PlusG SMART Solution をお試し頂きまして、ありがとうございます。本書は簡単なアプリケーションの製作を通して、PlusG SMART Solution を利用したアプリケーションの開発工程を理解して頂く事を目的としています。PlusG SMART Solution の評価や導入に、お役立てください。

1. 準備

1.1. システム要件

本書では、次の機器を使用します。要件を満たす機器及び環境をご用意ください。

■ Android タブレットまたはスマートフォン

専用クライアント「PGSMonitor」を動かすために使います。

[システム要件]

OS： Android 3.2 以降

NET： リソースエディタを実行する Window PC との TCP/IP 通信（例：無線 LAN）

■ Windows PC

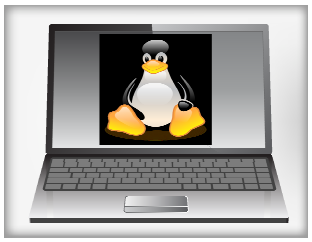
リソースエディタ「SGResourceEditor」、統合開発環境「SWEET」を動かすために使います。

[システム要件]

OS： Windows XP SP3 32bit, Windows Vista 32/64bit, Windows 7 32/64bit

NET： PGSMonitor を実行する Android 機器との TCP/IP 通信（例：無線 LAN または、アクセスポイントのある有線 LAN）、ビルドを実行する Linux PC との Telnet & Samba 通信

■ Linux PC



アプリケーションをビルドするために使います。

Linux PC をお持ちでない方は、仮想 PC ソフトウェアをご利用ください。弊社では「VirtualBox」を使用しております。VirtualBox 向けのアプライアンス（構築済イメージ）を用意しております。

VitualBox の導入手順は、別紙の「VirtualBox インストールガイド」（http://www.si-linux.co.jp/pub/SmartSolution/）をお読みください。VirtualBox のインストーラーは、公式サイト(https://www.virtualbox.org/)より、アプライアンスは弊社サイト

（http://www.si-linux.co.jp/pub/SmartSolution/）より、ダウンロードできます。

既存の Linux PC を利用する方は、システム要件をご確認ください。

[システム要件]

OS: Debian GNU/Linux 6.0（推奨）

PKG: gcc, make, telnetd, samba

NET: PGSMonitor を実行する Android 機器との TCP/IP 通信、
SWEET を実行する Windows PC との Telnet & Samba 通信

1.2. PGSMonitor のセットアップ

Android にアプリケーションの画面を表示するソフトウェア「PGSMonitor」をインストールします。下の 2 次元コードが示す URL から PGSMonitor.apk をダウンロード＆インストールしてください。

2 次元コードを利用できない方は、Android のブラウザから
http://www.si-linux.co.jp/pub/SmartSolution/PGSMonitor/CurrentVersion/ にアクセスしてください。

1.3. リソースエディタのセットアップ

Windows PC にアプリケーションの画面を作成するソフトウェア「SGResourceEditor」（以下、リソースエディタ）をインストールします。

Windows のブラウザから http://www.si-linux.co.jp/pub/SmartSolution/ResourceEditor/CurrentVersion/ にアクセスして、ResourceEditor*.*.*.*.zip(*はバージョン番号)をダウンロードしてください。ファイルが複数ある場合は、数字の大きい方をダウンロードしてください。

ダウンロード後、任意のフォルダ（インストール先）に展開してください。

1.4. SWEET のセットアップ

Windows PC に統合開発環境「SWEET」をインストールします。

Windows のブラウザから http://www.si-linux.co.jp/product/sweet/download/ にアクセスして、

SweetInstaller*_*_*_*.exe（*はバージョン番号）をダウンロードしてください。ファイルが複数ある場合は、数字の大きい方をダウンロードしてください。

ダウンロード後はファイルを実行して、ウィザードに従ってインストールしてください。

1.5. サンプルプロジェクトのダウンロード

Windows PC にサンプルのプロジェクトファイルをダウンロードします。

Windows のブラウザから http://www.si-linux.co.jp/pub/SmartSolution/Tutorial/ にアクセスして、「OKOnly.pgsspro」を任意のフォルダにダウンロードしてください。

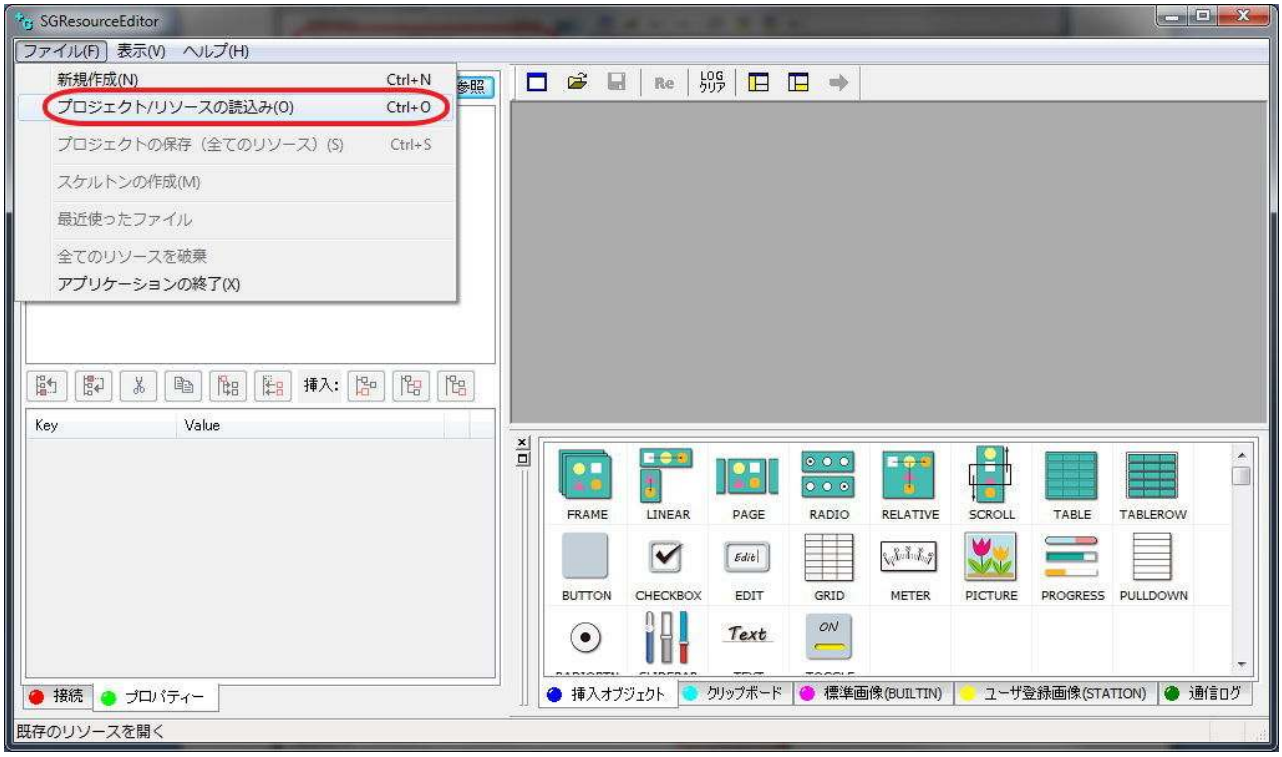
2. サンプルによる動作確認

2.1. サンプルプロジェクトの読み込み

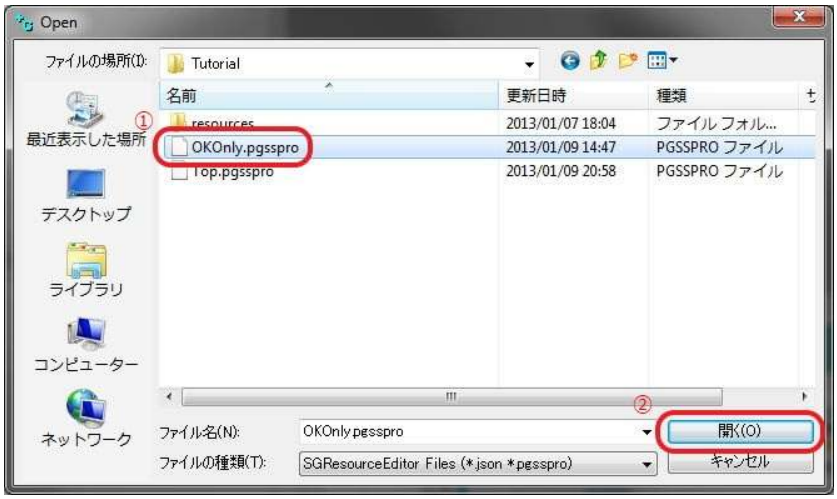
サンプルプロジェクトを利用して、お使いの環境でアプリケーションが動作するかを確認します。

