SHARP

# RoBoHoN
# アプリ開発スタートガイド

Version 2.0.0

Last update 2019/8/7

SHARP CORPORATION

SHARP

## 更新履歴

| Version | Description | Date |
|---|---|---|
| 2.0.0 | 0101_SR01MW_Summary_of_SDK<br>0301_SR01MW_System_Overview<br>0801_SR01MW_Instructions_for_Installing_RoBoHoN_SDK_Libraries<br>を統合して新規作成 | 2019/8/7 |
| | | |
| | | |
| | | |

SHARP

# 目次

# 1. はじめに

## 1.1 本資料の目的

本資料はロボホンのアプリケーション開発を始めるにあたって必要な情報を記載しております。ロボホンに関する基本的な情報を確認したい場合は、「2. ロボホンの基本情報」からご覧ください。今すぐアプリケーションを作成する場合は、「3. 開発環境の準備」からご覧ください。
本資料記載の内容は特別な記載のない限り、ロボホンのビルド番号 03.01.00 以降のバージョンを対象にしています。
ロボホンのアプリ開発をご利用の際は「設定 – 端末情報 – ソフトウェア更新」より最新のバージョンにアップデートしてからご利用ください。

## 1.2 著作権

本資料に関する著作権は、シャープ株式会社（以下、当社といいます）に帰属します。本資料に記載の内容の一部、または全部を無断で転載または複製することを禁じます。

## 1.3 免責事項

当社は、本資料の内容が将来にわたり正常に作動すること、ならびに、常時利用できることを保証しません。また、本資料の内容が正常に作動しないことおよび本資料の内容が利用できないことにより開発者が損害を被った場合、当社は当該損害に関して一切責任を負いません。
本資料の内容は予告なしに変更される場合があります。

## 1.4 表記関係について

本資料に記載されている会社名、製品名などは、各社の登録商標または商標、商品名です。
会社名、製品名については、本文中では©、®、™マークなどは表示していません。

## 1.5 用語の定義

本資料で出てくる用語の意味を示します。

| 用語 | 意味 |
|---|---|
| ロボホン | ロボホン(3G・LTE)、ロボホン(Wi-Fi)、ロボホンライトの総称 |
| 第 1 世代ロボホン | SR01MW/ SR02MW を指します |
| 第 2 世代ロボホン | SR03M/SR04M/SR05M を指します |
| SDK | Software Development Kit |
| Android | Google が開発したモバイルオペレーティングシステム |
| 対話 | ユーザの発話⇒ロボホンの応答のやり取り。HVML の実行開始からすべての HVML の topic 実行が終了される間を指します。 |
| モーション | ロボホンの身振りやダンスなどの動きを指します。 |
| (対話)シナリオ | 対話フローを記述した HVML ファイルを指します。 |
| HVML | Hyper Voice Markup Language の略語 |

| | |
|---|---|
| TTS | Text-to-Speech の略語。音声合成を利用してテキストを読み上げる機能です。 |
| IME | Input Method Editor の略語。文字の入力を補助するソフトウェアを指します。 |

## 1.6 参考資料

| No. | Title |
|---|---|
| [1] | RoBoHoN_API リファレンス |
| [2] | RoBoHoN_HVML リファレンス |
| [3] | RoBoHoN_アプリ開発ガイドライン |
| | |

# 2. ロボホンの基本情報

ロボホンは、主に音声入力および音声出力と身振り(モーション)などの対話をすることで様々な機能を提供します。これらの対話はアプリごとに開発者が実装することが可能です。
本章ではロボホンのアプリを開発するにあたって必要となる基本的な情報を記載します。ロボホンの利用にあたっては各種規約をご確認の上ご利用ください。

## 2.1 モデル種別

ロボホンは 3G・LTE モデル（SR01MW/SR03M）、電話機能のない Wi-Fi モデル(SR02MW/SR04M)、２足歩行機能のないロボホンライト(SR05M)の３タイプで構成されています。
SR01MW/SR02MW を第 1 世代ロボホン、SR03M/SR04M/SR05M を第 2 世代ロボホンと呼びます。

## 2.2 スペック

ロボホンのスペック一覧を記載します。

**表 2-1　スペック一覧**

| 項目 | SR01MW/SR02MW | SR03M/SR04M/SR05M |
|---|---|---|
| OS | Android 5.0.2 | Android 8.1 |
| CPU | Qualcomm Snapdragon 400 | Qualcomm Snapdragon 430 |
| メモリ | ROM 16GB／RAM 2GB | |
| ディスプレイ | 約 2.0 インチ QVGA (240×320 画素) | 約 2.6 インチ QVGA (240×320 画素) |
| プロジェクター | HD（1,280×720 画素）相当 | ― |
| カメラ | 約 800 万画素 CMOS | 約 800 万画素 CMOS |
| Wi-Fi | IEEE802.11b/g/n(2.4GHz のみ) | IEEE802.11a/b/g/n(2.4G/5GHz)/ac |
| Bluetooth | 4.0　BLE 対応(Peripheral 非対応) | 4.2 |
| GPS | 対応 | |
| センサー | 9 軸（加速度 3 軸、地磁気 3 軸、ジャイロ 3 軸）、照度センサー | |
| サーボモーター | 13 個 | 3G・LTE モデル・Wi-Fi モデル：13 個<br>ロボホンライト：7 個 |
| NFC/Felica | 非対応 | |
| ワンセグ/フルセグ | 非対応 | |
| SD カード | 非対応 | |

## 2.3 ハードウェア概要

ロボホンの各種パーツについて記載します。ハードウェアに関するソフト的な留意事項は参考資料[3]の「3. ハードウェア関連」を参照ください。

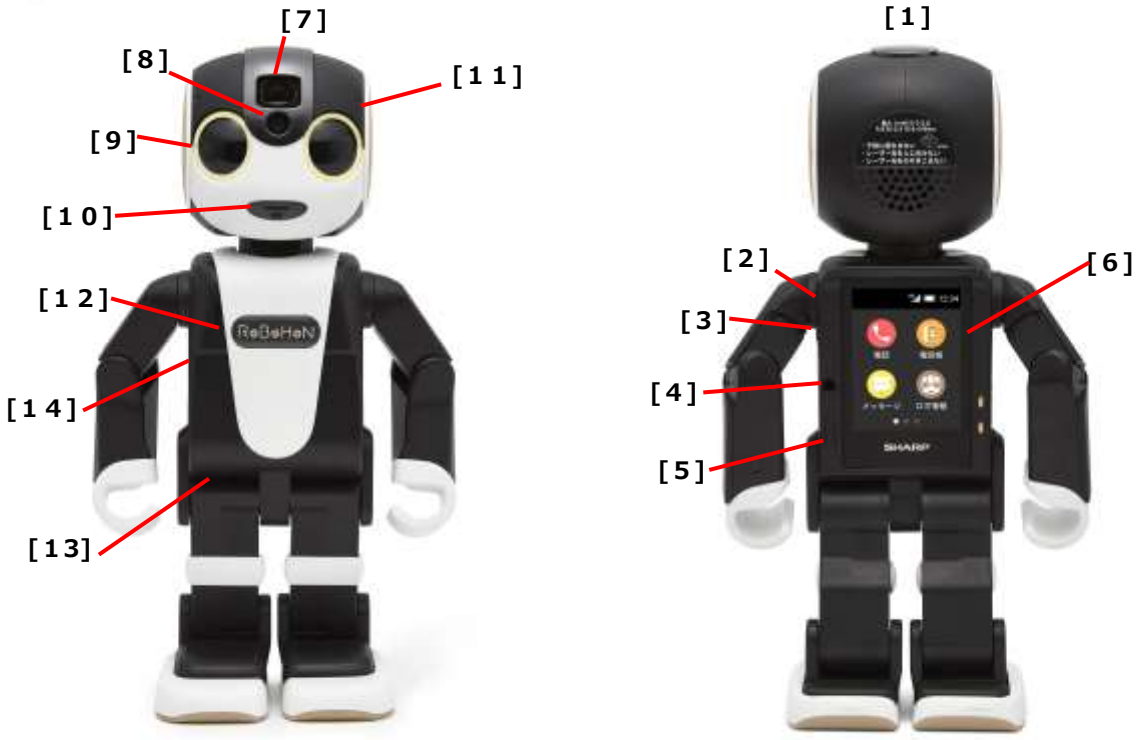



図 2-1 ロボホン(SR01M/SR02M)の外観(前後)


図 2-1 ロボホン(SR01M/SR02M)の外観(充電台)

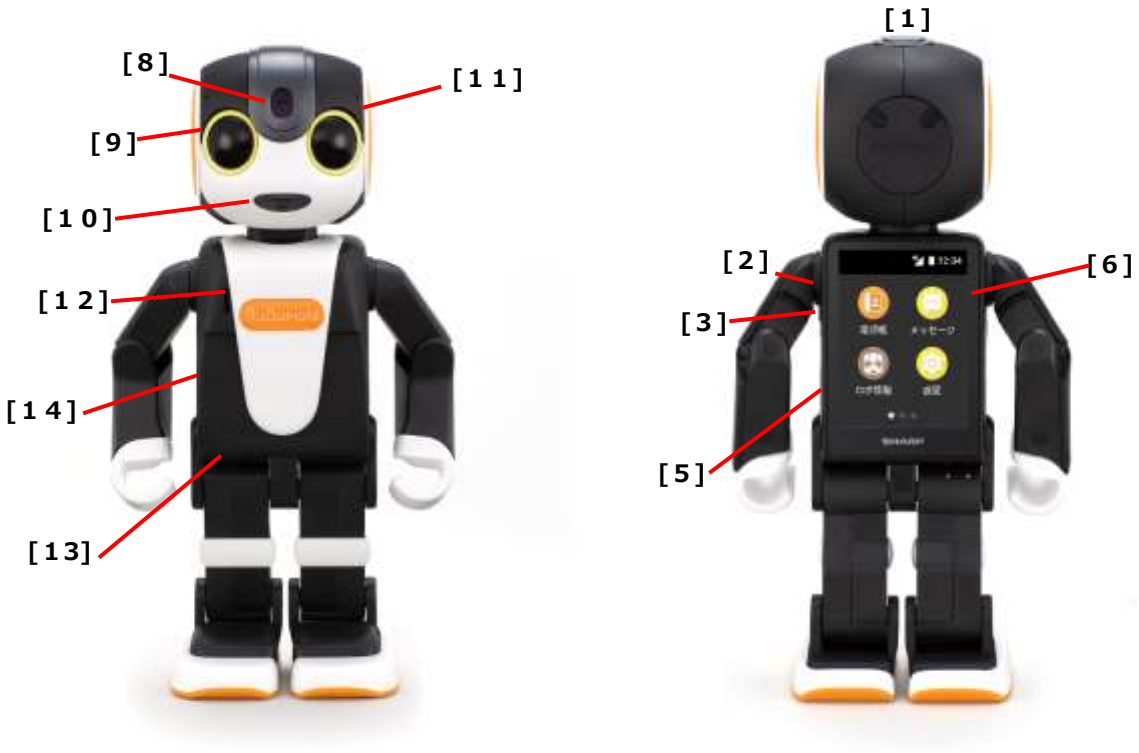



**図 2-1 ロボホン(SR03M/SR04M/SR05M)の外観前後**


**図 2-1 ロボホン(SR03M/SR04M/SR05M)の充電台外観**

**表 2.2　ロボホンの各部の説明**

| 部位 | 名称 | 機能概略 |
|---|---|---|
| [1] | 頭ボタン | アプリ終了、発話中断用ボタン |
| [2] | 電源ボタン | ディスプレイの消灯/点灯、長押しで電源 ON/OFF |
| [3] | マナースイッチ | マナーモード切替用スイッチ |
| [4] | イヤホンマイク端子 | 3.5Φ　SR03M/04M/05M は非搭載 |
| [5] | microUSB 端子 | microUSB ケーブルや充電器接続。SR03M/04M/05M は側面 |
| [6] | ディスプレイ | 表示およびタッチ操作 |
| [7] | プロジェクター | SR03M/04M/05M は非搭載 |
| [8] | カメラ | - |
| [9] | LED(目) | ３色 LED。モーション時や通知時に点灯、点滅。 |
| [10] | LED(口) | 単色 LED。発話に合わせて点滅。アプリ制御不可。 |
| [10] | レシーバー | 通話の際に相手の声が聞こえる |
| [10] | 照度センサー | - |
| [11] | マイク(音声認識、動画撮影) | 音声認識とカメラの動画撮影に使われるマイク |
| [12] | スピーカー | - |
| [13] | マイク(通話) | 通話用マイク |
| [14] | nanoSIM カードスロット | nanoSIM カード　3G・LTE モデルのみ |
| [15] | 充電台 | SR03M/04M/05M は別売り |

