SHARP

ロボホン認定試験チュートリアル
3.アプリ開発

Version 1.0.3

Last update 2018/04/25

SHARP CORPORATION

SHARP

更新履歴

Version	Description	Date
1.0.0	初版作成	2016/09/08
1.0.1	アプリアイコンの作成、適応方法のルールを修正	2017/05/09
1.0.2	テンプレートのインストール方法を修正	2017/06/20
1.0.3	ボタンでおしゃべりアプリのチュートリアル内容を修正	2017/04/25

SHARP

目次

更新履歴 2

目次 3

1. アプリの新規作成 4

1.1 ロボホンアプリ開発を始める準備 4
1.2 テンプレートのインストール 4
1.3 ロボホン用テンプレートファイルを利用して、アプリ開発を始める 5

2. アプリのライフサイクル 9

2.1 ANDROID のライフサイクルとは 9
2.2 ロボホンにおけるライフサイクル 9
ロボホンのライフサイクルにおける *Activity* *9*

3. ボタンでおしゃべりアプリ 11

3.1 はじめに 11
3.2 新しいプロジェクトの作成 11
3.3 レイアウトファイルの設定 12
3.4 ボタンを追加 13
3.5 STRINGS.XML の編集 15
3.6 オリジナル HVML ファイルの追加 16
3.7 MAINACTIVITY.JAVA の編集 17
3.8 SCENARIODEFINITIONS の編集 18
3.9 アプリの実行 19
3.10 ロボホンでの画面の作り方 20
3.11 アプリアイコンの作成、適応方法 23
3.12 タイトルバーの利用 27
3.13 ロボホンへのアプリのインストール・アンインストール方法 28
3.14 ANDROID アプリとロボホンアプリの機能の違い 29

SHARP

1. アプリの新規作成

1.1 ロボホンアプリ開発を始める準備

この章では初めてのロボホンアプリ開発を進めていきます。
ロボホンのアプリ開発では、音声 UI やプロジェクター、アドレス帳など、各種 API を呼び出すためのファイルや、音声起動のための HVML ファイルなど、Android の標準的なテンプレートとは別に複数のファイルを用意し、読み込み処理を行う必要があります。
手作業で読み込み処理を行うことも出来ますが、毎回決まったファイルの操作が必要なため、ロボホン SDK では Android Studio の専用テンプレートが用意されています。

ロボホン用の Android Studio テンプレートをインストールすると、プリインストールされたテンプレートと同じように、１クリックでロボホンアプリ開発の準備を行うことが出来ます。

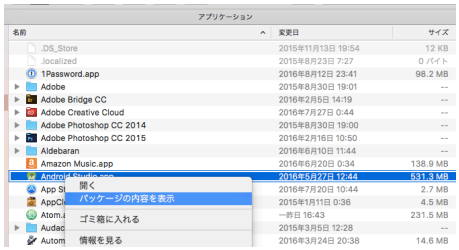
1.2 テンプレートのインストール

Android Studio にロボホン用のテンプレートファイルをインストールします。

注意
現時点では、Android Studio へオリジナルのテンプレートをインストールすると、Android Studio 本体のアップデートが出来なくなってしまう事があります。 ロボホン用テンプレートファイルを追加する前に、Android Studio 内の Plugin フォルダをバックアップしてください。 テンプレートをインストールする際に、Android Studio が既に立ち上がっている場合は、一旦終了させてから作業を行ってください。

[インストール方法]

1. Android Studioのインストールディレクトリ内の [plugins] ディレクトリをバックアップします。
   （pluginsディレクトリ自体をzip圧縮するなどして、ソフトウェアとは別の場所で保存してください。）



（macOSでのアプリケーションディレクトリの表示例）

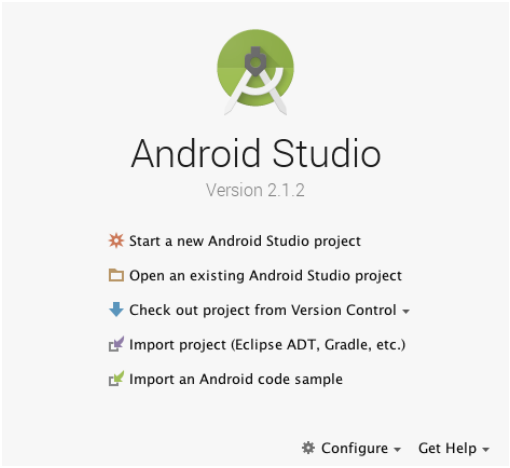
2. ロボホンSDKに付属する [robohon_templete.zip] を解凍します。
3. [plugins > android > lib > templates > activities] ディレクトリを開きます。
4. 手順②で解凍した、[robohon_template] 内の [activities] ディレクトリ直下にある、[RobohonActivity] ディレクトリを手順③で開いたディレクトリにコピーします。