リソースエディタを起動します。展開先フォルダの SGResourceEditor.exe を実行してください。

サンプルプロジェクトファイルを開きます。メニューから[ファイル]→[プロジェクト/リソースの読み込み]を実行してください。



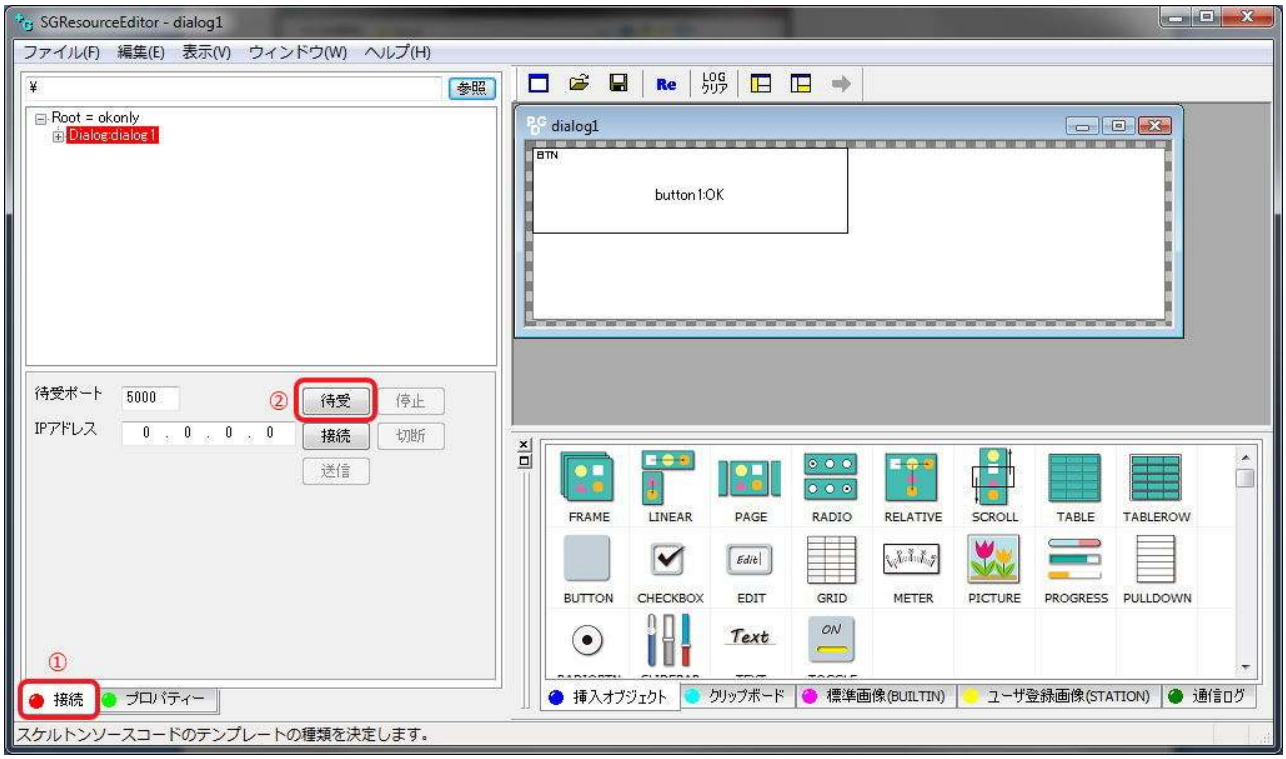
ダウンロードした”OKOnly.pgsspro”を開きます。



2.2. リソースエディター－PGSMonitor 接続

プロジェクトを開いたら、ウィンドウの左下にある[接続]タブを選択して、タブ内の[待受]ボタンをクリッ

クします。リソースエディタが PGSMonitor からの接続待ち受けを開始します。

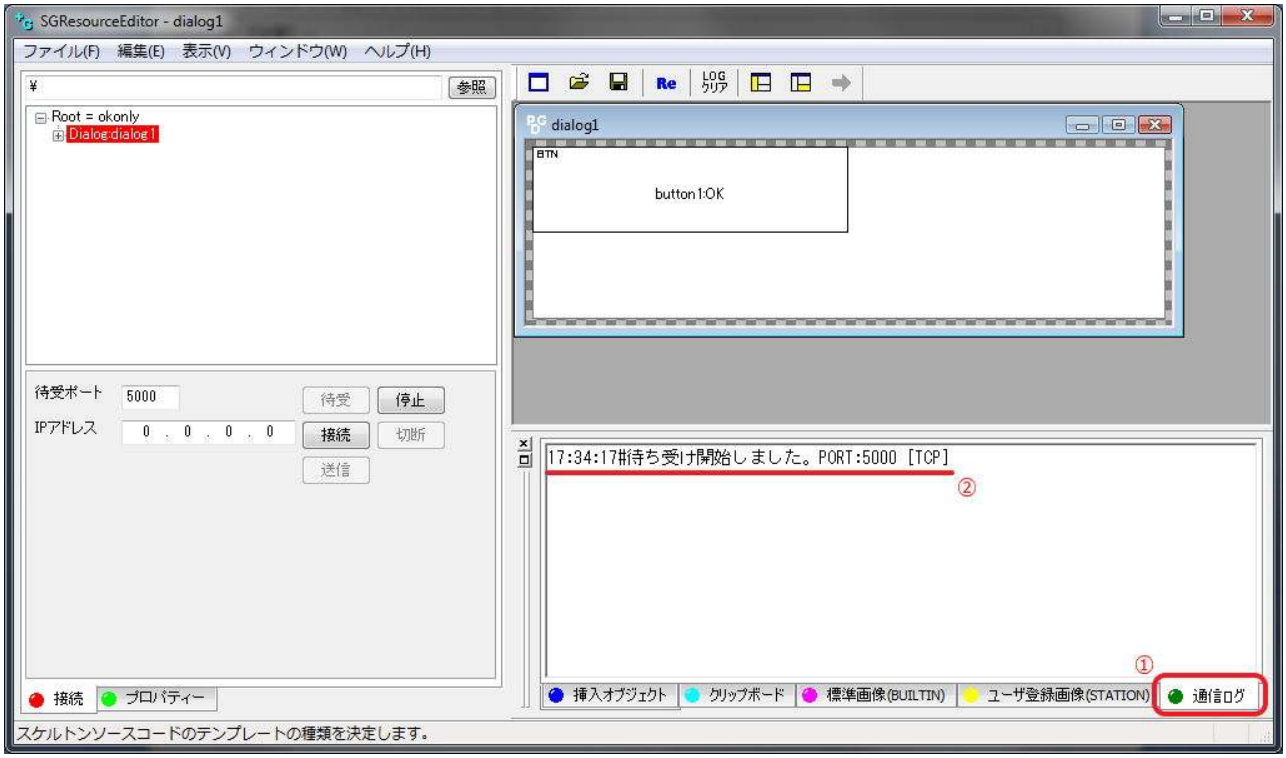


OS によっては初回実行時に[Windows セキュリティの重要な警告]が表示される事があります。その場合は[アクセスを許可する]ボタンをクリックしてください。

【注意】
[Windows セキュリティの重要な警告]は、デフォルトでは「プライベートネットワーク」のアクセスのみ許可するようになっています。後の工程で PGSMonitor と通信ができない場合は、この設定とネットワーク種類の不一致が原因の可能性がありますので、ネットワーク管理者やパワーユーザーに問い合わせて、設定を確認してください。

ウィンドウの右下にある[通信ログ]タブを選択します。待ち受けが開始されると、ログに次のメッセージが表示されます。

```
**:**:**#待ち受け開始しました。PORT:5000 [TCP] (*は数字)
```



リソースエディタに PGSMonitor を接続します。Android で PGSMonitor を起動してください。

アプリメニューを開いて [IP アドレス登録]をタップします。IP アドレス登録画面が開きます。

各フィールドに値を入力して[登録]ボタンをタップしてください。自動で再接続が開始されます。

ユーザー名：guest

パスワード：guest

接続方法：TCP 接続

TCP IP アドレス：Windows PC
（リソースエディタ）の IP アドレス

TCP ポート：5000

接続に成功すると PGSMonitor 上にポップアップで「TCP 接続：接続開始！」と表示されます。

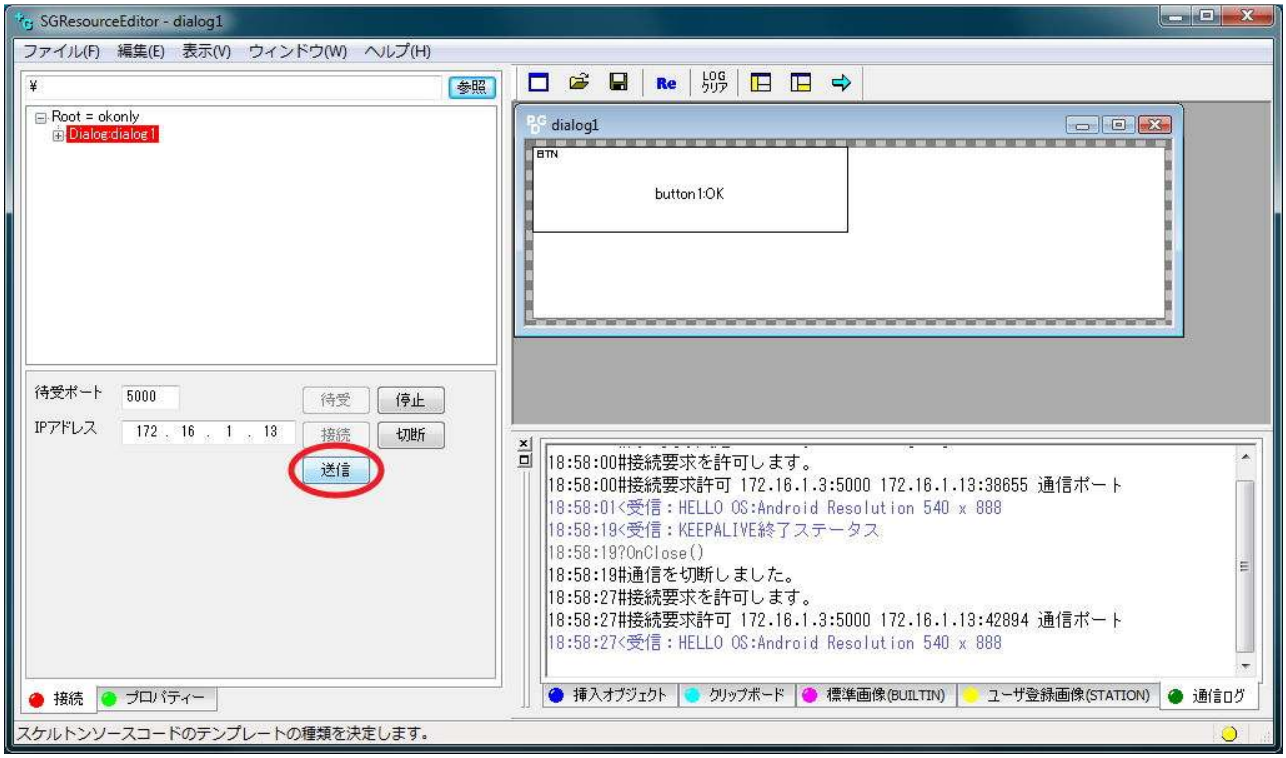
リソースエディタの通信ログに次のメッセージが表示されます。

```
**:**:**<受信：HELLO OS:Android Resolution *** x *** (*は数字)
```

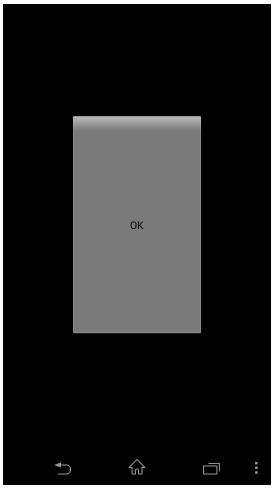
2.3. 画面送信－イベント受信

リソースエディタから PGSMonitor にサンプルダイアログを送信します。リソースエディタの[接続]タブ内にある[送信]ボタンをクリックしてください。

【注意】
[送信]ボタンが無効になっている場合、Android 側のスリープ等により接続が切れている事が考えられます。その時は、Android をスリープから復帰して、アプリメニューの[再接続]を実行してください。



ダイアログの送信－受信に成功すると、PGSMonitor に次の様なサンプルダイアログが表示されます。



サンプルダイアログが表示されたら、ダイアログ上の[OK]ボタンをタップしてください。リソースエディタの通信ログに次のメッセージが表示されます。このメッセージは、PGSMonitor で”button1”ボタンがタップされた事を通知しています。

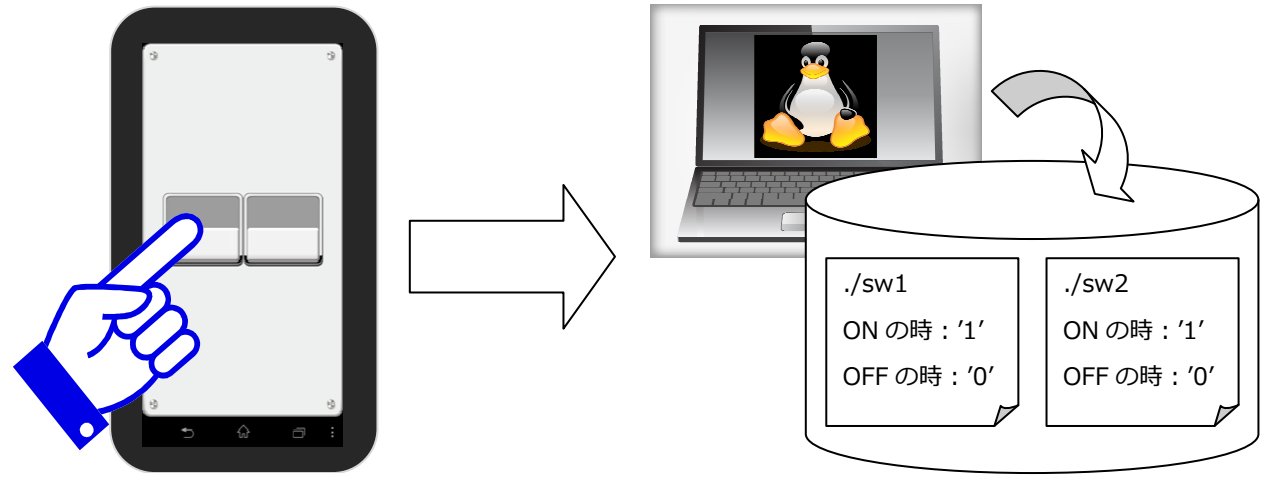
```
**:**:**<受信：CONDITION {'Command':'CONDITION','ActionName':'button1',... （*は数字）
```

このように PlusG SMART Solution では、ユーザーが Android の画面を操作すると、予め設定されたコマンドがサーバー（アプリケーション）に通知（送信）されるようになっています。サーバー（アプリケーション）は受け取ったコマンドに応じて機器の制御などの振る舞いを行い、必要であればコマンドを使ってクライアント（Android）にフィードバックを指示（送信）します。これらのやり取りが PlusG SMART Solution における基本的な処理の流れとなります。