## 2.4 ソフトウェア概要

ロボホンは音声認識、メッセージ機能、各種コンテンツ(ニュース等)、アプリのダウンロード・更新機能(アプリ管理)、バックアップ・リストアなどの機能をクラウド上で提供しています。クラウドサービスを利用するためにはココロプランもしくは法人向けのビジネスプランの契約が必要です。プラン契約についてはロボホンポータルサイトよりご確認ください。

ロボホンは Android をベースに対話機能を実現する為の当社独自のフレームワーク(音声 UI)を提供しています。各アプリは HVML 形式の対話フローなどを記載したファイル(対話シナリオ)を持ち、インストール時に音声 UI に登録することで、ユーザからの音声入力、ロボホンの発話やモーションなどを実行することができます。

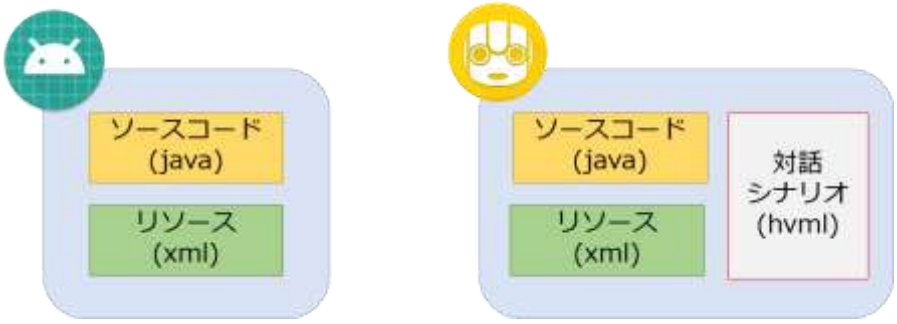



**図 2.2 Android アプリとロボホンアプリのイメージ図**

対話に関する簡単な流れを説明します。ロボホンでは音声 UI が常時外部からの音声入力を待ち受けています。ユーザからの音声を検出・認識(Speech to Text)したら、認識したテキストを用いて登録されているシナリオの中から最も適切なシナリオを選択します。その後、選択されたシナリオに記述されている発話やモーションを実行します。

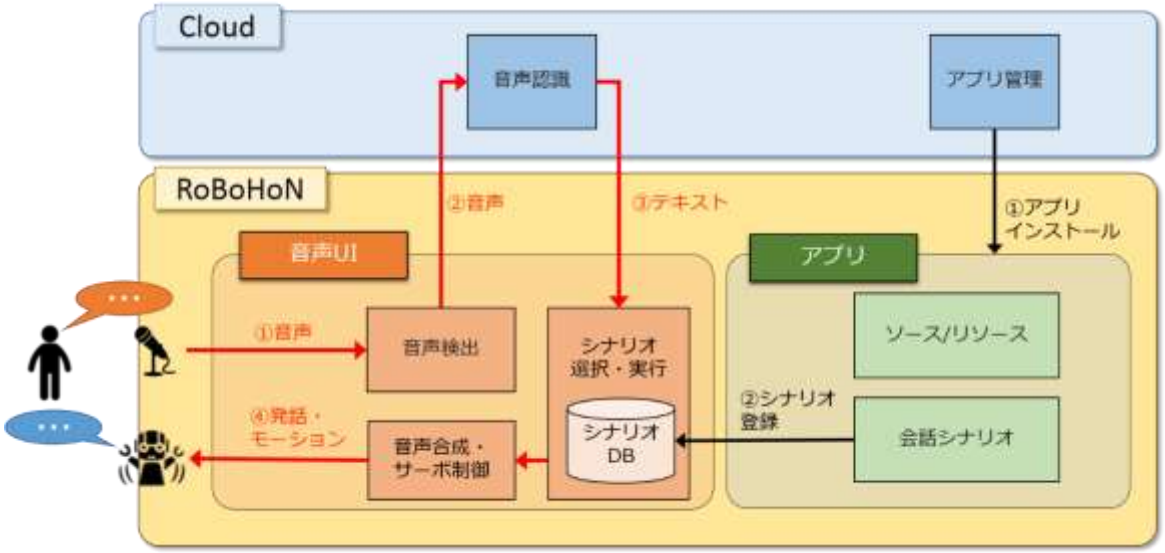



**図 2.3 アプリインストール時と対話の処理イメージ**

音声 UI の他にも、いくつかのロボホン固有の拡張 API を用いて様々なアプリを作成することができます。アプリの開発に関する詳細な情報、利用可能な各種 API、HVML の仕様、アプリの実装方法について参考資料を参照ください。

ロボホンの発話などに用いる記憶情報やインストール済のアプリ情報はロボホンのバックアップ対象データとなっており、バックアップ設定が ON になっている場合は、修理などでデータリセットされてもアプリは特に意識しなくても復元できます。バックアップに関しては取扱説明書を参照してください。

# 3. 開発環境の準備

ロボホンのアプリ開発には以下が必要です。

- ロボホン本体
  アプリの動作確認にはロボホン本体が必要です。エミュレータでの動作確認は出来ません。
- ネットワーク環境
  nanoSIM カードまたは無線 LAN 機器、通信回線等が必要です。
  インターネット接続サービスのプロバイダ料、通信費等、インターネットによる通信に必要な諸費用は開発者にてご負担ください。
- クラウドサービスの契約
  ココロプランもしくは法人向けのビジネスプランの契約が必要です。プラン契約については[ロボホンポータルサイト]よりご確認ください。
- PC
  システム要件は Android Studio のシステム要件に従います。
  https://developer.android.com/studio/

本章では PC(Android Studio 環境)/ロボホンのセットアップ手順を記載します。

## 3.1 PC(Android Studio 環境)のセットアップ

ロボホンは Android Studio でアプリを開発します。以下の順番で環境を設定してください。

1. JDK のダウンロード・インストール(必須)
2. Android Studio のインストール(必須)
3. SDK、Build Tools の追加インストール(必須)
4. adb driver のインストール(必要に応じて)
5. Instant Run 設定の無効化(必要に応じて)
6. HVML ファイルの拡張子を追加(推奨)
7. HVML 構文チェックツールの導入(推奨)

### 3.1.1 JDK のダウンロード・インストール

JDK（Java Development Kit）とは、Java 開発キットのことです。
以下のサイトからダウンロードし、インストールしてください。

http://www.oracle.com/technetwork/java/javase/downloads/index.html

JDK 7/8 で動作することを確認しています。

### 3.1.2　Android Studio のインストール

Android Studio は以下のサイトからダウンロードし、インストールしてください。

https://developer.android.com/studio/

なお、ロボホンのアプリ開発には Android Studio のバージョン 1.5 以降を利用してください。
ここでは、執筆時の最新版である Android Studio 3.3.2 の手順を記載します。

### 3.1.3　SDK、Build Tools の追加インストール

Android　SDK および、ビルドに必要となる SDK Build-Tools をインストールしてください。
Android Studio 画面の File → Settings メニューから、
Appearance & Behavior > System Settings > Android SDK
と進んで表示した画面で、下記の通りインストールします。

- SDK Platforms タブ
  “Android5.0(Lollipop) 21” の項目をチェックします。
- SDK Tools タブ
  “Android SDK Build Tools” の項目を選択して、表下部の ”Show Package Details” をチェックすると、
  バージョンごとに Build-Tools の項目が表示されるので、21 以降のバージョンをチェックします。

### 3.1.4　adb driver のインストール

以下に、adb driver のインストール手順を記載します。本手順は Windows でのみ必要で Mac OS X では不要です。
adb driver は Windows7, Windows 8.1, Windows10 にて動作することを確認しております。

1. ロボホンの USB デバッグの設定を ON にする。
2. USB ドライバ（usb_driver_SHARP_RoBoHoN_rx.x.zip）　をローカル PC に展開する。
3. ロボホンと PC を USB ケーブルで接続する。
4. ドライバの自動 Install が失敗するので、デバイスマネージャーを開いて Android ADB Interface の表示を確認する。
   ※Windows OS バージョンによって表示が異なります。「不明なデバイス」や「！マークが付いている(Android) ADB Interface」、！マークが付いていない 「ADB Interface」と表示されています。
5. Android ADB Interface を右クリックし、「ドライバーソフトウェアの更新」を選択する。
6. 表示が出たら、「コンピュータを参照してドライバーソフトウェアを検索します」を選択する。
7. 参照ボタンを押して 1 で展開したフォルダを選択し、「次へ」を押す。
8. 「”Sharp Corporation”からのソフトウェアを常に信頼する」にチェックを入れて、「インストール」を押す。
9. インストールが終了したら、「閉じる」を押す。

以上で、adb driver のインストールは終了です。

### 3.1.5　Instant Run 設定の無効化

Android Studio のバージョンが 2.x の場合、開発したアプリを debug 実行する際に 意図したシナリオが実行されない事があります。そのような場合、下記の手順に沿って Instant Run の設定を無効化してください。

1. メニューバーの[File]から[Settings…]をクリックする
2. Settings 画面から[Build, Execution, Deployment]をダブルクリックする
3. [Instant Run]をクリックする
4. [Enable Instant Run to hot swap code/resource changes on deploy (default enabled)]のチェックボックスをクリックしチェックを外す
5. OK ボタンをクリックする

上記設定を行っても不具合が改善しないときは、開発中のアプリをロボホンから一旦アンインストールしてください。設定変更が反映され、不具合が改善される場合があります。

### 3.1.6　HVML ファイルの拡張子を追加

ロボホンのシナリオは hvml ファイルを実装することにより実現します。以下の手順で Android Studio のファイルタイプに追加することで編集、構文チェックが可能になります。

1. Android Studio で File→Settings より Setting 画面を立ち上げます。
2. Editor - File Types を選択し、Recognized File Types から”XML”を選択します。
3. Registerd Patterns の＋ボタンを押下します。

