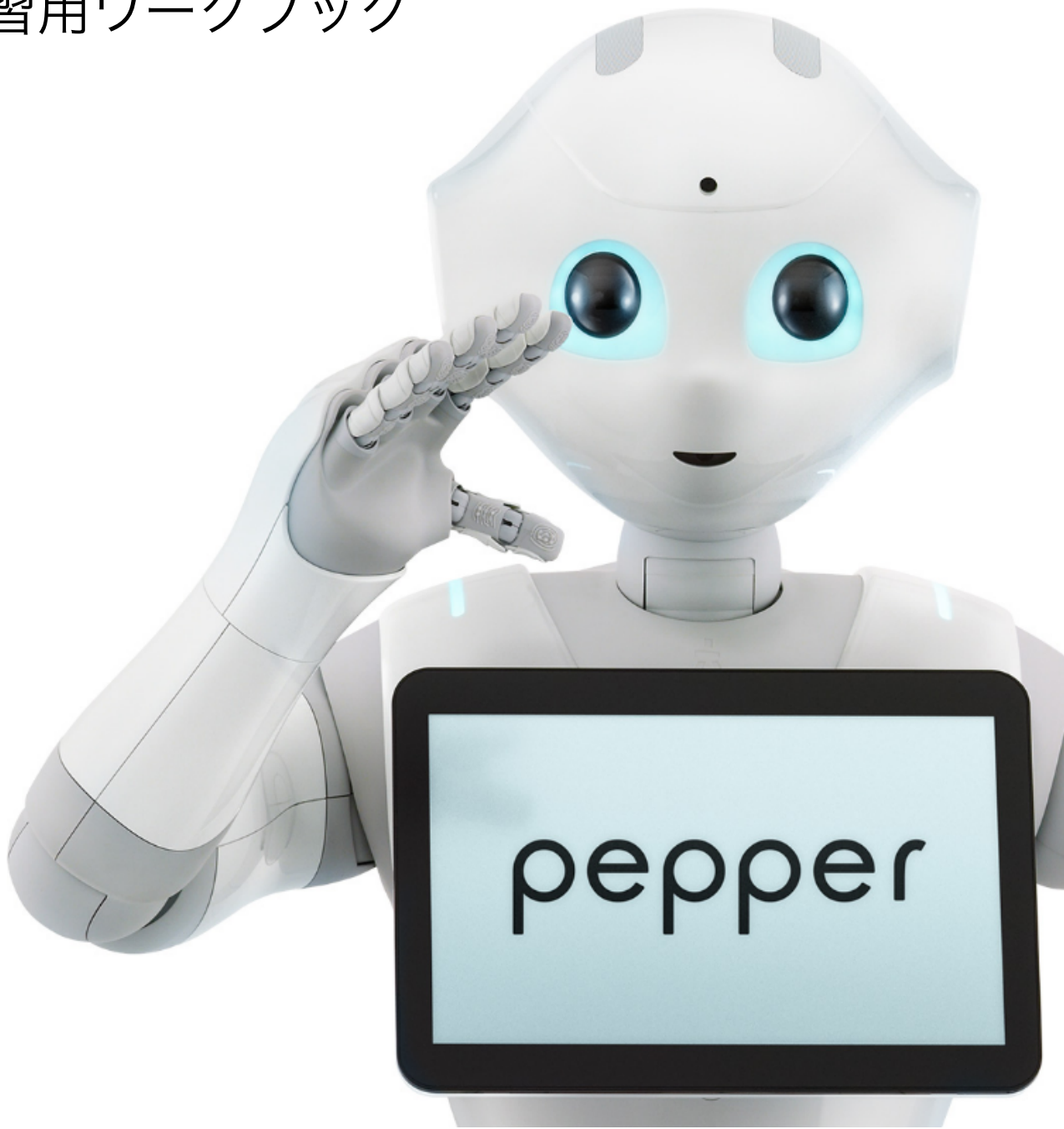


pepper

Pepper パートナープログラム

ロボアプリプロフェッショナル認定
学習用ワークブック



目次

1. はじめに.....	6
1.1. ワークブックについて.....	6
1.1.1. 用語の説明.....	6
1.1.2. 動作環境.....	6
1.2. 前提知識.....	6
1.3. Pepper の種類.....	7
2. Pepper のハードウェア仕様.....	7
2.1. Pepper の大きさ.....	8
2.2. CPU 周りの基本性能.....	9
2.3. マイク・スピーカー・カメラ・センサーの位置.....	10
2.4. 関節制御.....	16
2.5. LED ランプ.....	16
3. Pepper の取り扱い方法.....	17
3.1. 箱からの取り出し方.....	17
3.2. 梱包.....	17
3.3. 移動.....	18
3.3.1. セーフレスト状態.....	18
3.3.2. 手押し移動の場合.....	18
3.3.3. 本機を持ち上げて移動する場合.....	19
3.4. 起動と停止.....	19
3.5. 起動中のモード.....	20
3.6. エラー発生時の対応.....	21
4. Pepper SDK for Android の使い方.....	21
4.1. 動作環境.....	21
4.2. Pepper SDK for Android のインストール.....	22
4.3. Plugin で追加されるボタン.....	23
4.4. Robot SDK Manager.....	24
4.5. Emulator.....	25
4.5.1. Android Emulator.....	25
4.5.2. Robot view.....	26
4.5.3. Motion view.....	27
4.5.4. Log view.....	28
4.5.5. Dialog view.....	29

4.6.	Robot Browser.....	30
4.7.	Animation Editor.....	31
4.8.	Trajectory Edition Tool.....	33
4.9.	QiChat Editor.....	34
4.10.	Animation Browser	35
5.	基本的な開発手順.....	36
5.1.	プロジェクト作成.....	36
5.2.	Activity の作成.....	37
5.3.	Robot Application の設定.....	38
5.4.	Robot Focus と RobotLifecycle の実装.....	39
5.5.	QiSDK.register と unregister.....	41
5.6.	Action と発話の実装.....	42
5.7.	動作確認.....	44
6.	QiSDK の API.....	45
6.1.	Robot Focus と Robot Lifecycle.....	45
6.2.	Action.....	45
6.3.	同期と非同期.....	45
6.3.1.	非同期を利用するケース.....	46
6.3.2.	非同期の基本的な利用方法.....	46
6.3.3.	Java8 の lambda を利用しよう	47
6.4.	Future と Chaining.....	48
6.4.1.	Future メソッドの一覧.....	48
6.4.2.	then... と andThen...	48
6.4.3.	Consume と Apply と Compose.....	49
6.5.	Autonomous	50
6.6.	会話中の表示.....	50
6.7.	発話 (Say).....	51
6.8.	聞き取り (Listen)	51
6.9.	チャット (Chat).....	51
6.9.1.	基本的な使い方.....	51
6.9.2.	Topic ファイル.....	52
6.9.3.	QiChat の記述方法	53
6.9.4.	Dynamic concept、QiChatVariable	56
6.9.5.	Bookmark	56
6.9.6.	Executor.....	57
6.9.7.	独自の Chatbot.....	57

6.10.	Human.....	58
6.11.	Touch.....	58
6.12.	Camera	58
6.13.	動き (Animate)	58
6.14.	固定の移動 (Trajectory)	59
6.15.	LookAt.....	59
6.16.	GoTo	59
6.17.	Frame.....	60
6.18.	地図の作成(Localize)	60
7.	お仕事かんたん生成 3.0	60
8.	Robot Suite.....	61
9.	インタラクション分析.....	61
10.	UX.....	61
10.1.	イントネーションと抑揚の調整.....	61
10.1.1.	調整できる項目.....	62
10.1.2.	その他の調整方法.....	62
10.1.3.	調整済みテキストのサンプル集.....	63
10.2.	モーション作成時の注意点.....	63
10.2.1.	ポーズを1フレーム目に登録しない.....	64
10.2.2.	ポーズを連続で登録しない.....	64
10.2.3.	中途半端な姿勢を維持しない.....	64
10.2.4.	腰の角度を深くしすぎない.....	64
10.2.5.	タイムラインは500フレーム以内.....	64
10.2.6.	センサーの死角や予測できない利用者の動き.....	64
10.2.7.	人と目を合わせる.....	65
10.2.8.	開発時.....	65
10.3.	ディスプレイ表示の注意点.....	65
10.3.1.	字はできるだけ大きくする.....	65
10.3.2.	UI要素の配置が上下左右に偏らないようにする.....	66
10.3.3.	文字と背景のコントラスト比はできるだけ高くする.....	66
10.3.4.	ボタン連打に対応する.....	66
10.3.5.	操作は音声とディスプレイ両方でできるようにする.....	66
10.3.6.	画像作成時の注意点.....	66
10.4.	演出上の注意点.....	66
10.4.1.	人を惹きつける方法.....	67
10.4.2.	安全性の確保.....	67

11. その他.....	67
11.1. 運用時の注意点	67
11.1.1. ネットワーク環境.....	67
11.1.2. 設置場所.....	68
11.1.3. 複数台体制の検討.....	68
11.2. 規約上の注意.....	68
11.2.1. Pepper の商標・キャラクターについての制限.....	68
11.2.2. メディア掲載・公共環境での稼働について.....	69
11.2.3. 商標について.....	70
11.2.4. Pepper の画像について.....	70
11.3. [参考] qicli コマンド.....	71
11.3.1. SSH 接続の方法	71
11.3.2. NAOqi 側のログファイル.....	71
11.3.3. セーフティ機能解除.....	71
11.3.4. ログを確認する.....	72

1. はじめに

1.1. ワークブックについて

本書はソフトバンクロボティクス株式会社 Pepper パートナープログラムが、「ロボアプリプロフェッショナル認定試験」を受験する技術者を対象に作成した学習用ワークブックとなります。内容等確認して頂く場合、Pepper 本体が必要になる場合もありますので、ご了承ください。

1.1.1. 用語の説明

本書で使用する用語の一部を、以下で説明します。

表 1-1 用語の説明

用語	説明
ロボアプリ	Pepper 向けに開発されたアプリ。

1.1.2. 動作環境

本書に記載されているサンプルコードは以下の環境にて動作させることを前提に作成されています。

表 1-2 動作環境

環境	バージョン
NAOqi	2.9.1
Android	6.0
Android Studio	3.2
QiSDK	1.4.4

※ 2019/1/30 時点の最新版

1.2. 前提知識

本書は以下の前提知識を習得されている方向けに作成されています。前提知識の習得に関しては、事前に各自で実施しておく必要があります。

表 1-3 前提知識

前提知識	主な用途
Android	<p>Android Studio 対応版 Pepper 向けのロボアプリは、Pepper SDK for Android(QiSDK)を用いて、Android アプリとして開発を行います。このためロボアプリ開発には Android の基本的な知識が必要です。</p> <p>本書では、以下のようなキーワードについて、理解されていることを前提に説明を行います。</p> <ul style="list-style-type: none"> ● Java ● Activity ● Activity の LifeCycle ● View ● Listener ● Thread ● ANR

1.3. Pepper の種類

Pepper には、大きく以下 2 種類のサービスが存在します。

表 1-4Pepper の種類

種別	説明
Pepper for Home (一般販売モデル)	家庭向けに販売されているモデルです。会話アプリなど家庭向けのアプリが標準搭載されています。
Pepper for Biz	法人向けに販売されているモデルです。 お仕事かんたん生成と呼ばれる CMS にて、受付機能や接客機能のカスタマイズやロボアプリのリモートインストールなどを行うことができます。また、一般販売モデルと標準でインストールされているアプリが異なります。

2. Pepper のハードウェア仕様

本項では、Pepper に搭載されているデバイスやセンサーのなどのハードウェアについて説明します。

2.1. Pepper の大きさ

Pepper の身長は約 120 cm で、重さは 29kg 程度です。手を下げた状態だと横幅は 48cm 程度ですが、手を左右に広げると 120cm 程度まで広がります。

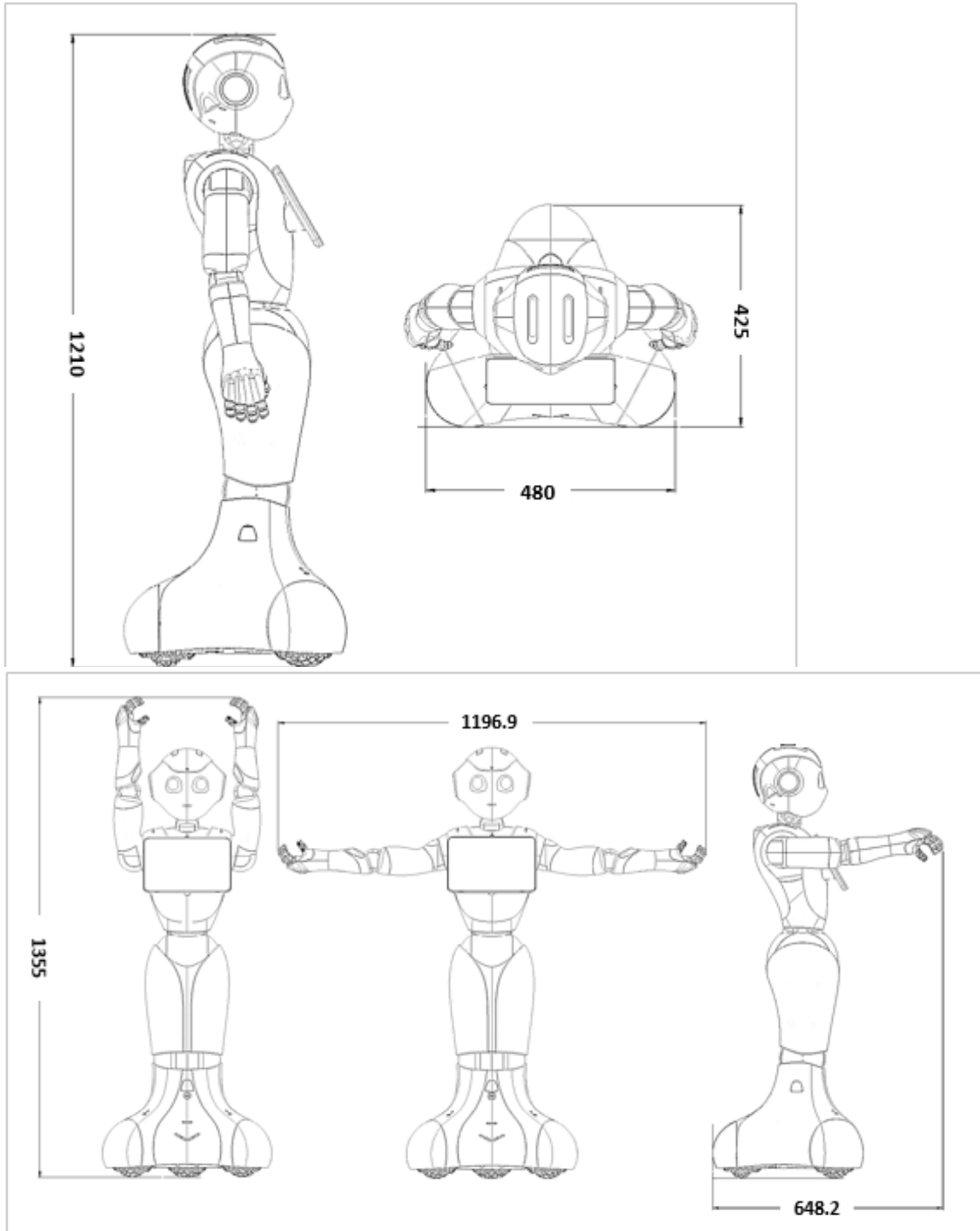


図 2-1 Pepper の大きさ

2.2. CPU 周りの基本性能

ロボアプリは Pepper のディスプレイにインストールされ動作しますが、Pepper の各種センサーやモーターは Pepper の頭にある CPU で制御されます。ディスプレイと Pepper の頭は USB ケーブルで接続されており、必要な通信を行っています。