以上でAndroid Studioへのロボホンテンプレートのインストールは完了です。

アンインストールを行う場合は、Android Studio内の [Plugins] ディレクトリを、バックアップした [plugins] ディレクトリを解凍した物に置き換えます。

1.3 ロボホン用テンプレートファイルを利用して、アプリ開発を始める

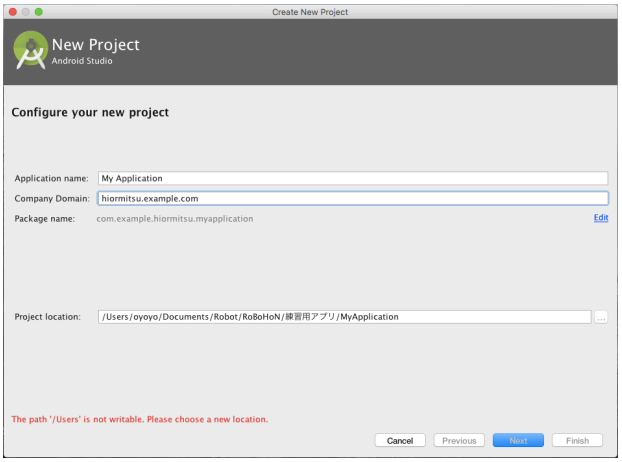
実際にアプリケーションの作成を行います。
Android Studioを起動して、ウェルカム画面が表示された状態にします。



(Android Studioウェルカム画面)

ウェルカム画面のメニューから、[Start a New Android Studio Project] を選択します。

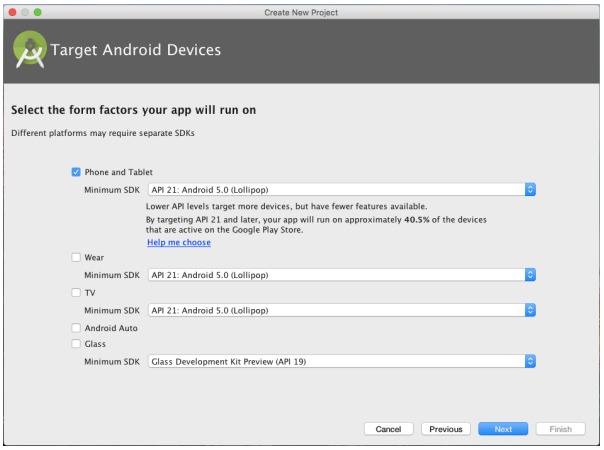
[Configure your new project] 画面が表示されます。



(Configure your new project画面)

[Application Name] [Company Domain] [Project Location] を任意の値へ変更し、[Next]ボタンをクリックします。(今回は練習用アプリですので、特に項目の内容は変更せず、そのまま続けても問題ありません。)

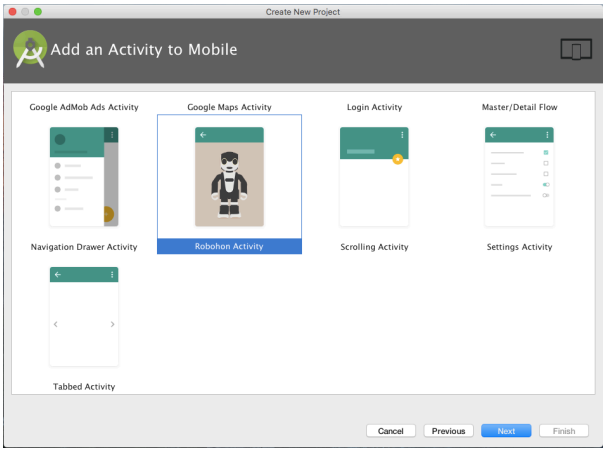
[Select the form factors your app will run on] 画面が表示されます。 この画面では対象とするAndroidデバイスを選択します。
ロボホンはAndroidスマートフォンがベースとなっているため、一番上の [Phone and Tablet] がチェックされていることを確認し、右側のセレクトメニューにて、[API: 21 Android 5.0 (Lollipop)] を選択し [Next] ボタンをクリックします。



(Select the form factors your app will run on画面)

[Add an Activity to Mobile] 画面が表示されます。

この画面では最初に使用する、画面アクティビティのテンプレートを選択します。
先ほど追加したロボホン用のテンプレートが表示されているはずですので、[Robohon Activity] を選択します。