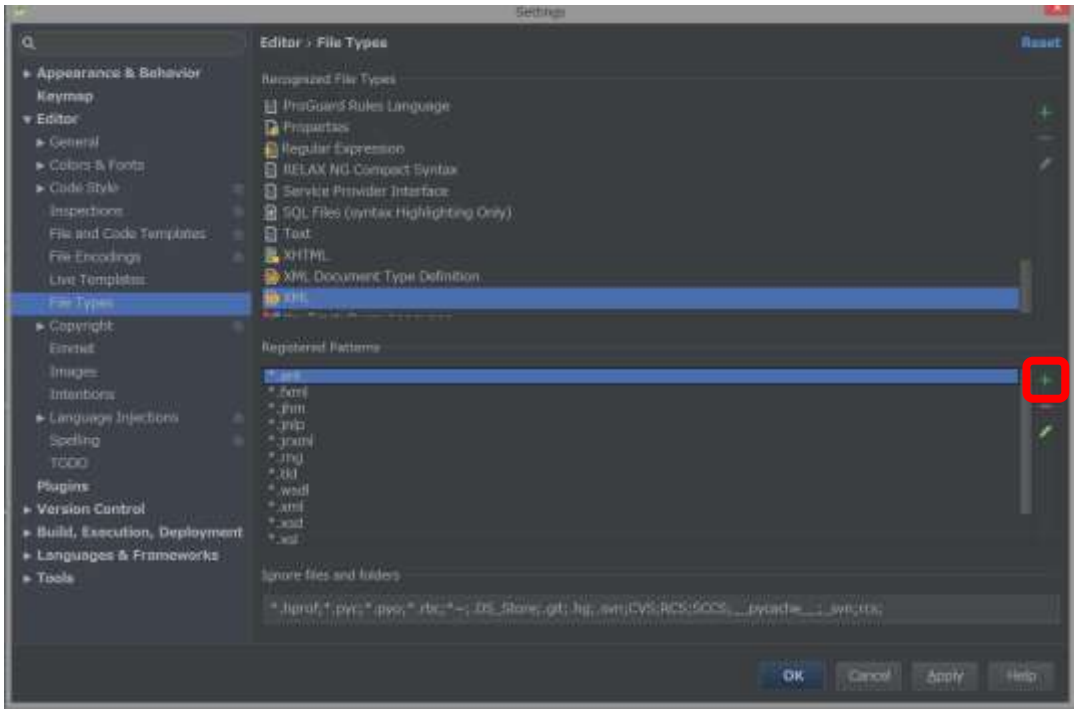



**図 3-1　Settings 画面**

表示されたテキストボックスに“*.hvml”を入力して OK を押下します。Setting 画面で再度 OK を押下で設定完了です。

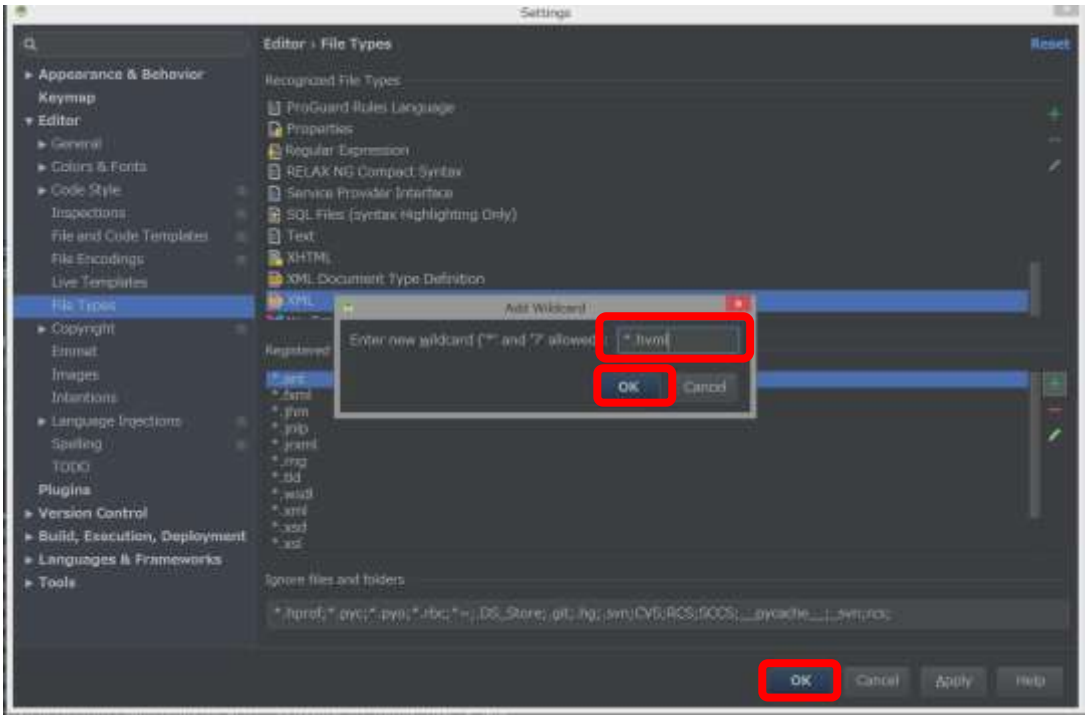



**図 3-2　Add Wildcard 画面**

### 3.1.7　HVML 構文チェックツールの導入

「3.1.6 HVML ファイルの拡張子を追加」の手順に加え、実際に hvml ファイルを作成・編集する際に以下の手順を追加で行うことにより、HVML 固有の構文チェックが可能になります。本手順は HVML ファイル毎に設定が必要です。
※完全なチェックができる訳ではありませんので、本ツールに加え、参考資料[2] [3]に従って作成してください。

1. schema.zip を PC に展開します。
2. hvml ファイルの編集画面の右上にある[レ]マークを右クリックし[Customize HighLighting Level]をクリックするか、または Android Studio 画面右下のアイコンをクリックします。

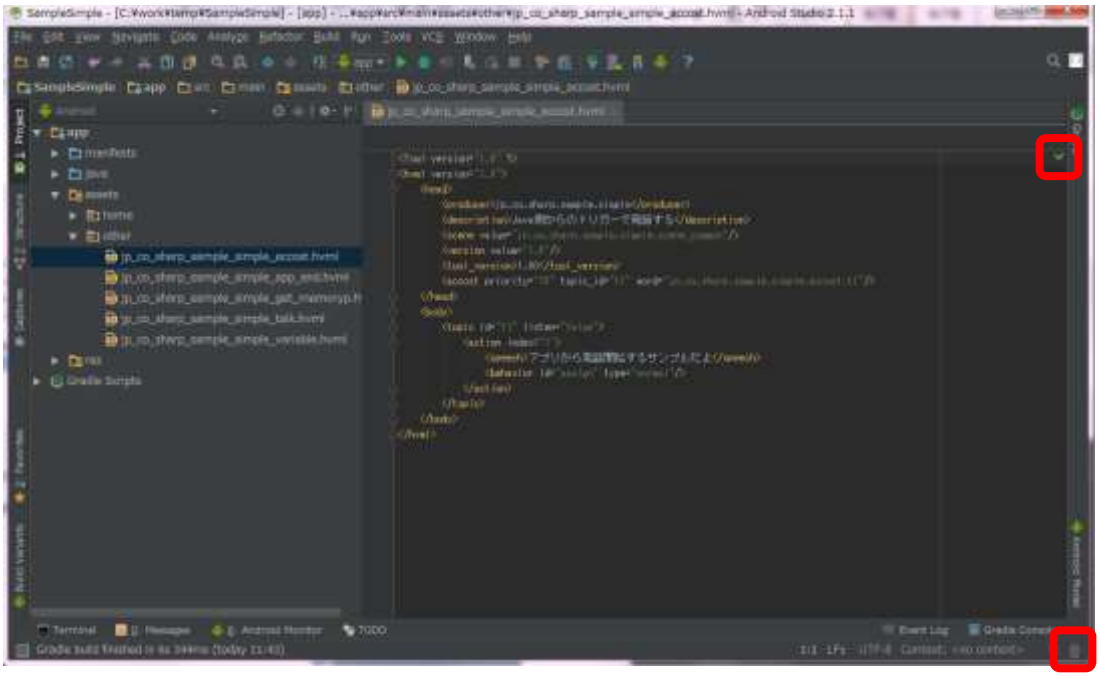


**図 3-3　HVML 編集画面**

3.　Relax-NG Schema Assosiation から 1.に含まれている"hvml_schema_Verxxx.xsd"を選択します。

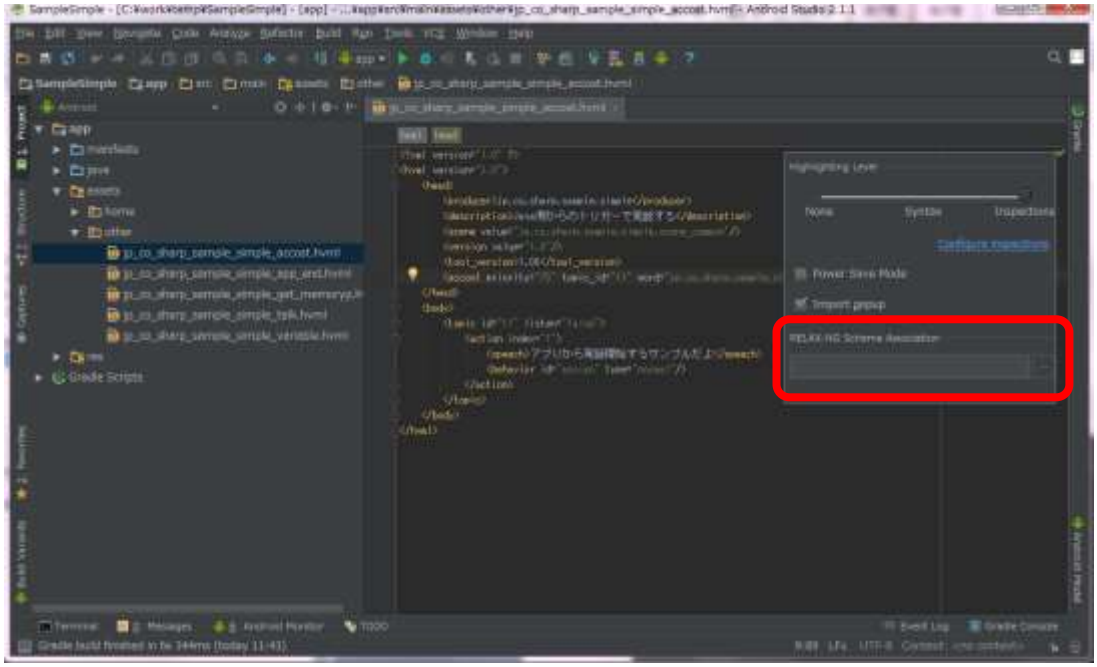


**図 3-4　Relax-NG Schema Assosiation 画面**

4.　hvml ファイルを編集します。以下のような記述があった場合にエラー検出が可能になります。

・hvml で定義されていないタグ
・タグの記述位置の不正
・各タグの必須属性が記述されていない

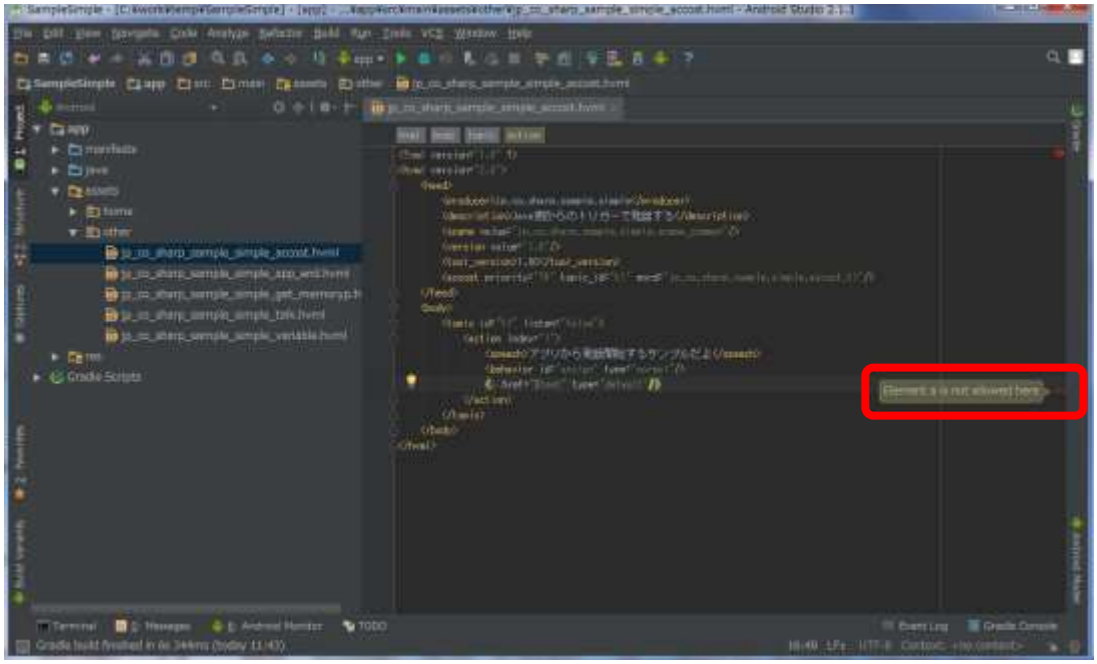


**図 3-5　HVML 編集画面(不正フォーマット)**

## 3.2　ロボホンの設定

動作確認用のロボホンで必要な設定です。接続するロボホン毎に必要な手順となります。一度設定しておくと設定を変更しない限り有効となります。

### 3.2.1　USB デバッグの設定

ロボホンが USB 接続されたことを PC が認識できるように、ロボホン側の設定を変更する必要があります。
以下に設定手順を記載します。

1. 背面 LCD のメニューの「設定」の欄をタップする。
2. 「端末情報」の欄をタップする。
3. 「ビルド番号」の欄を 7 回タップする。
4. 設定メニューに戻り、一番下の「その他」の欄をタップする。
5. 「開発者向けオプション」の項目が追加されているので、それをタップし、OFF を ON へ変更する。
6. 「開発用の設定を許可しますか」と聞かれるので、右下の OK をタップする。
7. 同メニュー内の「USB デバッグ」の項目をタップし、チェックを ON にする。
8. 「USB デバッグを許可しますか」と聞かれるので、右下の OK をタップする。
9. PC とロボホンを USB 接続した際に、「USB デバッグを許可しますか？」のダイアログが表示される場合は、「このパソコンからの USB がデバッグを常に許可する」にチェックを入れて、OK を選択する。