表 2-1 Pepper の頭の基本性能

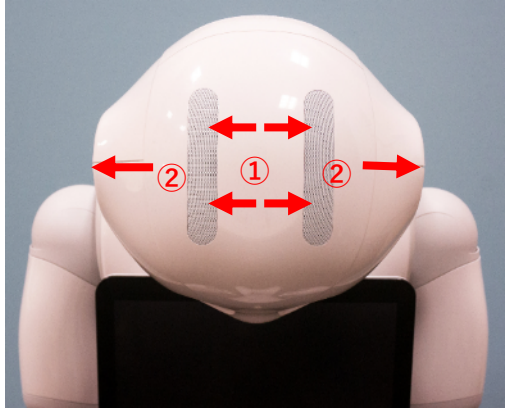
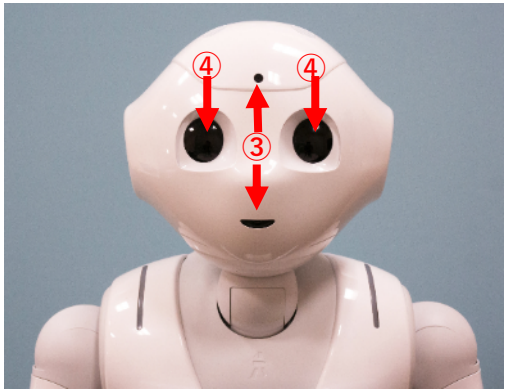
項目	性能
OS	NAOqi OS
CPU	1.91GHz quad-core Atom E3845
RAM	4GB DDR3

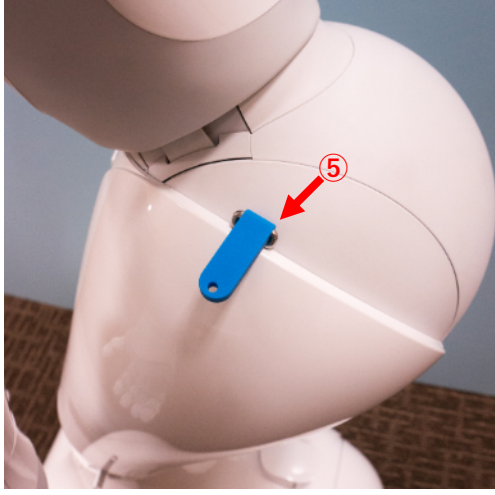

表 2-2 ディスプレイの基本性能

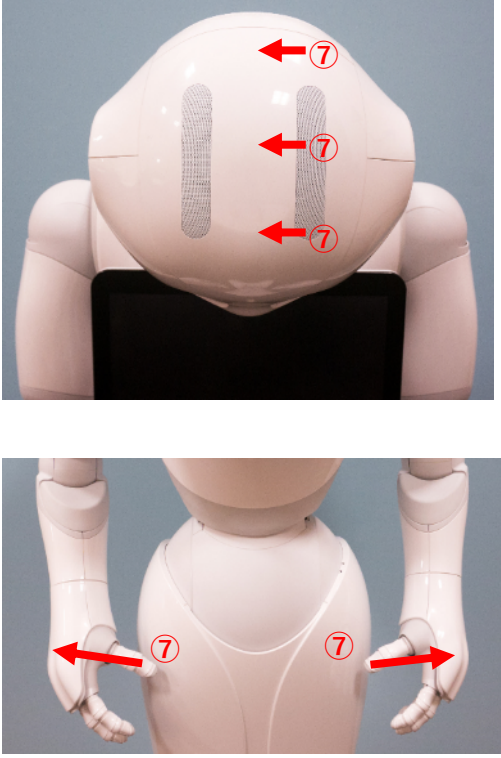

項目	性能
OS	Android 6.0
CPU	1.3 GHz quad-core ARM
メモリ(RAM)	DDR3 SDRAM: 1GB (512MB * 2)
ストレージ	32GB
ディスプレイ	サイズ：10 インチ、解像度：1280*800

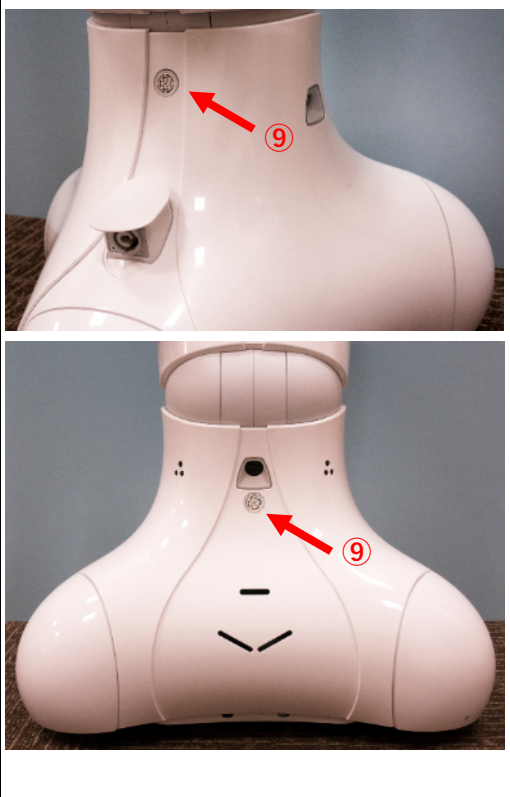
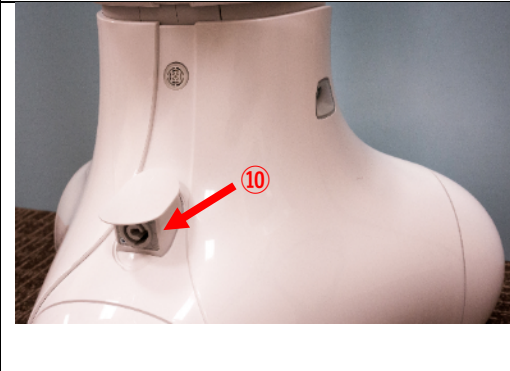
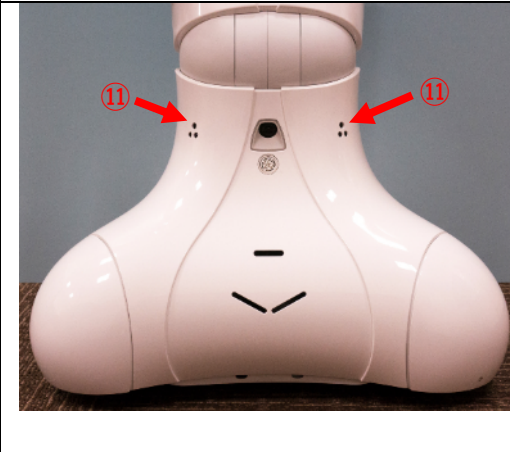
2.3. マイク・スピーカー・カメラ・センサーの位置

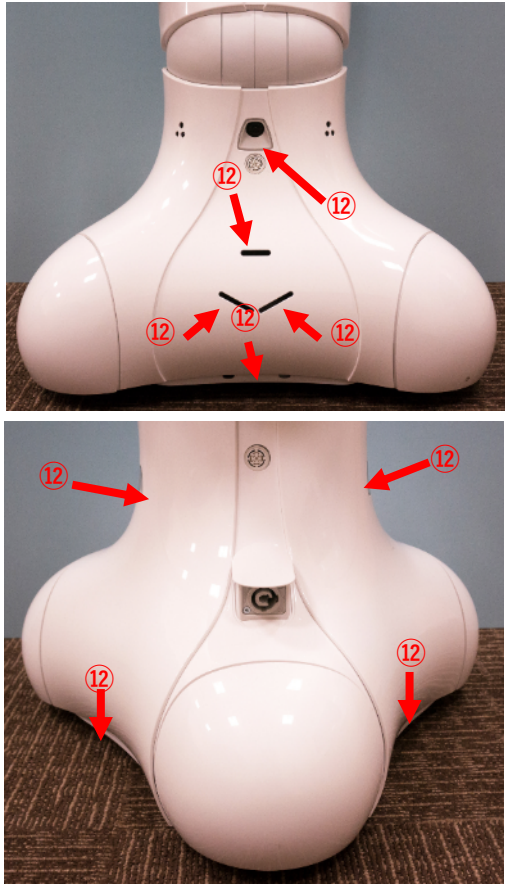
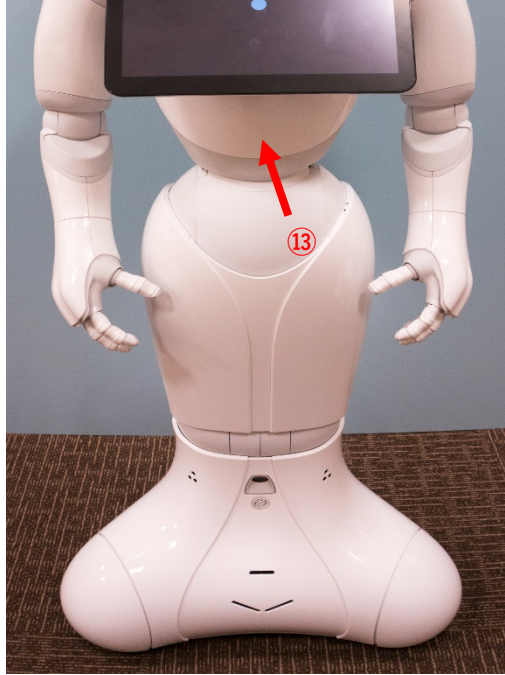
表 2-3 マイク・スピーカー・カメラ・センサーの位置

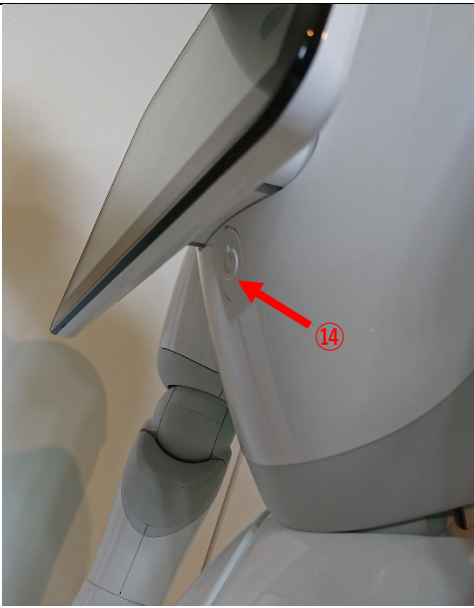
	①マイク	頭部の左の前後、 右の前後
	②スピーカー	頭部の左右の耳の部分
	③2D カメラ	額と口
	④3D カメラ	右目 (赤外線照射) 左目 (赤外線検出) ※ 機体のバージョンにより仕様 が異なります。 v18: 1280x720 / 15fps v18a: 320x240 / 20fps ※ 人認識などの精度には影響 しますが、ロボアプリ開発 上の差異はありません。 ※ V18 にはステレオカメラも 搭載されています。詳細に ついては以下のリンクをご 確認ください。 http://doc.aldebaran.com/2-5/family/pepper_technical/video_Stereo_pep.html

	<p>⑤解除キー 腰</p>	<p>腰のブレーキ解除キー</p>
	<p>⑥解除キー 脚</p>	<p>脚のひざ部のブレーキ解除キー</p>

	<p>⑦タッチセンサ ー</p>	<p>頭部中央の前方、中央、後方 左右の手の甲</p>
	<p>⑧バンパーセン サー</p>	<p>脚部の前方の左右と後方</p>

	<p>⑨ソナー</p>	<p>脚部の正面と後方</p>
	<p>⑩充電フラップ</p>	<p>脚部後部の AC アダプタの充電プラグを接続する</p>
	<p>⑪赤外線センサ</p> <p>—</p>	<p>脚部2か所</p>

	<p>⑫レーザーセンサー</p>	<p>脚部の正面と左右前方の3か所 穴が開いているところ 足元の前、左後方、右後方</p>
	<p>⑬ジャイロセンサー</p>	<p>胸部と脚部</p>

	⑭胸部ボタン	ディスプレイの裏側
---	--------	-----------

2.4. 関節制御

ロボアプリから制御できる Pepper の関節について把握しておきましょう。

表 2-4 Pepper の関節

部位		動き
首		<ul style="list-style-type: none"> ・首を支点にして頭を左右に振る動き(ヨ一) ・首を支点にして頭を上下に振る動き(ピッチ)
腕	肩	<ul style="list-style-type: none"> ・肩を支点にして腕全体を前後に回す動き(ピッチ) ・肩を支点にして腕全体を左右に開く動き(ロール)
	肘	<ul style="list-style-type: none"> ・肘を支点にして腕を上下にひく動き(ロール) ・肘を支点にして腕を左右に振る動き(ヨ一)
	腕	<ul style="list-style-type: none"> ・腕を支点にして手を裏表に回す動き(ヨ一)
	手	<ul style="list-style-type: none"> ・手のひらを開閉する動き
下半身	腰	<ul style="list-style-type: none"> ・腰を支点にして上半身を前後に傾ける動き(ピッチ) ・腰を支点にして上半身を左右に傾ける動き(ロール)
	膝	<ul style="list-style-type: none"> ・膝を支点にして上半身を前後に傾ける動き(ピッチ)

2.5. LED ランプ

Pepper には肩、目、耳に LED ランプが搭載されています。肩の LED ランプは NAOqi の状態に合わせて点灯します。

表 2-5 肩の LED ランプ

色 (光り方)	状態
白 (点灯)	正常
青 (点灯)	音声認識中
緑 (点滅)	通知情報あり
黄 (早く 2 回点滅)	警告
赤 (早く 2 回点滅)	エラー
赤 (遅く点滅)	使用不可

胸部ボタンを 1 回押すことで、Pepper が機体名、IP アドレスを発話した後、通知内容を発話します。複数の通知がある場合は緊急性の高いものから順に発話されます。通知の内容については、以下の通知情報一覧を参照してください。

https://doc.robot.softbank.jp/pepper_biz/manual/index/topics_detail16/id=86

3. Pepper の取り扱い方法

Pepper が届いたら梱包箱から安全に取り出す必要があります。取り出した後は Pepper を適切な場所へ移動してください。間違った手順で Pepper を扱うと故障や転倒の原因になりますので、本項で基本的な扱い方を確認しましょう。

3.1. 箱からの取り出し方

Pepper を箱から取り出す際の手順は、下記のとおりです。

1. 梱包箱を立てた状態で蓋を開けてください（梱包箱の上下を間違わないように注意）。
2. 梱包箱の上部のミミ（フラップ）を固定用の穴に差し込み固定します。
3. アクセサリーボックス（充電器一式）を取り出します。
4. Pepper を押さえながら緩衝材を取り出します（関節が固定されていないため、Pepper が前のめりになり、倒れやすいので注意）。
5. 首を前に倒して、脇を抱えて持ち上げ、外に出してください。

3.2. 梱包

Pepper を梱包する際の手順は、下記のとおりです。

1. 蓋を開けます。
2. 梱包箱の上部のミミ（フラップ）を固定用の穴に差し込み固定します。
3. 梱包箱から緩衝材を取り出します。
4. Pepper をセーフレストの姿勢にし、シャットダウンします。
5. 電源 OFF の状態で、膝と腰の部分に付属の解除キーを差し、関節の固定を解除します。
6. 緊急停止ボタンを押し、意図しない動作を防止してください。
7. 充電フラップを閉じます。（閉じない状態で梱包すると封緘時に緩衝材が浮き、故障の原因になりますので必ず閉じてください。）
8. Pepper の前面から脇を抱えて持ち上げ、足の方から押し込みます。
9. 梱包箱をゆっくり倒してください。

10. 頭、肩、手、腕を押し込みます。
11. アクセサリーボックス（充電器一式）を入れ、緩衝材（胸部のディスプレイ用および蓋）をはめます。
12. 梱包箱を閉じます。

3.3. 移動

Pepper を移動させる際の注意点は以下の通りです。

3.3.1. セーフレスト状態

セーフレスト状態は、Pepper の重心を低く保つことができ、倒れにくい姿勢です。Pepper はスリープモード、レストモード、もしくはシャットダウンの際にセーフレスト状態になります。



