

LR5-LAN ソケット通信
サンプルプログラム
(Windows Java)

内容

LR5-LAN ソケット通信 サンプルプログラム (Windows Java)	1
1. 概要	4
1.1. システム概要	4
2. 開発環境	4
3. アプリケーション概要	5
3.1. コマンド操作説明	5
3.1.1. コマンド一覧	5
3.1.2. 動作制御コマンド	6
3.1.3. クリアコマンド	6
3.1.4. 状態取得コマンド	6
3.2. メソッド説明	7
3.2.1. メソッド一覧	7
3.2.2. コンストラクタ	8
3.2.3. LR5-LAN に接続	8
3.2.4. ソケットをクローズ	9
3.2.5. コマンドを送信	9
3.2.6. PNS コマンドの動作制御コマンド送信	10
3.2.7. PNS コマンドのクリアコマンド送信	11
3.2.8. PNS コマンドの状態取得コマンド送信	12
3.3. 定数説明	13
3.3.1. 製品区分	13
3.3.2. PNS コマンド識別子	13
3.3.3. PNS コマンドの応答データ	13
3.3.4. 動作制御コマンドの LED ユニットパターン	13
3.3.5. 動作制御コマンドのブザーパターン	14
3.4. データクラス説明	15
3.4.1. 動作制御データクラス	15
3.4.2. 動作制御の状態データ	15
4. プログラム概要	16
4.1. LR5-LAN に接続	16
4.2. ソケットをクローズ	16
4.3. コマンドを送信	17
4.4. PNS コマンドの動作制御コマンド送信	18
4.5. PNS コマンドのクリアコマンド送信	19

4.6.	PNS コマンドの状態取得コマンド送信	20
------	---------------------------	----

1. 概要

LR5-LAN をソケット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 Java 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 LR5-LAN の機器をソケット通信で制御を行う。