3. アプリケーションの製作

3.1. 概要

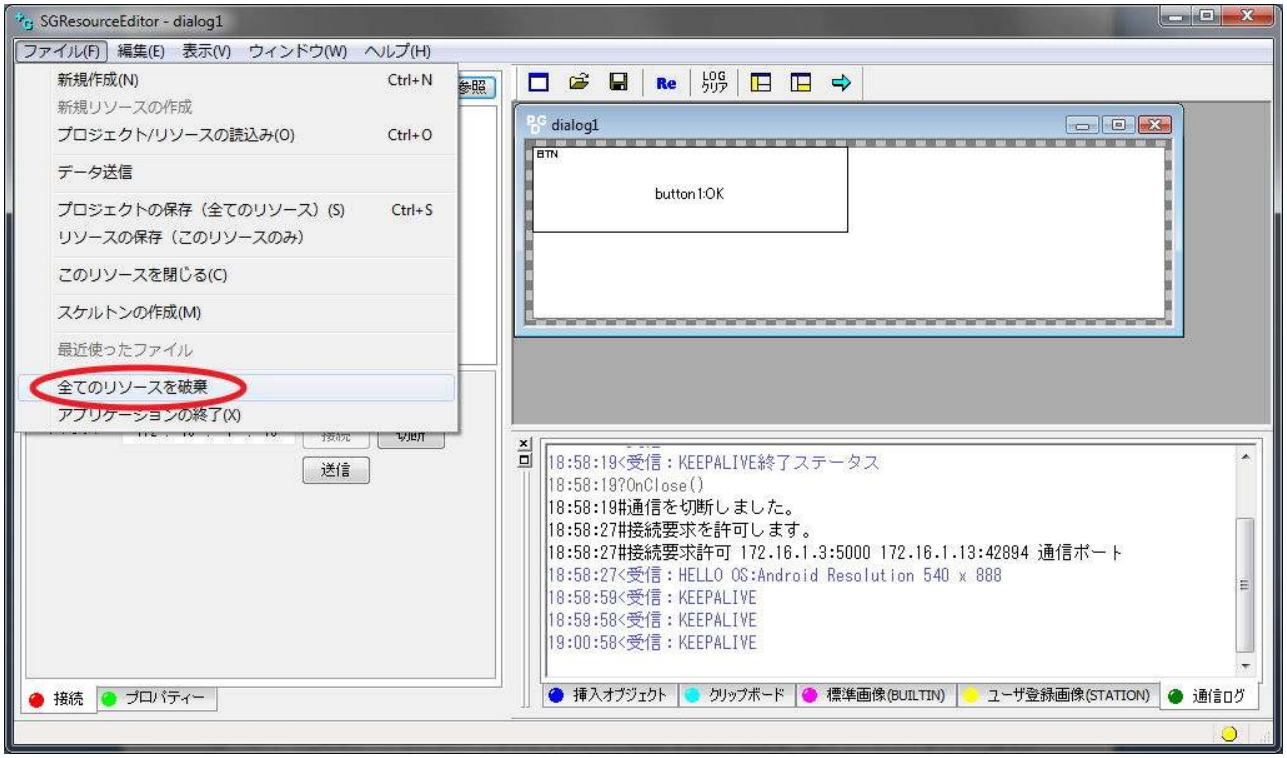
ここから、アプリケーションの製作手順を説明します。「画面にトグルを表示して、トグルの ON/OFF が切り替えられると、ファイルに値を書き込む。」というアプリケーションを製作していきます。



トグルをタップして ON/OFF を切り替えると…　　ファイルに値が書き込まれる。

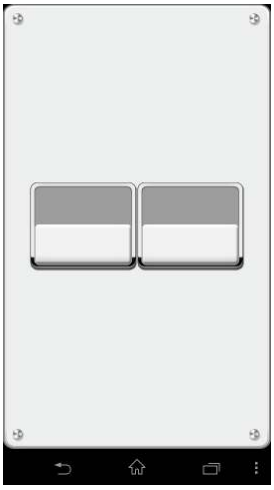
3.2. リソースの破棄

新しいプロジェクトを始める前に、既存のプロジェクト（サンプル）を破棄します。リソースエディタのメニューから[ファイル]→[全てのリソースを破棄]を実行してください。ツリーから全てのダイアログが削除されます。

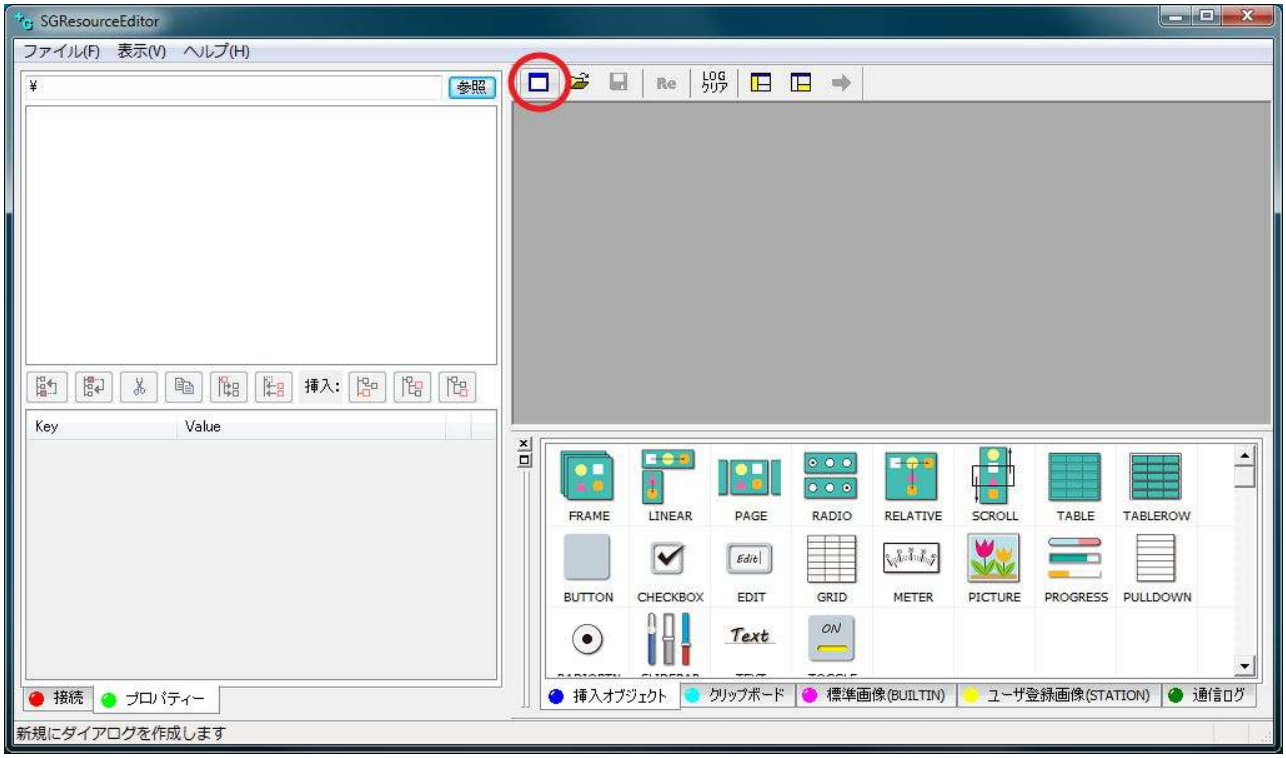


3.3. 画面の作成

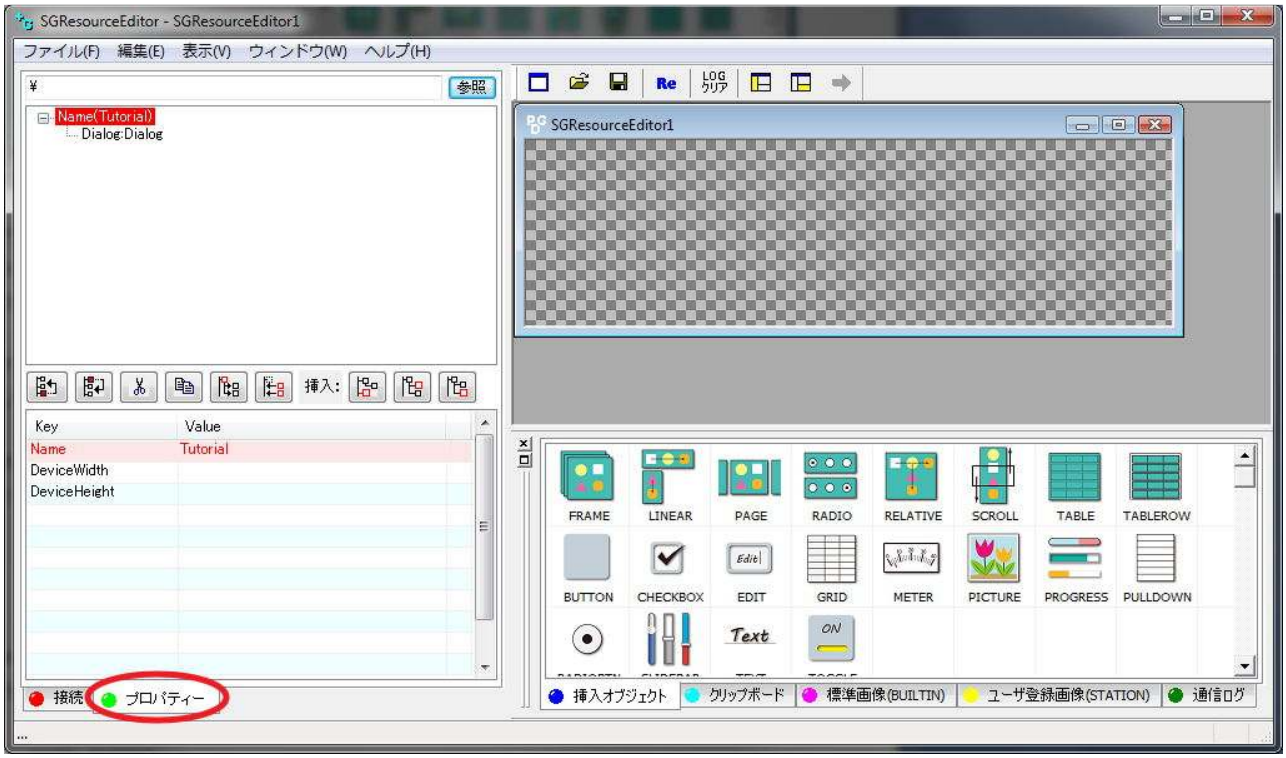
新しい画面を作成します。作成する画面のイメージは次の通りです。



ツールバーの[新規ダイアログ]ボタンをクリックしてください。リソースツリーにダイアログが追加されます。



アプリケーションのプロパティ値を変更します。プロパティ値を変更するには、[プロパティ]タブに切り替えます。



ツリーから対象のノードを選択します。リストにプロパティと値の一覧が表示されるので、変更するプロパティ行の Value 列をクリックします。値が反転（選択中）表示になり、変更できるようになります。

アプリケーションのプロパティを変更する場合は、「Root = ...」 を選択します。次のプロパティ値を変更してください。

Name:Tutorial

ダイアログのプロパティを変更します。ツリーの「Dialog:Dialog」を選択して、次のプロパティ値を変更してください。

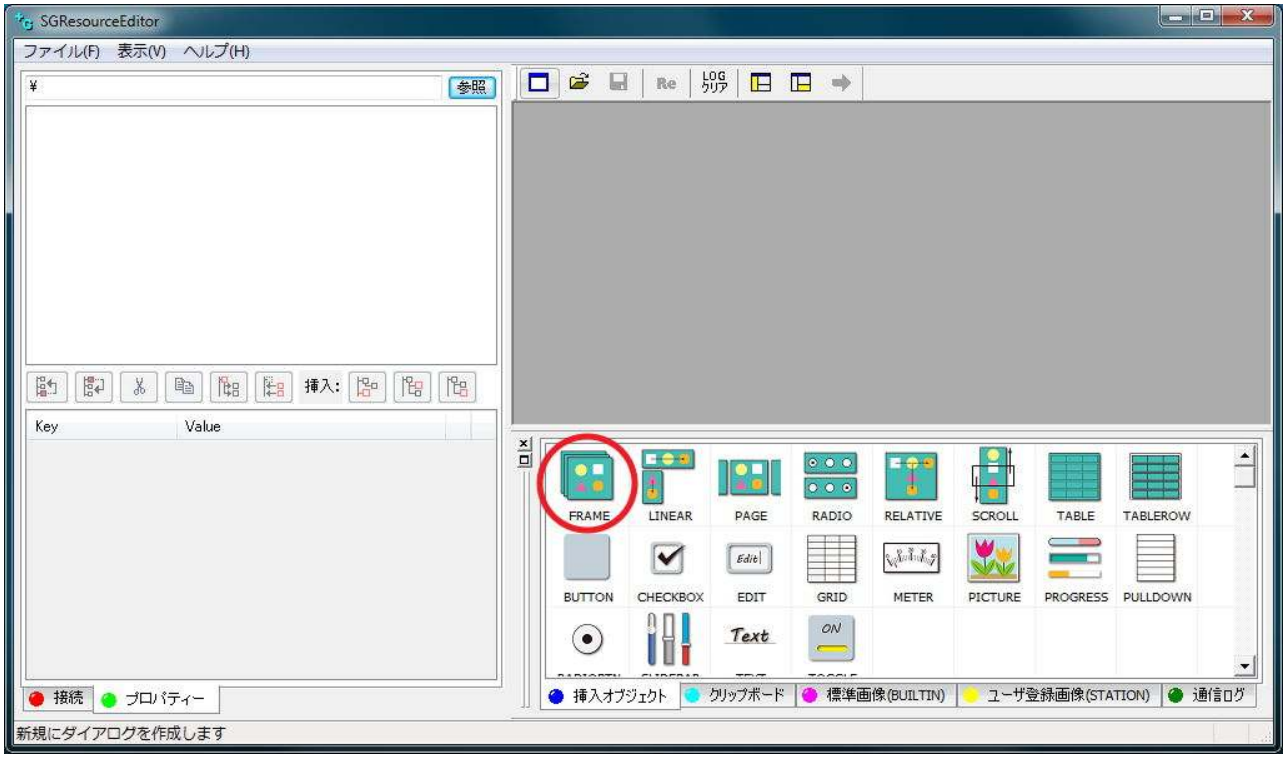
Name: Top

TemplateType: Shared

TargetLayoutWidth: 100

TargetLayoutHeight: 100

ダイアログに FRAME グループを追加します。[挿入オブジェクト]タブ内の[FRAME]をダブルクリックしてください。ダイアログに FRAME グループが追加されます。