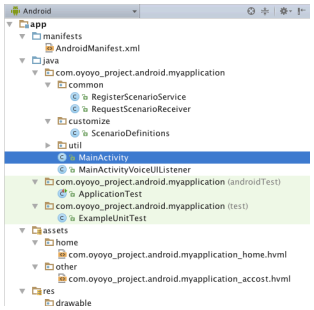


(Add an Activity to Mobile画面)

[Customize the Activity] 画面が表示されます。
この画面ではActivityの名前を任意の名前に変更することが出来ます。
今回は練習用のアプリですので、そのまま [Finish]ボタンを押して次へ進んでください。

[Finish]ボタンをクリックすると、プロジェクトの作成が始まります。
編集画面が表示されるまで少し時間がかかりますが、焦らないようにしてください。

編集画面が起動したら、左側のツリービューを確認してみてください。
様々なファイルが予め作成され、登録されていることが確認できると思います。



(Android Studio ツリービュー画面)

それでは、ロボホンで実行してみましょう。
Android Studioの上部メニュー内の再生ボタンをクリックします。
ロボホン上でサンプルアプリが起動すれば成功です。

(ロボホン上でアプリを実行した際の画面)

SHARP

2. アプリのライフサイクル

2.1 Android のライフサイクルとは

Androidアプリでは画面インターフェイスの単位を [Activity] と呼びます。
一つのAndroidアプリは複数の [Activity] を組み合わせた形で実装されており、Activity同士のやり取りの事を [Intent] と呼びます。 Androidでは、限られたハードウェアリソースを有効に活用するため、アプリ内の制御をActivityという、１画面ごとに管理し、Androidシステムが使用していない [Activity] を停止させたり、メモリを開放したりすることで、ハードウェアリソースの節約を行っています。
そのため、アプリ開発者の意図とは反して、アプリが終了してしまう事があります。
Androidでは、上記のActivity制御をコントロール出来るように、Activityの開始や終了等のイベントをアプリ側が簡単に取得できるようになっています。この仕組みをライフサイクルと呼んでいます。

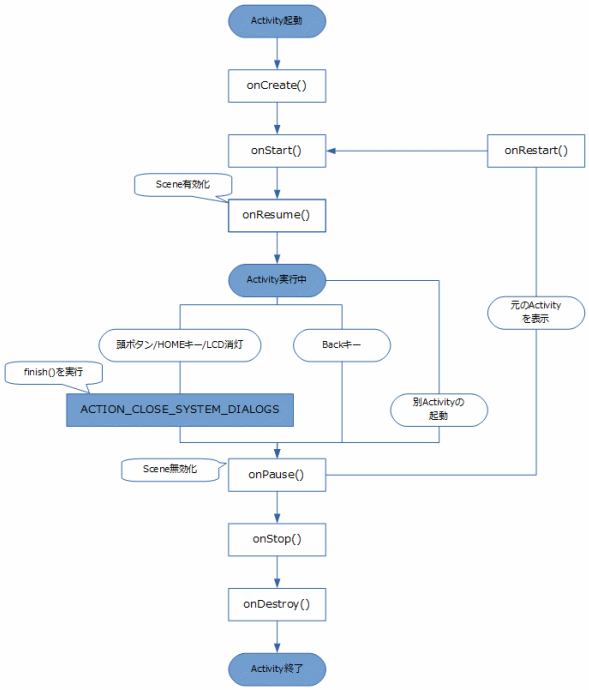
2.2 ロボホンにおけるライフサイクル

ロボホンのライフサイクルは、大きな流れは一般的なAndroid端末と変わりありませんが、ロボホンが音声での操作をメインとしているため、アプリを作成する上で、いくつか追加するべき実装や、制限があります。

ロボホンのライフサイクルにおける Activity
ロボホンのアプリを作成する際、アプリは必ず、起動時に背面 LCD を使用してください。
ロボホンにおいて、画面の存在しないアプリや、バックグラウンドで作動するアプリの作成は禁止されています。
また、ホーム用シナリオは、シンプルなものにしてください。複雑なシナリオを実行したい場合は、必ずアプリケーションが起動してから、その他のシナリオで行うようにしてください。

また、ロボホンは Android スマホにある Recent キーが存在しません。ユーザ操作により複数のアプリを切り替えたり、ホーム画面から再開することができないため、Activity が非表示になった際には可能な限り速やかに終了してください。

以下に記載する図のように、ホームキーイベント(ACTION_CLOSE_SYSTEM_DIALOGS)を受ける Broadcast レシーバークラスを実装し、同クラスの onReceive()内で finish()を実行し、自 Activity を終了してください。Back キーはアプリが特別な実装をしない限り Android 標準同様に onDestroy()まで実行されるため、特に考慮する必要はありません。



