LR-USB USB 通信
サンプルプログラム
（Linux Java）

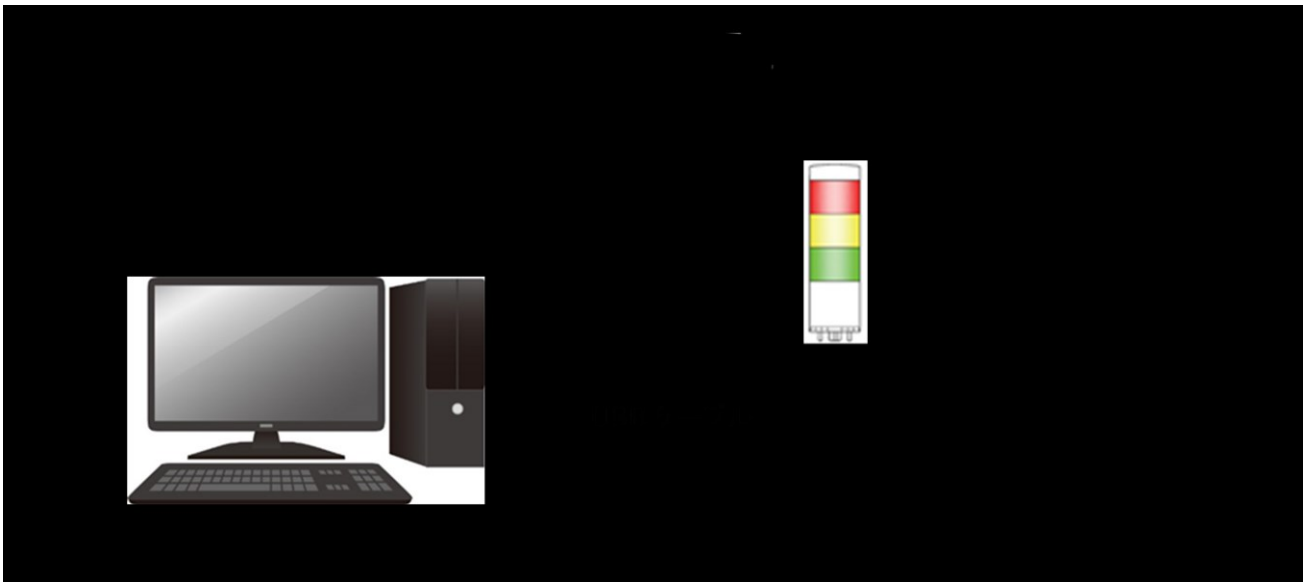
内容

1. 概要

LR-USB を USB 通信で制御するための、サンプルプログラムの概要を記載する。
本プログラムは、パトライトが提供する DLL を使用せずにでの制御をおこなうことを目的としている。

1.1. システム概要

本プログラムのシステム構成図は以下の通り。
本プログラムでは、1 台の LR-USB の機器を USB 通信で制御を行う。



2. 開発環境

サンプルプログラムの開発環境を以下に示す。

2.1. Linux 環境

開発環境		備考
開発 OS	Ubuntu	18.04
開発言語	Java	11.0 以降
パッケージ	usb4java	1.3.0 以降
ライブラリ	libusb,	1.0.21-2

2.1.1. 環境構築

・libusb のインストール

Ubunut(18.04)では標準パッケージとしてインストールされている。何らかの影響でインストールされていない場合は、apt-get コマンドにてインストールをする。

```
$ sudo apt-get update
$ sudo apt-get install libusb-1.0.0
```

・Java のインストール

apt-get コマンドにてインストールをする。

```
$ sudo apt-get update
$ sudo apt-get install default-jdk
```

・usb4java のインストール

パッケージをダウンロードする。

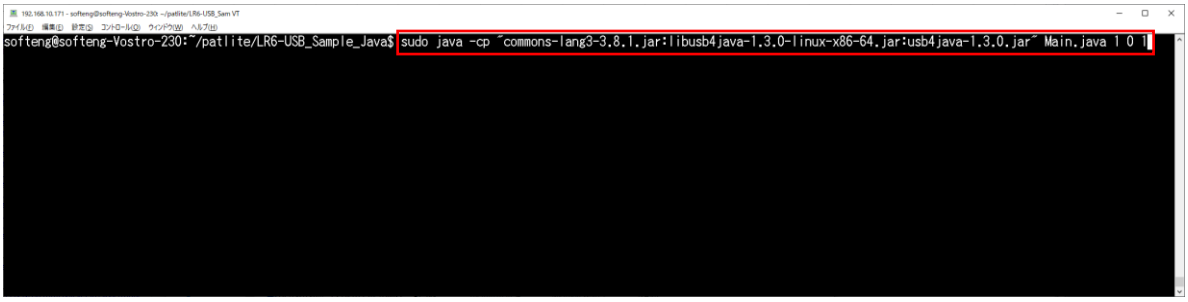
https://github.com/usb4java/usb4java/releases