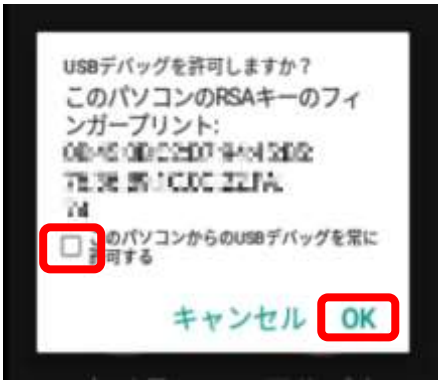



**図 3-6　USB デバッグの許可**

以上で、USB デバッグの設定手順は終了です。

### 3.2.2　スリープモードにしない設定

ロボホンは一定時間経過すると画面消灯してスリープモードになります。アプリ開発中にスリープモードに入らせたくない場合は、「3.2.1 USB デバッグの設定」に記載の「開発者向けオプション」より、「スリープモードにしない」にチェックをつけてください。

# 4. アプリのビルド手順

ロボホンのアプリを作成するためにはいくつかの決まった手順が必要です。簡略化するためにサンプルアプリやテンプレートアプリを用意しておりますので、目的に応じてそれぞれの手順でロボホンアプリをビルドしてください。
本章では以下の手順について記載します。

- サンプルアプリのビルド　　：　とりあえずサンプルアプリを試してみたい方
- テンプレートアプリのビルド　：　新規のアプリを作成したい方
- 新規プロジェクトのビルド　　：　完全にゼロからアプリを作成、既存のアプリをロボホン対応したい方

## 4.1　サンプルアプリのビルド

ここでは、SDK に同梱されているサンプル(sample.zip)をビルドする手順を記載します。同 zip 内には利用する API 毎にアプリのプロジェクト一式をサンプルコードとして同梱しています。まずはサンプルアプリの動作を確認されたい方はこちらの手順に従ってビルドしてください。

以下ではその中の SampleSimple を用いた手順を記載します。

1. sample.zip を PC の任意の場所に展開します。
2. Android Studio を起動し、「Open an existing Android Studio project」を選択します。

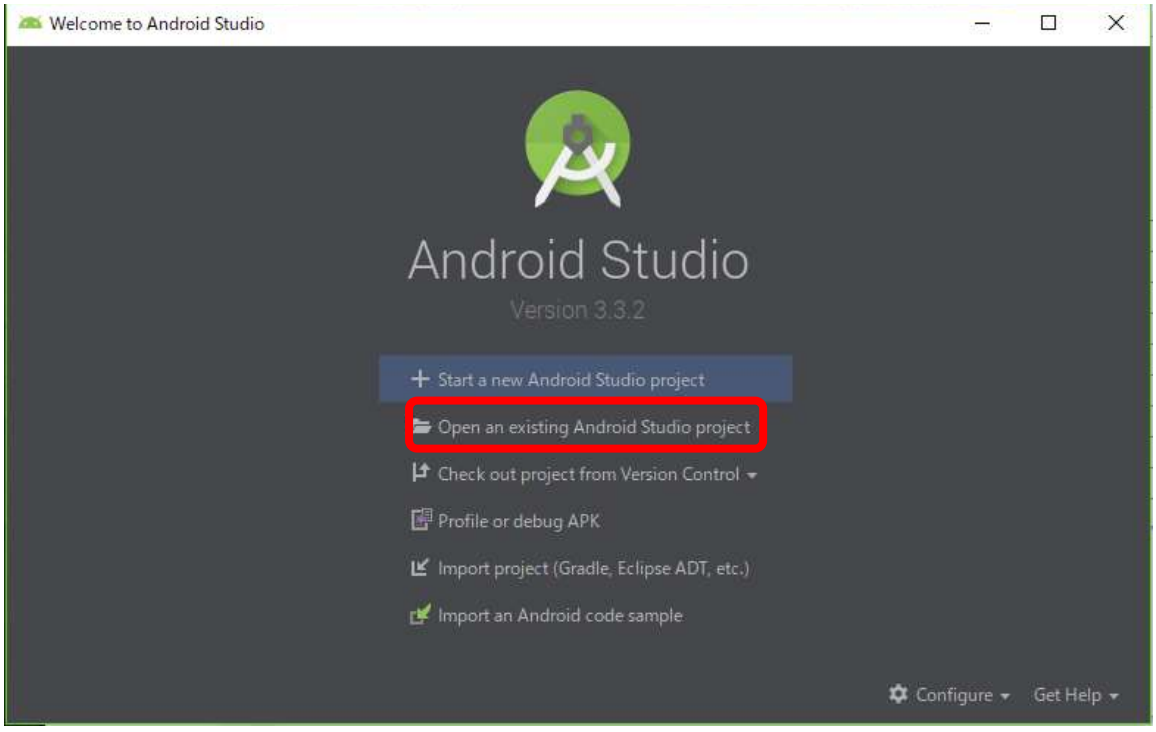



**図 4-1 開始画面**

3. 手順 1 で展開後のフォルダより「SampleSimple」を選択し、「OK」ボタンを押下します。

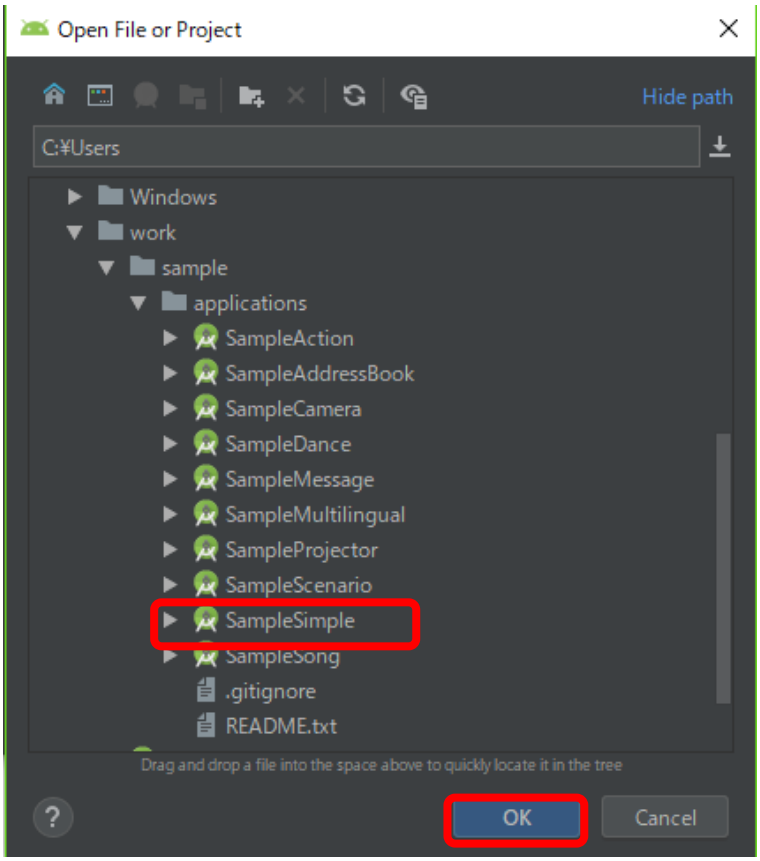



**図 4-2 Open File or Project 画面**

4. プロキシ設定画面出る場合には、必要な設定をして「OK」ボタンを押下します。
5. Gradle のアップデート画面が出るが無視します。
   ※Update する場合、エラーが Warning や Error が発生しますので、その際は build.gradle などを適切に修正してください。

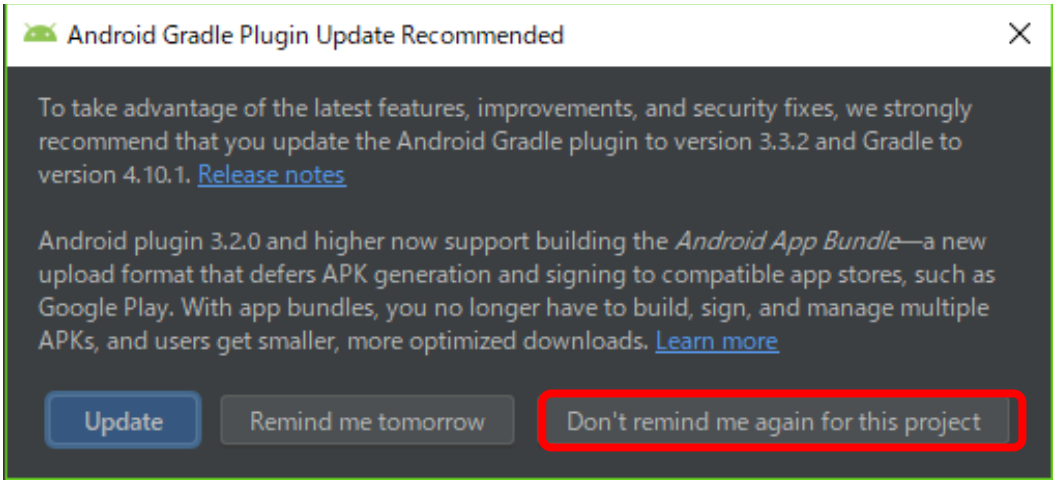



**図 4-3 Gradle Plugin Upadate Recommended 画面**

6. Gradle の Build が完了したら(左下部に Gradle build finished と表示される)、ロボホンを USB 接続した状態で Run 'app'ボタンを押下します。

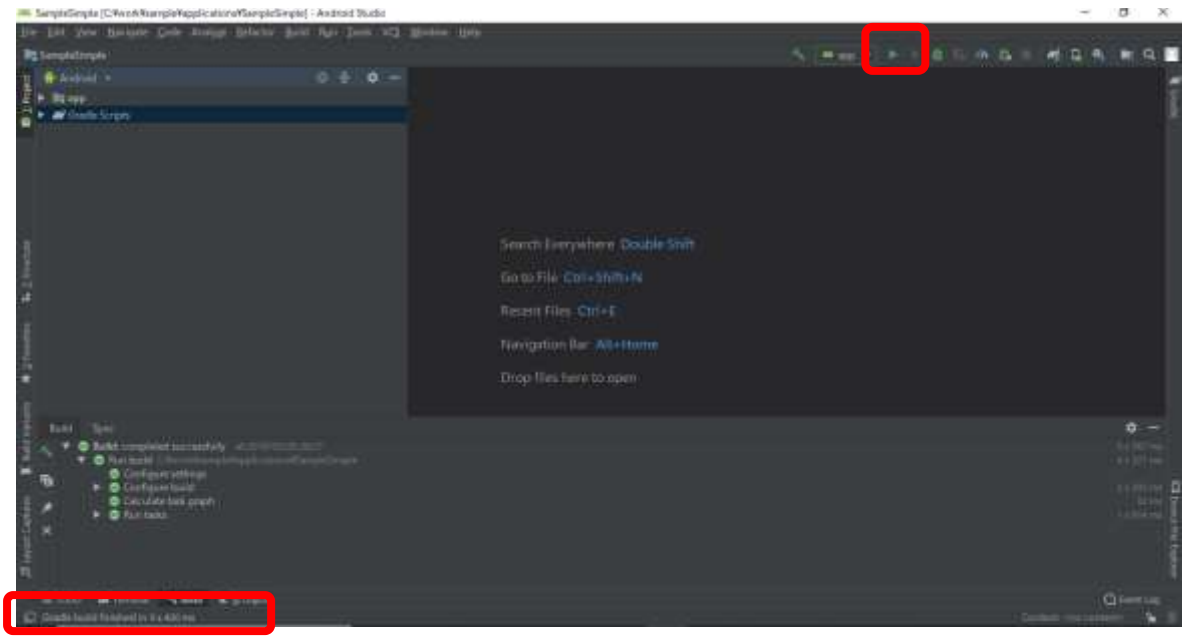


**図 4-4 Gradle build 完了**

7. Select Deployment Target 画面で Connected Devices に接続しているロボホン名(SHARP SR01MW など)が表示されていれば選択し、OK ボタンを押下します。表示されていない場合はロボホンが USB 接続されていない、ロボホンの adb ドライバがインストールされていない、もしくはロボホンの USB デバッグ設定が ON になっていない可能性がありますので、「3.2.1 USB デバッグの設定」「3.1.4 adb driver のインストール」を確認ください。