2. 開発環境

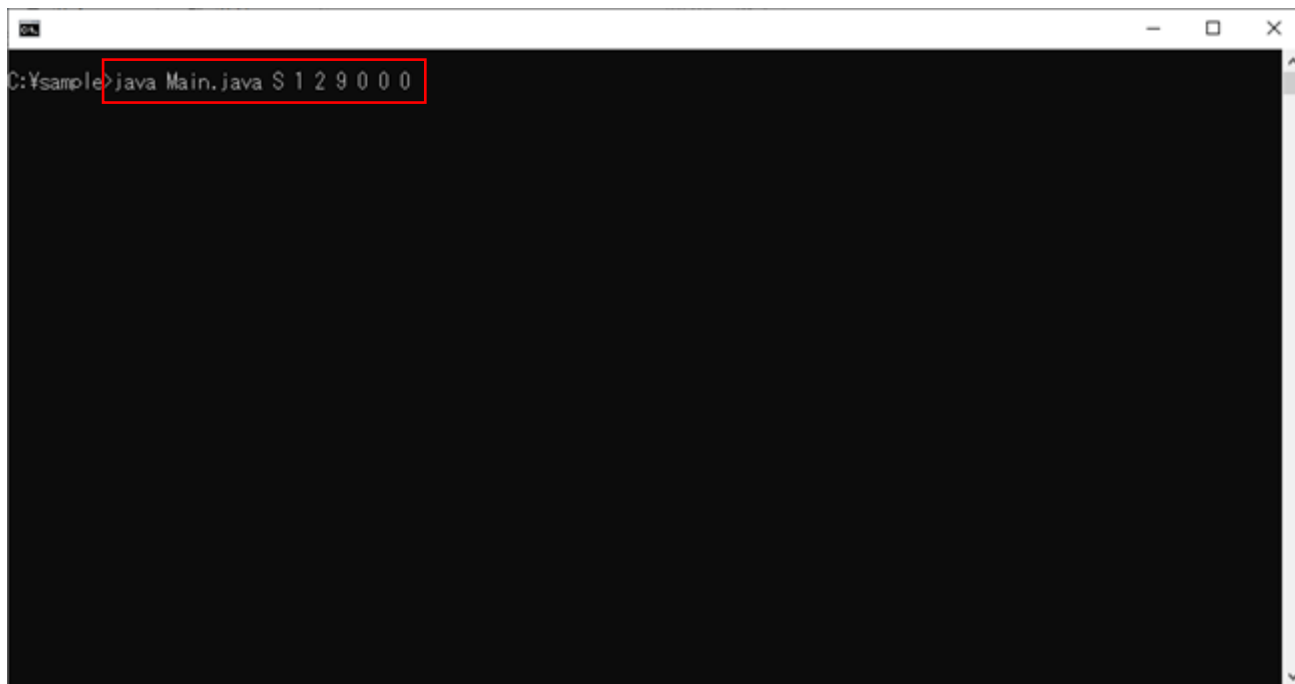
サンプルプログラムの開発環境を以下に示す。

開発環境		備考
開発 OS	Windows11 64bit	
開発言語	Java (Adopt Open JDK)	11 以降
開発ツール	Eclipse	2021-06

3. アプリケーション概要

3.1. コマンド操作説明

コマンドプロンプト上では、Main.java のファイルの場所に移動して、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
動作制御コマンド	LED ユニットの各色のパターンとブザー(吹鳴・停止)を制御する
クリアコマンド	LED ユニットを消灯し、ブザーを停止する
状態取得コマンド	LED ユニットおよびブザーの状態を取得する

3.1.2. 動作制御コマンド

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	S
2	LEDユニット赤	消灯:0 点灯:1 点滅(低速):2 点滅(中速):3 点滅(高速):4 シングルフラッシュ:5 ダブルフラッシュ:6 トリプルフラッシュ:7 変化なし:9
3	LEDユニット黄	
4	LEDユニット緑	
5	LEDユニット青	
6	LEDユニット白	
7	ブザーパターン	停止:0 吹鳴:1 変化なし:9

例:java Main.java S 1 2 9 0 0 1

3.1.3. クリアコマンド

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	C

例:java Main.java C

3.1.4. 状態取得コマンド

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	G

例:java Main.java G

3.2. メソッド説明

3.2.1. メソッド一覧

メソッド名	説明
Control	コンストラクタ
SocketOpen	LR5-LAN に接続する
SocketClose	ソケットをクローズする
SendCommand	コマンドを送信する
PNS_RunControlCommand	PNS コマンドの動作制御コマンド送信する
PNS_ClearCommand	PNS コマンドのクリアコマンド送信する
PNS_GetDataCommand	PNS コマンドの状態取得コマンド送信する

3.2.2. コンストラクタ

関数名	Control(final String ip, final int port)	
パラメータ	String ip	LR5-LAN の IP アドレス
	int port	LR5-LAN のポート番号
戻り値	Control のインスタンス	
説明	指定した IP アドレスの LR5-LAN を操作する Control クラスのインスタンスを作成する	
関数の使用方法	<pre>// メイン関数 public static void main(String[] args) { Control ctr = new Control ("192.168.10.1", 10000); }</pre>	
備考	プログラムの概要は「4.1LR5-LAN に接続」を参照	

3.2.3. LR5-LAN に接続

関数名	public int SocketOpen()	
パラメータ	なし	
戻り値	int	成功:0、失敗:0 以外
説明	指定した IP アドレスとポート番号の LR5-LAN にソケット通信で接続する	
関数の使用方法	<pre>// メイン関数 public static void main(String[] args) { Control ctr = new Control ("192.168.10.1", 10000); // サーバーに接続 int ret = ctr.SocketOpen(); if (ret == -1) { return; } }</pre>	
備考	プログラムの概要は「4.1LR5-LAN に接続」を参照	