圧縮ファイル内の commons-lang3-3.8.1.jar、libusb4java-1.3.0-linux-x86-64.jar、usb4java-1.3.0.jar の 3 つを Main.java と同じフォルダに配置する。

3. サンプルソース概要

3.1. コマンド操作説明

コマンドプロンプト上では、コマンドライン引数を指定することで各動作のコマンドを実行される。



3.1.1. コマンド一覧

コマンド名	内容
LED ユニットを制御	LED 色と LED パターンを指定して点灯、パターン点灯させる
複数の LED ユニットを制御	複数の LED 色と LED パターンを指定してパターン点灯させる
ブザーパターンでブザーを制御	ブザーパターンを指定してブザー吹鳴させる
ブザーパターンと音階でブザーを制御	ブザーの音階とパターンを指定してブザー吹鳴させる
リセット	LED ユニットをすべて消灯し，ブザーを停止させる

3.1.2. LED ユニットを制御

以下のコマンドライン引数を指定して、コマンドを実行する

No.	コマンドライン引数	値
1	コマンド ID	1
2	LED ユニット色	赤:0 黄:1 緑:2 青:3 白:4
3	LED パターン	消灯:0 点灯:1 LED パターン 1:2 LED パターン 2:3 LED パターン 3:4 LED パターン 4:5 現状の設定を維持:15

例:sudo java -cp "commons-lang3-3.8.1.jar:libusb4java-1.3.0-linux-x86-64.jar:usb4java-1.3.0.jar" Main.java 1 0 1

3.1.3. 複数の LED ユニットを制御

以下のコマンドライン引数を指定して、コマンドを実行する

No.	コマンドライン引数	値
1	コマンド ID	2
2	赤の LED のパターン	消灯:0 点灯:1 LED パターン 1:2 LED パターン 2:3 LED パターン 3:4 LED パターン 4:5 現状の設定を維持:15
3	黄の LED のパターン	
4	緑の LED のパターン	
5	青の LED のパターン	
6	白の LED のパターン	

例:sudo java -cp "commons-lang3-3.8.1.jar:libusb4java-1.3.0-linux-x86-64.jar:usb4java-1.3.0.jar" Main.java 2 1 2 3 4 5

3.1.4. ブザーパターンでブザーを制御

以下のコマンドライン引数を指定して、コマンドを実行する

No.	コマンドライン引数	値
1	コマンド ID	3
2	ブザーパターン	停止:0 吹鳴(連続) :1 ブザーパターン 1:2 ブザーパターン 2:3 ブザーパターン 3:4 ブザーパターン 4:5 現状の設定を維持:15
3	ブザーの連続動作・回数動作	連続動作:0 回数動作:1～15

例:sudo java -cp "commons-lang3-3.8.1.jar:libusb4java-1.3.0-linux-x86-64.jar:usb4java-1.3.0.jar" Main.java 3 1 15

3.1.5. ブザーパターンと音階でブザーを制御

以下のコマンドライン引数を指定して、コマンドを実行する

No.	コマンドライン引数	値
1	コマンド ID	4
2	ブザーパターン	停止:0 吹鳴(連続) :1 ブザーパターン 1:2 ブザーパターン 2:3 ブザーパターン 3:4 ブザーパターン 4:5 現状の設定を維持:15

3	ブザーの連続動作・回数動作	連続動作：0 回数動作：1～15
4	音 A のブザー音階	停止：0 A6：1 B♭6：2 B6：3 C7：4 D♭7：5 D7：6 E♭7：7 E7：8 F7：9 G♭7：10 G7：11 A♭7：12 A7：13 音 A のデフォルト値:D7：14 音 B のデフォルト値:(停止）：15
5	音 B のブザー音階	