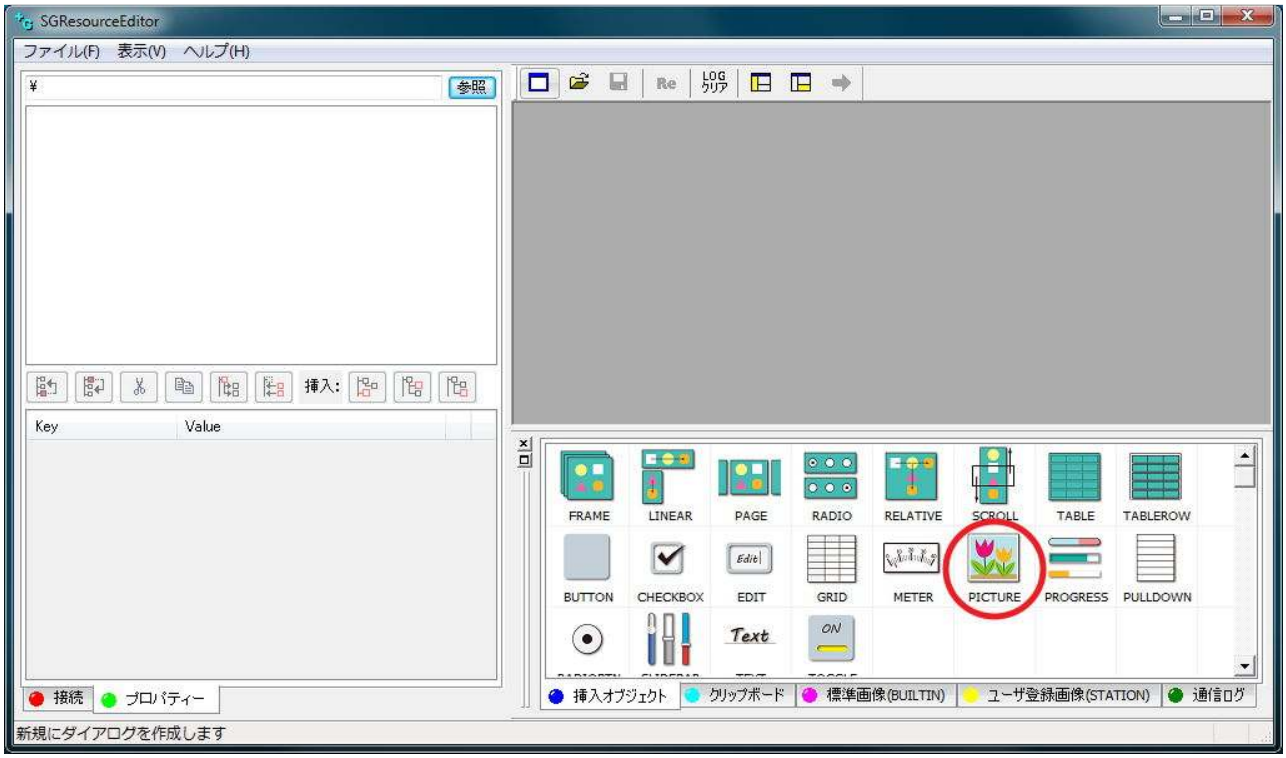
次のプロパティ値を変更します。

Name: frame1

Width: PARENT

Height: PARENT

FRAME グループに PICTURE オブジェクトを追加してください。



プロパティ値を変更してください。

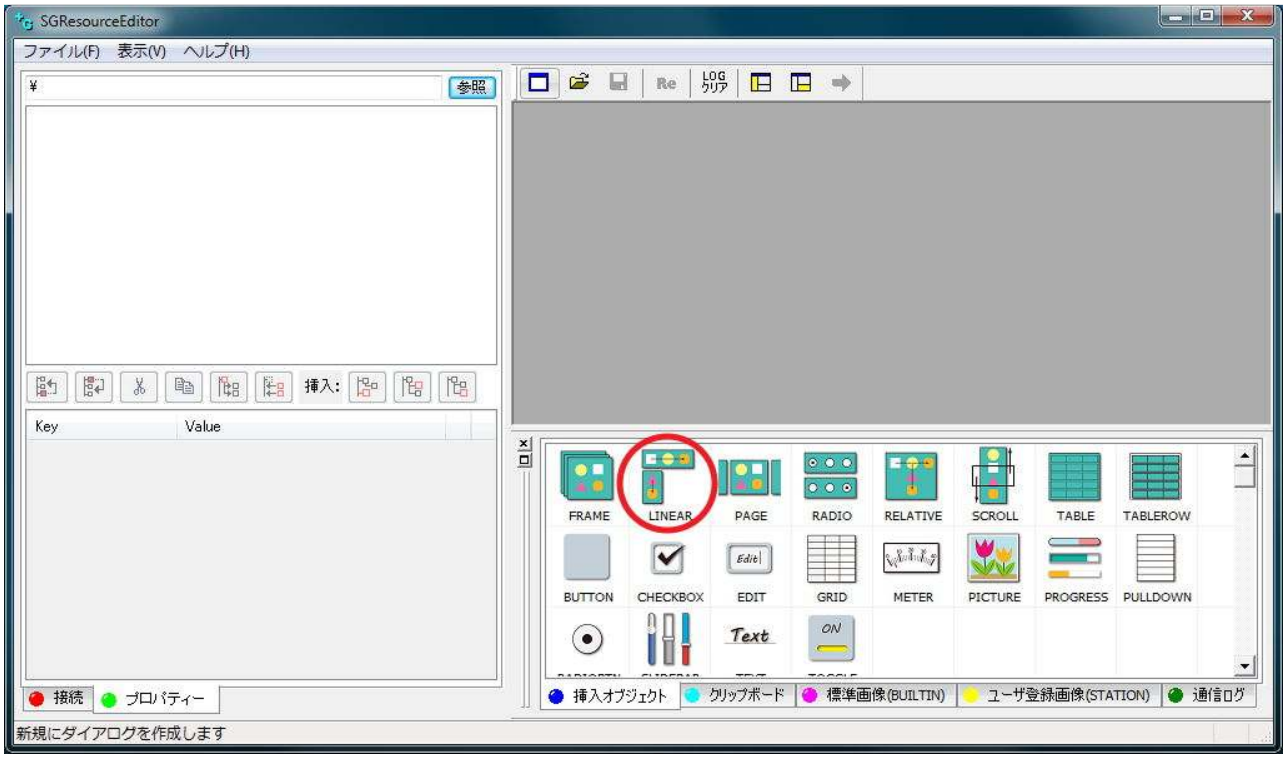
Name: picture1

Width: PARENT

Height: PARENT

File: frame.9.png

FRAME グループに LINEAR グループを追加してください。



プロパティ値を変更してください。

Name: linear1

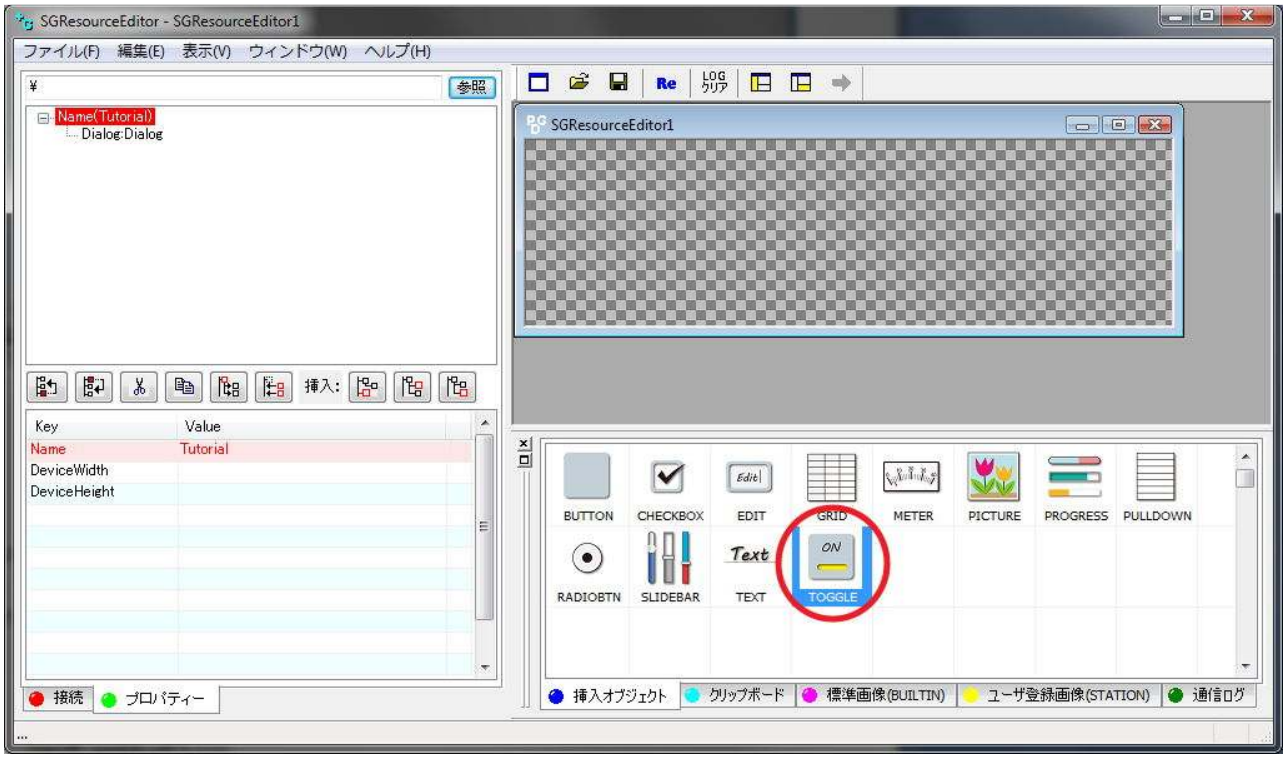
Width: PARENT

Height: PARENT

Padding: 10,20,10,20

Align: CENTER|MIDDLE

LINEAR グループに TOGGLE オブジェクト（１つ目）を追加してください。



プロパティ値を変更してください。

Name: sw1

Height: 20

Weight: 1

FileOff: switch_off.9.png

FileOn: switch_on.9.png

LINEAR グループに TOGGLE オブジェクト（２つ目）を追加して、プロパティ値を変更してください。

Name: sw2

Height: 20

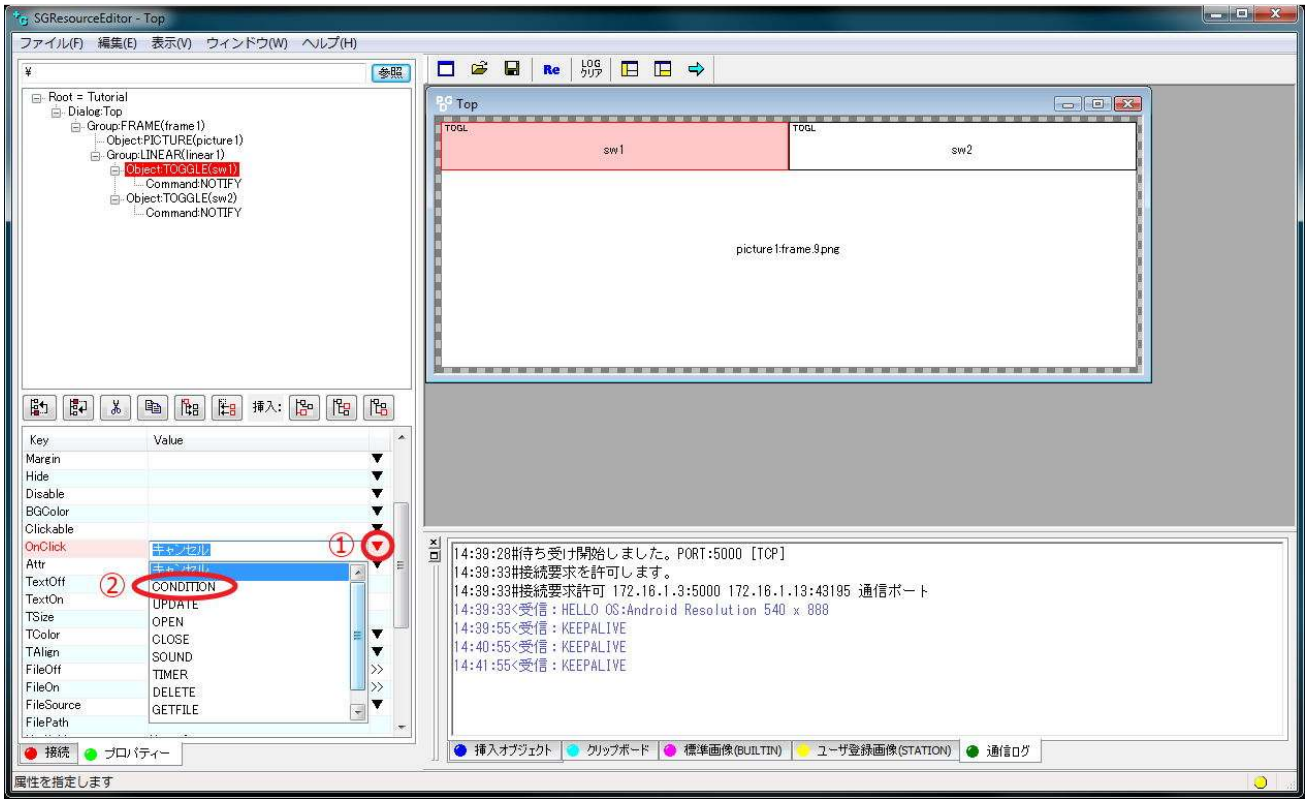
Weight: 1

FileOff: switch_off.9.png