3.2.4. ソケットをクローズ

関数名	public void SocketClose()
パラメータ	なし
戻り値	なし
説明	LR5-LAN に接続したソケットをクローズする
関数の使用方法	<pre>// メイン関数 public static void main(String[] args) { Control ctr = new Control ("192.168.10.1", 10000); // サーバーに接続 int ret = ctr.SocketOpen(); if (ret == -1) { return; } try { // 何れかの処理 } finally { // ソケットをクローズ ctr.SocketClose(); } }</pre>
備考	プログラムの概要は「4.2 ソケットをクローズ」を参照

3.2.5. コマンドを送信

関数名	private byte[] SendCommand(final byte[] sendData)	
パラメータ	byte[] sendData	送信データ
戻り値	byte[]	受信データ
説明	接続した LR5-LAN にデータを送信して、応答データを返す	
関数の使用方法	※Control クラス内でのみ使用可能	
備考	プログラムの概要は「4.3 コマンドを送信」を参照	

3.2.6. PNS コマンドの動作制御コマンド送信

関数名	public int PNS_RunControlCommand(final PNS_RUN_CONTROL_DATA runControlData)	
パラメータ	PNS_RUN_CONTROL_DATA runControlData	LED ユニットの各色のパターンとブザーを制御する送信データ 詳細は「3.4.1 動作制御データ」を参照
戻り値	Int	成功:0、失敗:0 以外
説明	PNS コマンドの動作制御コマンドを送信して、LED ユニットの各色のパターンとブザーを制御する	
関数の使用方法	<pre>// メイン関数 public static void main(String[] args) { Control ctr = new Control ("192.168.10.1", 10000); // サーバーに接続 int ret = ctr.SocketOpen(); if (ret == -1) { return; } try { // PNS コマンドの動作制御コマンド送信 // LED パターン 0: 消灯 // LED パターン 1: 点灯 // LED パターン 2: 点滅(低速) // LED パターン 3: 点滅(中速) // LED パターン 4: 点滅(高速) // LED パターン 5: シングルフラッシュ // LED パターン 6: ダブルフラッシュ // LED パターン 7: トリプルフラッシュ // LED パターン 9: 変化なし // ブザーパターン 0: 停止 // ブザーパターン 1: 吹鳴 // ブザーパターン 9: 変化なし Control.PNS_RUN_CONTROL_DATA runControlData = ctr.new PNS_RUN_CONTROL_DATA(); runControlData.ledRedPattern = Control.PNS_RUN_CONTROL_LED_ON; runControlData.ledAmberPattern = Control.PNS_RUN_CONTROL_LED_BLINKING_SLOW; runControlData.ledGreenPattern = Control.PNS_RUN_CONTROL_LED_NO_CHANGE; runControlData.ledBluePattern = Control.PNS_RUN_CONTROL_LED_OFF; runControlData.ledWhitePattern = Control.PNS_RUN_CONTROL_LED_FLASHING_TRIPLE; runControlData.buzzerPattern = Control.PNS_RUN_CONTROL_BUZZER_RING; ctr.PNS_RunControlComman(runControlData); } finally { // ソケットをクローズ ctr.SocketClose(); } }</pre>	
備考	プログラムの概要は「4.4PNS コマンドの動作制御コマンド送信」を参照	

3.2.7. PNS コマンドのクリアコマンド送信

関数名	public int PNS_ClearCommand()	
パラメータ	なし	
戻り値	Int	成功:0、失敗:0 以外
説明	PNS コマンドのクリアコマンドを送信して、LED ユニットを消灯し、ブザーを停止する	
関数の使用方法	<pre>// メイン関数 public static void main(String[] args) { Control ctr = new Control ("192.168.10.1", 10000); // サーバーに接続 int ret = ctr.SocketOpen(); if (ret == -1) { return; } try { // PNS コマンドのクリアコマンド送信 PNS_ClearCommand(); } finally { // ソケットをクローズ ctr.SocketClose(); } }</pre>	
備考	プログラムの概要は「4.5PNS コマンドのクリアコマンド送信」を参照	