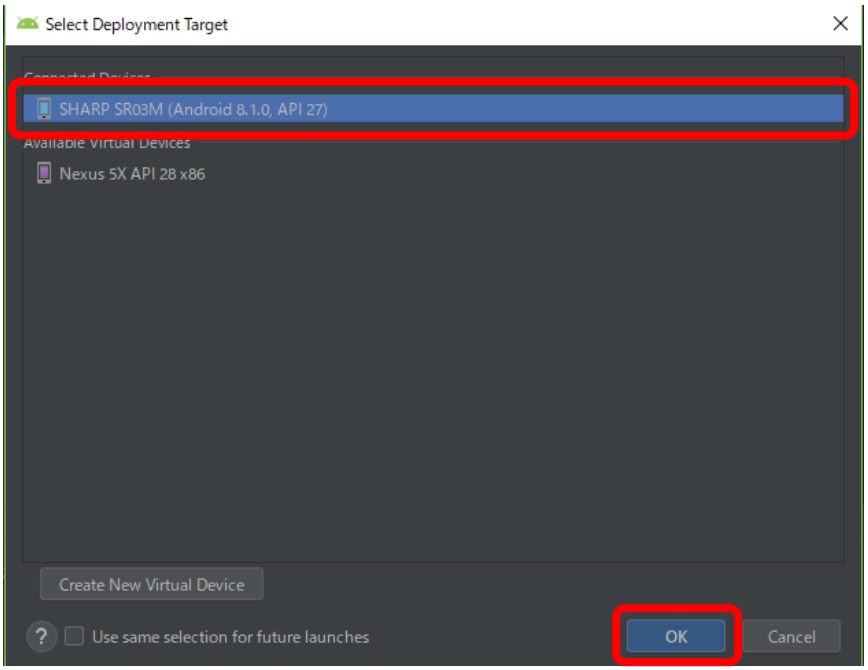



**図 4-5 Select Deployment Target 画面**

9. ロボホン上でアプリ画面が立ち上がり、発話(ACCOST)ボタン押下で発話されれば無事アプリインストール完了です。発話されない場合は、マナーモードになっていないか、「3.1.5 Instant Run 設定の無効化」等を確認してください。


**図 4-6 SampleSimple アプリ画面**

## 4.2 テンプレートアプリのビルド

ここでは、テンプレートアプリのパッケージ名を変更して新規アプリをビルドする手順を記載します。新規でアプリを作成されたい方はこちらの手順にしたがってビルドしてください。

テンプレートは音声 UI の API のみを利用する最低限のテンプレート(Template)と、すべての API を利用するためのテンプレート(TemplateFull)の 2 種類を用意しております。作成するアプリに都合のよい方を選択してご利用ください。

以下、Template(com.robohon.template)を RobohonApp(jp.co.sharp.robohon.app)に変更する場合の手順を記載します。

1. template.zip を PC の任意の場所に展開します。ここでは「C:¥work¥」配下に解凍しています。

2. プロジェクトのルートのフォルダ名を「C:¥work¥template¥**Template**」から「**RobohonApp**」に変更します。

3. java ソースのフォルダ構成をパッケージ名に合わせて
   C:¥work¥template¥RobohonApp¥app¥src¥main¥java**¥com¥robohon¥template¥**
   から
   C:¥work¥template¥RobohonApp¥app¥src¥main¥java**¥jp¥co¥sharp¥robohon¥app¥**
   へ変更します。

4. RobohonApp¥app¥src¥main¥AndroidManifest.xml の package 名を修正する。

【AndroidManifest.xml】

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package=" jp.co.sharp.robohon.app "
          android:versionCode="10000"
          android:versionName="1.0.0">
```

5. RobohonApp¥app¥build.gradle の applicationId を修正する。

【build.gradle】

