR Blaster. If LIR format is provided, it first converts the sequence from text format to binary format. Then it pads the sequence with header bytes and a trailer byte which are required by the IR Blaster.

3.1.1 LDSBus_IR_Blaster.py

LDSBus_IR_Blaster.py uses the LDSBus Python library, called pyliblds. Pyliblds takes care of all the LDSBus protocol communication, power, and overcurrent management via the "ldsbus.dll", which is a low-level library developed in C language. Figure 10 shows the LDSBus IR Blaster connected in the LDSU mode.

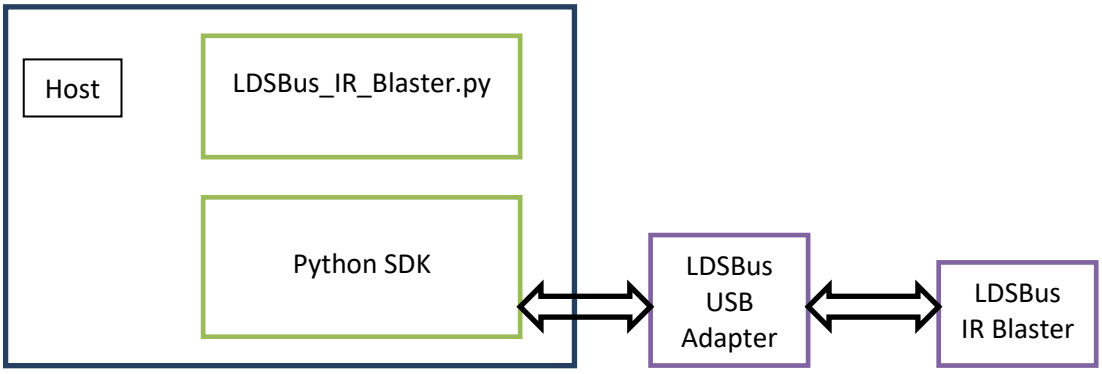

Figure 10 – IR Blaster Application Flow

3.1.2 Application Flow

The important variables in the sample application are:

a. *command*: This variable is the command for IR Blaster.
   i. Sample for "lir" format command:
   "26 0D90 06C8 01B6 01B4 01B4 0518 019C 270C 01 2F 2B 2F 17 22 21 12 F1 EF 1E 11 22 F1 53 01 2F 1B 2F 17 22 21 12 F1 E1 12 F1 52 12 12 F1 92 2F 14 21 22 12 2F 1D 22 2F 19 22 2F 1E 11 2F 16 2F 1E 11 21 21 21 22 1F"
b. *user_cfreq*: Carrier frequency; default = 38kHz and
c. *dutyCyclePerc*: Duty cycle of the carrier in percentage. If user specifies as zero, then firmware set to default 36 percent.

The sample application performs the following steps -

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)
4. Number of devices found.
5. Initialize LDSBusIRBlaster with device_id.
6. Get info of the device (device found and related info)
7. Waiting for user input LIR command.
8. Write command.
9. Inform device state after write command.

Sample code using "lir" command input:

```
import time
from liblds.ldsbus import LDSBus
from liblds.ldsdevice import LDSBusIRBlaster
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

        # print usb adapters
        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 = LDSBusIRBlaster(ldsbus, device_id)
        # Get info of the device
        log.info(f"UUID: {ldsu.descriptor['uuid']}")
        log.info(f"Last command: 0x{ldsu.info['last_command_status']}")
        log.ldsu_info(ldsu.info)
        log.ldsu_descriptor(ldsu.descriptor)
        log.sensor_actuator_list(ldsu.sa_list)

        while True:
            try:
                log.info("Enter LIR Command:")

                # Sample LIR Command
                # cmd = "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 2F 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"

                command = input()
                devstate = ldsu.write(command, user_cfreq=0x26, duty_cycle=0, format="lir")
                log.info(f"Device State: {devstate['Device State']}")
                if devstate["Error"]:
                    log.info(f"ERROR: {devstate['Error']}")
                if devstate["Busy"]:
                    log.info(f"BUSY: {devstate['Busy']}")
            except Exception as err:
                print(err)

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

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