3.2.8. PNS コマンドの状態取得コマンド送信

関数名	public PNS_STATUS_DATA PNS_GetDataCommand()	
パラメータ	なし	
戻り値	PNS_STATUS_DATA	状態取得コマンドの受信データ(LED ユニットおよびブザーの状態) 詳細は「3.4.2 動作制御の状態データ」を参照
説明	PNS コマンドの状態取得コマンドを送信して、LED ユニットおよびブザーの状態を取得する	
関数の使用方法	<pre>// メイン関数 public static void main(String[] args) { Control ctr = new Control ("192.168.10.1", 10000); // サーバーに接続 int ret = ctr.SocketOpen(); if (ret == -1) { return; } try { // PNS コマンドの状態取得コマンド送信 Control.PNS_STATUS_DATA statusData = ctr.PNS_GetDataCommand(); } finally { // ソケットをクローズ ctr.SocketClose(); } }</pre>	
備考	プログラムの概要は「4.66PNS コマンドの状態取得コマンド送信」を参照	

3.3. 定数説明

3.3.1. 製品区分

定数名	値	説明
PNS_PRODUCT_ID	0x4142	LR5-LAN の製品区分

3.3.2. PNS コマンド識別子

定数名	値	説明
PNS_RUN_CONTROL_COMMAND	X053	動作制御コマンド
PNS_CLEAR_COMMAND	0x43	クリアコマンド
PNS_REBOOT_COMMAND	0x42	再起動コマンド

3.3.3. PNS コマンドの応答データ

定数名	値	説明
PNS_ACK	0x06	正常応答
PNS_NAK	0x15	異常応答

3.3.4. 動作制御コマンドの LED ユニットパターン

定数名	値	説明
PNS_RUN_CONTROL_LED_ON	0x00	消灯
PNS_RUN_CONTROL_LED_OFF	0x01	点灯
PNS_RUN_CONTROL_LED_BLINKING_SLOW	0x02	点滅(低速)
PNS_RUN_CONTROL_LED_BLINKING_MEDIUM	0x03	点滅(低速)
PNS_RUN_CONTROL_LED_BLINKING_HIGH	0x04	点滅(低速)
PNS_RUN_CONTROL_LED_FLASHING_SINGLE	0x05	シングルフラッシュ
PNS_RUN_CONTROL_LED_FLASHING_DOUBLE	0x06	ダブルフラッシュ
PNS_RUN_CONTROL_LED_FLASHING_TRIPLE	0x07	トリプルフラッシュ
PNS_RUN_CONTROL_LED_NO_CHANGE	0x09	変化なし

3.3.5. 動作制御コマンドのブザーパターン

定数名	値	説明
PNS_RUN_CONTROL_BUZZER_STOP	0x00	停止
PNS_RUN_CONTROL_BUZZER_RING	0x01	吹鳴
PNS_RUN_CONTROL_BUZZER_NO_CHA NGE	0x09	変化なし

3.4. データクラス説明

3.4.1. 動作制御データクラス

名前	PNS_RUN_CONTROL_DATA
定義	<pre> public class PNS_RUN_CONTROL_DATA { /** LED ユニット赤色のパターン */ public byte ledRedPattern = 0; /** LED ユニット黄色のパターン */ public byte ledAmberPattern = 0; /** LED ユニット緑色のパターン */ public byte ledGreenPattern = 0; /** LED ユニット青色のパターン */ public byte ledBluePattern = 0; /** LED ユニット白色のパターン */ public byte ledWhitePattern = 0; /** ブザーのj 状態 */ public byte buzzerMode = 0; } </pre>
説明	動作制御コマンドで送信するデータエリアの LED ユニットの各色のパターンとブザーの状態