ロボホンにおけるライフサイクル

SHARP

3. ボタンでおしゃべりアプリ

3.1 はじめに

新規プロジェクトを作成し、ロボホン用テンプレートに、オリジナルのボタン、シナリオファイル(HVML)などを追加して、ロボホンが挨拶をするアプリを作ってみましょう。
このアプリを作成しながら、以下の項目を合わせてご説明致します。

[このアプリで確認する項目]

- ロボホンでの画面の作り方
  (画面の解像度、レイアウトの取り方、レイアウトファイルの設定方法)
- 背中の画面とプロジェクターの違い
  (画面のスケーリング、ポートレート・ランドスケープの設定)
- アプリアイコンの作成、適応方法
  (アイコンのサイズとレギュレーション)
- ロボホンへのアプリのインストール・アンインストール方法
  (作成したアプリの転送方法と削除方法)

3.2 新しいプロジェクトの作成

それではアプリの作成を始めましょう。
前項目の「ロボホン用テンプレートファイルを利用して、アプリ開発を始める」を参考に新規プロジェクトを作成してください。

新しいプロジェクトで設定する値

設定項目	設定する値
Application Name	TapOnTalk

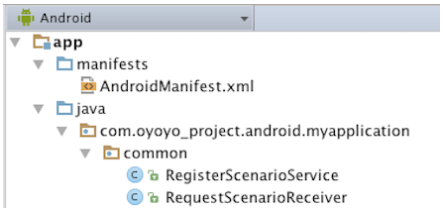
※上記の項目以外は任意にご指定ください。

SHARP

3.3　レイアウトファイルの設定

Android Studioが起動したら、レイアウトファイルの設定を行います。
レイアウトファイルの画面サイズ、APIレベルをロボホン開発用に変更します。

[設定方法]
左側に表示されているツリービューの表示モードをAndroidに設定します。



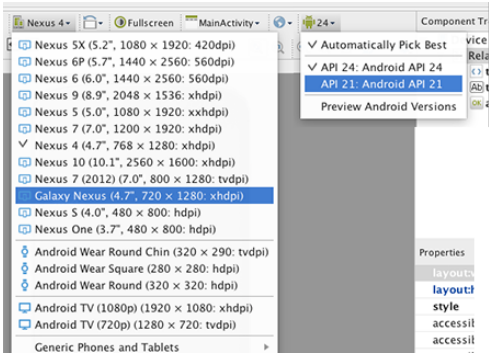
（Androidモードのツリービュー画面）

ツリービューより [res > layout > activity_main.xml] 項目を開きます。

表示されているAndroid端末のシミュレーターを変更します。

シミュレーターの選択プルダウンメニューから、[Galaxy Nexus] を選択します。

同様にAPI Levelも［21］に設定します。
（詳しくは認定試験チュートリアル資料2番目を御覧ください。）

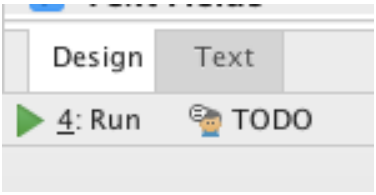


（レイアウトファイルの設定画面）

SHARP

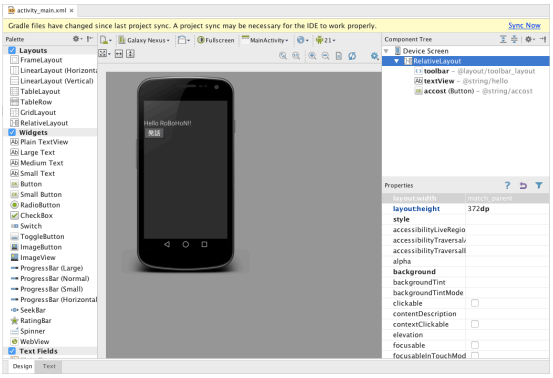
3.4　ボタンを追加

続いて、ロボホンを喋らせるためのボタンを追加していきます。
レイアウトファイルの下部に表示されている [Design / Text] タブをクリックし、Design編集モードに切り替えます。



(Design/Text切り替えタブ画面)

Text編集モードに切り替えると、ロボホンの背中の画面に表示させるパーツがGUI形式で表示されます。
左側の部品一覧よりButton要素をドラックしてサンプルとして配置されている「発話」ボタンの下に配置します。



(activity_main.xmlの編集画面)

SHARP

発話ボタンを設置した後、[Design / Text] タブをクリックし、Text編集モードへ切り替え、サンプルとして設定されている部品、<include />、<TextView />、<Button />の３つのタグ下に、新たに、<Button>タグが追加されていることを確認します。