```
Device 0
       Description: LDSBus USB Adapter-LA0201
       Serial: 0189VQCH
2023-05-25 16:49:34:392 INFO Successfully opened the port...
2023-05-25 16:49:35:484 INFO Scanning...
2023-05-25 16:49:48:535 INFO Number of devices found : 1, ID(s) is/are :  [15]
2023-05-25 16:49:49:481 INFO UUID: LC02010102032100083
2023-05-25 16:49:49:496 INFO Last command: 0x1
2023-05-25 16:49:49:530 INFO LDSU Info
2023-05-25 16:49:49:530 INFO =========
2023-05-25 16:49:49:530 INFO     RESET_SOURCE : NONE
2023-05-25 16:49:49:530 INFO     FIRMWARE_VERSION : 1.4
2023-05-25 16:49:49:530 INFO     LAST_COMMAND_STATUS : 1
2023-05-25 16:49:49:530 INFO     BOARD_VOLTAGE : 4961
2023-05-25 16:49:49:530 INFO     NUMBER_OF_RECEIVED_QUERIES_COMMANDS : 428
2023-05-25 16:49:49:544 INFO     NUMBER_OF_RESPONSES : 477
2023-05-25 16:49:49:544 INFO     NUMBER_OF_ERROR_PACKETS_RECEIVED : 0
2023-05-25 16:49:49:544 INFO LDSU Descriptor
2023-05-25 16:49:49:544 INFO ===============
2023-05-25 16:49:49:544 INFO     MANUFACTURER : BRT Systems Pte Ltd.
2023-05-25 16:49:49:544 INFO     PRODUCT_NAME : LDSBus IR Blaster
2023-05-25 16:49:49:544 INFO     UUID : LC02010102032100083
2023-05-25 16:49:49:544 INFO     PRODUCT_VERSION : 1.0
2023-05-25 16:49:49:544 INFO     NUMBER_OF_SENSORS_ACTUATORS : 2
2023-05-25 16:49:49:544 INFO Sensors and Actuators
2023-05-25 16:49:49:544 INFO ====================
2023-05-25 16:49:49:544 INFO     ID : 0
2023-05-25 16:49:49:544 INFO     MANUFACTURER : Bridgetek Pte. Ltd.
2023-05-25 16:49:49:544 INFO     PART_NUMBER : BRT-VDEV
2023-05-25 16:49:49:544 INFO     I2C_ADDRESS : 0
2023-05-25 16:49:49:544 INFO     REPORT_RATE_MS : 1000
2023-05-25 16:49:49:544 INFO     ID : 1
2023-05-25 16:49:49:544 INFO     MANUFACTURER : Bridgetek Pte. Ltd.
2023-05-25 16:49:49:544 INFO     PART_NUMBER : IR Transmitter
2023-05-25 16:49:49:544 INFO     I2C_ADDRESS : 88
2023-05-25 16:49:49:544 INFO     REPORT_RATE_MS : 1000
2023-05-25 16:49:49:544 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 2F 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
2023-05-25 16:50:02:546 INFO Device State: 0
2023-05-25 16:50:02:546 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 2F 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
2023-05-25 16:50:06:890 INFO Device State: 0
2023-05-25 16:50:06:894 INFO Enter LIR Command:
```

Figure 11 – Sample Output

3.1.3 Verifying LDSBus IR Blaster Transmission

If it is necessary to verify that the LDSBus IR Blaster is indeed transmitting correctly, the transmission from the IR Blaster may be captured using the LearnIR application and compared against the original capture from the remote controller.

1. Start and initialize LearnIR application and LearnIRv2 dongle.
2. Point the LDSBus IR Blaster at the LearnIRv2 dongle.
3. Initiate a transmission using LDSBus_IR_Blaster.py. Capture using LearnIR and save this capture to file let's say "recaptured_lir_file.txt". Copy the command sent to "lir_input.txt".
4. Update the global variable inside the IRBlaster helper function python file – "irBlasterAppHelperFunctions.py".

a. lir_file: This variable stores the name of file where captured sequence is saved using LearnIRv2. In this note, the sequence is taken from 'lir_input.txt' which are captured after remote transmission.
b. recaptured_lir_file: this variable stores the name of file where captured sequence is saved using LearnIRv2. In this note, for example, the sequence is taken from 'recaptured_lir_file.txt' which are captured after IRBlaster transmission.
c. Run "irBlasterAppHelperFunctions.py" to compare these two files:

C:/../verification> python irBlasterAppHelperFunctions.py

The result of the comparison is then displayed.