3.4.2. 動作制御の状態データ

名前	PNS_STATUS_DATA
定義	<pre> public class PNS_STATUS_DATA { /** LED パターン 1~5 */ public byte[] ledPattern = new byte[8]; /** ブザーモード */ public byte buzzer = 0; } </pre>
説明	動作制御の状態取得コマンドの応答データの LED ユニットおよびブザーの状態のデータクラス

4. プログラム概要

プログラムの動作を要点のみ記載する。

4.1. LR5-LAN に接続

プログラム	説明
<pre>main.java /** Socket */ private Socket sock;</pre>	→ソケットのメンバ変数を定義
<pre>main.java SocketOpen() /** * Connect to LR5-LAN * * @return success: 0, failure: non-zero */ public int SocketOpen() { try { // Create a socket this.sock = new Socket(this.ip, this.port); // Obtaining the transmitted and received streams this.out = this.sock.getOutputStream(); this.in = this.sock.getInputStream(); } catch (IOException ex) { ex.printStackTrace(); return -1; } return 0; }</pre>	→ソケットの作成し、接続する →送受信ストリームの取得

4.2. ソケットをクローズ

プログラム	説明
<pre>main.java SocketClose() public void SocketClose() { try { if (this.in != null) { this.in.close(); this.in = null; } if (this.out != null) { this.out.close(); this.out = null; } if (this.sock != null) { this.sock.close(); this.sock = null; } } catch (IOException ex) { ex.printStackTrace(); } }</pre>	→受信ストリームのクローズメソッドを呼び出す →送信ストリームのクローズメソッドを呼び出す →ケットのクローズメソッドを呼び出す

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作成し、LR5-LAN 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は「4.4PNS コマンドの動作制御コマンド送信」以降を参照

プログラム	説明
<pre>main.java SendCommand() try { if (this.sock == null) { System.err.println("socket is not"); return null; } // Send this.out.write(sendData); // Receive response data byte[] recvData = new byte[1024]; int size = this.in.read(recvData); // Truncate the incoming data to the size you re recvData = Arrays.copyOf(recvData, size); return recvData; } catch (IOException ex) { ex.printStackTrace(); return null; }</pre>	<p>→作成した送信データを send メソッドで送信</p> <p>→送信後に recv メソッドで機器からのレスポンスを取得</p>

4.4. PNS コマンドの動作制御コマンド送信

プログラム	説明
<pre> main.java PNS_RunControlCommand() ByteBuffer sendData = ByteBuffer.allocate(12); // Product Category (AB) sendData.putShort(PNS_PRODUCT_ID); // Command identifier (S) sendData.put(PNS_RUN_CONTROL_COMMAND); // Empty sendData.put((byte) 0x00); // Data size (data area) byte[] data = { runControlData.ledRedPattern, // LED Red pattern runControlData.ledAmberPattern, // LED Amber pattern runControlData.ledGreenPattern, // LED Green pattern runControlData.ledBluePattern, // LED Blue pattern runControlData.ledWhitePattern, // LED White pattern runControlData.buzzerMode, // Buzzer mode }; sendData.putShort((short) data.length); sendData.put(data); // Send PNS command byte[] recvData = this.SendCommand(sendData.array()); if (recvData == null) { System.err.println("failed to send data"); return -1; } // check the response data if (recvData[0] == PNS_NAK) { // receive abnormal response System.err.println("negative acknowledge"); return -1; } return 0; </pre>	<p>以下の順で送信データを作成</p> <ul style="list-style-type: none"> →1 バイト目: 製品区分 (A: 0x41) →2 バイト目: 製品区分 (B: 0x42) →3 バイト目: 識別子 (S: 0x53) →4 バイト目: 空き (0x00) →5 バイト目: データサイズ (0x00) →6 バイト目: データサイズ (0x06) →7~12 バイト目: データエリア <p>データサイズは 6 バイト</p> <p>データエリアには「3.4.1 動作制御データ」の値を設定する</p> <p>→「4.3 コマンドを送信」を呼び出し、機器にデータを送信</p> <p>→送信後に応答データを確認</p> <p>正常応答: ACK(0x06)</p> <p>異常応答: NAK(0x15)</p>