```
    defaultConfig {
        applicationId " jp.co.sharp.robohon.app "
        minSdkVersion 21
        targetSdkVersion 21
    }
```

6. C:¥work¥template¥RobohonApp¥app¥src¥main¥assets¥hvml 配下の HVML ファイル名を HVML 命名ルールに従って変更します。詳細は参考資料[3]の「4.1 HVML ファイル命名規則」を参照ください。

   **com_robohon_template_home.hvml → jp_co_sharp_robohon_app_home.hvml**
   **com_robohon_template_hello.hvml → jp_co_sharp_robohon_app_hello.hvml**

7. Java/HVML ファイル内のパッケージ名記載箇所を「**com.robohon.template**」から「**jp.co.sharp.robohon.app**」へ置換します。※grep して一括で置換すると楽です。

【変更ファイル名一覧　(カッコ内は行数) 色塗部が置換必要箇所】

jp_co_sharp_robohon_app_home.hvml(4):
<producer>**jp.co.sharp.robohon.app**</producer>
jp_co_sharp_robohon_app_home.hvml(22):
<data key="package_name" value="**jp.co.sharp.robohon.app**"/>
jp_co_sharp_robohon_app_home.hvml(23):
<data key="class_name" value="**jp.co.sharp.robohon.app**.MainActivity"/>
jp_co_sharp_robohon_app_hello.hvml(4):
<producer>**jp.co.sharp.robohon.app**</producer>
jp_co_sharp_robohon_app_hello.hvml(6):
<scene value="**jp.co.sharp.robohon.app**.scene_common"/>
jp_co_sharp_robohon_app_hello.hvml(11):
<accost priority="75" topic_id="say" word="**jp.co.sharp.robohon.app**.hello.say"/>
MainActivity.java(1):
package **jp.co.sharp.robohon.app;**
MainActivity.java(13):
import **jp.co.sharp.robohon.app**.voiceui.ScenarioDefinitions;
MainActivity.java(14):
import **jp.co.sharp.robohon.app**.voiceui.VoiceUIListenerImpl;
MainActivity.java(15)
import **jp.co.sharp.robohon.app**.voiceui.VoiceUIManagerUtil;
RegisterScenarioService.java(1)
package **jp.co.sharp.robohon.app**.voiceui;
RequestScenarioReceiver.java(1)
package **jp.co.sharp.robohon.app**.voiceui;
ScenarioDefinitions.java(1)
package **jp.co.sharp.robohon.app**.voiceui;
ScenarioDefinitions.java(42)
protected static final String PACKAGE = "**jp.co.sharp.robohon.app**";
VoiceUIListenerImpl.java(1)
package **jp.co.sharp.robohon.app**.voiceui;
VoiceUIManagerUtil.java(1)
package **jp.co.sharp.robohon.app**.voiceui;
VoiceUIManagerUtil.java(12)
import static **jp.co.sharp.robohon.app**.voiceui.ScenarioDefinitions.TAG_ACCOST;
VoiceUIVariableUtil.java(1)
package **jp.co.sharp.robohon.app**.voiceui;

8. Android Studio を起動し、「Open an existing Android Studio project」を選択します。

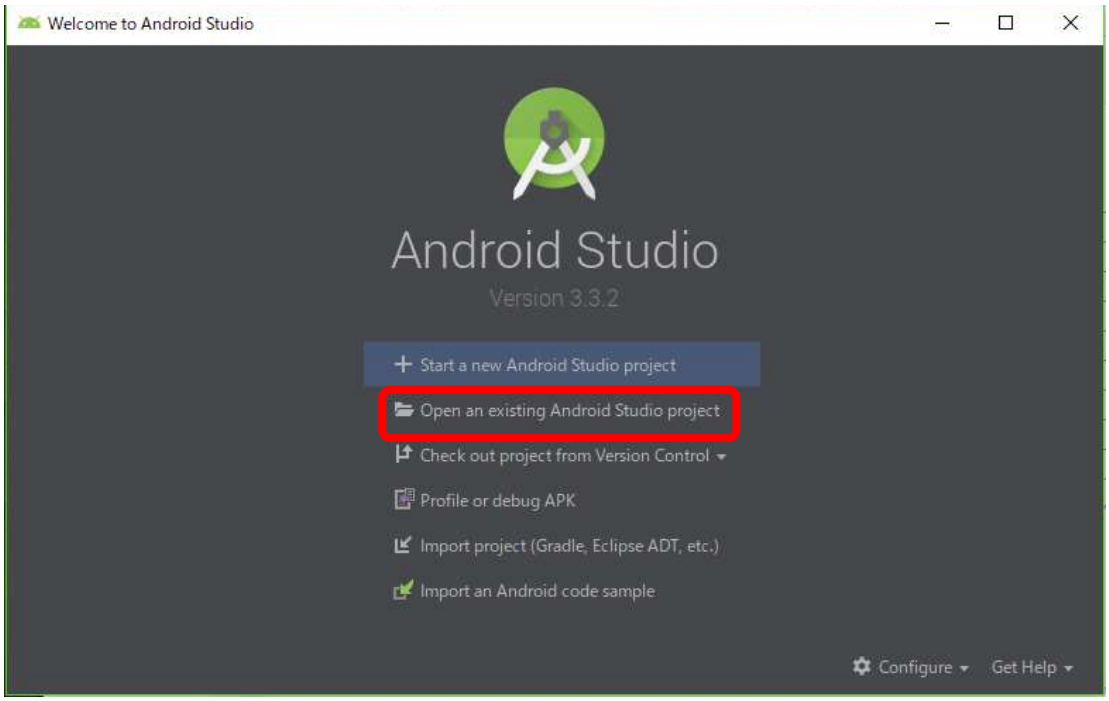



**図 4-7 開始画面**

9. 「RobohonApp」を選択し、「OK」ボタンを押下します。以降、「4.1 サンプルアプリのビルド」手順 4.以降と同様です。

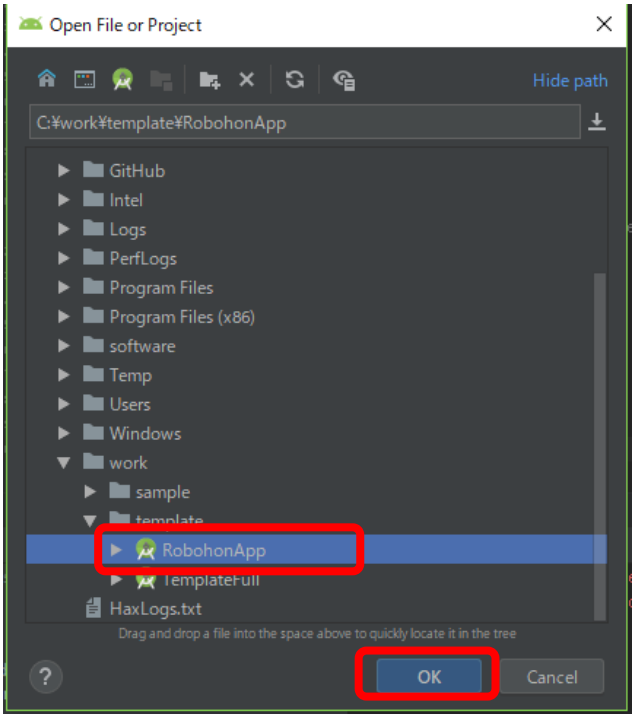



**図 4-8 Open File or Project 画面**

## 4.3 新規プロジェクトのビルド

ここでは、新規のプロジェクトでアプリを作成する場合や、既存のプロジェクトにロボホンの API を導入してビルドする手順を記載します。ロボホンアプリ開発上級者の方向けです。

1. Android Studio でプロジェクトを開きます。ここでは Start a new Android Studio project を選択した場合の手順を記載します。既存のパッケージを流用する場合は、Open an existing Android Studio project よりパッケージを選択してください。

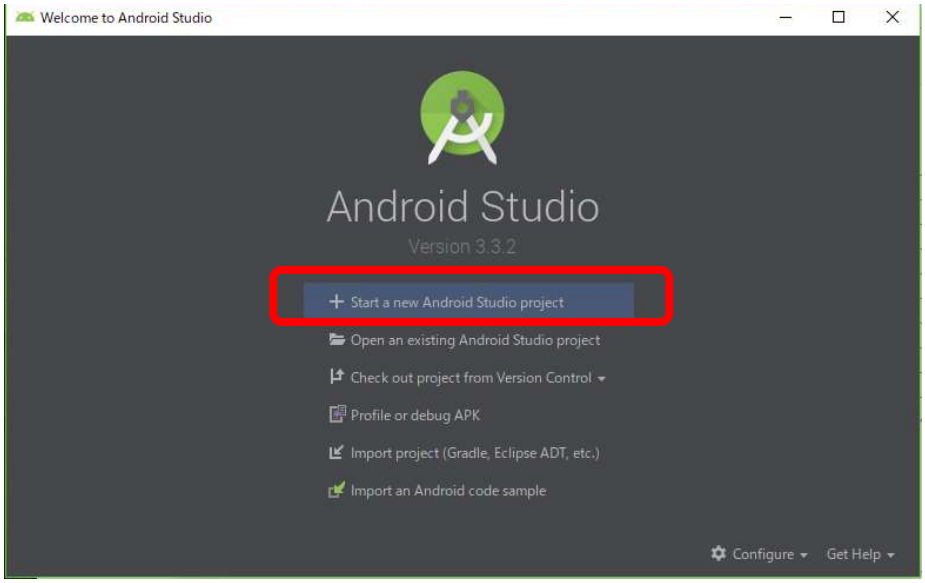



**図 4-9 開始画面**

2. 「Empty Activity」を選択し、「Next」ボタンを押下します。

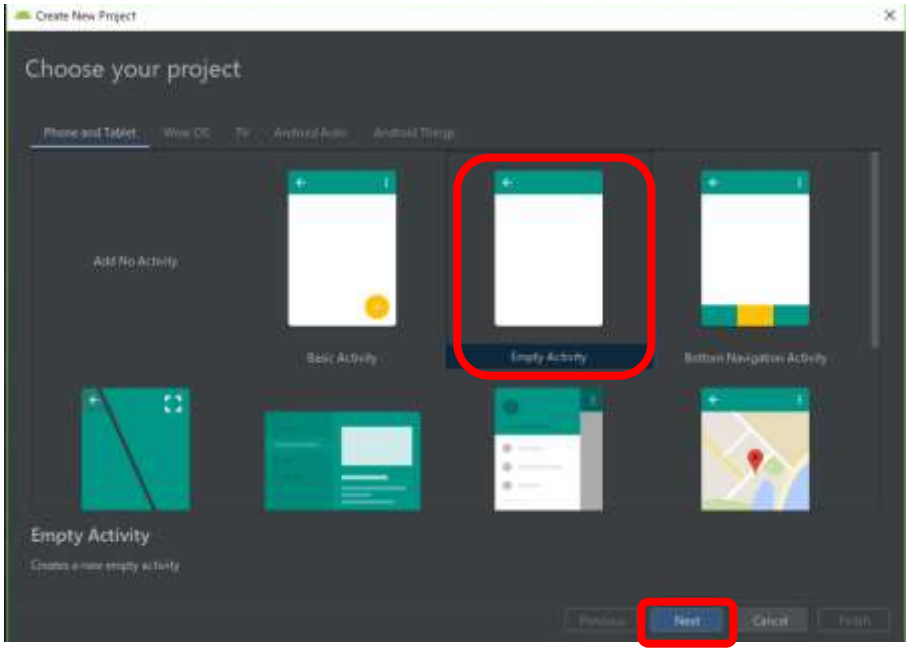



**図 4-10 Choose your project 画面**

3. 適当なパッケージ名を入力し、Language は「Java」、Minimum API level は API レベル 21 を選択してください。