図 3-1 セーフレスト状態の Pepper

3.3.2. 手押し移動の場合

1. 姿勢を安定させるため、セーフレストの姿勢にします。
2. 腰とひざの解除キーがついている場合は取り外します。
3. 充電フラップを開きます。
4. 重心が足元にあるため、肩に手を置き、もう一方の手をおしりにあてて静かに前に押し移動させます。

5. 周囲（特に進行方向の足元）に障害物がないか確認しながら移動させます。
- ※ 移動をしている最中に緊急停止ボタンや胸部ボタンを押さないよう注意してください。
 - ※ 電源 ON 時は、Pepper のホイール以外は動作が継続しているので注意してください。
 - ※ 移動は電源 OFF、もしくはレストモードで行ってください。スリープモードでもセーフレスト状態になりますが、頭部のタッチセンサーを触るだけで動き出してしまいうため危険です。
 - ※ 充電フラップを開けることでオムニホイールを停止することが出来ます。手押しで移動する際に、オムニホイールが急に動作すると、転倒などに繋がりやすく非常に危険です。手押しで移動する際は充電フラップを開けるのを忘れないようにしましょう。安全上、オムニホイールの動作を制限したい場合にも、充電フラップの開閉を利用することが出来ます。

3.3.3. 本機を持ち上げて移動する場合

1. シャットダウンし、セーフレスト姿勢にします。
2. 緊急停止ボタンを押します。
3. 背後から本機を支えながら腰とひざの解除キーを取り付けます。
4. 背後から胸の下に手を入れ、抱きかかえるようにして持ち上げて移動させてください。

3.4. 起動と停止

各種起動と停止の方法や注意事項について以下に記載します。

表 3-1 起動と停止

名称	概要
通常起動	電源 OFF のときに胸部ボタンを 1 回押すと起動します。
長押し起動	電源を入れる時、電源ボタンを 4 秒長押しして、肩の LED が青く点滅し始めたらずちに離します。この方法は、各部位にある基盤のファームウェアの再書き込みを行います。Pepper 本体のエラー発生後に、この方法で起動しても問題が改善されない場合は、ハードウェア的な故障を疑ってください。
シャットダウン	電源 ON 時に胸部ボタンを 3 秒間押すとシャットダウンを行います。

強制シャットダウン	電源 ON 時に胸部ボタンを 8 秒以上押しと強制シャットダウンを行います。強制シャットダウンを行うと、アプリの終了処理などが呼ばれずデータが保存されないことがありますので注意してください。
緊急停止	<p>緊急時は首の後ろのカバーの中にある緊急停止ボタンを押すと、頭および体への電気供給がすべて停止し、即座に動作を終了することができます。</p> <p>緊急停止ボタンを押すと、起動にロックがかかり、ロックを解除するまで電源は入りません。緊急停止ボタンをつまんで時計回りに回すことでロックを解除することができます。</p> <p>※ 緊急停止ボタンは緊急時以外に使用しないでください。</p> <p>※ 緊急停止ボタンはカバーの上からたたき押すことも可能です。</p> <p>※ 移送時には通常のシャットダウンを行った後に、緊急停止ボタンを押して起動のロックをかけてください。移送時に箱の中で電源が入ると故障の原因となります。</p>

3.5. 起動中のモード

Pepper 起動中の基本的な状態について、以下に記載します。

表 3-2 起動中のモード

名称	概要
ウェイクアップモード	Pepper 起動後の状態です。
スリープモード	<p>ウェイクアップモード時に額のカメラを隠しながら、前頭部の一番手前の頭部タッチセンサーを 3 秒タッチすると、スリープモードになります。スリープモード時に頭部のタッチセンサーを触ると、ウェイクアップモードになります。</p> <p>スリープモード中、ロボアプリは発話やモーションが行えません。また、画面は黒くなり操作はできません。</p>
レストモード	胸部ボタンを 2 回押しとレストモードになります。レストモード時に胸部ボタンを 2 回押しとウェイクアップモードになります。

	レストモード中、ロボアプリは発話やモーションが行えません。ただし、画面は黒くならず、ディスプレイ上でロボアプリの操作を続けることが可能です。
--	--

3.6. エラー発生時の対応

Pepper 利用中には様々なエラーが発生する可能性があります。よく見かけるエラーについて説明します。

表 3-3 エラーの一覧

名称	概要
高温時のエラーを確認した場合	Pepper の関節(モーター)やプロセッサに負荷がかかり Pepper の内部温度が高くなった場合、エラーを検出する可能性があります。電源を切り 30 分以上休ませたあと、Pepper の長押し起動をお試してください。また起動後にアプリケーションを最新版にアップデートしてください。
充電に関するエラーを確認した場合	充電残量が低下すると Pepper の肩の LED が黄色点滅になり、充電が必要な旨を Pepper が発声します。充電を十分に行ったあと、Pepper を長押し起動してください。
機体のエラーの場合、もしくはエラー内容が不明な場合	Pepper の長押し起動をお試してください。改善しない場合は最新版にアップデートください。

4. Pepper SDK for Android の使い方

Android Studio 対応版 Pepper のロボアプリは Android Studio を使用して開発します。Pepper SDK for Android は Android Studio Plugin で、一連のグラフィカルツールと Java ライブラリである QiSDK を提供します。QiSDK が提供する API を利用することで Android の Activity から簡単に Pepper を制御することが出来ます。

4.1. 動作環境

Pepper SDK for Android のインストールに必要な動作環境は、下記のとおりです。

表 4-1 Pepper SDK for Android の動作環境

項目名		動作環境
OS	Linux	Ubuntu 16.04 Xenial Xerus - 64bits のみ
	Windows	Microsoft Windows 10 - 64bits のみ
	Mac	Mac OS X 10.12 Sierra
Android Studio		バージョン 2.3 以上

4.2. Pepper SDK for Android のインストール

Pepper SDK for Android のインストール手順については公式サイトで最新の情報を確認の上、インストールを行ってください。Plugin 自体は Android Studio の設定にある Plugin から簡単にインストール可能ですが、事前準備が煩雑なため注意が必要です。

[Pepper SDK for Android -インストール手順]

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch1_gettingstarted/installation.html

4.3. Plugin で追加されるボタン

Pepper SDK for Android の Plugin をインストールすることで、Toolbar に5つのボタンが追加されます。Plugin をインストールしたのにボタンが表示されない場合は、ViewメニューのToolbarにチェックが入っているか確認してください。追加されるボタンの概要は以下の通りです。

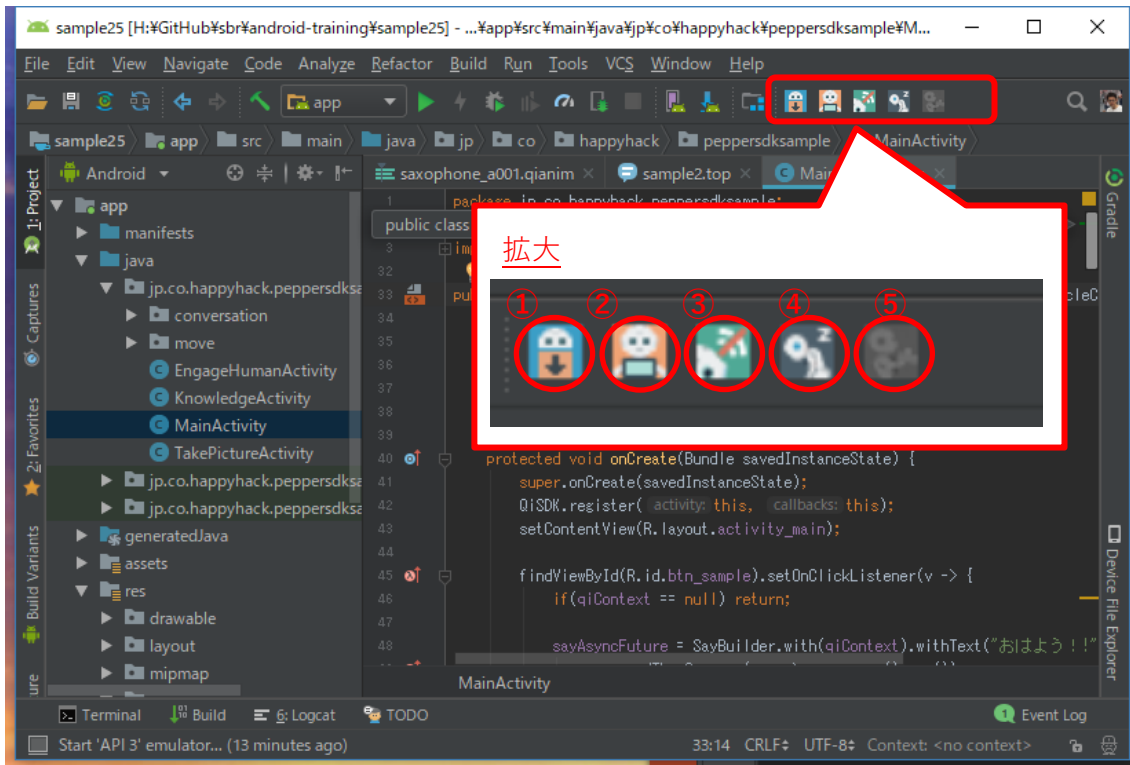


図 4-1 Plugin で追加されるボタン

表 4-2 Plugin で追加されるボタン一覧

No	ボタン名称	概要
①	Robot SDK Manager	Robot SDK Manager を開くためのボタン。
②	Emulator	Emulator を起動するためのボタン。
③	Connect	Robot Browser を開くためのボタン。
④	Rest	接続中の Pepper を Rest 状態にするためのボタン。
⑤	Enable animation mode	アニメーションモードの ON/OFF を切り替えるためのボタン。

4.4. Robot SDK Manager

Robot SDK Manager はインストールされている Pepper SDK のバージョンを管理する際に使用します。

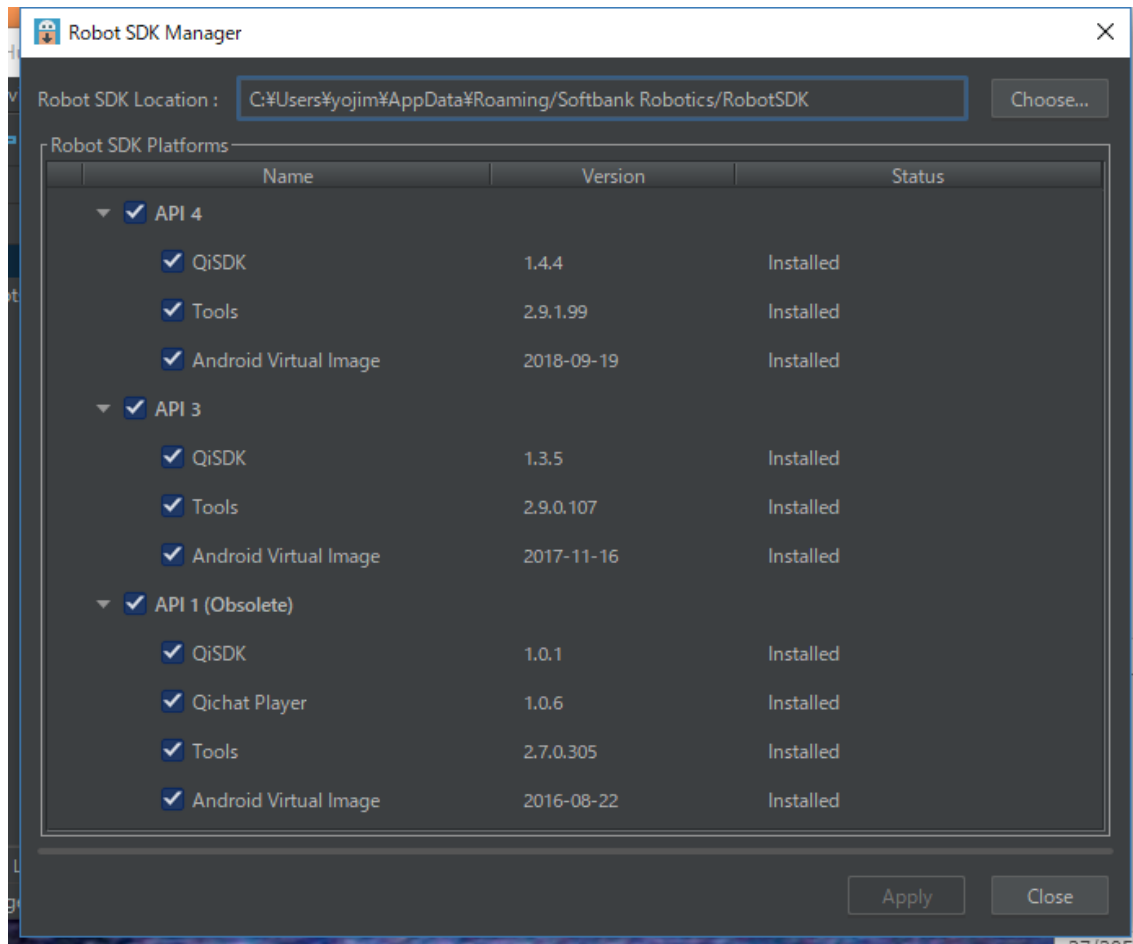


図 4-2 Robot SDK Manager

4.5. Emulator

Emulator ボタンをクリックすることで Pepper の Emulator を起動することができます。大きく Android Emulator と Robot Viewer の二つが開き、Robot Viewer は Robot view、Motion view、Log view、Dialog view のタブから構成されます。

4.5.1. Android Emulator

Android Emulator は Pepper のディスプレイをエミュレートします。

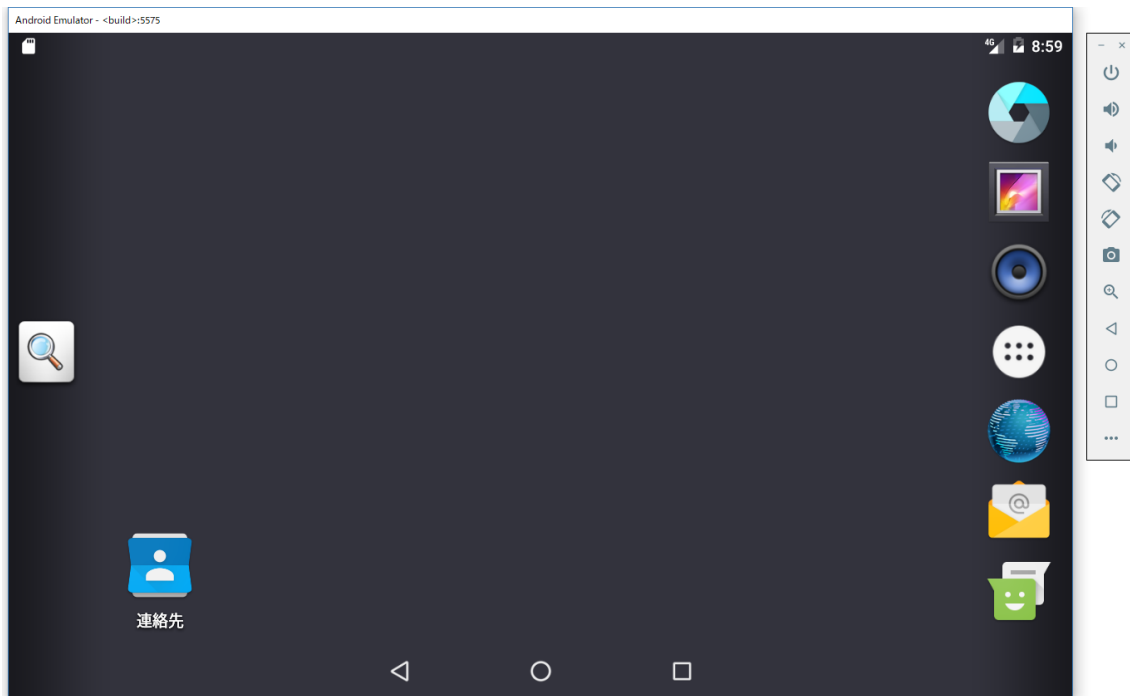


図 4-3 Android Emulator

4.5.2. Robot view

Robot view は Pepper 本体の動きを 3D モデルでシミュレートします。Pepper が発話した場合は、吹き出しにて表示されます。また、Pepper 本体と接続している際は、背景が緑色になり、Pepper 本体の状態を再現します。