4.5. PNS コマンドのクリアコマンド送信

プログラム	説明
<pre> main.java PNS_ClearCommand() ByteBuffer sendData = ByteBuffer.allocate(6); // Product Category (AB) sendData.putShort(PNS_PRODUCT_ID); // Command identifier (C) sendData.put(PNS_CLEAR_COMMAND); // Empty sendData.put((byte) 0x00); // Data size sendData.putShort((short) 0); // Send PNS command byte[] recvData = this.SendCommand(sendData.array()); if (recvData == null) { System.err.println("failed to send data"); return -1; } // check the response data if (recvData[0] == PNS_NAK) { // receive abnormal response System.err.println("negative acknowledge"); return -1; } return 0; </pre>	<p>以下の順で送信データを作成</p> <ul style="list-style-type: none"> →1 バイト目: 製品区分 (A: 0x41) →2 バイト目: 製品区分 (B: 0x42) →3 バイト目: 識別子 (C: 0x43) →4 バイト目: 空 (0x00) →5 バイト目: データサイズ (0x00) →6 バイト目: データサイズ (0x00) <p>データサイズは 0 バイト データエリアは無し</p> <p>→「4.3 コマンドを送信」を呼び出し、機器にデータを送信</p> <p>→送信後に応答データを確認 正常応答: ACK(0x06) 異常応答: NAK(0x15)</p>

4.6. PNS コマンドの状態取得コマンド送信

プログラム	説明
<pre> main.java PNS_GetDataCommand() ByteBuffer sendData = ByteBuffer.allocate(6);↓ // Product Category (AB)↓ sendData.putShort(PNS_PRODUCT_ID);↓ // Command identifier (G)↓ sendData.put(PNS_GET_DATA_COMMAND);↓ // Empty↓ sendData.put((byte) 0x00);↓ // Data size↓ sendData.putShort((short) 0);↓ // Send PNS command↓ byte[] recvData = this.SendCommand(sendData.array());↓ if (recvData == null) {↓ System.err.println("failed to send data");↓ return null;↓ }↓ // check the response data↓ if (recvData[0] == PNS_NAK) {↓ // receive abnormal response↓ System.err.println("negative acknowledge");↓ return null;↓ }↓ PNS_STATUS_DATA statusData = new PNS_STATUS_DATA();↓ // LED Pattern 1 to 5↓ System.arraycopy(recvData, 0, statusData.ledPattern, 0, statusl // buzzer Mode↓ statusData.buzzer = recvData[5];↓ return statusData;↓ </pre>	<p>以下の順で送信データを作成</p> <ul style="list-style-type: none"> →1 バイト目: 製品区分 (A:0x41) →2 バイト目: 製品区分 (B:0x42) →3 バイト目: 識別子 (G:0x47) →4 バイト目: 空 (0x00) →5 バイト目: データサイズ (0x00) →6 バイト目: データサイズ (0x00) <p>データサイズは 0 バイト データエリアは無し</p> <p>→「4.3 コマンドを送信」を呼び出し、機器にデータを送信</p> <p>→送信後に応答データを確認</p> <p>正常応答: ACK(0x06) 異常応答: NAK(0x15)</p> <p>以下の処理で応答データの各データの取得</p> <p>→LED ユニットの状態</p> <ul style="list-style-type: none"> ・1 バイト目: LED ユニット赤色の状態 ・2 バイト目: LED ユニット黄色の状態 ・3 バイト目: LED ユニット緑色の状態 ・4 バイト目: LED ユニット青色の状態 ・5 バイト目: LED ユニット白色の状態 ・6 バイト目: ブザーの状態