例：sudo java -cp "commons-lang3-3.8.1.jar:libusb4java-1.3.0-linux-x86-64.jar:usb4java-1.3.0.jar" Main.java 4 1 15 1 13

3.1.6. リセット

以下のコマンドライン引数を指定して、コマンドを実行する

No.	コマンドライン引数	値
1	コマンド ID	5

例：sudo java -cp "commons-lang3-3.8.1.jar:libusb4java-1.3.0-linux-x86-64.jar:usb4java-1.3.0.jar" Main.java 5

3.2. 関数定義

3.2.1. 関数一覧

関数名	説明
usb_open	USB 制御積層信号灯へ USB 通信で接続する
usb_close	USB 制御積層信号灯との USB 通信を終了する。
send_command	コマンドを送信する。
set_light	LED 色と LED パターンを指定して USB 制御積層信号灯を点灯、パターン点灯させる。
set_tower	複数の LED 色と LED パターンを指定して USB 制御積層信号灯をパターン点灯させる。
set_buz	ブザーのパターンを指定して USB 制御積層信号灯をブザー吹鳴させる。
set_buz_ex	ブザーの音階とパターンを指定して USB 制御積層信号灯をブザー吹鳴させる。
reset	LED ユニットをすべて消灯し，ブザーを停止させる。

3.2.2. usb_open 関数

関数名	int usb_open()	
パラメータ	なし	
戻り値	int	実行結果。失敗なら 0 未満が返る。
説明	ベンダーID が「0x191A」とデバイスが「0x8003」のデバイスをオープンする	

関数の使用方法	# メイン関数
	void main(){
	　　# LR-USB へ USB 通信で接続する
	　　ret = usb_open();
	}	

備考	関数のプログラムの概要は「4.1LR-USB に接続」を参照	

3.2.3. usb_close 関数

関数名	usb_close()
パラメータ	なし
戻り値	なし
説明	LR-USB との USB 通信を終了する

関数の使用方法	void main(){
	　　# LR-USB との USB 通信を終了する
	　　usb_close();
	}

備考	関数のプログラムの概要は「4.2LR-USB との切断」を参照

3.2.4. send_command 関数

関数名	int send_command(final byte[] sendData)	
パラメータ	sendData	送信データ
戻り値	int	成功：送信したバイト数、失敗：0 以下
説明	接続した LR-USB にデータを送信する	

関数の使用方法	# メイン関数
	void main(){
	　　# LR-USB へ USB 通信で接続する
	　　ret = usb_open();
	
	　　# 送信データを作成
	　　byte[] data = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00. 0x00, 0x00};
	
	　　# 送信
	　　ret = sed_command(data);
	　　if (ret <= 0){
	　　　　System.out.println("failed to send data");
	　　}
	}	

備考	関数のプログラムの概要は「4.3 コマンドを送信」を参照	

3.2.5. set_light 関数

関数名	int set_light(final byte color, final byte state)	
パラメータ	color	制御する LED 色(赤:0、黄:1、緑:2、青:3、白:4)
	state	LED パターン(消灯:0x00、点灯:0x01、LED パターン 1:0x02、LED パターン 2:0x03、LED パターン 3:0x04、LED パターン 4:0x05、現状の設定を維持:0x06～0x0F)
戻り値	int	成功:送信したバイト数、失敗:0 以下
説明	LED 色と LED パターンを指定して USB 制御積層信号灯を点灯、パターン点灯させるブザーおよび、指定された LED 色以外の LED ユニットは現在の状態を維持する	

関数の使用方法	# メイン関数
	void main(){
	　　# LR-USB へ USB 通信で接続する
	　　ret = usb_open();
	