追加された<Button />タグ内のandroid:text=”New Button"をandroid:text="@string/greeting"へ書き換えます。

Text モードで追加された<button />コード

```
<Button
    android:id="@+id/talk"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_below="@+id/accost"
    android:text="@string/greeting"
    android:textSize="27sp" />
```

編集するファイル: [res > layout > activity_main.xml]

SHARP

3.5 strings.xml の編集

次に<Button>部品に表示するテキストの定義を行います。
Androidではアプリの国際化を容易にするため、アプリ内で使用する言葉を定義するXMLファイル、strings.xmlが用意されています。
アプリ内で使用する言葉はstrings.xmlで定義し、言葉を使いたい場所で定義した言葉のIDを指定することで呼び出すことが出来ます。

左側のツリービューより [res > values > strings.xml] を選択し開きます。
Stringsの設定項目に、追加する<Button>用のテキストを追加します。
<string>の最後の要素の下に、以下の<string>要素を追加します。

追加するコード

```
<string name="greeting">こんにちは</string>
```

編集するファイル: [res > values> settings.xml]

SHARP

3.6　オリジナル HVML ファイルの追加

ボタンを押した際に実行するHVMLのシナリオファイルを作成します。
[assets > other] ディレクトリをツリービューに表示します。
このディレクトリ内に、会話用のシナリオファイルを追加します。
以下の命名規則に沿って、追加してください。

HVMLファイルの命名規則

「パッケージ名(.を_に変更)＋_(アンダーバー)＋シナリオ名＋.hvml」

例えば、パッケージ名が「jp.co.sharp.robohon.example.tapontalk」で、シナリオ名が「scenario」の場合は

「jp_co_sharp_robohon_example_scenario_tapontalk.hvml」となります。

今回はシナリオ名を [talk] としますので、上記のパッケージ名の場合は

「jp_co_sharp_robohon_example_tapontalk_talk.hvml」となります。

ファイルの作成が完了したら、HVMLファイルを編集します。

ロボホンの背中の画面に表示されているボタンをタップすると「こんにちは。これからよろしくね。」と喋るHVMLシナリオファイルを作成します。

以下の内容を作成したHVMLシナリオファイルに書き込んでください。

ファイルの変更例

```
<?xml version="1.0" ?>
<hvml version="2.0">
    <head>
        <producer>jp.co.sharp.robohon.example.tapontalk</producer>
        <!-- TODO このシナリオの説明文を入力してください(プログラムに影響はありません) -->
        <description>挨拶ボタンを押した時の処理</description>
        <scene value="jp.co.sharp.robohon.example.tapontalk.talk" />
        <version value="1.0"/>
        <accost priority="75" topic_id="0001"
word="jp.co.sharp.robohon.example.tapontalk.talk.greet" />
    </head>
    <body>
        <topic id="0001" listen="false">
            <action index="1"><speech>こんにちは。これからよろしくね。</speech></action>
        </topic>
    </body>
</hvml>
```

※パッケージ名「jp.co.sharp.robohon.example」となっている箇所はご自分の環境に合わせて書き換えてください。

編集するファイル: [assets> home > パッケージ名+_home.hvml]

3.7 MainActivity.java の編集

MainActivity.javaファイルを編集します。
今回のアプリでは、ボタンをタップすると、「こんにちは。これからよろしくね。」とロボホンが喋るアプリですので、先ほどレイアウトファイルの最後に追加した<Button>部品にイベントを追加していきます。
ツリービューより [java > jp.co.shap.robohon.example(パッケージ名) > util > MainActivity] を開きます。
onCreateメソッドの最後の行に、新しく追加したボタンのイベントを設定していきます。

追加するコード

```
//あいさつボタン
Button ButtonGreet = (Button) findViewById(R.id.talk);
ButtonGreet.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (mVoiceUIManager != null) {
            VoiceUIVariableListHelper helper = new
VoiceUIVariableListHelper().addAccost(ScenarioDefinitions.ACC_GREET);
            VoiceUIManagerUtil.updateAppInfo(mVoiceUIManager, helper.getVariableList(),
true);
        }
    }
});
```

編集するファイル: [java > パッケージ名 > MainActivity]

シナリオファイルを追加したので、シーンの有効化を行います。シナリオファイル内に定義してあるシーンを有効にすることで、イベントなどが動作するようになります。onResume()の最後の行に追加してください。

追加するコード

```
//Scene 有効化.
VoiceUIManagerUtil.enableScene(mVoiceUIManager, ScenarioDefinitions.SCENE_COMMON);
VoiceUIManagerUtil.enableScene(mVoiceUIManager, ScenarioDefinitions.SCENE01);
VoiceUIManagerUtil.enableScene(mVoiceUIManager, ScenarioDefinitions.SCENE_TALK);
```