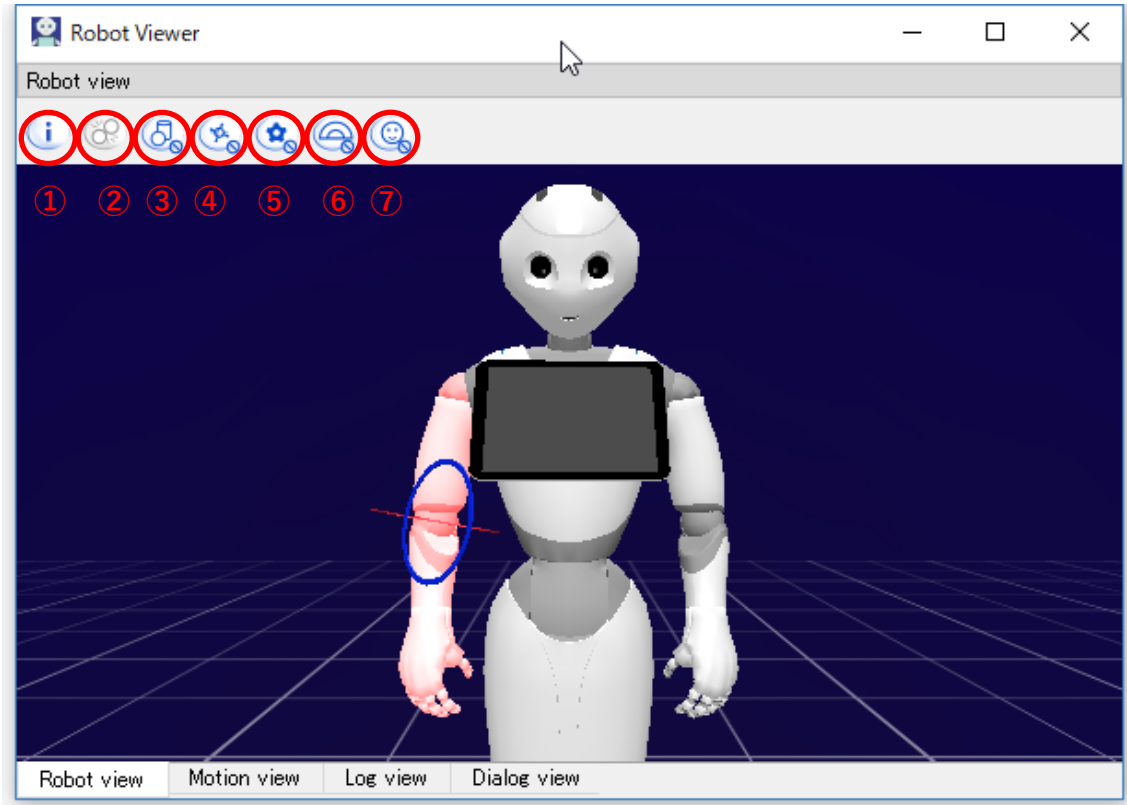


図 4-4 Robot view

画面上部にある各ボタンは以下の動作を行います。

表 4-3 Robot view のボタン

No.	概要
①	3D モデル情報の表示の ON、OFF を切り替える。
②	障害物検知情報の表示の ON、OFF を切り替える。
③	オブジェクトの表示の ON、OFF を切り替える。
④	ロボットの位置情報の表示の ON、OFF を切り替える。
⑤	モーターの位置情報の表示の ON、OFF を切り替える。
⑥	エンゲージメントゾーンの表示の ON、OFF を切り替える。
⑦	認識中の人の表示の ON、OFF を切り替える。

4.5.3. Motion view

Motion view は Robot view で選択された部位の関節の角度を閲覧、編集することが出来ます。



図 4-5 Motion view

Pepper 本体と接続中は Motion view の操作に合わせて本体も動作するので注意して下さい。

4.5.4. Log view

Log view は Pepper の頭側の CPU で動作している NAOqi フレームワークの一部ログを確認することができます。

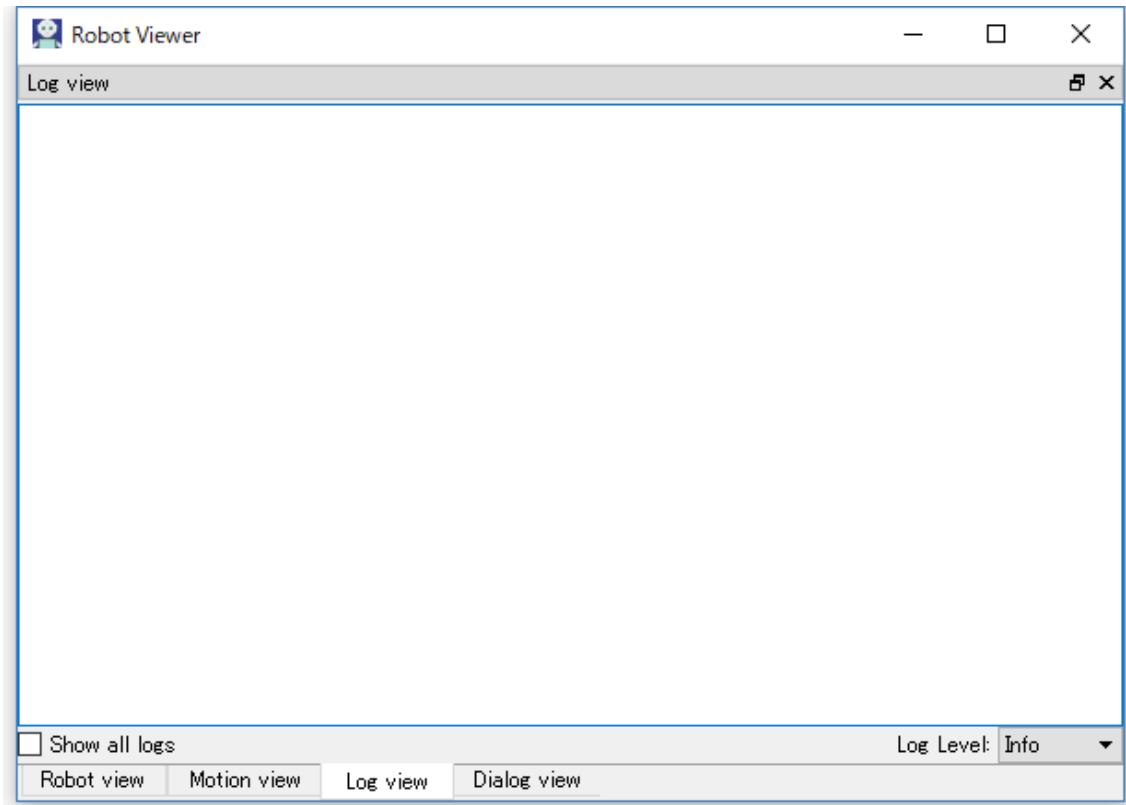


図 4-6 Log view

4.5.5. Dialog view

Dialog view は Pepper の発話内容と音声認識の内容を確認することができます。また、画面下部の入力フォームを用いることで、音声認識をシミュレートすることも可能です。

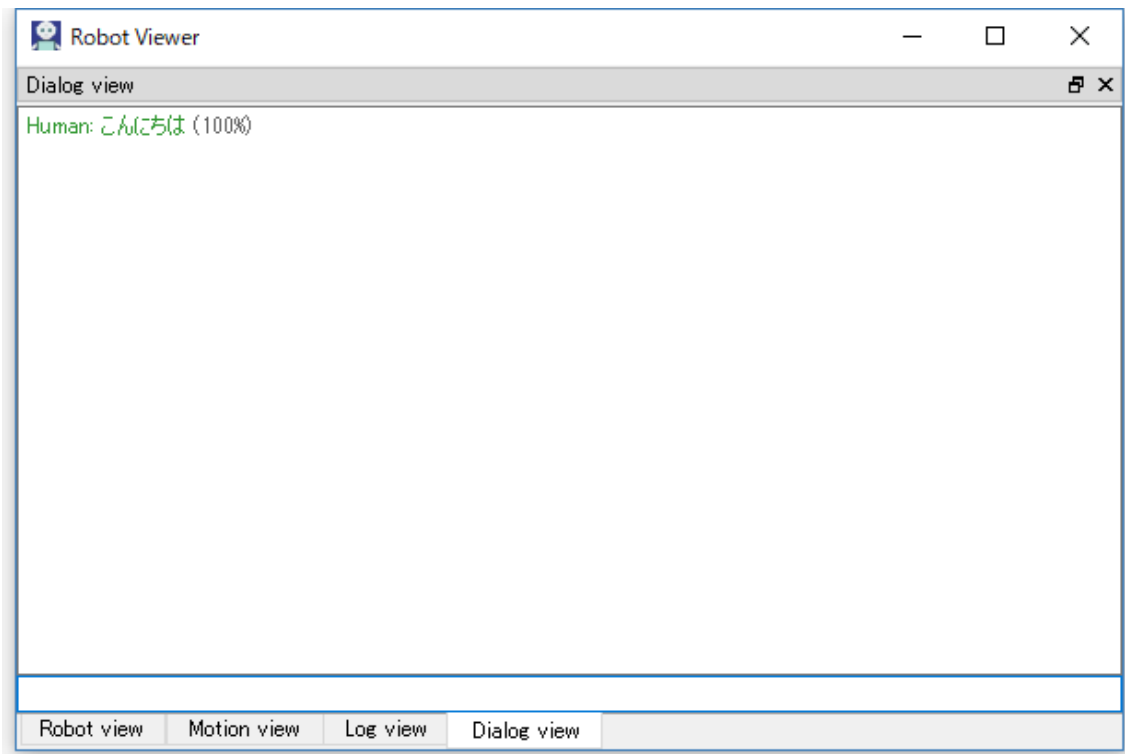


図 4-7 Dialog view

4.6. Robot Browser

Robot Browser は、Pepper の実機に接続することができます。Robot Browser から直接 Pepper の IP を指定して Pepper に接続します。

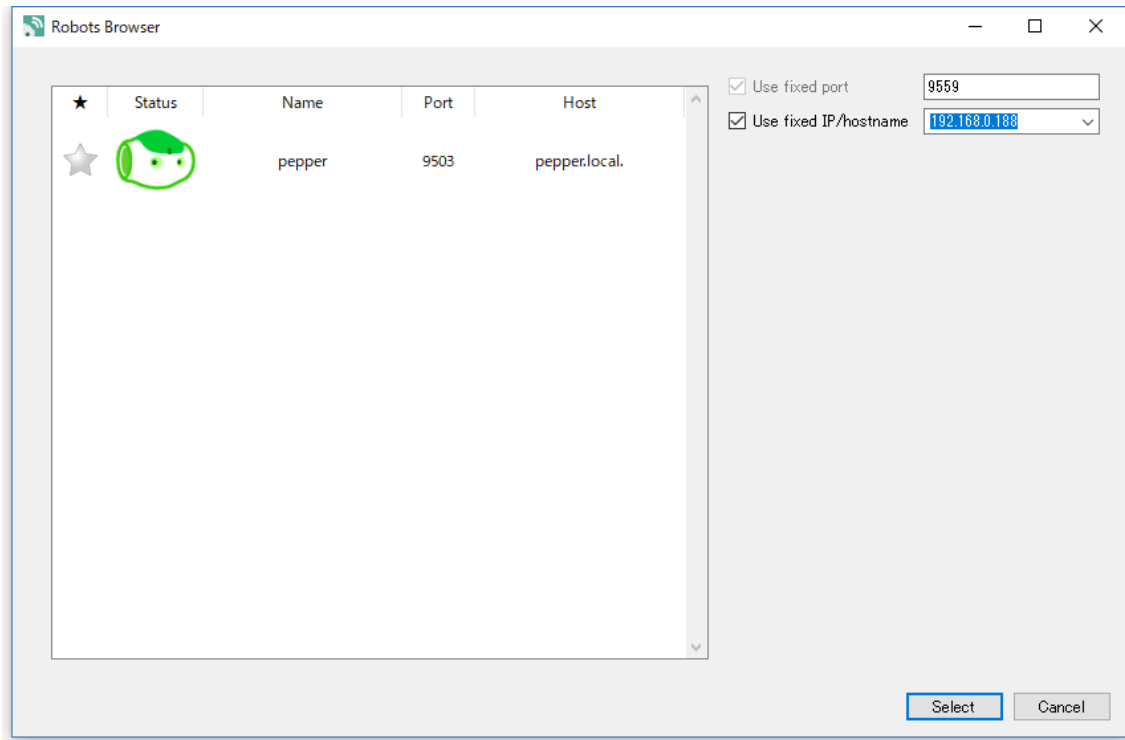


図 4-8 Robot browser

4.7. Animation Editor

Animation Editor は Pepper のアニメーションを作成する際に使用し、関節の角度や動きの速さなど細かく設定することが可能です。

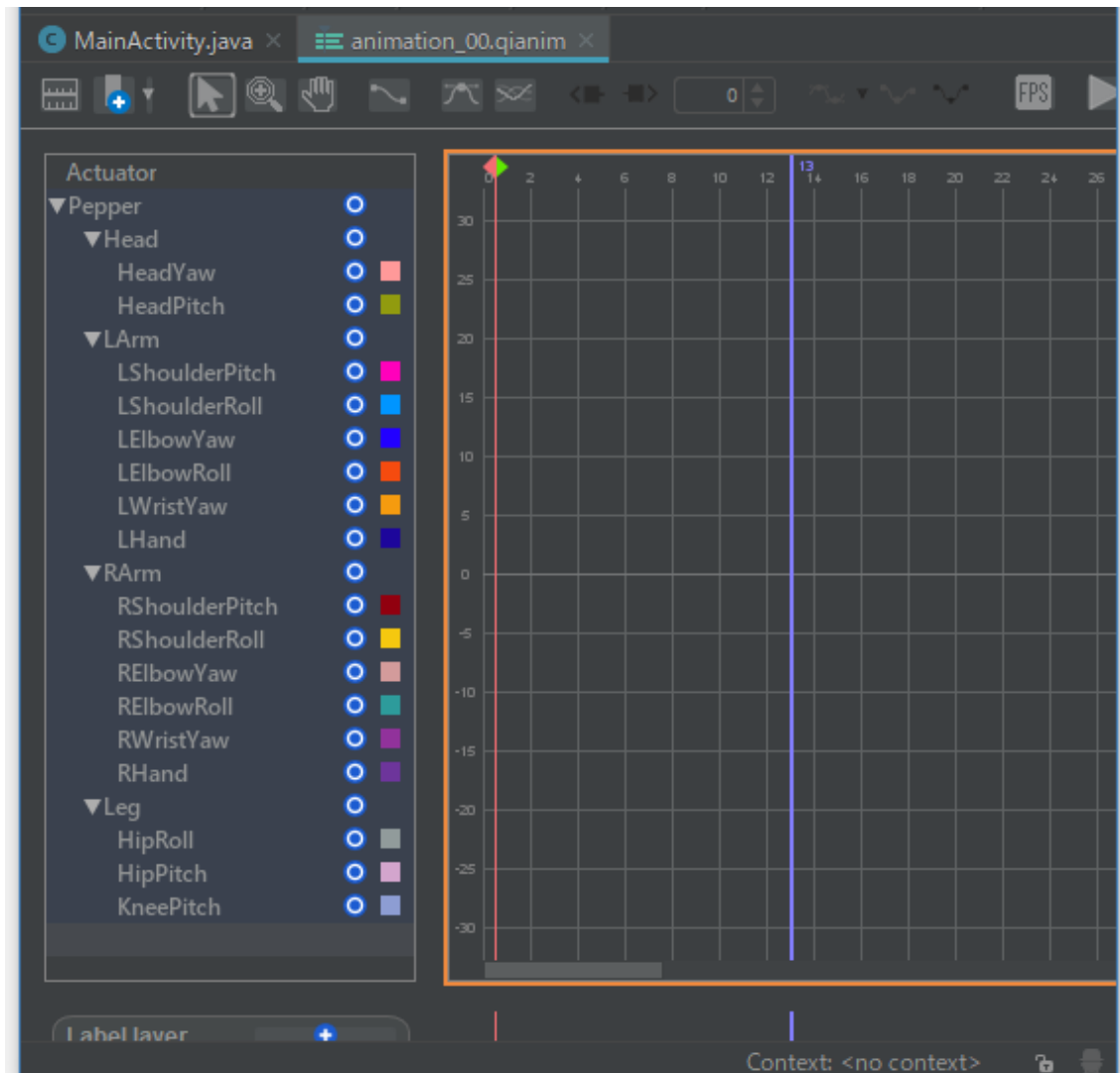


図 4-9 Animation Editor

独自の Animation を作成する場合やインポートした Animation を編集する際は Animation Editor を使用します。Animation Editor の使い方は以下で説明されています。

https://qjsdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch3_tools/animation_editor.html