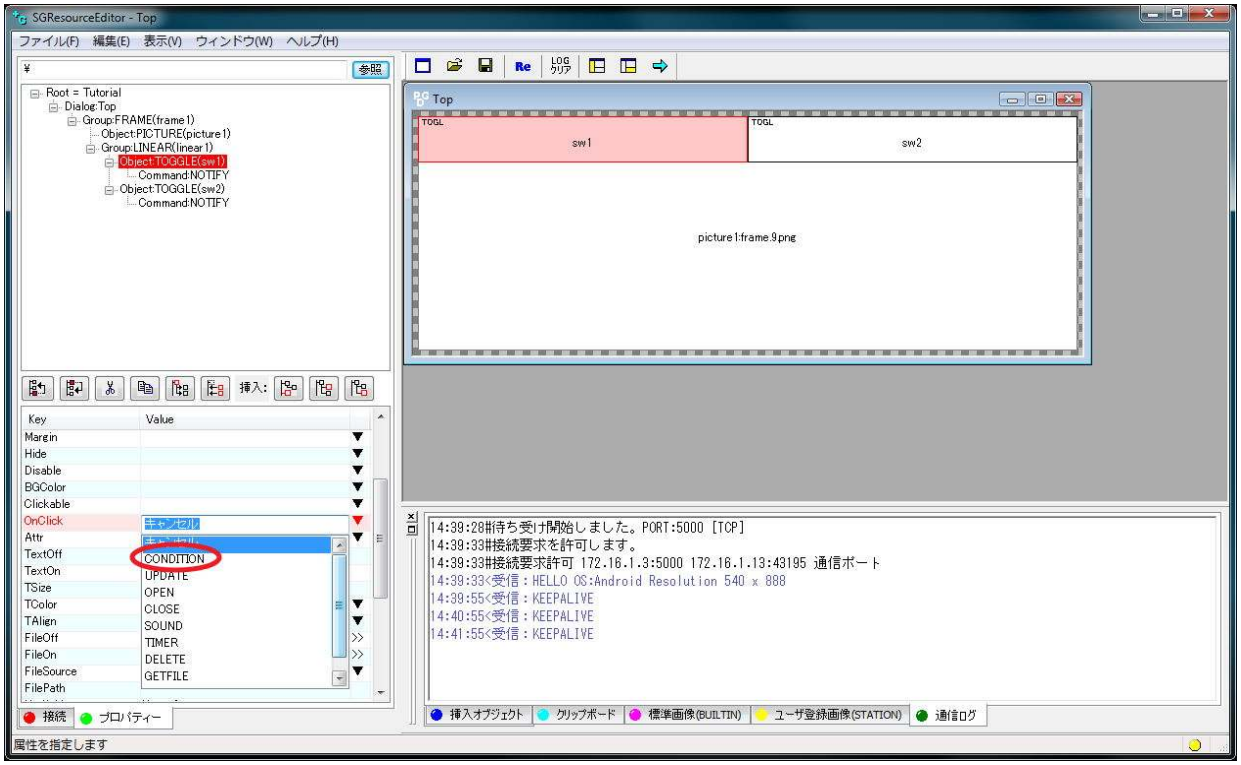
FileOn: switch_on.9.png

3.4. 動作の追加

２つのトグルにタップ時のアクションを設定します。リソースツリーの Dialog/frame1/linear1/sw1 を選択して、OnClick プロパティ行の[▼]をクリックしてください。コマンドのリストが表示されます。



リストから[CONDITION]をクリックしてください。ツリーに CONDITION コマンドが追加されます。



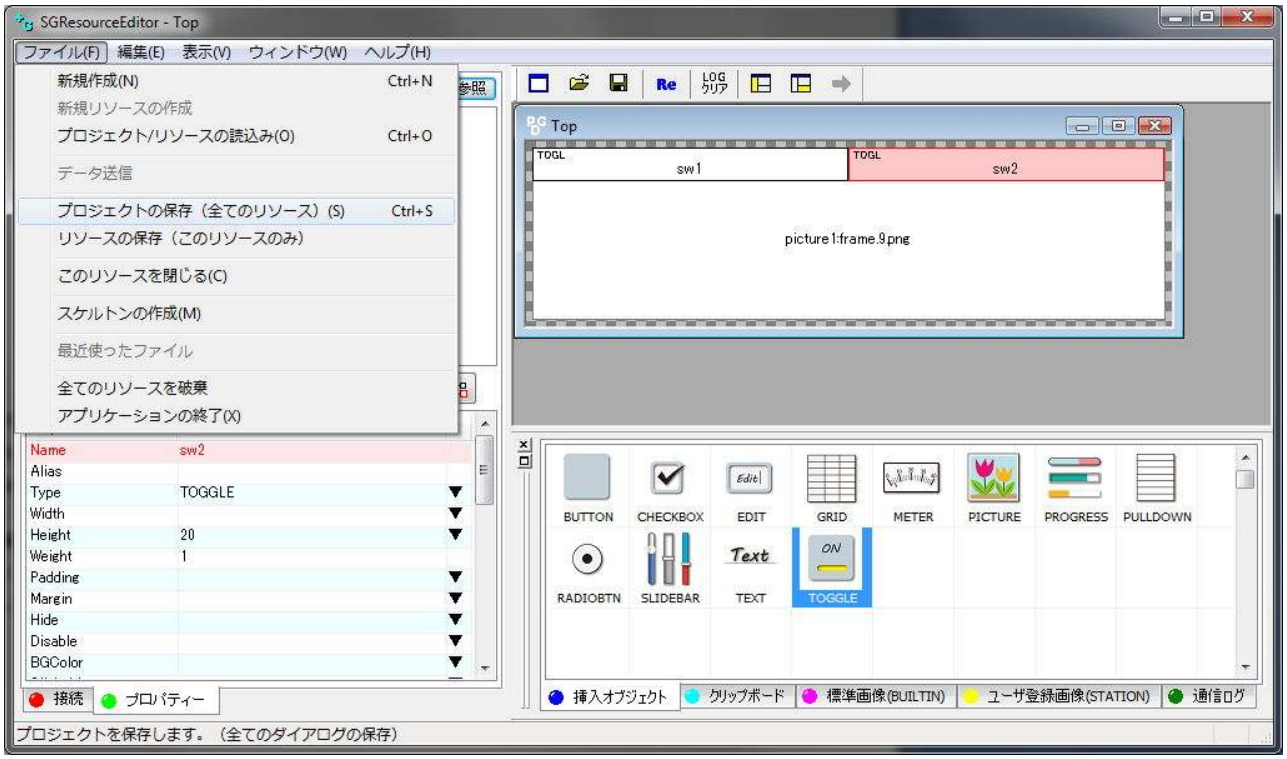
CONDITION コマンドの Name プロパティの値を”sw1”に変更してください。

同様に Dialog/frame1/linear1/sw2 にも CONDITION コマンドを追加してください。Name プロパティ

の値は”sw2”にしてください。

3.5. プロジェクトの保存

作成したリソース（ダイアログ）を保存します。リソースエディタのメニューから[ファイル]→ [プロジェクトの保存（全てのリソース）]を実行してください。ファイル名は”Top”にしてください。

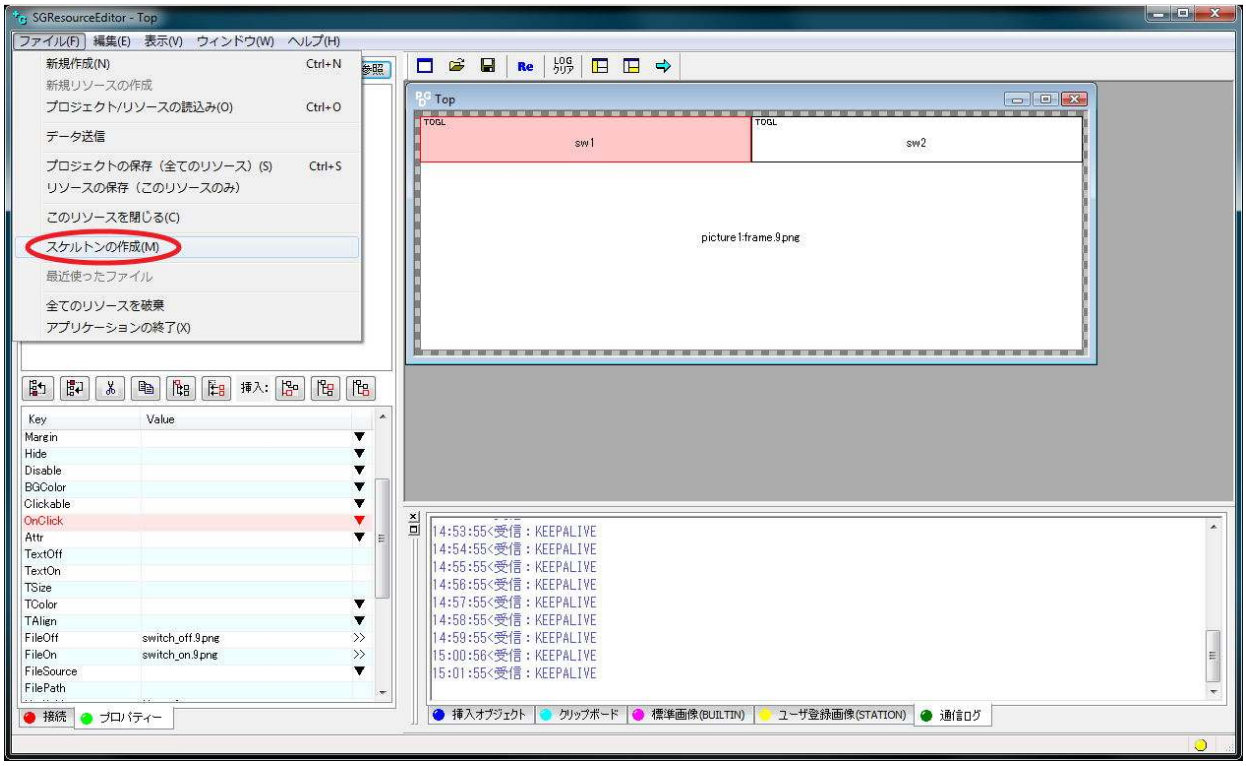


3.6. 画面／動作の確認

リソースエディタと PGSMonitor を接続して、作成したダイアログを送信してください。
ダイアログが表示されたら、いずれかのトグルをタップしてください。リソースエディタの通信ログに「'Command':'CONDITION'」を含む行が表示されれば、動作も正しく設定されています。