	　　# 赤色のユニットを点灯
	　　ret = set_light(LED_COLOR_RED, 0x01);
	}	

備考	関数のプログラムの概要は「4.4LED 色と LED パターンを指定したコマンドの送信」を参照	

3.2.6. set_tower 関数

関数名	int set_tower(final byte red, final byte yellow, final byte green,final byte blue, final byte white)	
パラメータ	red	赤の LED パターン(消灯:0x00、点灯:0x01、LED パターン 1:0x02、LED パターン 2:0x03、LED パターン 3:0x04、LED パターン 4:0x05、現状の設定を維持:0x06～0x0F)
	yellow	黄の LED パターン(消灯:0x00、点灯:0x01、LED パターン 1:0x02、LED パターン 2:0x03、LED パターン 3:0x04、LED パターン 4:0x05、現状の設定を維持:0x06～0x0F)
	green	緑の LED パターン(消灯:0x00、点灯:0x01、LED パターン 1:0x02、LED パターン 2:0x03、LED パターン 3:0x04、LED パターン 4:0x05、現状の設定を維持:0x06～0x0F)
	blue	青の LED パターン(消灯:0x00、点灯:0x01、LED パターン 1:0x02、LED パターン 2:0x03、LED パターン 3:0x04、LED パターン 4:0x05、現状の設定を維持:0x06～0x0F)
	white	白の LED パターン(消灯:0x00、点灯:0x01、LED パターン 1:0x02、LED パターン 2:0x03、LED パターン 3:0x04、LED パターン 4:0x05、現状の設定を維持:0x06～0x0F)
戻り値	int	成功:送信したバイト数、失敗:0 以下
説明	複数の LED 色と LED パターンを指定して USB 制御積層信号灯をパターン点灯させる	

関数の使用方法	# メイン関数
	void main(){
	　# LR-USB へ USB 通信で接続する
	　ret = usb_open();
	
	　# 赤色のユニット:パターン 1
	　# 黄色のユニット:パターン 3
	　# 緑色のユニット:パターン 4
	　# 青色のユニット:点灯
	　# 白色のユニット:消灯
	　ret = set_light(0x02, 0x04, 0x05, 0x01, 0x00);
	}	

備考	関数のプログラムの概要は「4.5 複数の LED 色と LED パターンを指定したコマンドの送信」を参照	

3.2.7. set_buz 関数

関数名	set_buz(final byte buz_state, final byte limit)	
パラメータ	buz_state	ブザーパターン(停止:0x00、吹鳴(連続):0x01、ブザーパターン 1:0x02、ブザーパターン 2:0x03、ブザーパターン 3:0x04、ブザーパターン 4:0x05、現状の設定を維持:0x0F)
	limit	連続動作:0、回数動作:1～15
戻り値	int	成功:送信したバイト数、失敗:0 以下
説明	ブザーのパターンを指定して USB 制御積層信号灯をブザー吹鳴させる LED ユニットは現在の状態を維持する。音階はデフォルト値で動作する	

関数の使用方法	# メイン関数
	void main(){
	# LR-USB へ USB 通信で接続する
	ret = usb_open();
	
	# ブザーをパターン 1 で、2 秒吹鳴
	ret = set_buz(0x02, 2);
	}	

備考	関数のプログラムの概要は「4.6 ブザーのパターン指定したコマンドの送信」を参照	

3.2.8. set_buz_ex 関数

関数名	int set_buz_ex(final byte buz_state, final byte limit, final byte pitch1, final byte pitch2)	
パラメータ	buz_state	ブザーパターン(停止:0x00、吹鳴(連続):0x01、ブザーパターン 1:0x02、ブザーパターン 2:0x03、ブザーパターン 3:0x04、ブザーパターン 4:0x05、現状の設定を維持:0x0F)
	limit	連続動作:0、回数動作:1～15
	pitch1	音 A のブザー音階(停止:0x00、A6:0x01、B♭6:0x02、B6:0x03、C7:0x04、D♭7:0x05、D7:0x06、E♭7:0x07、E7:0x08、F7:0x09、G♭7:0x0A、G7:0x0B、A♭7:0x0C、A7:0x0D、音 A のデフォルト値:D7:0x0E、音 B のデフォルト値:(停止):0x0F)
	pitch2	音 B のブザー音階(停止:0x00、A6:0x01、B♭6:0x02、B6:0x03、C7:0x04、D♭7:0x05、D7:0x06、E♭7:0x07、E7:0x08、F7:0x09、G♭7:0x0A、G7:0x0B、A♭7:0x0C、A7:0x0D、音 A のデフォルト値:D7:0x0E、音 B のデフォルト値:(停止):0x0F)
戻り値	int	成功:送信したバイト数、失敗:0 以下
説明	ブザーの音階とパターンを指定して USB 制御積層信号灯をブザー吹鳴させる	

関数の使用方法	# メイン関数
	void main(){
	　# LR-USB へ USB 通信で接続する
	　ret = usb_open();
	