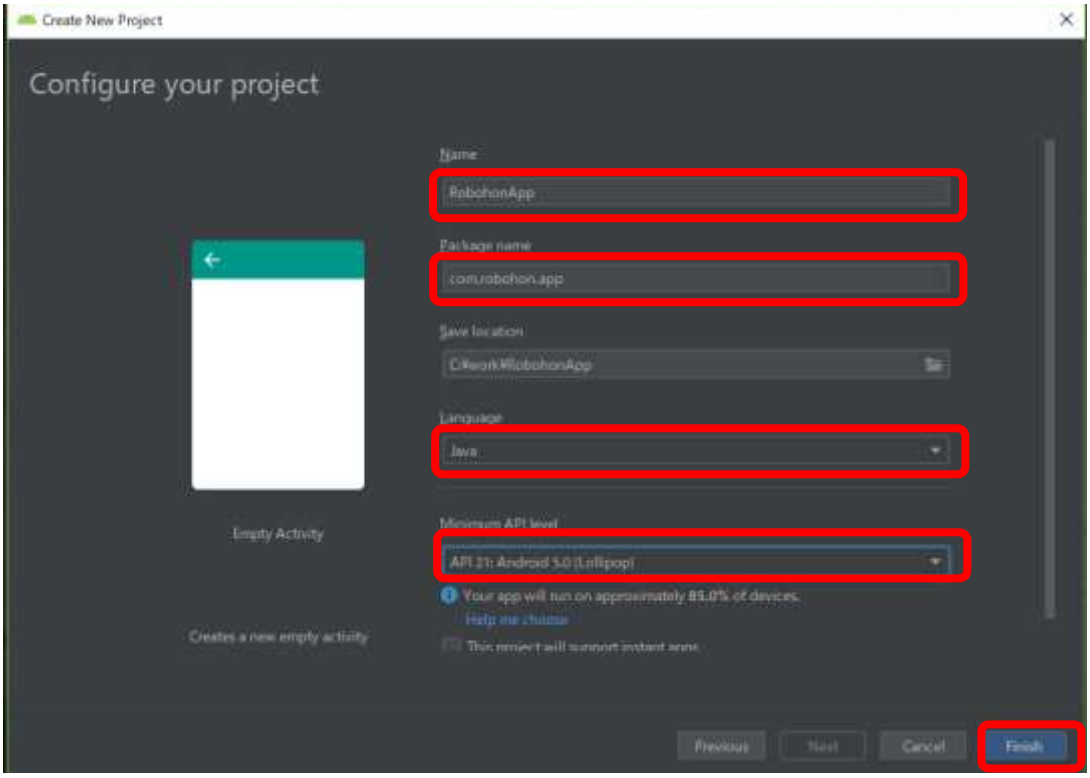



**図 4-11 Configure your project 画面**

4. Project タブの表記を左上のプルダウンより「Android」から「Project」に変更します。

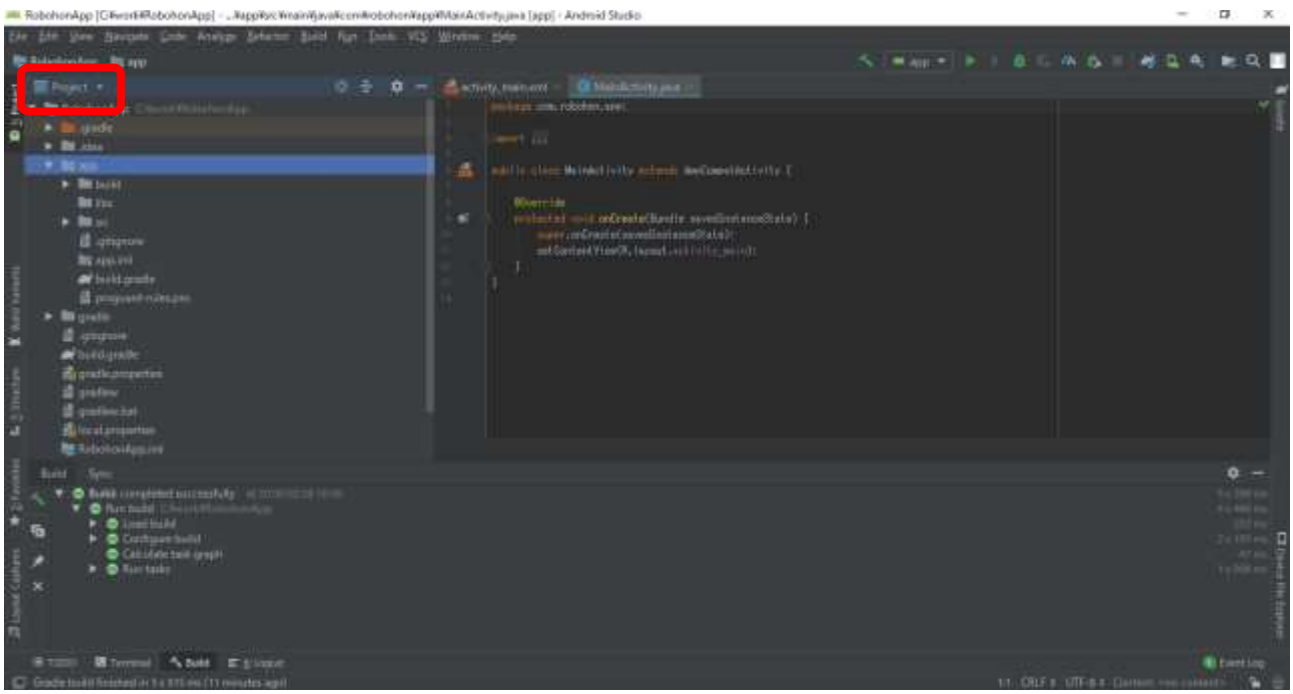
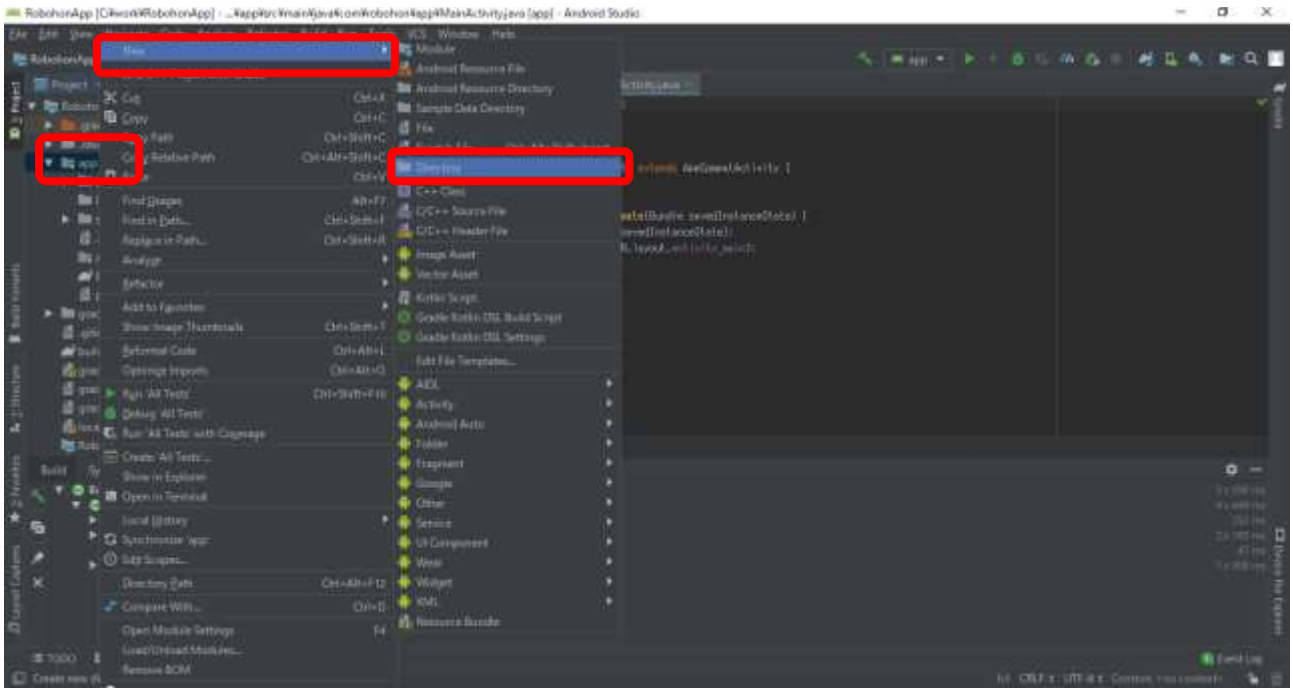


**図 4-12 開発画面**

5. 「アプリ名¥app¥」を選択し右クリック – New – Directory を選択し、「jar」フォルダを作成します。



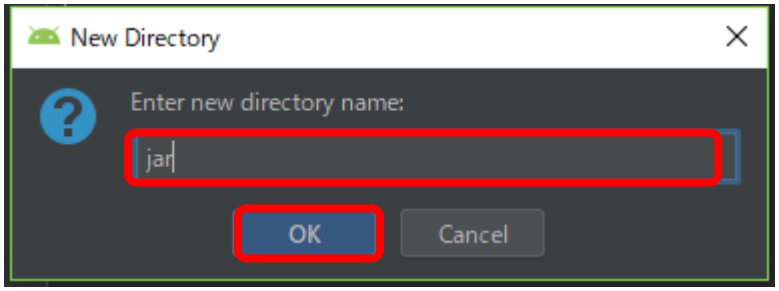



**図 4-13 Directory 作成画面**

6. 作成した jar フォルダ配下に、必要な API の jar ファイルをコピーします。ロボホン用の jar ファイルは、SDK 内に同梱されている jar.zip 内から利用する API に応じて jar ファイルを選択して下さい。 ここでは音声 UI のみコピーします。

   音声 UI：jp.co.sharp.android.voiceui.framework.jar
   プロジェクター：jp.co.sharp.android.rb.projector.framework.jar
   電話帳：jp.co.sharp.android.rb.addressbook.framework.jar
   カメラ：jp.co.sharp.android.rb.cameralibrary.jar
   ダンス：jp.co.sharp.android.rb.rbdance.framework.jar
   メッセージ：jp.co.sharp.android.rb.messaging.framework.jar
   アクション：jp.co.sharp.android.rb.action.framework.jar
   歌：jp.co.sharp.android.rb.song.framework.jar

7. コピーした jar ファイルを右クリックで選択し、「Add As Library…」を選択しダイアログの「OK」ボタン押下します。

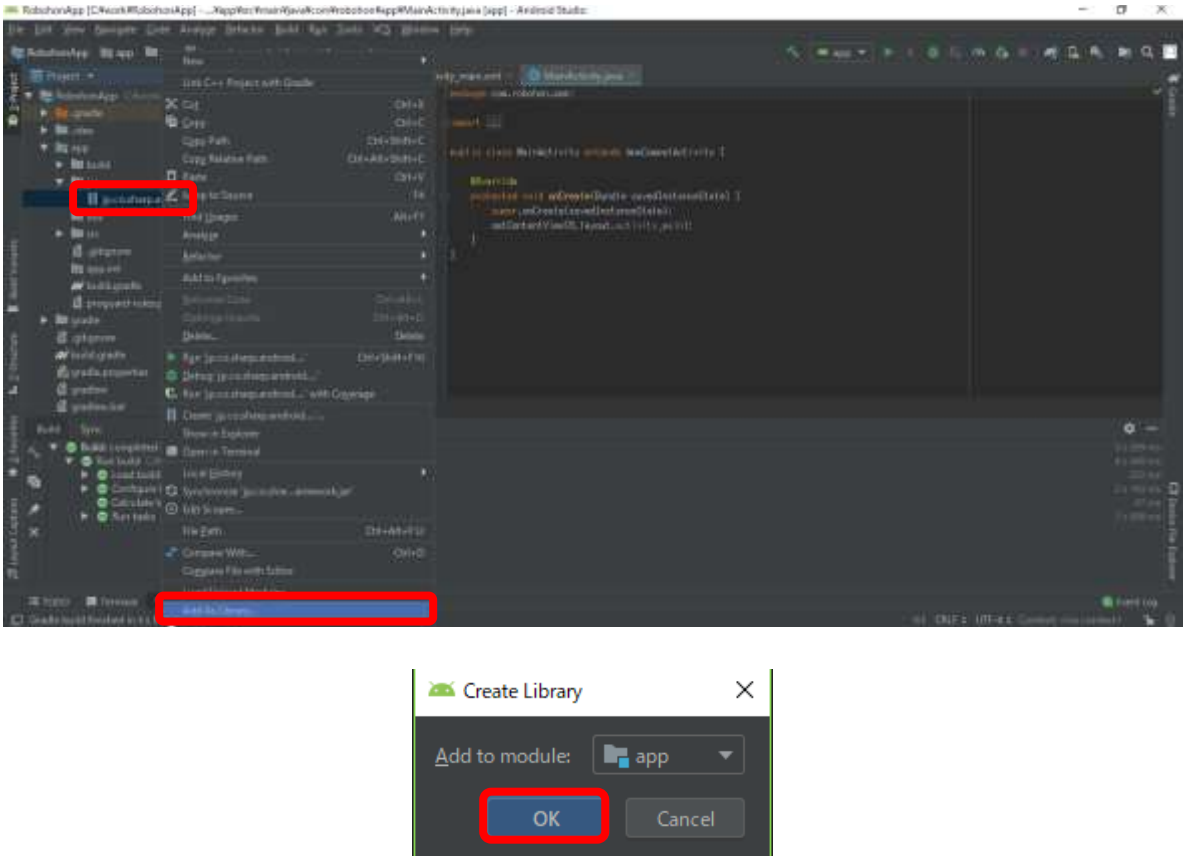



**図 4-14 Library 追加画面**

8. File – Project Structure… を選択し app の Dependencies タブを表示します。
9. 追加した jar ファイルを選択して、Scope を「Compile only」に変更して「OK」を押下します。

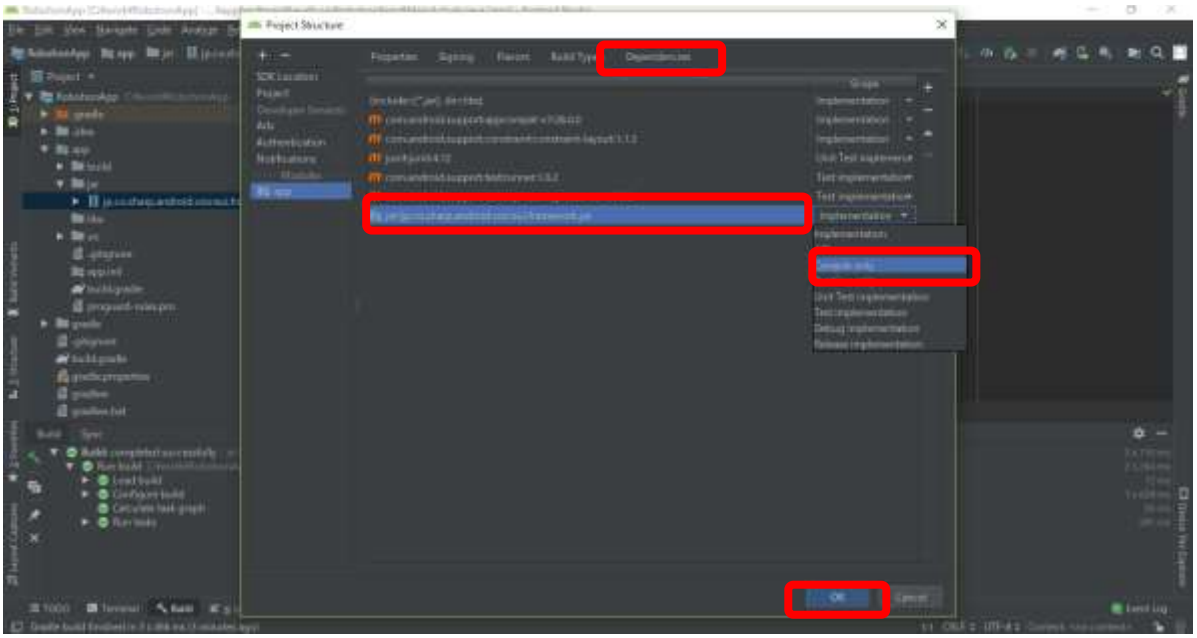



**図 4-15 Project Structure 画面**

10. 音声 UI を利用するための実装一式はテンプレートの実装を流用します。
「¥Template¥app¥src¥main¥java¥com¥robohon¥template¥voiceui」フォルダをそのまま
「アプリ名¥app¥src¥main¥java¥パッケージ名¥」配下にコピーします。
Package 名の違いでエラーが出ている箇所は作成された Package 名に合わせて修正してください。
ScenarioDefinitions に定義している Package 名も合わせて修正してください。

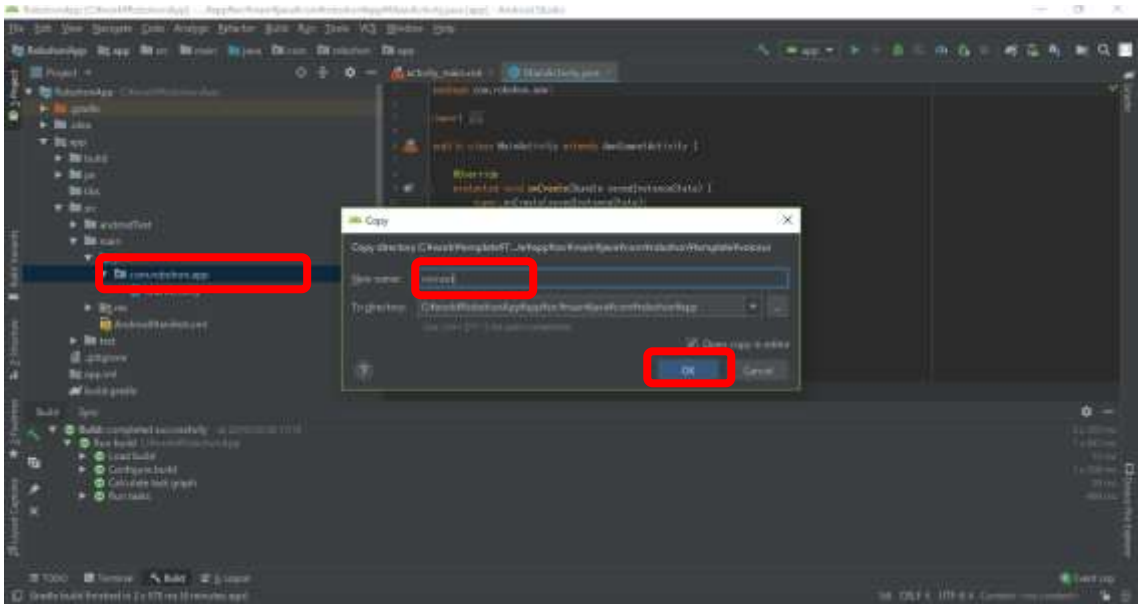


**図 4-16 Voiceui フォルダコピー画面**

11. HVML ファイル一式もテンプレートを流用します「¥Template¥app¥src¥main¥assets」をそのまま
「アプリ名¥app¥src¥main¥」配下にコピーします。その後、「4.2 テンプレートアプリのビルド」手順 6. 7. に従ってファイル名のリネームと Package 名の記載がある箇所を修正します。

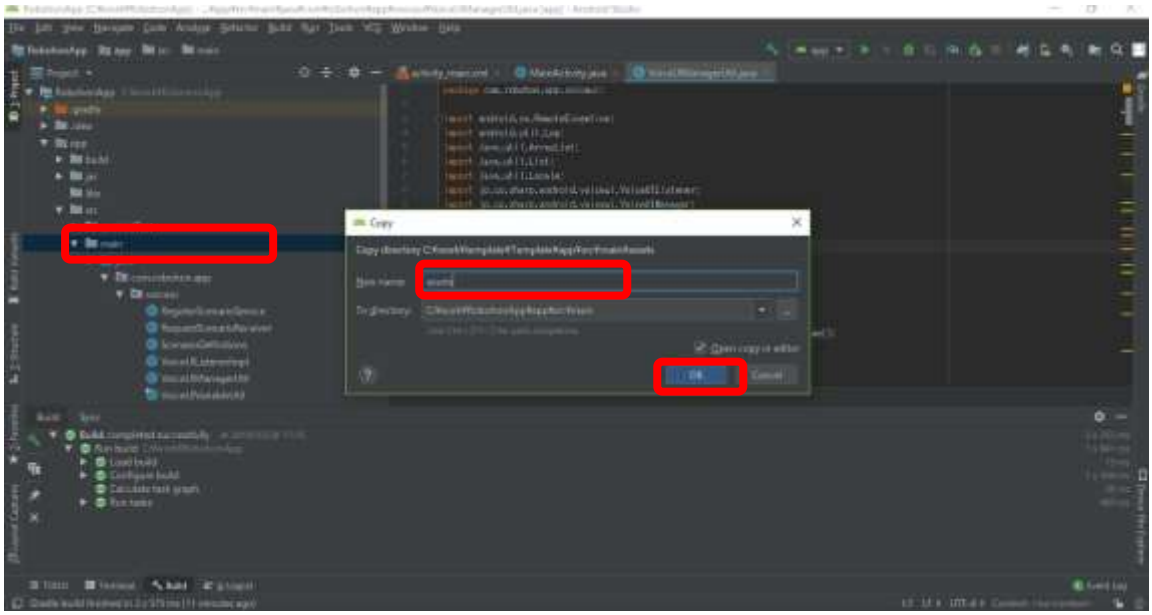


**図 4-17 assets フォルダコピー画面**

12. AndroidManifest.xml に uses-permission、uses-library、アプリアイコン表示用の intent-filter、シナリオ登録用の receiver、service を追加します。追加する内容の詳細については参考資料[1][3]などを確認ください。

【AndroidManifest.xml】

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.robohon.app">

    <uses-permission android:name="jp.co.sharp.android.permission.VOICEUI" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
                <category android:name="jp.co.sharp.android.rb.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <service
            android:name=".voiceui.RegisterScenarioService"
            android:enabled="true"
            android:exported="false" >
        </service>
        <receiver android:name=".voiceui.RequestScenarioReceiver" >
            <intent-filter>
                <action android:name="jp.co.sharp.android.voiceui.REQUEST_SCENARIO" />
            </intent-filter>
        </receiver>
        <uses-library
            android:name="jp.co.sharp.android.voiceui.framework"
            android:required="true" />
    </application>
</manifest>
```