編集するファイル: [java > パッケージ名 > MainActivity]

SHARP

3.8 ScenarioDefinitions の編集

ScenarioDefinitionsファイルを編集します。
「こんにちは。これからよろしくね。」と喋るシナリオを呼び出すための定数を定義します。
ツリービューより [java > jp.co.shap.robohon.example(パッケージ名) > customize > ScenarioDefinitions] を開きます。こんにちは発行実行のaccost名の上に、新しく追加したaccost名の定義を追加します。

追加するコード

```
/**
 * accost 名：挨拶用ボタン実行.
 */
public static final String ACC_GREET = ScenarioDefinitions.PACKAGE + ".talk.0001";
```

編集するファイル: [java > パッケージ名 > MainActivity]

同様にシーン名の定義を追加します。

追加するコード

```
/**
 * scene 名：挨拶シーン
 */
public static final String SCENE_TALK = PACKAGE + ".talk";
```

編集するファイル: [java > パッケージ名 > MainActivity]

SHARP

3.9　アプリの実行

それでは実際にアプリを実行してみましょう。
今回追加した「こんにちは」ボタンをタップしたら、ロボホンが、「こんにちは。これからよろしくね。」と喋れば成功です。

（ロボホン 背中の画面）

SHARP

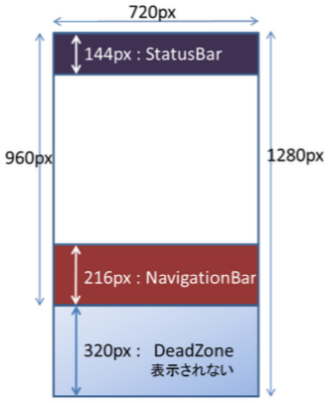
3.10　ロボホンでの画面の作り方

ロボホンには背中の画面と、プロジェクターの画面の２種類の画面レイアウトが存在します。注意点等を記載します。

背中の画面（背面LCD)

ロボホンの背中の画面（背面LCD)はアプリの起動や文字の入力、設定等、音声UIを補助する形で使用します。
背面LCDは通常のAndroidと異なり、画面の向きがポートレート固定(縦向き)となっています。
画面自体のサイズ(解像度)は720×960(px)ですが、上部にStatusBar、下部にNavigationBarが存在し、実際にアプリ表示に使用できるサイズは小さくなりますので、注意してください。
アプリ作成時の解像度は720×960(px)ですが、実際のハードウェア上の解像度は240×320(px)なので、ロボホン実機の背面LCDでは、1/3に縮小されて表示されます。dpi値の密度はxhdpiとなります。

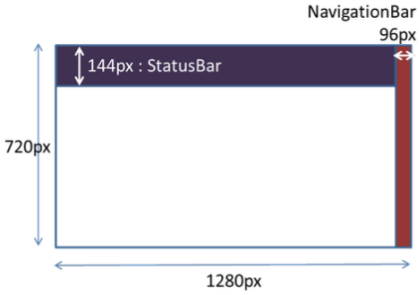
また、アプリを作成する際、レイアウトファイルで指定する端末GalaxyNexusは720×960(px)ですが、実際のロボホンは画面の下部320pxが表示されない領域（DeadZone）となっています。DeadZone部分にレイアウト部品を置いても、ロボホン実機では表示されませんので注意してください。



(背面LCD レイアウト図)

頭部のプロジェクター

ロボロボホン頭部のプロジェクターは写真や動画、アプリ画面の投影などに使用できます。
プロジェクターを起動する度に、ロボホン所有者の顔認証確認、もしくはロボナンバーの入力が必要です。
プロジェクター投影中は、背中の画面を操作することにより、プロジェクター画面のカーソルを操作することが出来ます。これらの昨日はSDKに内蔵されており、変更はできません。
画面自体のサイズ(解像度)は1280×720(px)で、画面の向きはポートレート固定(縦画面)となっています。



(頭部プロジェクター レイアウト図)

背面LCD・プロジェクター 共通の注意点

作成するアプリケーション内ではアクションバー・ステータスバーを非表示にしてください。
[Styles.css] にて [Theme.DeviceDefault.NoActionBar.Fullscreen] を指定し、[AndroidManifest.xml] 内で適応することで、作成するアプリケーション内の全ての画面において、アクションバー／ステータスバーを非表示に設定します。

ロボホンでは、Darkテーマの仕様を推奨しているため、特別な理由がない限り、AndroidのテーマはDarkテーマを使用するようにしてください。

SHARP

部品パーツのサイズについて

ロボホンではAndroidの標準的なレイアウト部品を使用することが出来ますが、一部の部品は、通常のAndroid端末で使用した場合と異なるサイズが指定されております。