3.7. スケルトンの出力

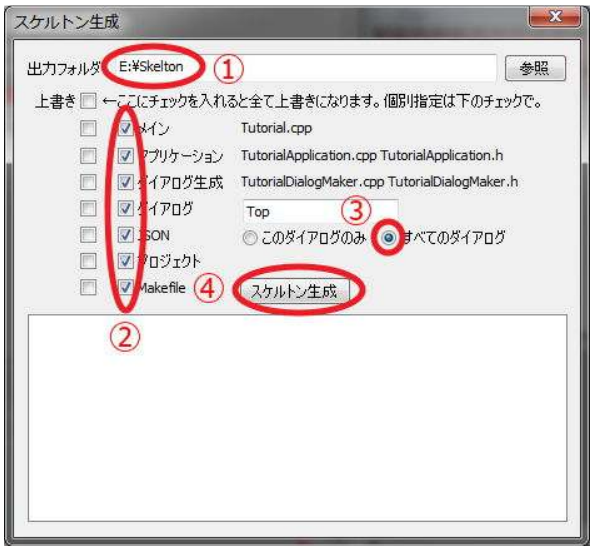
実行ファイルを作るために、ソースコードを出力します。リソースエディタのメニューから[ファイル]→[スケルトンの作成]を実行してください。[スケルトン作成]ダイアログが開きます。



[スケルトン作成]が開いたら、次の操作を実施してください。

1. [出力フォルダ]に任意のフォルダ（ソースフォルダ）を指定する。
2. チェックボックスの右側全てにチェックを点ける。
3. [すべてのダイアログ]ラジオボタンを選択する。

操作が終わったら[スケルトン作成]ボタンをクリックしてください。出力フォルダにソースファイル（.cpp, .h）が作成されます。



3.8. コーディング

出力されたソースは基本構造のみを備えている（スケルトン）ため、これにトグルがタップされた時の動作を付加します。

TutorialDialogTop.h の次の箇所を変更してください。

24 行目付近：

```
void ResourceToDevice();
```

↓

```
void ResourceToDevice(PGJsonObject*);
```

TutorialDialogTop.cpp の次の箇所を変更してください。

3 行目付近：

```
（なし）
```

↓

```
#include <fcntl.h>
```

40 行目付近、OnInit 関数内

```
ResourceToDevice();
```

↓

```
//ResourceToDevice();
```

87 行目付近、OnCondition 関数内

```
ResourceToDevice();
```

↓

```
//ResourceToDevice();
```

97 行目付近、OnUpdate(PGJsonObject*,PGJsonObject*) 関数内

```
PGDialog::OnUpdate(to,from);
```

↓

```
PGDialog::OnUpdate(to,from);
ResourceToDevice(to);
```

155 行目付近、ResourceToDevice() 関数定義

```
ResourceToDevice()
```

↓

```
ResourceToDevice(PGJsonObject* to)
```

157 行目付近、ResourceToDevice() 関数内

```
（なし）
```

↓

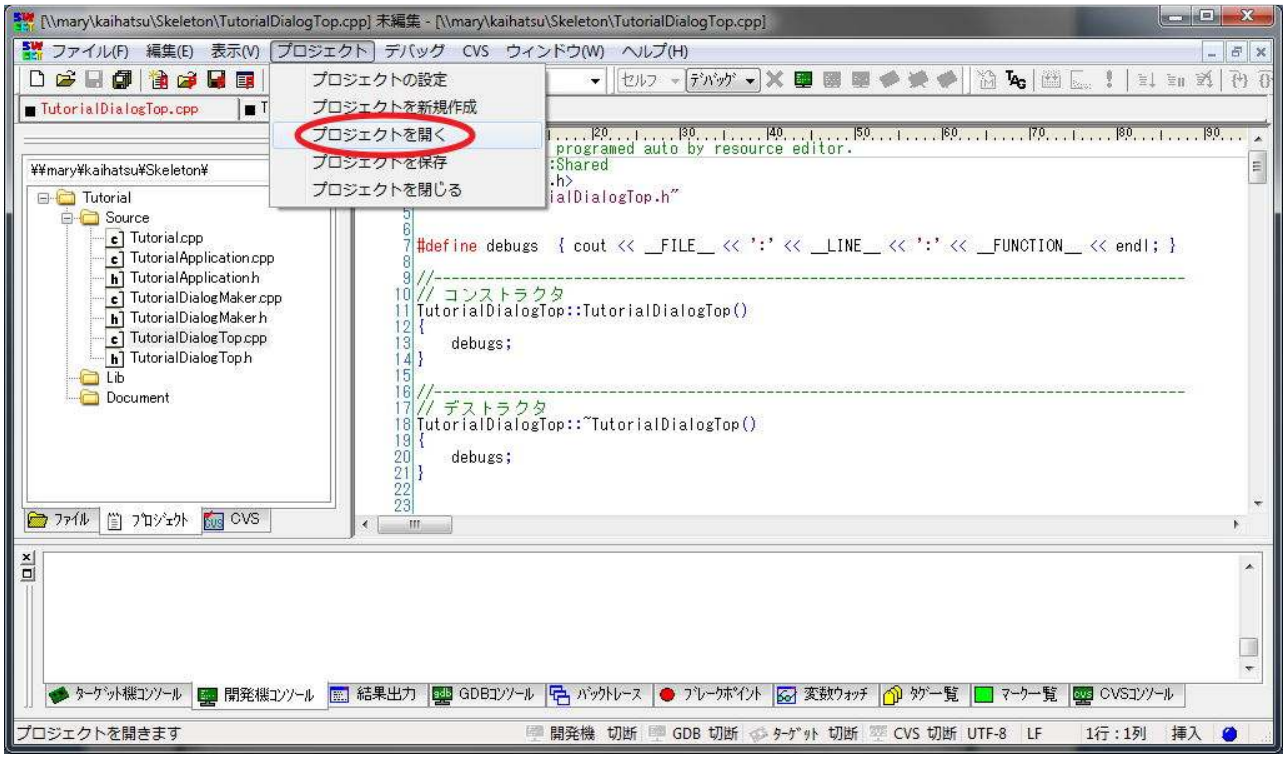
```
SGString name = to->GetValue("Name").GetString();
SGString attr = to->GetValue("Attr").GetString();

cout << "¥n   ResourceToDevice() " << "Name:" << name << " Attr:" << attr << endl;

if(name == "sw1"){
	char c = '?';
	if(attr == "ON"){
		c = '1';
	}
	else{
		c = '0';
	}
	int fd = open("./sw1", O_WRONLY | O_CREAT, 0644);
	if(fd >= 0){
		int l = write(fd, &c, sizeof(c));
		if(l > 0){
```

```
                }
                else{
                        perror("write");
                }
                close(fd);
        }
}
else if(name == "sw2"){
        char c = '?';
        if(attr == "ON"){
                c = '1';
        }
        else{
                c = '0';
        }
        int fd = open("./sw2", O_WRONLY | O_CREAT, 0644);
        if(fd >= 0){
                int l = write(fd, &c, sizeof(c));
                if(l > 0){
                }
                else{
                        perror("write");
                }
                close(fd);
        }
}
```

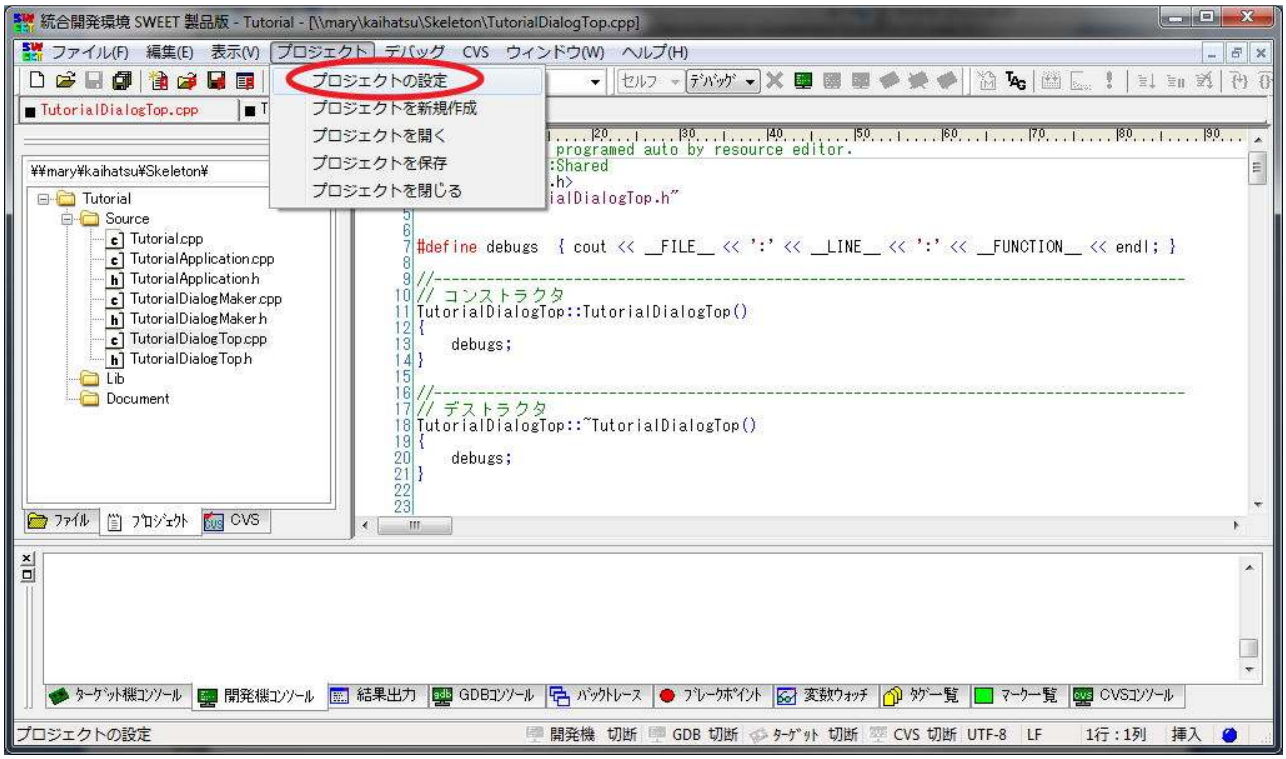
3.9. ビルド

アプリケーションのビルドは Linux で行います。ソースフォルダを Linux PC の任意のディレクトリ（以下、ビルドディレクトリ）に移動してください。

移動後、Windows PC で SWEET を起動して、スケルトンフォルダにある Tutorial.slpro を開いてください。



プロジェクトの設定を、お使いの環境に合わせて変更してください。



[開発機]