4.8. Trajectory Edition Tool

Trajectory Edition Tool は Pepper の足の動き(オムニホイール)を作成する際に使用します。

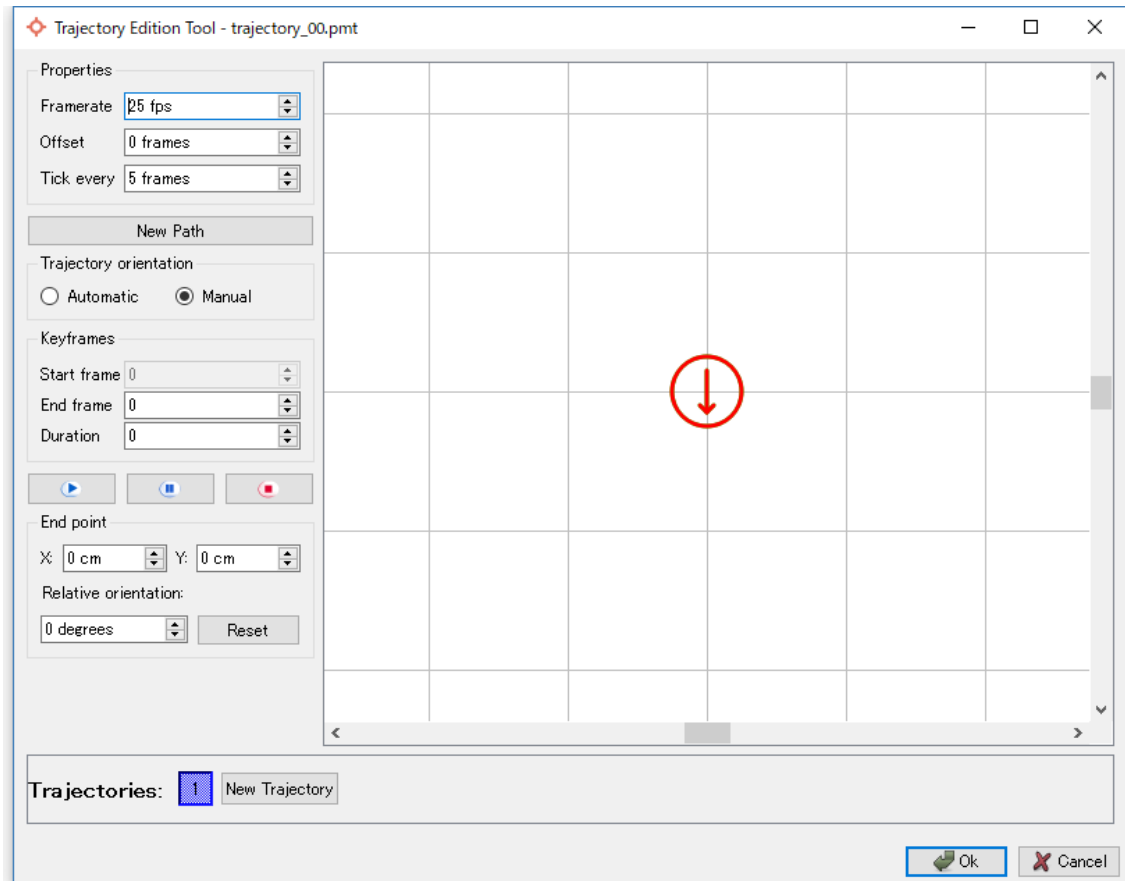


図 4-10 Trajectory Edition Tool

Trajectory Editor の使い方については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch3_tools/trajectory_editor.html?highlight=trajectory

4.9. QiChat Editor

QiChat Editor は Pepper の発話内容を定義するトピックファイルを作成する際に使用します。

```

1  topic: ~sample()
2  ‡ Defining extra concepts out of words or group of words
3  ‡concept:(hello) [hello hi hey "good morning" greetings]
4
5  ‡ Replying to speech
6  ‡u:(~hello) ~hello
7

```

図 4-11 QiChat Editor

4.10. Animation Browser

Animation Browser は QiSDK に含まれている既成のモーションをプロジェクトにインポートする際に使用します。File メニュー > New > Import animation…を選択することで表示されます。

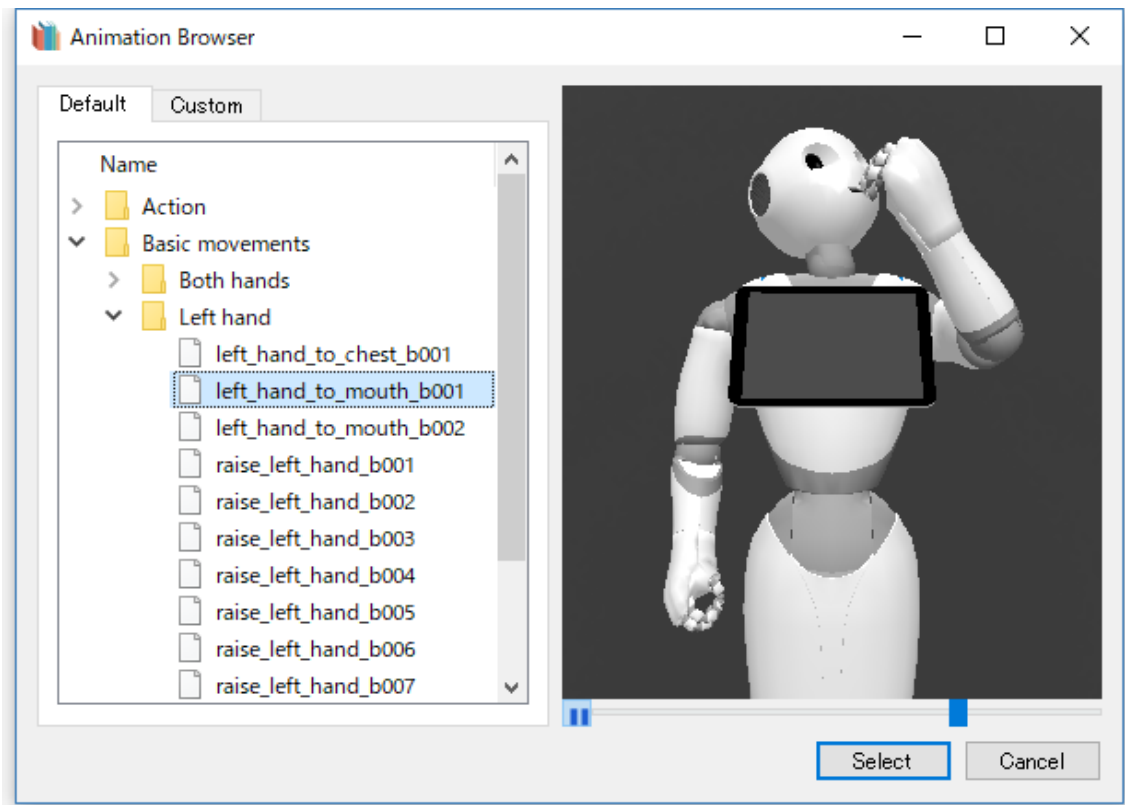


図 4-12 Animation Browser

5. 基本的な開発手順

Pepper SDK for Android を使った基本的なアプリの開発手順を説明します。

5.1. プロジェクト作成

まずは、Android アプリのプロジェクトを作成しましょう。

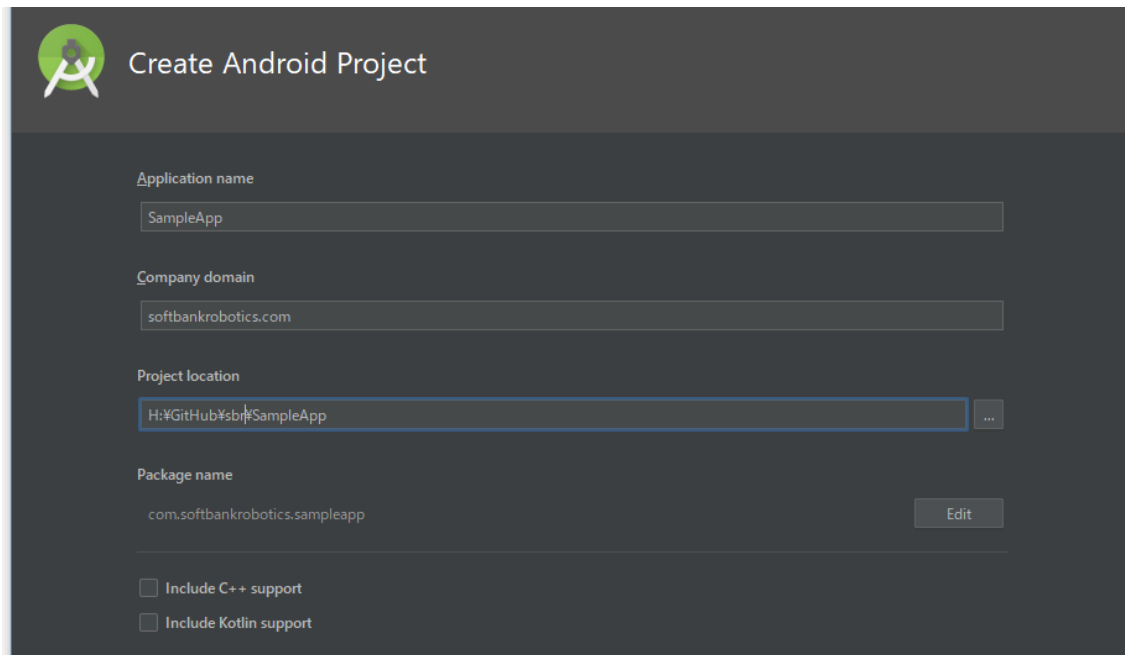


図 5-1 プロジェクト作成

SDK のバージョンは API 23 を選択します。

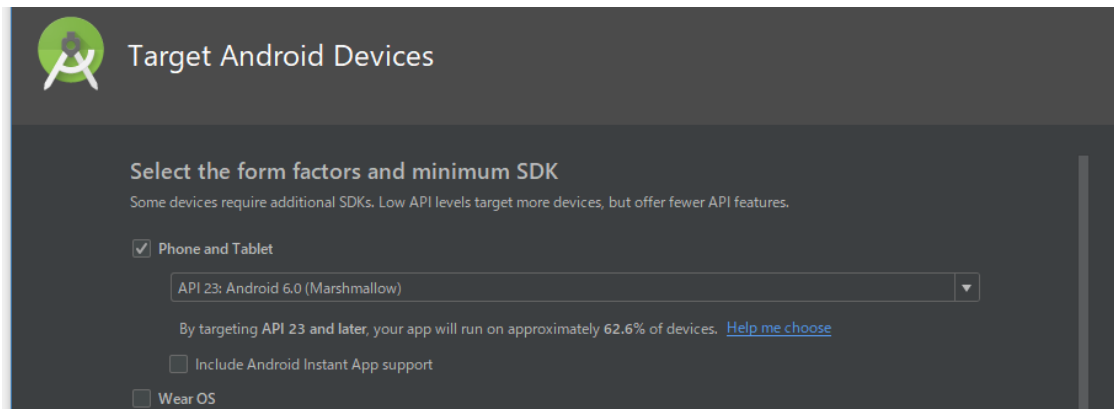


図 5-2 SDK バージョンの選択

5.2. Activity の作成

各アプリがバックグラウンドから自由に発話を行うと、発話が競合して会話が成り立たなくなることが考えられます。この問題を回避するため、Pepper に指示を出せるのはフォアグラウンドにいる Activity に制限されています。このため、画面を表示する場合はもちろんですが、発話などを行う場合も Activity を作成する必要があります。

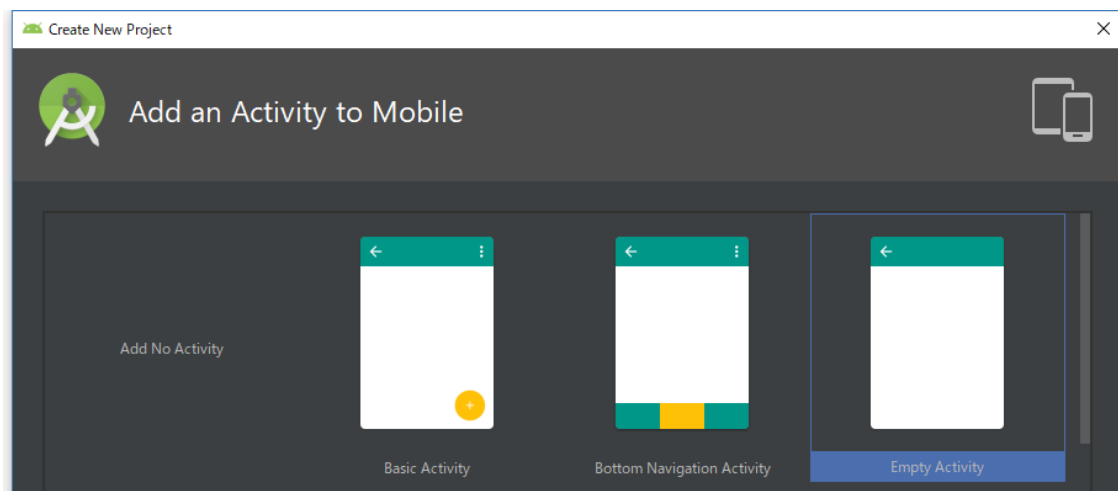


図 5-3 Activity の作成

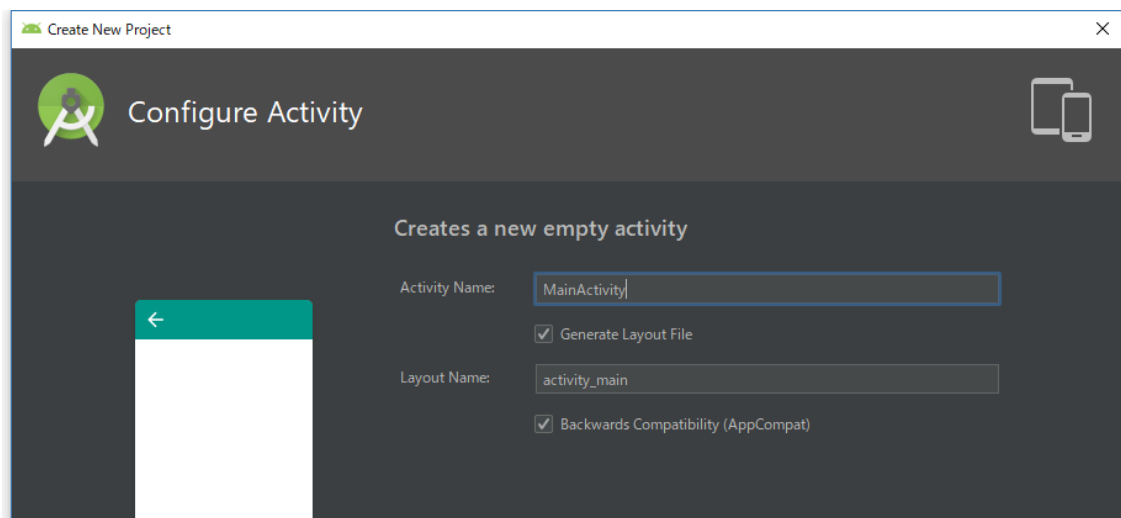


図 5-4 レイアウトファイル

5.3. Robot Application の設定

プロジェクトメニューから Robot Application を追加することで、Gradle ファイルなどを自動で設定してくれます。追加が終わったら Gradle の同期を忘れないようにしましょう。