13. app¥build.gradle の targetSdkVersion を 21 にします。※Google Play では API レベル 26 以上を要求されますがロボホンでは新旧ロボホンの互換性を優先するため原則 API レベル 21 を利用します。

14. MainActivity.java に下記の最低限必要な実装を追加していきます。Template からコピペで OK です。
    ・音声 UI のインスタンス生成
    ・HOME イベントの処理
    ・シーンの有効、無効化
    あとは、作成するアプリに応じて参考資料を参照いただき実装してください。以降、「4.1 サンプルアプリのビルド」手順 6.以降と同様です。

【MainActivity.java】

```
public class MainActivity extends AppCompatActivity {
    /**  音声 UI 制御  */
    private VoiceUIManager mVUIManager = null;
    /**  ホームボタンイベント検知. */
    private HomeEventReceiver mHomeEventReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //ホームボタンの検知登録.
        mHomeEventReceiver = new HomeEventReceiver();
        IntentFilter filterHome = new IntentFilter(Intent.ACTION_CLOSE_SYSTEM_DIALOGS);
        registerReceiver(mHomeEventReceiver, filterHome);
    }

    @Override
    public void onResume() {
        super.onResume();
        //VoiceUIManager インスタンス生成.
        if(mVUIManager == null){
            mVUIManager = VoiceUIManager.getService(this);
        }
        //Scene 有効化.
        VoiceUIManagerUtil.enableScene(mVUIManager, ScenarioDefinitions.SCENE_COMMON);
        //アプリ起動時に発話実行
        VoiceUIManagerUtil.startSpeech(mVUIManager, ScenarioDefinitions.ACC_HELLO);
    }

    @Override
    public void onPause() {
```

```
        super.onPause();
        //バックに回ったら発話を中止する.
        VoiceUIManagerUtil.stopSpeech();
        //Scene 無効化.
        VoiceUIManagerUtil.disableScene(mVUIManager, ScenarioDefinitions.SCENE_COMMON);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        //ホームボタンの検知破棄.
        this.unregisterReceiver(mHomeEventReceiver);
        //インスタンスのごみ掃除.
        mVUIManager = null;
    }

    /**
     * ホームボタンの押下イベントを受け取るための Broadcast レシーバークラス.<br>
     */
    private class HomeEventReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
            Log.v(TAG, "Receive Home button pressed");
            // ホームボタン押下でアプリ終了する.
            finish();
        }
    }
}
```

## 4.4　その他の注意点

### 4.4.1　レイアウトの編集

Android Studio の GUI を用いたレイアウト編集時は、仮想デバイスは Galaxy Nexus を、API レベルは 21 を設定してください。

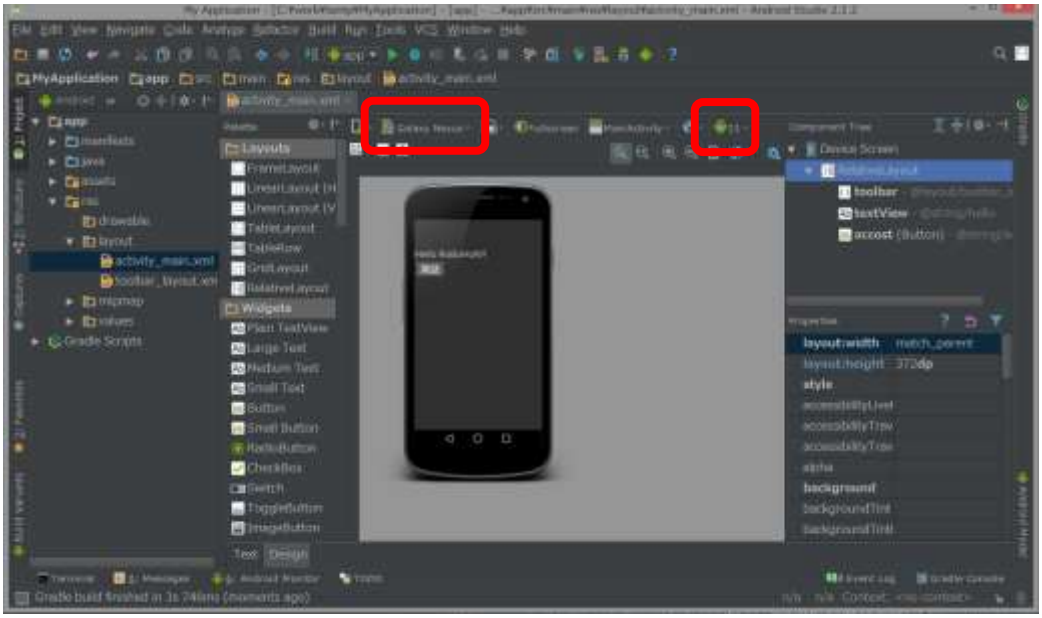


**図 4-18 layout 編集画面**

### 4.4.2　HVML ファイル新規作成時の文字コード

Android Studio で hvml ファイルを新規作成する際には、文字コードを「UTF-8」に指定してください。
テンプレートやサンプルアプリのシナリオをコピーして作成する場合は、UTF-8 が設定されているのでそのまま修正してください。
以下に新規作成する際の具体的な文字コード指定方法を記載します。

1.　hvml ファイルを追加するフォルダで右クリック→New→File を選択し、ファイル名を入力します。

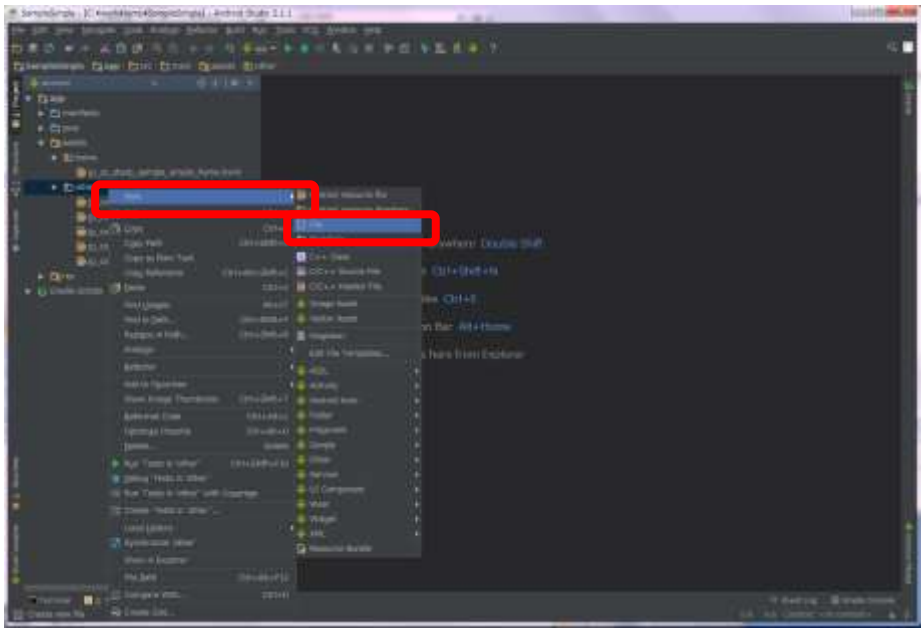


**図 4-19 HVML ファイル新規作成画面**

2.　編集画面の先頭行に”<?xml version="1.0" encoding="utf-8" ?>”または”<?xml version="1.0" ?>”と記述してください。

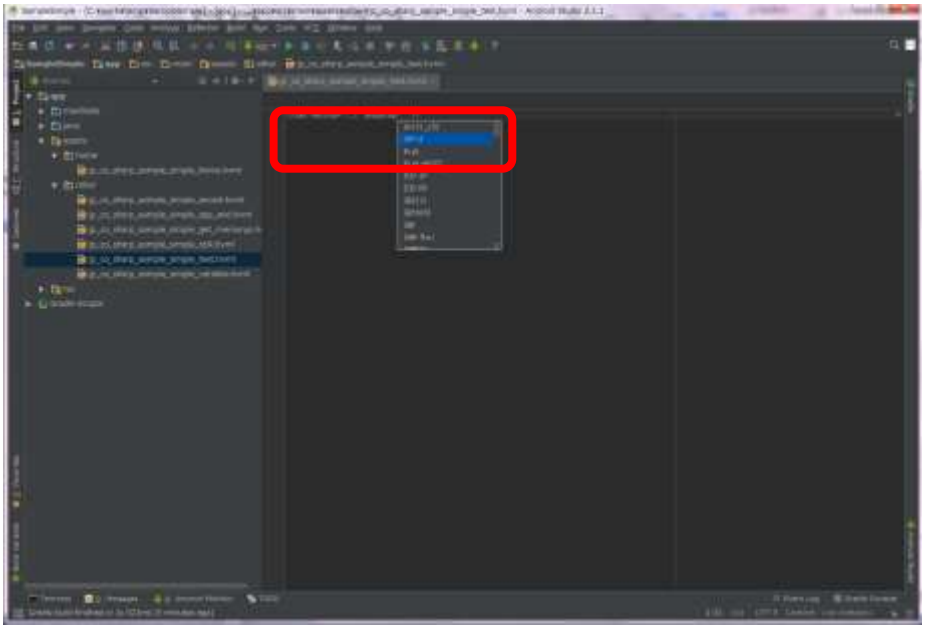


**図 4-20 HVML ファイル編集画面**

## 5. 最後に

ロボホンではアプリ間で共通の UX(User Experience)を維持するために、キャラクター性や世界観、対話の言い回し、アプリの実装方法などのルール(参考資料[3] RoBoHoN_アプリ開発ガイドライン)を定めています。ロボホンのアプリ開発に役立つ情報も記載しておりますので、是非ご参照ください。