	　# ブザーをパターン 1 で、2 秒吹鳴
	　# 音 A:B♭6
	　# 音 B:E7
	　ret = set_buz_ex(0x02, 2, 0x02, 0x08);
	}	

備考	関数のプログラムの概要は「4.7 ブザーのパターンと音階を指定したコマンドの送信」を参照	

3.2.9. reset 関数

関数名	reset()	
パラメータ	なし	
戻り値	int	成功：送信したバイト数、失敗：0 以下
説明	LED ユニットをすべて消灯し，ブザーを停止させる	

関数の使用方法	# メイン関数
	void main(){
	　　# LR-USB へ USB 通信で接続する
	　　ret = usb_open();
	
	　　# LR-USB の状態をリセット
	　　ret = reset();
	}	

備考	関数のプログラムの概要は「4.8 リセットコマンドの送信」を参照	

3.3. 定数定義

3.3.1. ベンダーID

定数名	値	説明
VENDOR_ID	0x191A	LR-USB のベンダーID

3.3.2. デバイスID

定数名	値	説明
DEVICE_ID	0x8003	LR-USB のデバイス

3.3.3. コマンドバージョン

定数名	値	説明
COMMAND_VERSION	0x00	LR-USB へコマンドを送信する時のコマンドバージョン

3.3.4. コマンド ID

定数名	値	説明
COMMAND_ID	0x00	LR-USB へコマンドを送信する時のコマンドID

3.3.5. ホスト→USB 制御積層信号灯に送信するためのエンドポイントアドレス

定数名	値	説明
ENDPOINT_ADDRESS	0x01	PC から LR-USB へ送信するためのエンドポイント

3.3.6. コマンド送信時のタイムアウト時間

定数名	値	説明
SEND_TIMEOUT	1000	コマンドを送信する時のタイムアウト時間 単位はミリ秒

3.3.7. プロトコルデータ領域サイズ

定数名	値	説明
SEND_BUFFER_SIZE	8	送信するデータのバッファサイズ

3.3.8. LED ユニット色

定数名	値	説明
LED_COLOR_RED	0	赤
LED_COLOR_YELLOW	1	黄
LED_COLOR_GREEN	2	緑

LED_COLOR_BLUE	3	青
LED_COLOR_WHITE	4	白

3.3.9. LED パターン

定数名	値	説明
LED_OFF	0x00	消灯
LED_KEEP	0x0F	現状の設定を維持

3.3.10. ブザーパターン

定数名	値	説明
BUZZER_OFF	0x00	停止
BUZZER_KEEP	0x0F	現状の設定を維持

3.3.11. ブザー音階

定数名	値	説明
BUZZER_PITCH_OFF	0x00	停止
BUZZER_PITCH_DFLT_A	0x0E	音 A のデフォルト値:D7
BUZZER_PITCH_DFLT_B	0x0F	音 B のデフォルト値:(停止)

4. プログラム概要

起動後のプログラムの動作を要点のみ記載する。

4.1. LR-USB に接続

プログラム	説明

Main.java usb_open()
 	
`int ret = 0;`
`// Initialization process`
`context = new Context();`
`int initret = LibUsb.init(context);`
`if(initret != LibUsb.SUCCESS) {`
`  return -1;`
`}`
 	
`// Device open`
`devHandle = LibUsb.openDeviceWithVidPid(context, (short)VENDOR_ID, (short)DEVICE_ID);`
`if(devHandle == null) {`
`  return -1;`
`}`
 	
`// Interface acquisition`
`LibUsb.detachKernelDriver(devHandle, 0);`
`int IntRet = LibUsb.claimInterface(devHandle, 0);`
`if(IntRet != 0) {`
`  return -2;`
`}`
 	
`return ret;`
	→libusb ライブラリの初期化
	
	→ベンダーID とデバイス ID で LR-USB デバイスをオープンしてハンドルを取得する
	
	→カーネルドライバのデタッチを行い、LR-USB デバイスのインターフェースを取得する

4.2. LR-USB との切断

プログラム	説明

Main.java usb_close()
// End processing
LibUsb.close(devHandle);
LibUsb.exit(context);
	→デバイスをクローズする