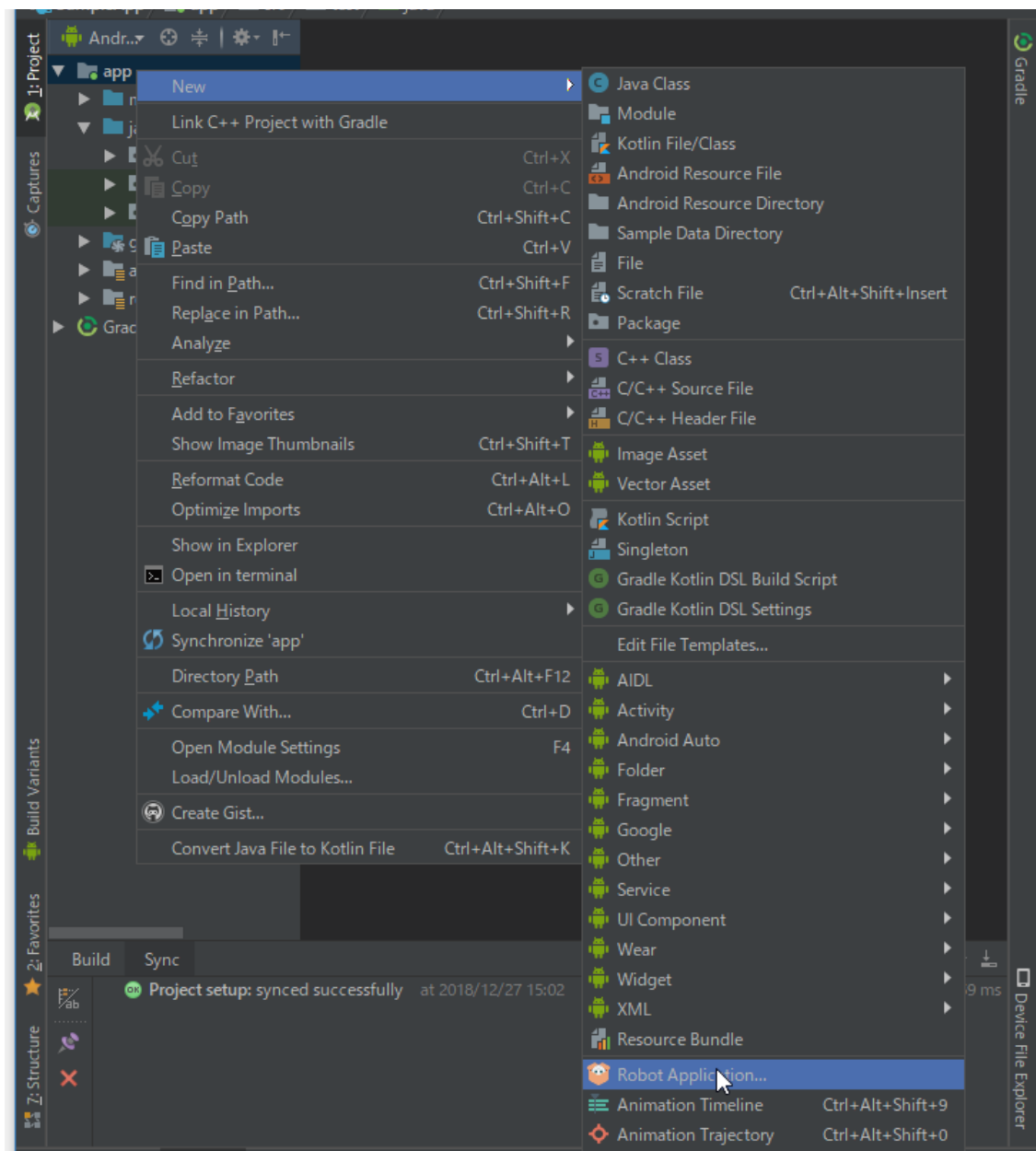


図 5-5 Robot Application の設定

5.4. Robot Focus と RobotLifecycle の実装

作成した Activity に RobotLifecycleCallbacks インターフェースを実装しましょう。

アプリが Pepper に指示を出せるようになるためには、Robot Focus を獲得する必要があります。RobotLifecycleCallbacks は Robot Focus の獲得や喪失を通知するためのコールバックメソッドが定義されたインターフェースです。

```
package com.softbankrobotics.sampleapp;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import com.aldebaran.qi.sdk.QiContext;
import com.aldebaran.qi.sdk.RobotLifecycleCallbacks;

public class MainActivity extends AppCompatActivity implements
RobotLifecycleCallbacks {

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

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    @Override
    public void onRobotFocusGained(QiContext qiContext) {

    }

    @Override
    public void onRobotFocusLost() {

    }

    @Override
    public void onRobotFocusRefused(String reason) {

    }

}
```

図 5-6 RobotLifecycle の実装

RobotLifecycleCallbacks の各メソッドがコールバックされるタイミングは以下の通りです。

表 5-1 RobotLifecycleCallbacks のメソッド

メソッド名	メソッドが呼ばれるタイミング
onRobotFocusGained	Activity がフォアグラウンドに遷移して、Robot Focus を獲得した時
onRobotFocusLost	Activity がバックグラウンドに遷移するなどして、Robot Focus を失った時
onRobotFocusRefused	Activity がフォアグラウンドになったのに、何らかの理由で Robot Focus が獲得できなかった時

5.5. QiSDK.register と unregister

Robot Focus は、Activity がフォアグラウンドに遷移したタイミングで獲得し、バックグラウンドに遷移したタイミングで失います。Robot Focus の獲得や喪失の通知を受けるには、Activity と共に RobotLifecycleCallbacks のインスタンスを QiSDK に登録する必要があります。今回は、Activity に RobotLifecycleCallbacks を実装していますので、register の引数が両方 this となります。

```

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

    QiSDK.register(this, this);
}

@Override
protected void onDestroy() {
    QiSDK.unregister(this, this);
    super.onDestroy();
}
    
```

図 5-7 QiSDK.register と unregister の実装

Robot Focus の獲得や喪失の通知は Activity の onResume や onPause を契機に動作しているため、登録や登録解除の処理は onCreate、onDestroy に実装する必要があります。

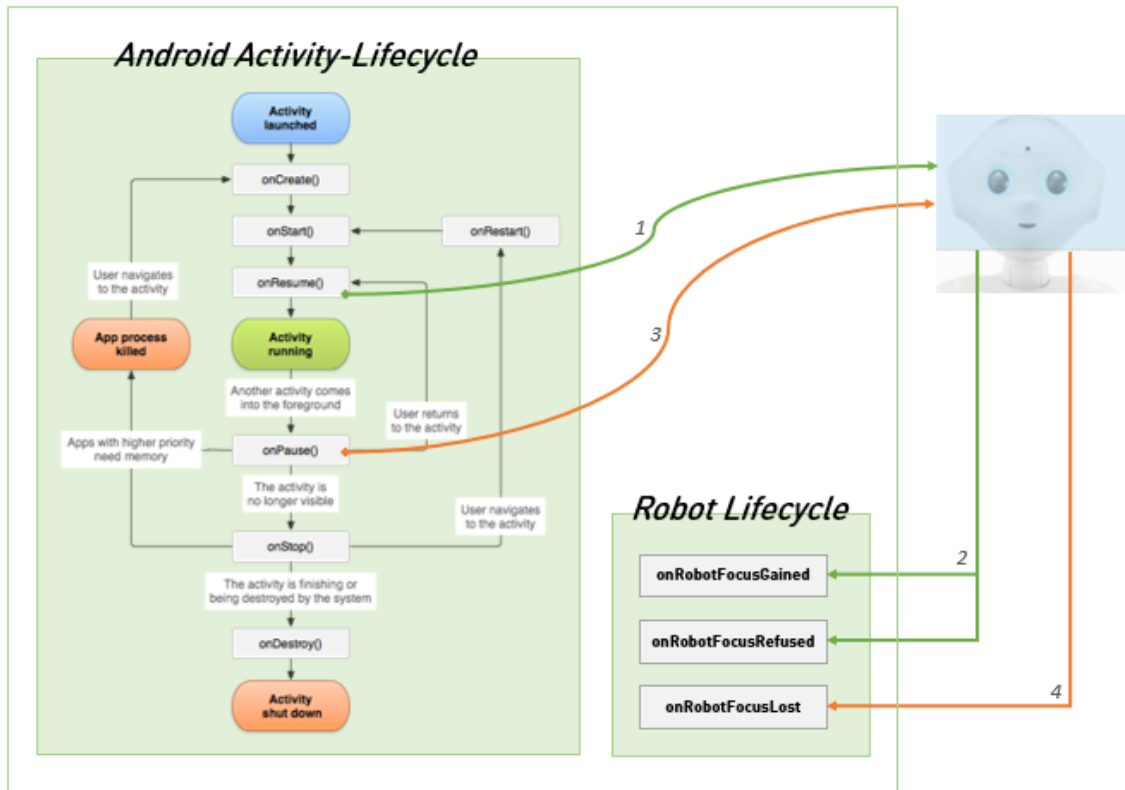


図 5-8 Activity の Lifecycle と Robot Lifecycle

5.6. Action と発話の実装

Action は Pepper と連携するためのコンポーネントです。発話、聞き取り、移動など多くの機能で Action を利用します。Action の作成には QiContext が必要ですが、Robot Focus を獲得した時にコールバックされる onRobotFocusGained メソッドの引数として受け取ることが出来ます。build メソッドで何らかの Action オブジェクトを作成し、それを実行することで Pepper に何らかの指示を送ることが出来ます。Action の作成や実行は Pepper との通信が発生するため UI Thread で利用するとエラーとなるので注意が必要です。Pepper には様々な Action が用意されていますがその詳細については、次章以降で説明します。以下は簡単な発話を行うサンプルコードです。

```
import com.aldebaran.qi.sdk.builder.SayBuilder;
import com.aldebaran.qi.sdk.object.conversation.Say;

(省略)
@Override
public void onRobotFocusGained(QiContext qiContext) {
    Say say = SayBuilder.with(qiContext).withText("ハローヒューマン
").build();
    say.run();
}
```

図 5-9 onRobotFocusGained の実装

SayBuilder に QiContext を渡して build メソッドで Action オブジェクトを生成し、Action オブジェクトの run メソッドで処理を実行しています。

5.7. 動作確認

エミュレーターを起動し、アプリを実行すると、アプリ起動後 Dialog view にハローヒューマンと表示されるのが確認出来ます。

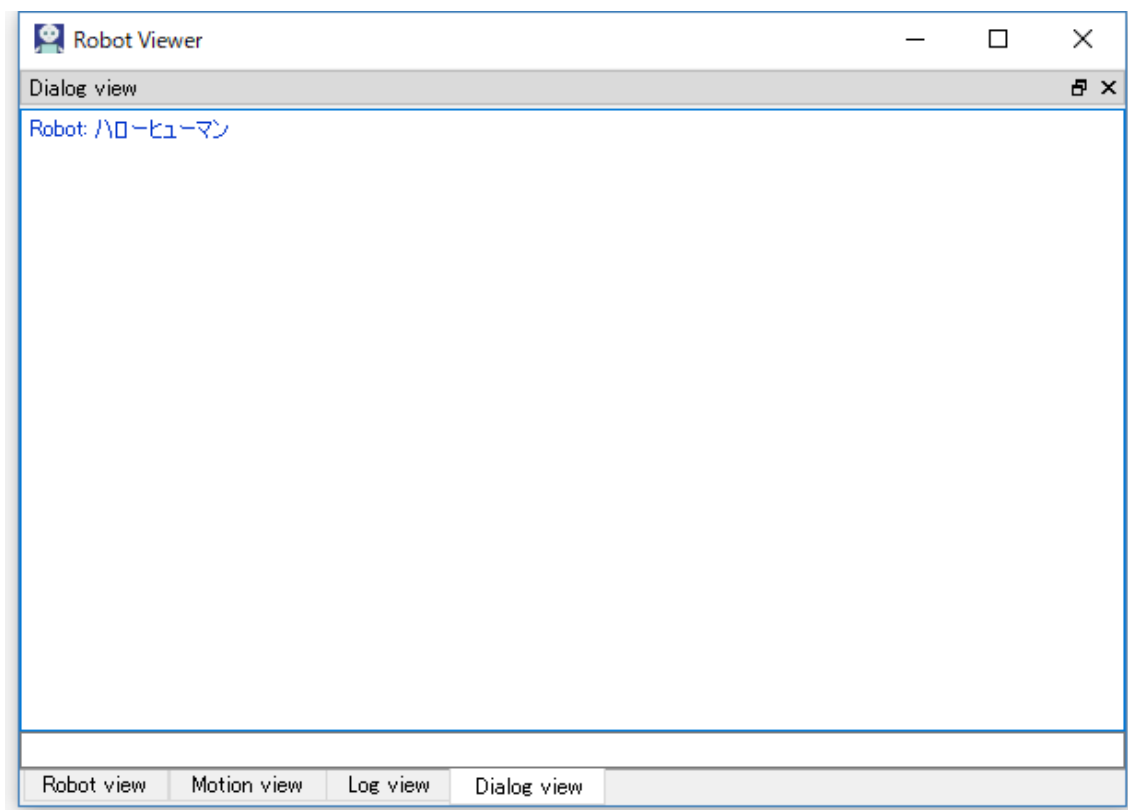


図 5-10 ハローヒューマンの表示

Robot Browser から Pepper と接続することで実機でも動作確認することが可能です。Pepper が「ハローヒューマン」と発話するのを確認してください。

6. QiSDK の API

QiSDK の API 仕様については以下でまとまっています。

[チュートリアル]

<https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/index.html>

[Javadoc]

<https://qisdk.softbankrobotics.com/sdk/doc/qisdk/index.html>

本書には特に重要な点やチュートリアル上で助けになりそうな情報のみを記載します。

6.1. Robot Focus と Robot Lifecycle

前章で少し触れた RobotFocus や RobotLifecycle の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch2_principles/focus_lifecycle.html

6.2. Action

前章で少し触れた Action の詳細については、以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch2_principles/action_getting_started.html

6.3. 同期と非同期

Action の作成や実行には、非同期で実行するための仕組みが用意されています。同期と非同期の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch2_principles/sync_async.html

6.3.1. 非同期を利用するケース

Action の非同期による作成や実行は、以下のようなケースで役立ちます。

- UI Thread 上から、Action の作成や実行を行う場合
- Action のキャンセルが出来る必要がある場合
- Action を複数同時に実行する必要がある場合

Pepper の場合、発話と同時にアニメーションを実行する場合やディスプレイ操作で発話を中断する場合など、多くの場合で非同期を利用する必要があります。

6.3.2. 非同期の基本的な利用方法

Action を非同期で作成するには、build メソッドの代わりに buildAsync メソッドを利用します。また、実行する際は直接 run を呼ぶのではなく、async メソッドを挟んで呼ぶことで非同期となります。以下は [図 5 9 ONROBOTFOCUSGAINED の実装](#) と同じ動作を非同期で実行する場合のサンプルコードです。

```

Future future;

@Override
public void onRobotFocusGained(QiContext qiContext) {
    Future future = SayBuilder.with(qiContext)
        .withText("ハローヒューマン").buildAsync()
        .thenConsume(new Consumer<Future<Say>>() {
            @Override
            public void consume(Future<Say> sayFuture) throws
                Throwable {
                    sayFuture.getValue().async().run();
            }
        });
}
    
```

図 6-1 同期と非同期

Future の requestCancellation メソッドを利用することで、Action をキャンセルすることが出来ます。以下はボタン(R.id.btn_cancel)がクリックされ場合に発話をキャンセルする場合のサンプルコードです。

```

Future future;

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