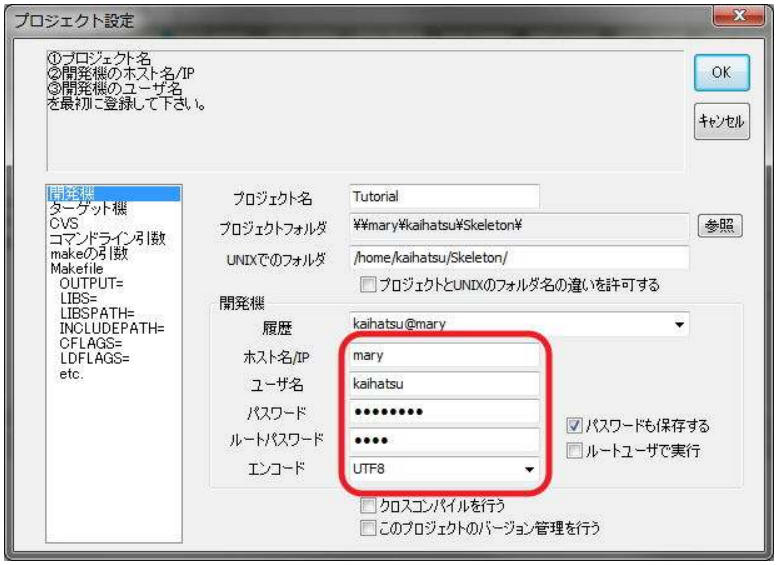
ホスト名/IP：（Linux PC の IP アドレス）

ユーザー名：（Linux PC のログインユーザー名）

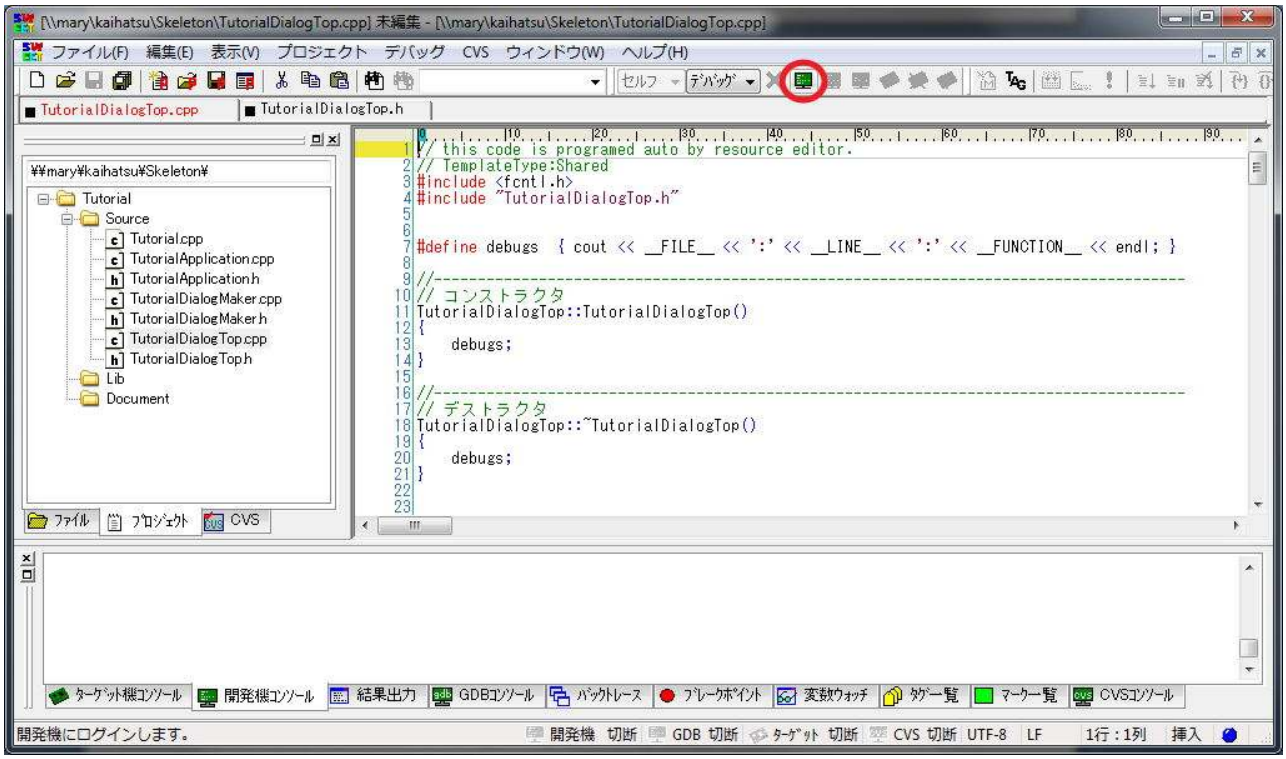
パスワード：（ログインユーザーのパスワード）

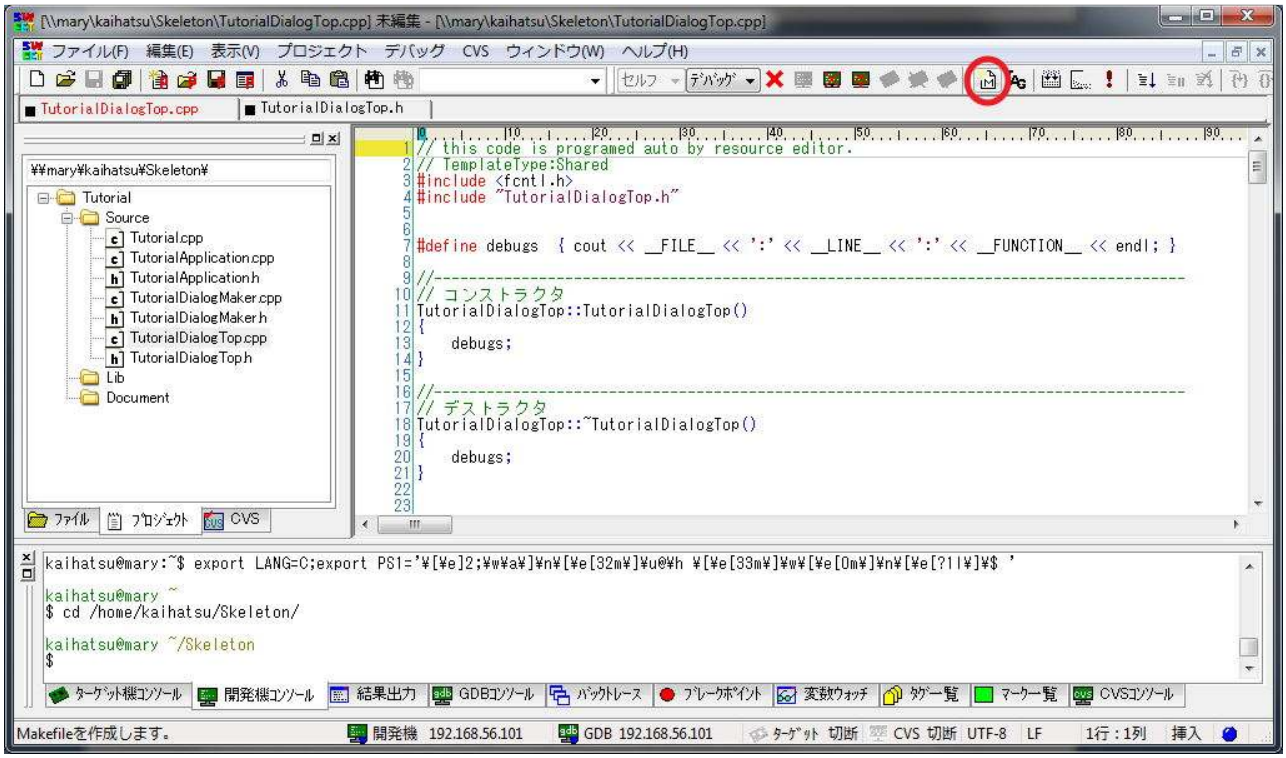
ルートパスワード：（Linux PC の管理者パスワード）

エンコード：（Linux PC の標準文字コード）

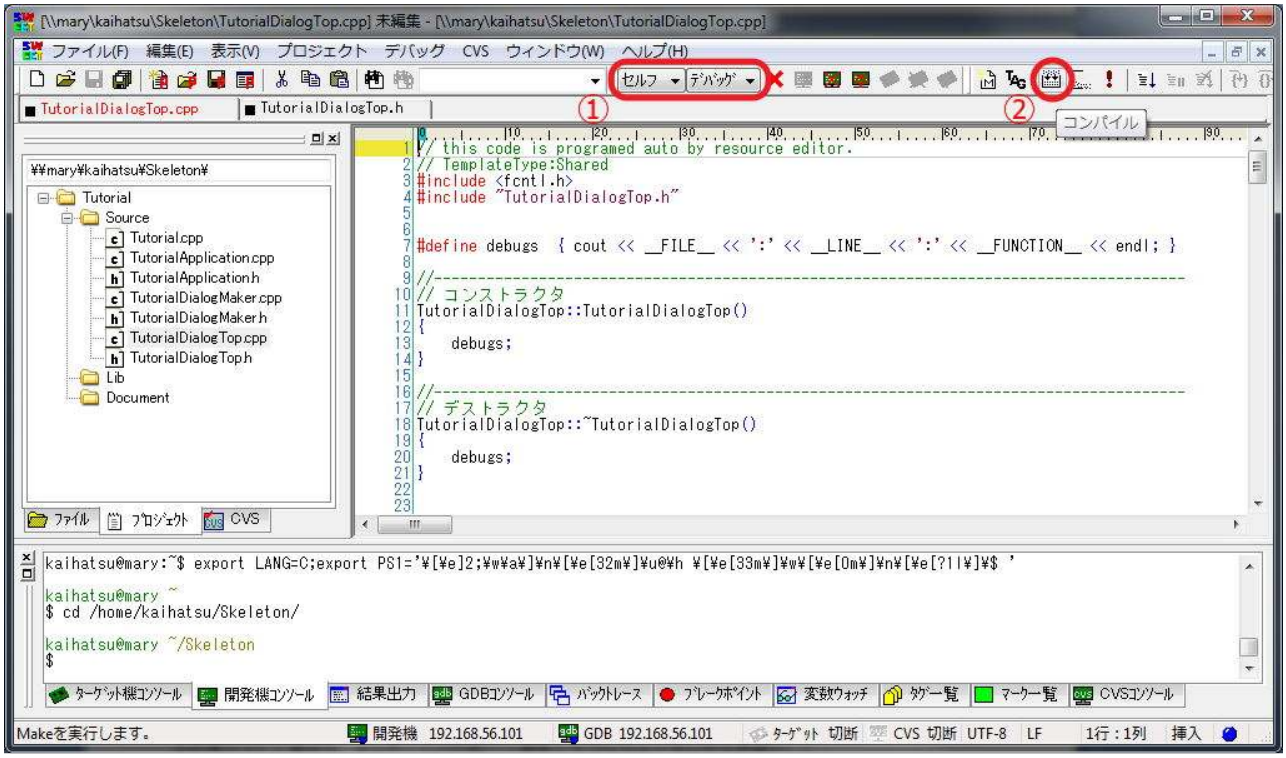


プロジェクトの設定変更が済んだら、開発機にログインして、Makefile 作成を実行してください。

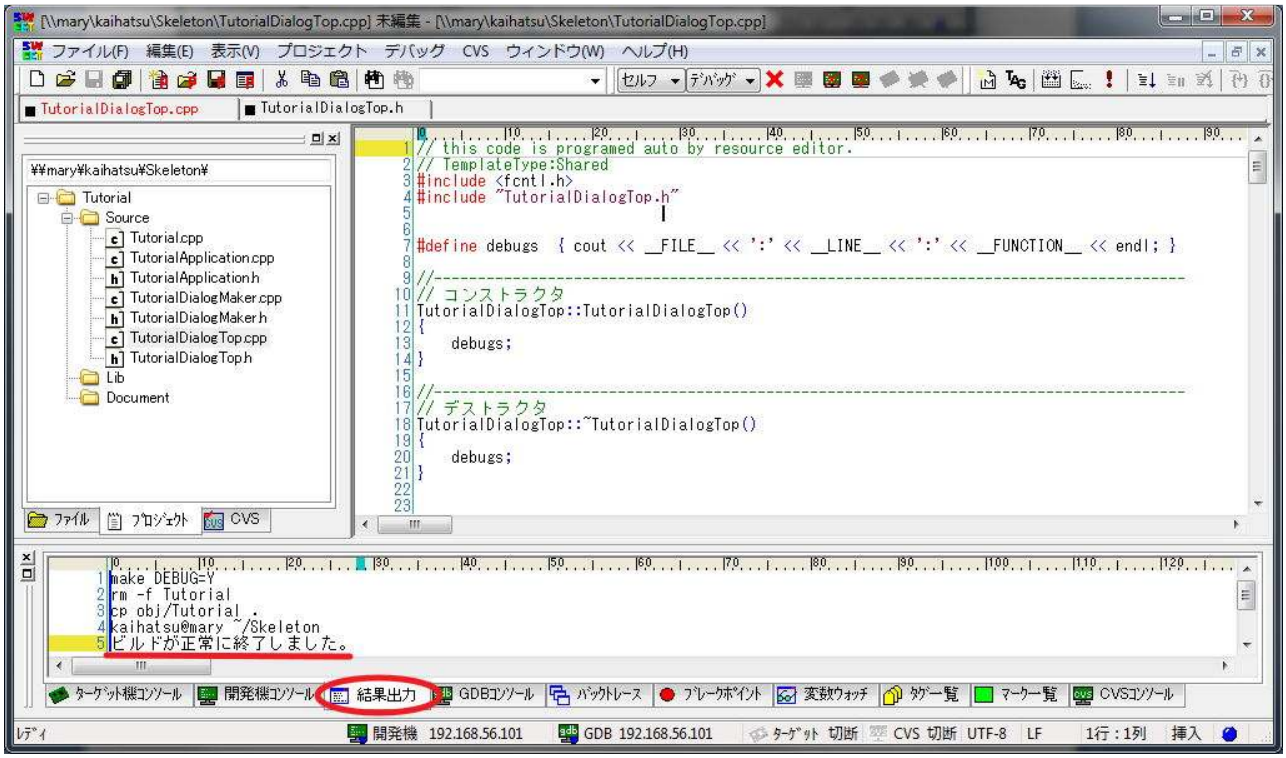




Makefile が作成されたら、ビルドタイプを[セルフ]＆[デバッグ]に変更して、コンパイルを実行してください。

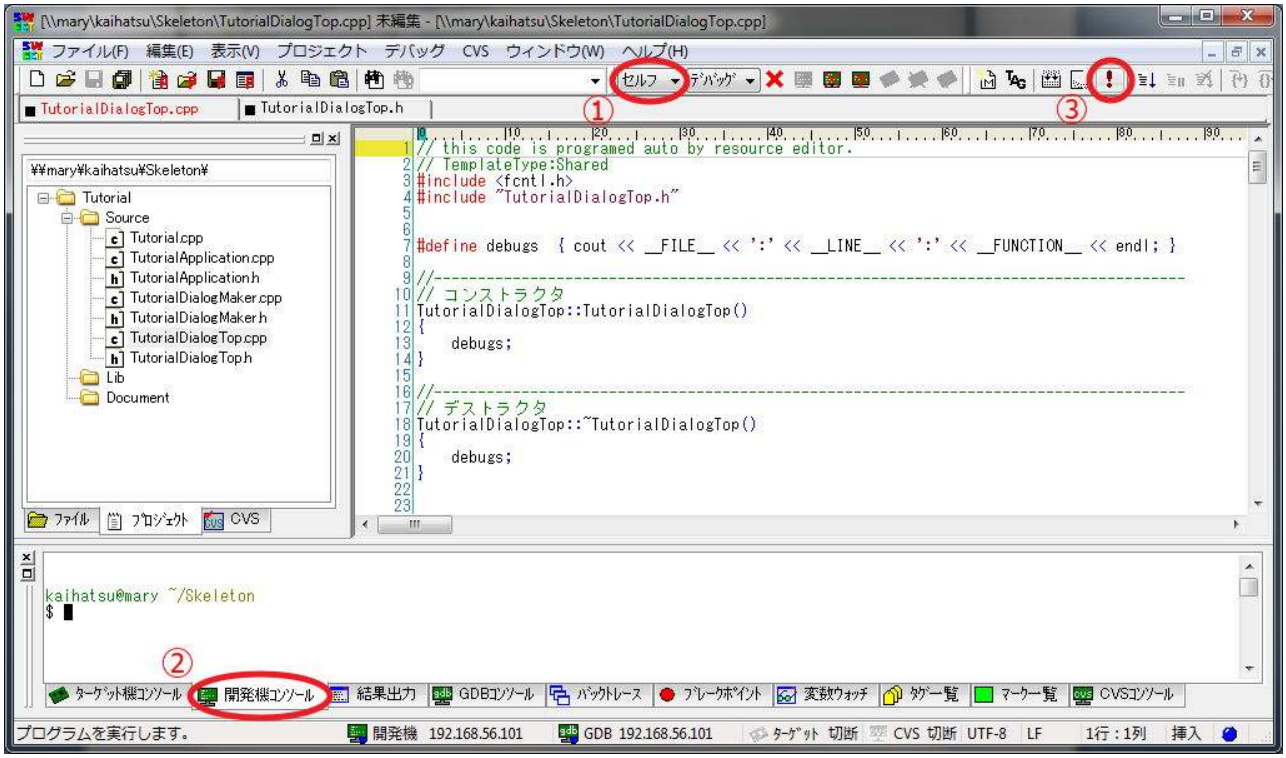


ビルドが成功すると[結果出力]タブに「ビルドが正常に終了しました。」と表示されます。その他のメッセージが表示された場合は、メッセージの内容から適宜トラブルシュートを行ってください。