```
C:\MyFolder\Gitlab\BS_SW_000066_LDSBUS_SDKPython\src\python_sample\verification>python irBlasterAppHelperFunctions.py
lir_input.txt
['26', '0D90', '06C8', '01B6', '01B4', '01B4', '0518', '019C', '270C', '01', '2F', '2B', '2F', '17', '22', '21', '12', '
F1', 'EF', '1E', '11', '22', 'F1', '53', '01', '2F', '1B', '2F', '17', '22', '21', '12', 'F1', 'E1', '12', 'F1', '52', '
12', '12', 'F1', '92', '2F', '14', '21', '22', '12', '2F', '1D', '22', '2F', '19', '22', '2F', '1E', '11', '2F', '16', '
2F', '1E', '11', '21', '21', '21', '22', '1F']


recaptured_lir_file.txt
['26', '0D90', '06C8', '01B6', '01B4', '01B4', '0518', '019C', '270C', '01', '2F', '2B', '2F', '17', '22', '21', '12', '
F1', 'EF', '1E', '11', '22', 'F1', '53', '01', '2F', '1B', '2F', '17', '22', '21', '12', 'F1', 'E1', '12', 'F1', '52', '
12', '12', 'F1', '92', '2F', '14', '21', '22', '12', '2F', '1D', '22', '2F', '19', '22', '2F', '1E', '11', '2F', '16', '
2F', '1E', '11', '21', '21', '21', '22', '1F']


Compare result :
Success - LearnIR sequence transmitted from Remote buttton and IRBlaster are matched.
```

Figure 12 – Result Verification

3.1.4 Further Ideas

Further ideas to enhance the sample app are:

1. Create classes extending this LDSBusIRBlaster with specific commands like "AC_ON" with specified 'lir' command.
2. Read the captured samples from LearnIR directly and transmit via LDSBus IR Blaster.

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

class Aircon(LDSBusIRBlaster):

    def ac_on(self):
        cmd = "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 2F 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"
        self.write(cmd, format="lir")

    def ac_off(self):
        cmd = "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 2F 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"
        self.write(cmd, format="lir")
```

4 Conclusion

In this app note, we have shown how to use the LearnIRv2 dongle and LearnIR application to capture IR sequences. We have also introduced the LDSBus Python SDK and how it may be used to control and communicate with the LDSBus IR Blaster.

5 Contact Information

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

Appendix A – References

Document References

BRTSYS_AN_001_LDSBus_Configuration_Utility_User_Guide

BRTSYS_API_001_LDSBus_Python_SDK_Guide

Acronyms and Abbreviations

Terms	Description
AC	Air Conditioner
COM	Communication Port
IR	Infrared
LDSBus	Long Distance Sensor Bus
LDSU	Long Distance Sensor Unit
LIR	Learn IR
PC	Personal Computer
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_AN_002 LDSBus IR Blaster Application
Document Reference No.: BRTSYS_000015
Clearance No.: BRTSYS#018
Product Page: https://brtsys.com/ldsbus/product/irblaster/
Document Feedback: Send Feedback

Revision	Changes	Date
Version 1.0	Initial Release	26-11-2021
Version 1.1	Updated release under BRT Systems	15-09-2022
Version 1.2	Updated as per Python SDK v3.1.0; Updated the Singapore address	21-09-2023