```

        QiSDK.register(this,this);

        findViewById(R.id.btn_cancel).setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(future != null) future.requestCancellation();
            }
        });
    }

    @Override
    public void onRobotFocusGained(QiContext qiContext) {
        future = SayBuilder.with(qiContext).withText("ハローヒューマン
").buildAsync()
            .thenConsume(new Consumer<Future<Say>>() {
                @Override
                public void consume(Future<Say> sayFuture) throws
Throwable {
                    sayFuture.getValue().async().run();
                }
            });
    }
}

```

図 6-2 Action のキャンセル

6.3.3. Java8 の lambda を利用しよう

Action の非同期実行は非常に便利ですが、ソースコードが冗長になりやすいため、Java8 の lambda を利用することをお勧めします。図 6 2 ACTION のキャンセルのサンプルコードも lambda を利用することでソースコードの見通しが良くなります。

```

Future future

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

    QiSDK.register(this,this);

    findViewById(R.id.btn).setOnClickListener(v -> {
        if(future != null) future.requestCancellation();
    });
}

```

```

    }

    @Override
    public void onRobotFocusGained(QiContext qiContext) {
        future = SayBuilder.with(qiContext).withText("ハローヒューマン")
            .buildAsync()
            .thenConsume(sayFuture -> sayFuture.getValue().async().run());
    }

```

図 6-3lambda の利用

6.4. Future と Chaining

QiSDK には Future と Chaining という仕組みがあり、Callback による非同期処理をラップしてくれるため、ソースコードの見通しが良くなります。Future と Chaining の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch2_principles/futures.html

6.4.1. Future メソッドの一覧

Future には以下のような非同期を連動させるメソッドが用意されています。

- thenConsume
- thenApply
- thenCompose
- andThenConsume
- andThenApply
- andThenCompose

6.4.2. then…と andThen…

Future メソッドは then もしくは andThen で始まるメソッドの 2 種類に分類出来ません。

種別	概要
then…	最初の処理の結果にかかわらず、2 つの処理を順次実行します。2 つの処理が独立している場合に有効です。

andThen...	最初の処理が正しく実行された場合にのみ、2つ目の処理を実行されます。2つめの処理が1つ目の処理に依存する場合に有効です。
------------	--

表 6-1 then と andThen

サンプルコードで2つの違いを見比べてみます。以下は then..を用いたサンプルコードです。1行目の Action の build に失敗しても5から7行目が実行されます。

```

SayBuilder.with(qiContext).withText("ハローヒューマン").buildAsync()
    .thenConsume(new Consumer<Future<Say>>() {
        @Override
        public void consume(Future<Say> sayFuture) throws
Throwable {
            if(sayFuture.isSuccess()){
                sayFuture.getValue().async().run();
            }
        }
    });

```

図 6-4 then で始まるメソッド

以下は andThen...を用いたサンプルコードです。1行目の Action の build に失敗した場合、5行目は実行されません。

```

SayBuilder.with(qiContext).withText("ハローヒューマン").buildAsync()
    .andThenConsume(new Consumer<Say>() {
        @Override
        public void consume(Say say) throws Throwable {
            say.async().run();
        }
    });

```

図 6-5 andThen で始まるメソッド

6.4.3. Consume と Apply と Compose

Future メソッドは Consume、Apply、Compose で終わる3種類にも分類出来ます。2つ目の処理の戻り値に応じて使い分ける必要があります。

種別	概要
...Consume	2つ目の処理が戻り値を返さない場合に使用します。
...Apply	2つ目の処理で戻り値がオブジェクトを返す場合に使用します。

…Compose	2 回目の処理が Future を返す場合に使用します。
----------	------------------------------

表 6-2 Consume, Apply, Compose

以下は Compose と Consume を利用して、2 つの発話を実行し、途中で発話にエラーが発生した場合はアプリを終了するサンプルコードです。andThen…で繋いだ処理のいずれかでエラーが発生しても最後の then…は実行されることに注意してください。発話の成否によらず画面を終了したい場合などに有用です。

```
Future future = SayBuilder.with(qiContext).withText("一つ目の発話")
    .buildAsync()
    .andThenCompose(say -> say.async().run())
    .andThenCompose(aVoid ->
        SayBuilder.with(qiContext)
            .withText("二つ目の発話").buildAsync())
    .andThenCompose(say -> say.async().run())
    .thenConsume(voidFuture -> {
        if(voidFuture.hasError())finish();
    });
```

図 6-6 chaining のサンプルコード

6.5. Autonomous

Pepper には呼吸や瞬き、人を見つけて、人のほうを向いたりするなど自律的に動作を行う Autonomous abilities という機能がついていますが、この機能の ON/OFF を制御することが可能です。Autonomous abilities の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch2_principles/autonomous_abilities_mgt.html

6.6. 会話中の表示

Pepper とユーザが会話をするにあたり、Pepper が音声認識中なのか、また、何と音声認識したのかをユーザに伝えることは UX 的に重要です。QiSDK では SpeechBar という音声認識の状態や認識内容を表示する機能が用意されています。SpeechBar の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/conversation_feedbacks.html

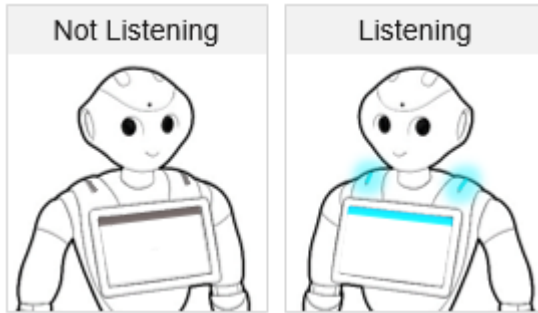


図 6-7 SpeechBar の表示

6.7. 発話 (Say)

簡単な発話については、Say action を使用します。Say action の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/reference/say.html

6.8. 聞き取り (Listen)

簡単な聞き取りについては、Listen action を使用します。Listen action の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/reference/listen.html

6.9. チャット (Chat)

複雑な会話を構築するには Say や Listen の組み合わせではなく、Chat action を利用することをお勧めします。

6.9.1. 基本的な使い方

Chat action を利用するには、まず会話の内容を管理する Chatbot を生成する必要があります。QiSDK には QiChatbot という Chatbot の機能を提供するクラスが用意されており、QiChatbot を使って Chat action を実行するのが簡単です。QiChatbot を使った Chat action の実行の流れは以下のようになります。

1. topic ファイルを作成する。
2. Topic を build する。
3. QiChatbot を build する。
4. Chat action を build する。
5. Chat action を実行する。

1 の topic ファイルは会話の内容を QiChat スクリプトで記述したファイルです。2~5 の流れを実装したサンプルコードです。このサンプルではあらかじめ raw ディレクトリ以下に shop.top という名前で topic ファイルを配置しています。

```
Topic topic =  
TopicBuilder.with(qiContext).withResource(R.raw.shop).build();  
QiChatbot qichatbot =  
QiChatbotBuilder.with(qiContext).withTopic(topic).build();  
Chat chat = ChatBuilder.with(qiContext).withChatbot(qichatbot).build();  
Future<Void> fchat = chat.async().run();
```

図 6-8 Chat action のサンプルコード

6.9.2. Topic ファイル

QiChatbot で扱う会話の内容は QiChat というスクリプト言語にて記述します。

QiChat で記述した会話ファイルのことを topic ファイルと呼びます。topic ファイルは File メニュー > New > Chat Topic から、追加することが出来ます。また、編集時には QiChat Editor が自動で起動します。

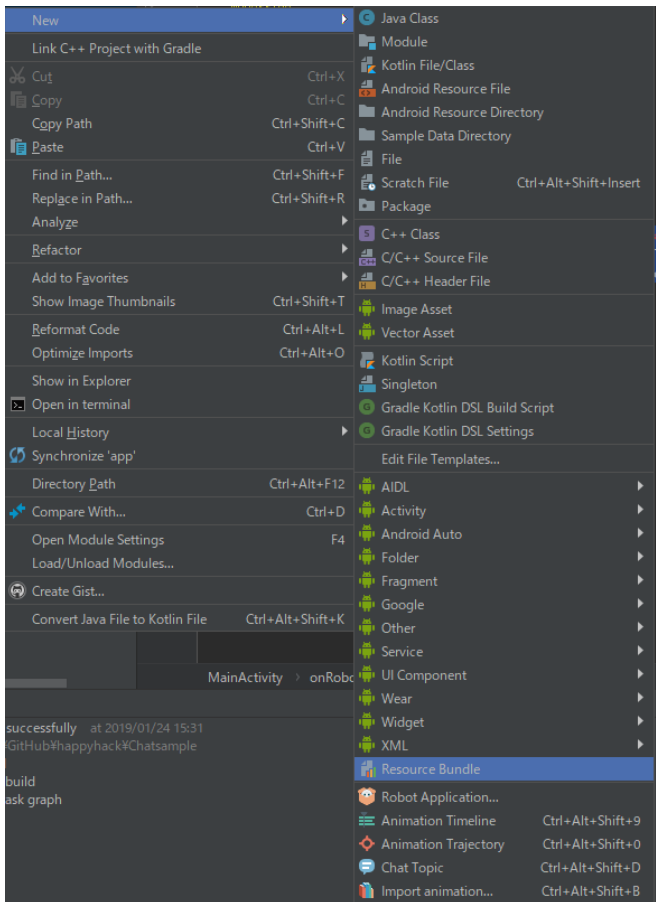


図 6-9 プロジェクトパネルのコンテキストメニュー

6.9.3. QiChat の記述方法

Topic ファイル生成時に使用する QiChat の記述方法について説明します。ここで紹介する記述方法以外にも QiChat には様々な記述方法が存在します。QiChat の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/qichat/qichat_index.html

6.9.3.1. 基本書式

簡単な一問一答を行う場合の書式は以下の通りです。

u:(ユーザの発話) Pepper の発話

図 6-10 QiChat の基本書式

ユーザが「こんにちは」と発話したら「ようこそ」と発話するサンプルコードは以下のようになります。1行目は topic ファイルを追加した際に自動で出力されますので編集しないようにしましょう。

```
topic: ~shop ()
u:(こんにちは) ようこそ
```

図 6-11 ようこそと返すサンプルコード

6.9.3.2. subrule

“u1”、“u2”、“u3”などを使うことで、階層を持つことも出来ます。以下のサンプルコードでは、1行目の “u:(こんにちは) ようこそ” が実行されるまで、2行目、3行目は無効となります。

```
u:(こんにちは) ようこそ
  u1:(何がお勧めですか?) 赤い薔薇がお勧めです。
  u1:(カーネーションはありますか?) すいません。在庫を切らしています。
```

図 6-12 subrule のサンプルコード

6.9.3.3. concept

concept を使えば、よく使う単語のグループを取りまとめておくことが出来ます。concept を定義する書式は以下の通りです。定義済みの concept を使用する場合は concept 名の先頭に~を付けることで使用出来ます。

```
concept:(concept 名) [フレーズ1 フレーズ2・・・]
```

図 6-13 concept の書式

以下のサンプルコードの場合は、ユーザが hello で定義されたフレーズのいずれかを発話すると、Pepper が hello で定義されたフレーズから順番に発応答を返します。ユーザが「よっす」と発話すると Pepper は「こんにちは」と応答し、もう一度「よっす」と発話すると2番目の「こんばんは」と応答します。

```
concept:(hello) [こんにちは こんにちは おはよう やあ 元気 よっす]
u:(~hello) ~hello
```

図 6-14 concept のサンプルコード

6.9.3.4. 入力記憶と変数

会話中にはユーザの発話内容を記憶しておきたい場合があります。_を使用することでユーザ入力を記憶することが出来ます。また、\$を使うことで変数を扱うことも可能です。以下のサンプルコードを見てみましょう。

```
concept:(color)[赤 青 黄]
u:(私は _~color が好きです) $1 色ですね
```

図 6-15 入力記憶のサンプルコード

~color は concept ですので、3行目は「私は赤が好きです」、「私は青が好きです」、「私は黄が好きです」の3パターンのユーザ入力に対応することが出来ます。~colorの前に_がついていますのでユーザ入力は記憶され、自動的に\$1に保存されます。例えば、ユーザが「私は赤が好きです」言えば、\$1には"赤"が入り、Pepperの応答内容も「赤色ですね」となります。

次は、オプションや変数を使ったサンプルコードです。

```
concept:(color)[赤 青 黄]
u:(私は _~color {色}が好きです) $1 色ですね $favoriteColor=$1
u:(私の好きな色はなに) ^first["$favoriteColor 色です" "知りません"]
```

図 6-16 変数のサンプルコード

入力にある { } で括られた文字列がオプションとなります。"色"の部分が入力オプションとなっており、「私は赤が好きです」、「私は赤色が好きです」の両方のユーザ入力を受けることが出来るようになります。

\$favoriteColor は変数で、ユーザ入力された色を保存しています。「私は赤色が好きです」と発話したのであれば、\$favoriteColor には "赤" が入ります。

Pepper の応答内容にある ^first は、あとに続く [] 内のフレーズを先頭から順に検索し最初の発話可能な内容を返す書き方です。このサンプルコードでは \$favoriteColor が未設定の場合、2 番目の「知りません」が選択されることとなります。

6.9.4. Dynamic concept、QiChatVariable

QiChat 内の concept や変数をプログラムから取得したり、動的に設定することも可能です。Concept や変数の動的な取り扱いについては以下で説明されています。

[Concept]

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/tuto/dynamic_concepts_tutorial.html

[QiChatVariable]

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/tuto/qichatvariable_tutorial.html#

6.9.5. Bookmark

Bookmark は QiChat 上にタグを挿入し、挿入したタグに会話到達したことを、リスナーを通じてプログラム側で検知できるようになる仕組みです。また、指定のタグまで会話をジャンプさせるようなことも可能です。

具体的には以下のように%で始まるタグを追加しておきます。ユーザが "赤" と入力した場合に、会話が "%color_red" に到達しますので、リスナーを通して検知し画面を赤色に変化させたりすることが出来ます。

u: (赤) 情熱的ですね。 %color_red u: (青) 聡明ですね。 %color_blue
--

図 6-17Bookmark のイメージ

Bookmark の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/tuto/bookmarks_tutorial.html

6.9.6. Executor

Executor は QiChat 上に ^execute を挿入し、用意しておいた処理を実行させることが出来る仕組みです。Bookmark と異なり Executor は、Executor の処理が完了するまで会話を中断します。

以下は executor を使うサンプルです。Executor で呼び出したい処理をあらかじめ用意しておく必要があります。今回は写真撮影用のプログラムを takePhoto として用意しています。「写真撮って」のユーザ入力があると、「わかりました。」と発話ののちに takePhoto を実行します。takePhoto が完了したら「撮りました。」と会話を再開します。