4.3. コマンドを送信

各コマンドの送信データフォーマットの送信データを LR-USB にコマンドデータを送信する。

各コマンドの送信データフォーマットの作成は「4.4LED 色と LED パターンを指定したコマンドの送信」以降を参照

プログラム	説明

Main.java send_command()
// Convert data
ByteBuffer setData = ByteBuffer.allocateDirect(SEND_BUFFER_SIZE);
setData.put(sendData);
// Check the handle
if(devHandle == null) {
return 0;
}
// data transfer
IntBuffer sendLength = IntBuffer.allocate(SEND_BUFFER_SIZE);
int TranRet = LibUsb.interruptTransfer(devHandle, ENDPOINT_ADDRESS, setData, sendLength, (long)SEND_TIMEOUT);
if(TranRet == 0) {
ret = sendLength.get();
}else{
ret = -1;
}
return ret;
	→LR-USB に送信データを送信する。
	LR-USB のデバイスハンドル
	エンドポイントアドレス(1)
	送信データ
	送信したデータのサイズ
	タイムアウトの秒数(ミリ秒)

4.4. LED 色と LED パターンを指定したコマンドの送信

プログラム	説明

Min.java set_light()
 	
`byte[] sendData = new byte[Control.SEND_BUFFER_SIZE];`
 	
`// Command version`
`sendData[0] = Control.COMMAND_VERSION;`
 	
`// Command ID`
`sendData[1] = Control.COMMAND_ID;`
 	
`// Buzzer control`
`sendData[2] = Control.BUZZER_KEEP;`
 	
`// Buzzer scale`
`sendData[3] = Control.BUZZER_PITCH_OFF;`
 	
`// LED (red / yellow)`
`sendData[4] = (byte) ((Control.LED_KEEP << 4) | Control.LED_KEEP);`
`if(color == Control.LED_COLOR_RED) {`
`  sendData[4] = (byte) ((state << 4) | Control.LED_KEEP);`
`}`
`if(color == Control.LED_COLOR_YELLOW) {`
`  sendData[4] = (byte) ((Control.LED_KEEP << 4) | state);`
`}`
 	
`// LED (green / blue)`
`sendData[5] = (byte) ((Control.LED_KEEP << 4) | Control.LED_KEEP);`
`if(color == Control.LED_COLOR_GREEN) {`
`  sendData[5] = (byte) ((state << 4) | Control.LED_KEEP);`
`}`
`if(color == Control.LED_COLOR_BLUE) {`
`  sendData[5] = (byte) ((Control.LED_KEEP << 4) | state);`
`}`
 	
`// LED (white)`
`sendData[6] = (byte) ((Control.LED_KEEP << 4) | Control.LED_OFF);`
`if(color == Control.LED_COLOR_WHITE) {`
`  sendData[6] = (byte) ((state << 4) | Control.LED_OFF);`
`}`
 	
`// openings`
`sendData[7] = Control.BLANK;`
 	
`// Send command`
`int ret = this.send_command(sendData);`
`if (ret <= 0) {`
`  System.err.println("failed to send data");`
`}`
 	
`return ret;`
	→引数の赤と黄に対応する 4bit に引数の LED パターンを設定する。(※)
	
	→引数の緑と青に対応する 4bit に引数の LED パターンを設定する。(※)
	
	→引数の白に対応する 4bit に引数の LED パターンを設定する。(※)
	
	→「4.3 コマンドを送信」を呼び出し、機器にデータを送信

※指定されなかった色は LED_KEEP で設定を維持する。

4.5. 複数の LED 色と LED パターンを指定したコマンドの送信

プログラム	説明

Main.java set_tower()
`byte[] sendData = new byte[Control.SEND_BUFFER_SIZE];`
 	
`// Command version`
`sendData[0] = Control.COMMAND_VERSION;`
 	
`// Command ID`
`sendData[1] = Control.COMMAND_ID;`
 	
`// Buzzer control`
`sendData[2] = Control.BUZZER_KEEP;`
 	
`// Buzzer scale`
`sendData[3] = Control.BUZZER_PITCH_OFF;`
 	
`// LED (red / yellow)`
`sendData[4] = (byte) ((red << 4) | yellow);`
 	
`// LED (green / blue)`
`sendData[5] = (byte) ((green << 4) | blue);`
 	
`// LED (white)`
`sendData[6] = (byte) ((white << 4) | Control.LED_OFF);`
 	
`// openings`
`sendData[7] = Control.BLANK;`
 	
`// Send command`
`int ret = this.send_command(sendData);`
`if (ret <= 0) {`
`System.err.println("failed to send data");`
`}`
 	