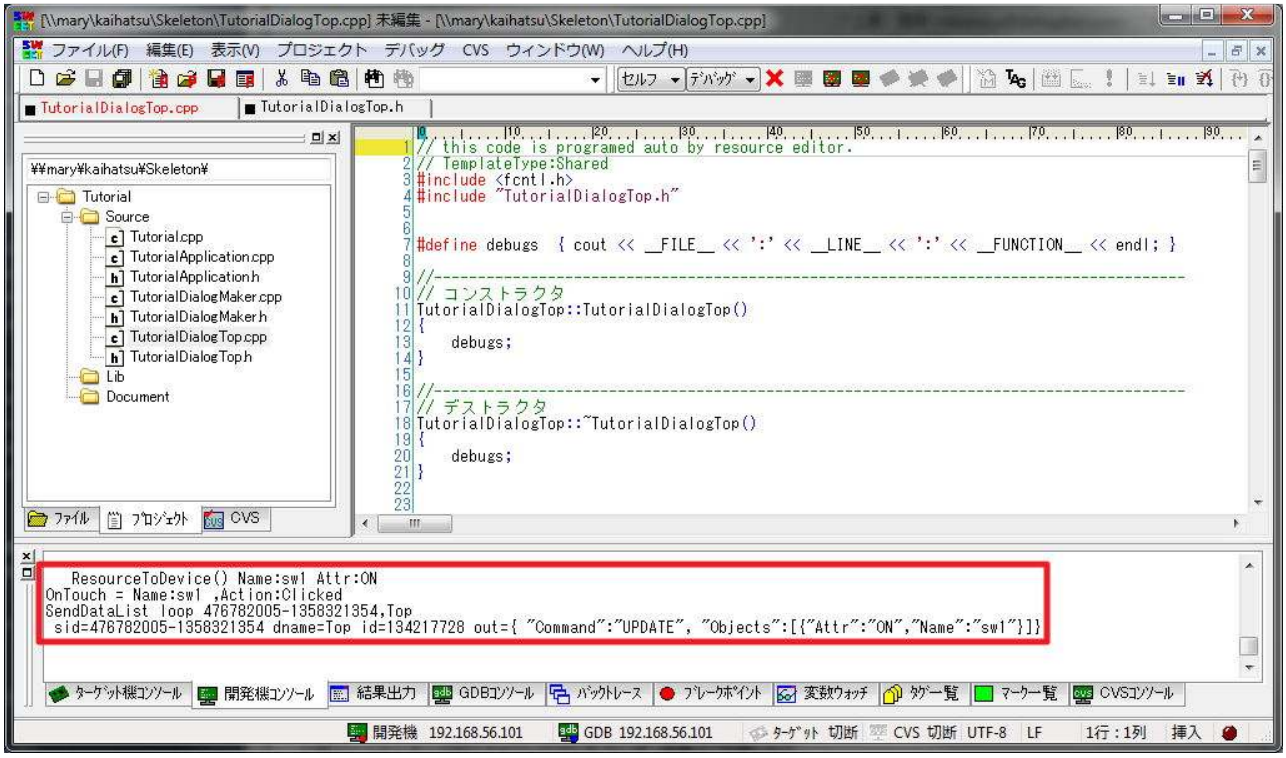
3.10. 動作確認

アプリケーションを実行します。[開発機コンソール]タブ内のシェルで"./sample"を実行してください。

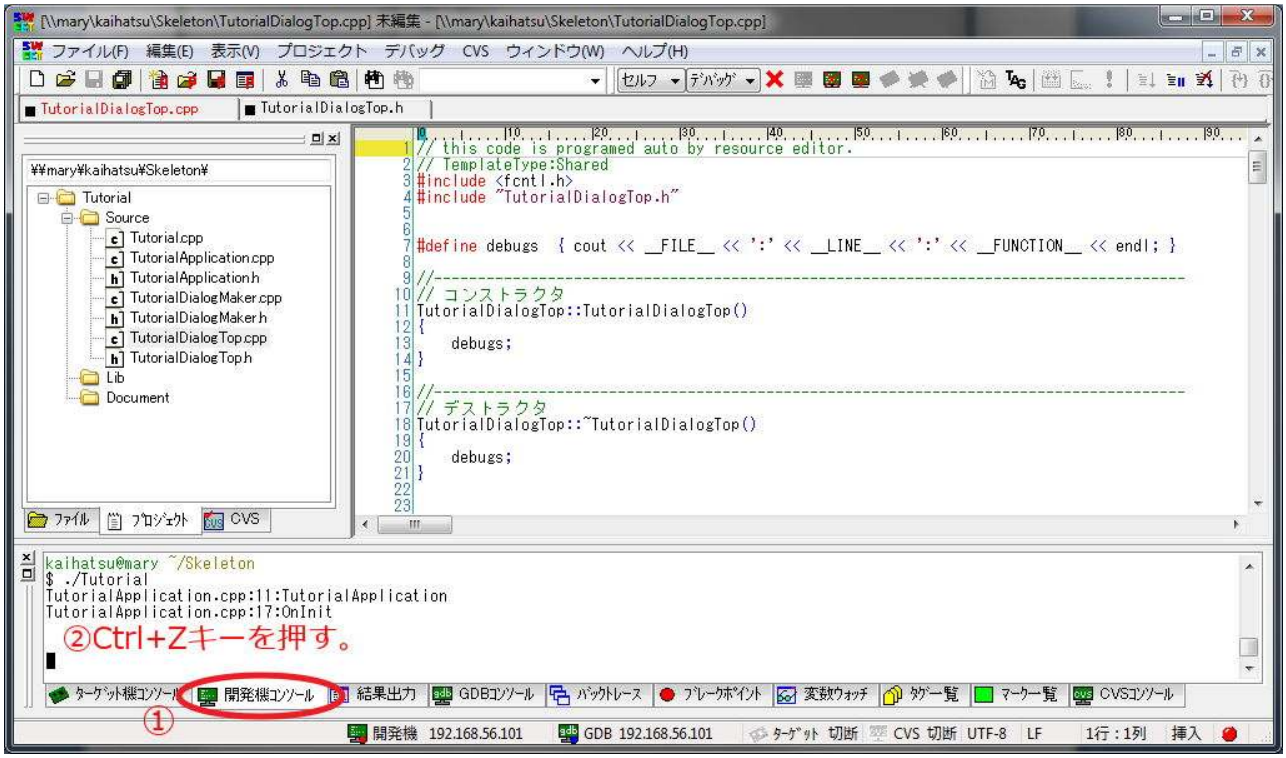


アプリケーション実行後、Android で PGSMonitor を起動して、IP アドレス登録に Linux PC の IP アドレスを設定してください。接続に成功した場合、先程作成したダイアログが表示されます。

左側のトグルをタップしてください。トグルの表示が変わり、開発機コンソールにコマンド受信メッセージが表示されます。



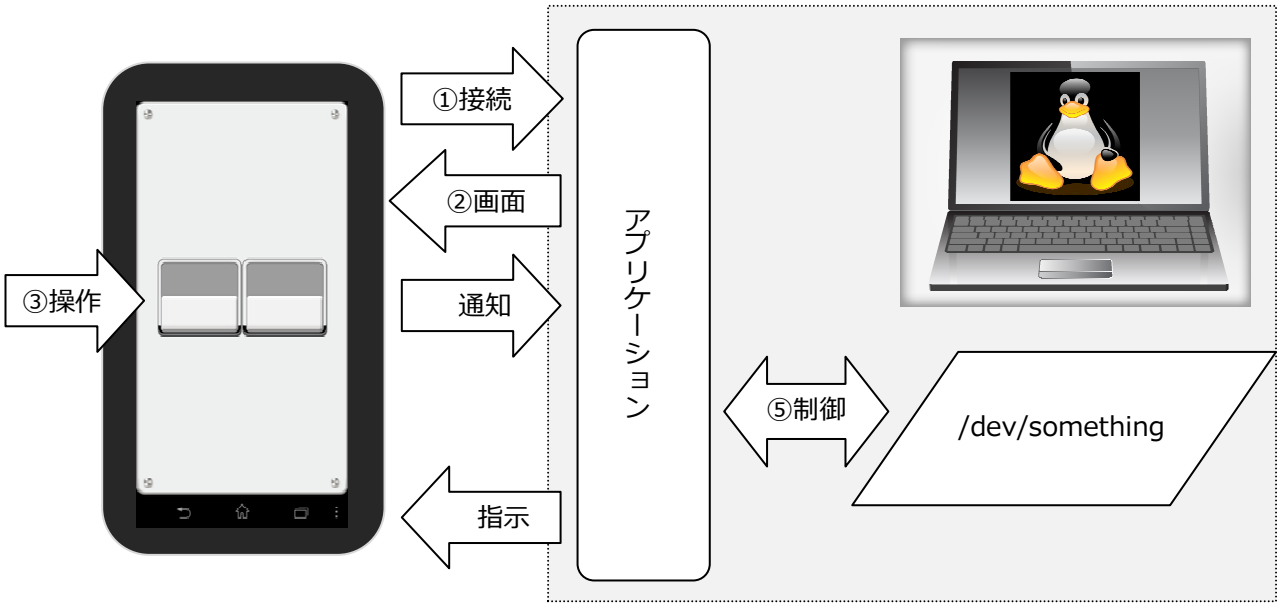
受信メッセージが表示されたら、開発機コンソールで Ctrl+Z を入力してアプリケーションを停止します。



ls コマンドを実行してください。sw1 というファイルが作成されています。

“cat sw1”を実行して中身を見てください。’0’または’1’が表示されます。このファイルはアプリケーションの ResourceToDevice 関数に記述したコードにより sw1 トグルの状態を反映するようになっています。

実際のアプリケーション開発では、このファイルに代わってデバイスファイルを読み書きする事で、周辺機器の制御ができます。



以上