以下のサイズリストを参照しながらアプリ開発を進めてください。

	背面 LCD(縦)	プロジェクター(横)	Android 標準
タイトルバー			
高さ(アプリ指定)	81dp	36dp	縦:56dp 横:48dp
タイトル文字サイズ	27dp	14dp	縦:20dp 横:14dp
ボタン領域(アプリ指定)	63×63(dp)	28×28(dp)	-
アイコンサイズ(アプリ指定)	72×72(px)	32×32(px)	-
アラートダイアログ			
タイトル文字サイズ	27sp	標準と同じ	20sp
メッセージ文字サイズ	27sp	標準と同じ	16sp
ボタン文字サイズ	24sp	標準と同じ	14sp
スイッチ			
幅	186px	標準と同じ	94px
高さ	108px	標準と同じ	54px
シークバー			
高さ	108px	標準と同じ	64px
EditText			
文字サイズ	27sp	標準と同じ	18sp
チェックボックス			
幅	72px	標準と同じ	64px
高さ	72px	標準と同じ	64px
ラジオボタン			
幅	72px	標準と同じ	64px
高さ	72px	標準と同じ	64px
Date/TimePicker			
モード	spinner	spinner	calendar/clock

(レイアウト部品のサイズ表)

SHARP

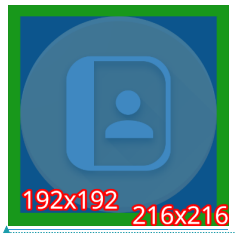
3.11　アプリアイコンの作成、適応方法

作成するアプリの起動は音声でのアプリ起動の他に、背面LCDのアプリランチャーを使用してのアプリ起動も可能です。
背面LCDのアプリランチャーでは、アプリごとにアイコンを設定することが出来ます。
この章ではアプリアイコンの作成方法についてご紹介します。

ロボホンのアプリアイコンは、座布団エリアと呼ばれる正円形の背景サークルの上に、アイコンのモチーフをのせて作成します。座布団エリアの色は、決められた８色から選ぶ必要があります。同じ製作者が複数アイコンを作る場合は、極力色が被らないようにする必要があります。

(アプリアイコンの配色例)

アイコンのサイズは以下のとおりで作成してください。

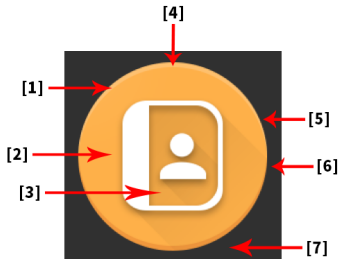
アイコン全体のサイズ **216×216 px** 背景サークル **192×192 px**	 (アイコンのサイズ情報)

書式変更: フォント : 太字

SHARP

デザイン上の注意点

アイコンをデザインする際は以下の項目に注意してください。



番号	説明文
[1]	アイコンの背景は正円のサークルとしてください。背景サークルは必ず以下の色を使用してください。 #ff6663 / #feb049 /#e3c89f /#ffd507 /#ffed57 /#ffd290 /#b08c71 /#ec96a9
[2]	左上から右下にかけFinish処理があります。 種類：円形 角度：45° カラー：#FFFFFF グラデーション中間点の位置：33% スライダー1: 不透過度10% 位置0% スライダー2: 不透過度0% 位置100%
[3]	アイコンのカラーは以下を使用してください。 アイコン内で使用するカラー：#FFFFFF
[4]	光沢のエッジ処理があります。 高さ：5px 不透明度：20% カラー：#FFFFFF
[5]	アイコンにドロップシャドウがついています。 モード：通常 不透明度：20% Xオフセット: 0px Yオフセット：4px ぼかし：4px カラー：背景サークルの色によってシャドウ/影のエッジ処理の色が変わります。 背景サークルカラー：シャドウカラー #ff6663 ： #3e2723

	#feb049 ： #bf360c #e3c89f ： #3e2723 #ffd507 ： #bf360c #ffed57 ： #bf360c #ffd290 ： #bf360c #b08c71 ： #3e2723 #ec96a9 ： #3e2723
[6]	45度フラットシャドウが入ります。光源が左上にある前提で実際の影と同様になるように配置します。 種類：線形 角度：-45° カラー：#000000 グラデーション中間点の位置：50% スライダー1: 不透過度10% 位置0% スライダー2: 不透過度0% 位置100%
[7]	影のエッジ処理があります。 高さ:5px 不透明度：20% カラー：背景サークルの色によって対応するシャドウカラーが変わります。 背景サークルカラー：シャドウカラー #ff6663 ： #3e2723 #feb049 ： #bf360c #e3c89f ： #3e2723 #ffd507 ： #bf360c #ffed57 ： #bf360c #ffd290 ： #bf360c #b08c71 ： #3e2723 #ec96a9 ： #3e2723