`return ret;`
	→引数の赤と黄の LED パターンからデータを作成
	
	→引数の赤と黄の LED パターンからデータを作成
	→引数の白の LED パターンからデータを作成
	
	→「4.3 コマンドを送信」を呼び出し、機器にデータを送信

4.6. ブザーのパターン指定したコマンドの送信

プログラム	説明

Main.java set_buz()
`byte[] sendData = new byte[Control.SEND_BUFFER_SIZE];`
 	
`// Command version`
`sendData[0] = Control.COMMAND_VERSION;`
 	
`// Command ID`
`sendData[1] = Control.COMMAND_ID;`
 	
`// Buzzer control`
`sendData[2] = (byte) ((limit << 4) | buz_state);`
 	
`// Buzzer scale`
`sendData[3] = (byte) ((Control.BUZZER_PITCH_DFLT_A << 4) | Control.BUZZER_PITCH_DFLT_B);`
 	
`// LED (red / yellow)`
`sendData[4] = (byte) ((Control.LED_KEEP << 4) | Control.LED_KEEP);`
 	
`// LED (green / blue)`
`sendData[5] = (byte) ((Control.LED_KEEP << 4) | Control.LED_KEEP);`
 	
`// LED (white)`
`sendData[6] = (byte) ((Control.LED_KEEP << 4) | Control.LED_OFF);`
 	
`// openings`
`sendData[7] = Control.BLANK;`
 	
`// Send command`
`int ret = this.send_command(sendData);`
`if (ret <= 0) {`
`  System.err.println("failed to send data");`
`}`
 	
`return ret;`
	→引数のブザーパターンと回数動作から制御データを作成する。
	→ブザー音階はデフォルト値で作成する
	
	→LED 制御は設定維持で作成する
	
	→「4.3 コマンドを送信」を呼び出し、機器にデータを送信

4.7. ブザーのパターンと音階を指定したコマンドの送信

プログラム	説明

Main.java set_buz_ex()
`byte[] sendData = new byte[Control.SEND_BUFFER_SIZE];`
 	
`// Command version`
`sendData[0] = Control.COMMAND_VERSION;`
 	
`// Command ID`
`sendData[1] = Control.COMMAND_ID;`
 	
`// Buzzer control`
`sendData[2] = (byte) ((limit << 4) | buz_state);`
 	
`// Buzzer scale`
`sendData[3] = (byte) ((pitch1 << 4) | pitch2);`
 	
`// LED (red / yellow)`
`sendData[4] = (byte) ((Control.LED_KEEP << 4) | Control.LED_KEEP);`
 	
`// LED (green / blue)`
`sendData[5] = (byte) ((Control.LED_KEEP << 4) | Control.LED_KEEP);`
 	
`// LED (white)`
`sendData[6] = (byte) ((Control.LED_KEEP << 4) | Control.LED_OFF);`
 	
`// openings`
`sendData[7] = Control.BLANK;`
 	
`// Send command`
`int ret = this.send_command(sendData);`
`if (ret <= 0) {`
`  System.err.println("failed to send data");`
`}`
 	
`return ret;`
	→引数のブザーパターンと回数動作から制御データを作成する
	→引数の音 A のブザー音階と音 B のブザー音階から音階データを作成する
	→LED 制御は設定維持で作成する
	
	→「4.3 コマンドを送信」を呼び出し、機器にデータを送信

4.8. リセットコマンドの送信

プログラム	説明

Main.java reset()
`byte[] sendData = new byte[Control.SEND_BUFFER_SIZE];`
 	
`// Command version`
`sendData[0] = Control.COMMAND_VERSION;`
 	
`// Command ID`
`sendData[1] = Control.COMMAND_ID;`
 	
`// Buzzer control`
`sendData[2] = Control.BUZZER_OFF;`
 	
`// Buzzer scale`
`sendData[3] = Control.BUZZER_PITCH_OFF;`
 	
`// LED (red / yellow)`
`sendData[4] = Control.LED_OFF;`
 	
`// LED (green / blue)`
`sendData[5] = Control.LED_OFF;`
 	
`// LED (white)`
`sendData[6] = Control.LED_OFF;`
 	
`// openings`
`sendData[7] = Control.BLANK;`
 	
`// Send command`
`int ret = this.send_command(sendData);`
`if (ret <= 0) {`
`  System.err.println("failed to send data");`
`}`
 	
`return ret;`
	→ブザーパターンを停止にする
	
	→ブザー音階を停止にする
	
	→LED ユニットを消灯にする
	
	→「4.3 コマンドを送信」を呼び出し、機器にデータを送信