SDK資料内にPSD形式のサンプルデータが存在します。
[160328a_launcherIc_sample.psd] を参考にしながらアイコンを作成してください。
アイコンに使用するモチーフはGoogleが提供する [Material icons] を上手く活用すると簡単に作成できます。

[Google Material icons]
https://design.google.com/icons/

SHARP

3.12　タイトルバーの利用

ロボホンアプリではAndroidの標準的なアクションバーは使用せず、Toolbarをアクションバーとして利用します。
アプリ内で使用する場合は以下の項目を確認してください。

[確認項目]

- Toolbarをアプリレイアウト上に配置し、高さを [81dp] に指定します。
- アクションボタンを表示する場合は、アクションボタンをImageButtonとして使用します。オプションメニューのアクションアイテムとすると、意図通りのレイアウトに配置されません。
- アクションボタンのグラフィックは、72×72(px) サイズで drawable-xhdpi に配置してください。
- アクションボタンは、Activity の onCreate 内で setActionBar を実行して、Toolbar をアクションバーとして指定してください。
- タイトルには、アプリ名が表示されます。変更する場合は、getActionbar().setTitle()で変更してください。
- アプリ領域に重ねて、タイトルバーを透過することも可能です。

(タイトルバーの表示例)

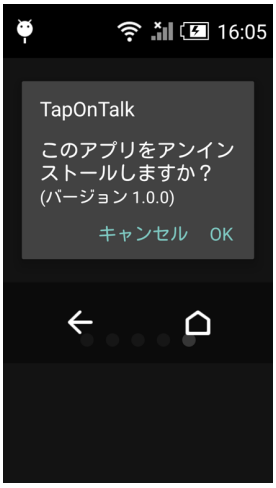
※　タイトルバーについての詳細はSDK資料[0401_SR01MW_Application_Programming_Guide 3.4.3 タイトルバー]の項目を参照してください。

SHARP

3.13　ロボホンへのアプリのインストール・アンインストール方法

作成したアプリをロボホンにインストールするには、Android Studio上の再生ボタンをクリックします。
再生ボタンをクリックするとアプリがビルドされ、ロボホンにインストールされます。

ロボホンに転送したアプリをアンインストールしたい場合は、ロボホン実機のアプリランチャーで、アンインストールしたいアプリのアイコンをホールドタップすると、アンインストールメニューが表示されます。



(アプリのアンインストール画面)

SHARP

3.14 Android アプリとロボホンアプリの機能の違い

この項目では、通知・マナーモード・文字入力など、Androidアプリ開発で頻繁に使用する項目が、ロボホンの場合どのように異なるかを記載します。

NOTIFICATION (通知)機能

ロボホンでもAndroidのNotificationAPIの利用が可能です。
ロボホンの背中の画面は通知アイコンが一つしか表示できません。そのため、必ず表示させるためには、Priority設定を高くする必要があります。
またロボホンでは、背中の画面がOFFの状態でも、眼の色を変えてNOTIFICATION通知をお知らせすることも可能です。
ロボホンでのNOTIFICATIONの詳しい利用方法はSDK資料の [0401_SR01MW_Application_Programming_Guide]をご確認ください。

マナーモード

ロボホンには本体側面にマナーモードのハードウェアスイッチが付いています。
このスイッチをマナーモードに設定すると、背面LCDでマナーモードの種類を選択することが出来ます。

マナーモードの種類

会話のみOFF	電話の着信音やアラームなどの電子音のみ再生させる
全てOFF	会話・電子音を含めすべての音を慣らさない。

2種類のマナーモードの違いはSTREAM_MUSIC を MUTEに指定するかしないかの違いです。
どちらのマナーモード状態でも、Android側では通常のマナーモードとして認識されており、
[AudioManager#getRingerMode()]は、どちらの場合でも [RINGER_MODE_SILENT]となります。
なお、マナーモードに設定している際は、ロボホンのシナリオファイルを実行することが出来ません。

文字入力

ロボホンは文字入力アプリとして手書き入力IMEパットとAndroidキーボード(English)を搭載しています。
Androidキーボードは主に半角英数字での利用を想定しており、日本語の入力は手書き入力IMEパッドを利用します。
入力フィールドのInputTypeを設定することで、表示させる入力方式を選択することが出来ます。
詳しくはSDK資料[0401_SR01MW_Application_Programming_Guide]をご覧ください。