```
u: (写真撮って) わかりました。^execute(takePhoto) 撮りました。
```

図 6-18 Executor のイメージ

Executor の詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/tuto/execute_tutorial.html#execute-tutorial

6.9.7. 独自の Chatbot

QiChatbot の使い方についてはすでに触れましたが、Dialogflow やその他のチャットボット API サービスなどと連携する独自の Chatbot を定義して利用することも可能です。独自の Chatbot の作成方法については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/reference/basechatbot.html

また、一つの Chat action に複数の Chatbot を登録することも可能です。複数の Chatbot が登録された場合は、各 Chatbot に応答の可否を問合せ、応答可能な Chatbot が応答を行うこととなります。Chatbot への問い合わせの順序は引数の並びで決定します。Chat の詳細については、以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/conversation/reference/chat.html

6.10. Human

Pepper には人認識機能がついており、周辺の人々の位置だけでなく年齢や性別、感情状態などを認識しています。人認識の詳細については、以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/perception/tuto/people_characteristics_tutorial.html

6.11. Touch

Pepper の頭と手にはタッチセンサーがついています。また、足元には3つのバンパーがあります。これらのセンサーについては、センサーの状態を取得、監視することが出来ます。タッチセンサーの詳細については以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/perception/reference/touchsensor_touchstate.html

6.12. Camera

ディスプレイのカメラであれば一般的な Android 端末と同様に利用することが可能ですが、Pepper のおでこのカメラで写真を撮りたい場合は、TakePicture action を利用する必要があります。TakePicture の詳細については、以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/perception/tuto/take_picture_tutorial.html

6.13. 動き (Animate)

Pepper は手や頭を動かして会話を自然に見せたり、ダンスをしたりすることが出来ます。Pepper の手や頭を動かすには Animate action を使用します。Animate action の実行の流れは以下のようになります。

1. 動きの内容を定義した qianim ファイルを作成する。
2. qianim ファイルを指定して Animation を build する。
3. Animation を指定して Animate action を build する。
4. Animate を実行する。

1.の qianim ファイルは Android Studio の File メニュー> New > Animation Timeline から追加することが出来ます。また、QiSDK にはあらかじめよく使われる

動きや面白い動きが用意されており、Animation Browser からインポートして利用することができます。Animate の詳細については以下で説明されています。

[Animate]

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/reference/animate.html

[チュートリアル]

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/tuto/animate_tutorial.html

6.14. 固定の移動 (Trajectory)

現在位置から前に 1m など固定の動きをさせたい場合は、移動の内容を定義した pmt ファイルを作成し、qinami ファイルと同様の方法で Animate action で実行します。pmt ファイルの作成には Trajectory Editor を使います。

qianim ファイルは Pepper の手、腰、頭の関節の動きを定義しており、pmt ファイルは Pepper のホイールの動きを定義します。それぞれ独立しているため 2 つの Animate action を同時に実行することができます。

6.15. LookAt

Pepper に指定の方向を向けさせたい場合は LookAt action を使用します。LookAt action の詳細は以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/tuto/lookat_tutorial.html

6.16. GoTo

Pepper に指定の場所に移動させたい場合は Goto action を使用します。Goto action の詳細は以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/tuto/goto_tutorial.html

6.17. Frame

方向や位置を指定するには Frame を使用します。Frame の詳細は以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/reference/frame.html

また、Pepper に Frame には FreeFrame と AttachedFrame があります。それぞれの詳細については以下で説明されています。

[FreeFrame]

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/tuto/goto_world_tutorial.html

[AttachedFrame]

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/tuto/follow_human_tutorial.html

6.18. 地図の作成(Localize)

Pepper は周辺の地図情報を持つことが出来るようになりました。これにより、より精度の高い移動や背後にいる人の認識を行うことなどが出来ます。Localize の詳細については、以下で説明されています。

https://qisdk.softbankrobotics.com/sdk/doc/pepper-sdk/ch4_api/movement/reference/localize_and_map.html

7. お仕事かんたん生成 3.0

- 以下の内容について、Pepper for Biz 3.0 の取扱説明書にて確認をお願いいたします。サービス概要
- Pepper にアプリ/お仕事の動作をさせる
 - アプリ/お仕事の動作の概要
 - お仕事の新規作成
 - お仕事の管理
 - アプリ/お仕事の追加/削除
 - Pepper 側でお仕事を更新されるタイミング
 - Pepper が実行するアプリ/お仕事の設定

8. Robot Suite

以下の内容について、Pepper for Biz 3.0 の取扱説明書にて確認をお願いいたします。

- Robot Suite 利用の準備
 - Robot Suite について
 - 表示モード
- ロボット/アカウント管理
 - ロボット管理
 - ロボットグループ
 - ユーザー管理
 - アカウント種別
- 基本操作
 - 遠隔管理
 - ヘルスチェック
 - 稼働レポート

9. インタラクション分析

- 以下の内容について、Pepper for Biz 3.0 の取扱説明書にて確認をお願いいたします。インタラクション分析について
- データの閲覧

10. UX

ロボアプリを開発するうえで、UX 上注意が必要な事項について説明します。

10.1. イントネーションと抑揚の調整

Pepper はバージョンアップを重ねる度にイントネーションの精度は上がっていますが、まだ違和感が残る場合があります。以下のように調整すると自然な発声に出来ます。

```
\rspd=110\\vct=160\ええっ、 \pau=700\ \vct=145\ それ、
\pau=200\ \vct=155\ 本当なの？
```

図 10-1 「ええっ、それ本当なの？」の調整済みのフレーズ

10.1.1. 調整できる項目

イントネーションや抑揚の調整のために使用できる項目は以下の通りです。

表 10-1 調整項目

キー名	値の範囲	用途
vct	50~200 (デフォルト : 100)	声の高さを設定する。
rspd	50~400 (デフォルト : 100)	声の早さを設定する。
pau	ミリ秒	一時停止時間を設定する。単位はミリ秒。
vol	0~100%	音量を設定する。

Pepper の標準の声は vct=135, rspd=110 です。Say ボックスなどの発話をさせるボックス内の初期値は vct、rspd とともに 100 なので、標準値に調整するか、意図的に高い声や低い声に調整して発話させましょう。

10.1.2. その他の調整方法

イントネーションを調整するには、以下の様な方法もあります。

- 漢字をひらがなにする。
- ひらがなの一部をカタカナにする。
- ひらがなの一部を別の漢字にする。
- 漢字を一音ずつ別の漢字にする。
- 「、」やスペースで間を空ける。
- 語尾を上げるには「？」を付ける。
- 語尾に勢いを付けるには「っ」や「ッ」を付けたたり、「！」を1つ以上付ける。
- 語尾を伸ばすには「ー」を1つ以上付けたたり、「あ」「い」「う」「え」「お」を複数並べる。
- 「や」「ゆ」「よ」を使ってみる。
- 「、」「。」「(スペース)」「|」「^」を使ってみる。

声の高さを調整した後は、アプリが終了しても、その状態を維持しますので、自分が作成したロボアプリの前に動作していたロボアプリの設定を引き継いでしまいます。これを回避するために、発話テキストの先頭に声の高さや早さを設定するようにしましょう。

表 10-2 調整例の一覧

セリフ	調整後のセリフ
やったー！	\rspd=102\\vct=155\ヤッ田————ツ
いってらっしゃい	\rspd=115\\vct=140\イッテラッしゃーあ伊っ
違うでしょ	\rspd=115\\vct=130\ちが \vct=155\ 鶯でしょー？
それじゃあ、いきますよ！	\rspd=115\\vct=135\それじゃあ、\vct=140\いきますよー？？
3、2、1	\rspd=120\\vct=140\さーん、、、 \pau=700\\rspd=115\\vct=140\にー、、、 \pau=700\\rspd=115\\vct=140\いーち、、

10.1.3. 調整済みテキストのサンプル集

調整済みテキストについて、いくつかサンプルがありますので参考にしてください。

- Pepper セリフ集_Ver1.1.xlsx (本ドキュメントとともに配布しております。)
- Pepper for Biz (法人向けモデル) サポートの Pepper セリフ集
(<https://www.softbank.jp/robot/biz/support/tool/intonation/>)

10.2. モーション作成時の注意点

モーションを作成する際の注意点を紹介します。注意点を守らないと、短時間で関節のモーターがオーバーヒートする場合や転倒する場合があります。

10.2.1. ポーズを1フレーム目に登録しない

これから実行しようとするモーションが始まる前に Pepper がどのような姿勢になっているか予想できない場合、タイムラインの1フレーム目にポーズ（モーションのキーフレーム）を登録すると、急激に姿勢を変化させようとして関節のモーターに高負荷がかかったり、ユーザが驚くことがあるので避けてください。

10.2.2. ポーズを連続で登録しない

タイムラインでモーションのキーフレームの間隔を短くすると、動作が激しくなります。モーターがオーバーヒートして Pepper が停止する原因になるので避けてください。

10.2.3. 中途半端な姿勢を維持しない

腕を上げた状態など、中途半端な姿勢を長時間維持すると、肩のモーターが加熱して Pepper が停止します。特にアプリ終了時には、直立姿勢に戻るようプログラムしてください。

10.2.4. 腰の角度を深くしすぎない

腰の関節の角度を深く曲げすぎると転倒につながります。特に後方、斜め前方は注意が必要です。

10.2.5. タイムラインは500フレーム以内

タイムラインは、右にスクロールするといくらでも長くすることができます。しかし、長すぎるタイムラインは非常に編集がしにくいため、1つのタイムラインは500フレームを目安にしてください。それ以上長くなる場合は、qianim ファイルを分割して、順次実行することで実現することをお勧めします。

10.2.6. センサーの死角や予測できない利用者の動き

ロボットには接触回避機構（セーフティ）が搭載されていますが、あらゆる場面において接触を回避できるわけではありません。予測できない利用者の動き（子どもなど）、ロボットセンサーの死角での動作は、接触回避機構では十分にカバーできません。リスク回避の観点からスピードを落としてください。以下のようなモーションは特に注意が必要です。

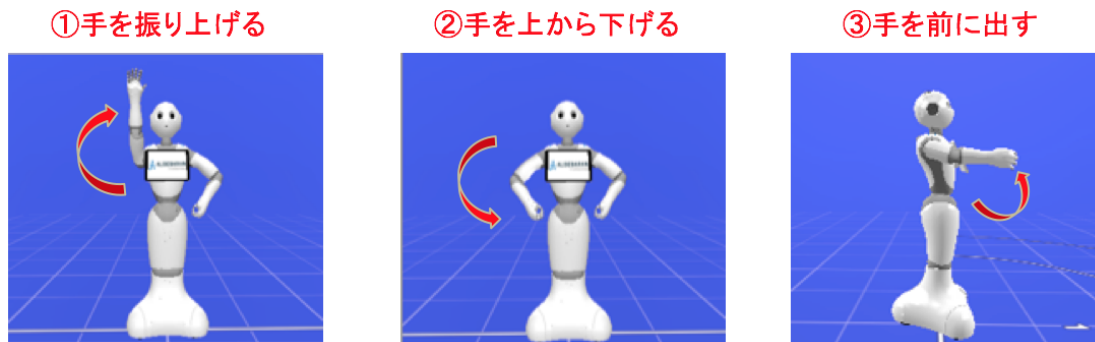


図 10-2 特に注意が必要なモーション

10.2.7. 人と目を合わせる

人と人同士が会話を行うときに、顔を見ながら話をしないとお互いに不自然な印象を受けてしまいます。ロボットが人と対話するロボアプリでも、ロボットの目線が利用者の顔を追わないと不自然な印象を利用者が受けてしまいます。Autonomous Ability を利用し、利用者の顔を追いながら会話するようにしてください。

10.2.8. 開発時

開発中に実機を利用していない場合は、レストモードにしておきます。通常の直立状態でも負荷がかかっているためです。また、動作を確認するときは広いところで動作させてください。障害物が近くにあると、接触回避機構が働き、通常の動きと異なる結果になる可能性があります。アプリ終了時には、Pepper を基本姿勢に戻すように実装してください。別のアプリを起動した際に、急なモーションを行う原因となります。

10.3. ディスプレイ表示の注意点

ディスプレイを使用するロボアプリは、以下の点に注意して作成するように心がけてください。

10.3.1. 字はできるだけ大きくする

Pepper のディスプレイは、ユーザが立った状態で見える可能性が高いため、一般的なディスプレイアプリよりディスプレイと目の距離が遠くなります。そのため、ディス

プレイに表示する文字はデザインが破綻しない範囲で、できるだけ大きくすべきです。

10.3.2. UI 要素の配置が上下左右に偏らないようにする

文字、ボタンなどの UI 要素は可能な限り大きく配置して、上下左右に偏ったり、無意味な空間が存在しないようにしてください。

10.3.3. 文字と背景のコントラスト比はできるだけ高くする

文字と背景のコントラスト比が低いと、色弱者には判断できないことがあります。できるだけコントラスト比が高くなるような色使いを心がけてください。

10.3.4. ボタン連打に対応する

ディスプレイ内のボタンを連打すると、同じイベントが複数回発生し、同じ処理を連続で実行してしまうことがあります。それを回避するため、ボタンは短時間で連打されても 1 回しか反応しないように設計および実装してください。

10.3.5. 操作は音声とディスプレイ両方でできるようにする

Pepper のメインユーザインターフェイスは会話ですが、環境音などの影響でユーザの言葉を聞き取れないことがあります。どのような環境下でもアプリを途中で止めてしまうことがないように、操作は音声とディスプレイの両方でできるようにしてください。

10.3.6. 画像作成時の注意点

Pepper のディスプレイの解像度は、1280px(W)×800px (H) です。適切な画像サイズを使用してください。不必要に大きいサイズの画像はデータ量の肥大化を招き、著しく小さい画像はジャギーが目立ちロボアプリの見栄えが悪くなります。

10.4. 演出上の注意点

Pepper の振る舞いの自然さや面白さと、その動作をさせているアプリの質は必ずしも比例しません。とても複雑な動作が可能なアプリだとしても、それが不自然であったり、あるいは使用する人々にとって不親切であったりすれば、ユーザからするとそれはよくないアプリとなります。Pepper のアプリを開発する場合は、そうした

Pepper とユーザとの関係を確認し、どのようなユーザ体験を与えられるのか、綿密に設計・デザインする必要があります。

10.4.1. 人を惹きつける方法

Pepper で動作するアプリは、ユーザの要求(操作・声かけ)に応じて動作させるものだけではありません。Pepper 自身が周辺にいる人に対して声をかけ、認識した人の状態を判断して適切なアプリを起動するなど、Pepper 側から能動的にユーザに働きかけを行うアプリを作成することも出来ます。例えば、店舗に Pepper を配置した場合は、定期的に店内を巡回し、認識した顧客に対して声かけを行うようなアプリで注目を集められます。また、子供が集まるような場所であれば、Pepper の目の部分の LED を光らせたり、音楽を再生したりすると楽しんでもらえるでしょう。

10.4.2. 安全性の確保

人を惹きつけるアプリを作成する際に注意しなければならないことは、安全性を確保することです。特に Pepper が移動するようなアプリを使用したい場合は、周囲半径 90cm 以内に障害物がない場所に配置することが望ましいです。また、インパクトのある動きをさせることで注目を集めようとして、突然大きな声で話したり、人が近づいてきたところで突然大きな音楽を再生したりすることはトラブルの原因になりますので避けましょう。Pepper に派手なアクションをさせようと無理な姿勢を長時間つづけるようなことも故障の原因となります。ロボアプリの開発は、人を惹きつけることと同時に安全性を確保するという、相反する要求を同時に達成するための工夫が求められると言えるでしょう。

11. その他

その他の注意事項やロボアプリを開発上で役立つ情報について説明します。

11.1. 運用時の注意点

Pepper を実運用する際の注意点を説明します。

11.1.1. ネットワーク環境

多くのロボアプリはインターネットに接続されていないと正しく動作することができません。Pepper の設置場所のネットワーク環境について、運用開始前に確認してお

きましょう。特にイベント会場などの場合は Wi-Fi の電波が届かないケースやセキュリティ設定で Pepper 本体と Android Studio が接続できないようなケースもあります。

来場者のスマートフォンなどの接続により Wi-Fi ルーターやネットワークが重くなるようなことも考えられますので、モバイル Wi-Fi などを準備しておくとう良いでしょう。

11.1.2. 設置場所

設置場所に関する注意点を以下に列挙します。

- 強い逆光があるような場所だと顔認識や表情認識などカメラを使った機能が正しく動作しません。
- 狭い場所だとセーフティ機能が頻繁に動作してしまいよくわからないモーションになってしまうことがあります。
- 実際に使用する位置に Pepper を設置後、一連の動作が正しく動作するか確認する必要があります。
- Pepper が壁に反射する光を顔だと認識してしまうと、ずっと壁の方を向いてしまう場合があります。
- 電源フラップを開き、移動機能を停止してしまうことで、問題を回避することができる場合もあります。

11.1.3. 複数台体制の検討

ダンス系のロボアプリを連続して使用する場合やイベント会場でのプレゼンなどで失敗が許容されにくい場合では、Pepper を複数用意することを検討しましょう。万が一、モーターの温度が上昇して動作できなくなる場合や急に動作が不調になった際に、スワップ機の準備があればリスクを減らすことができます。

11.2. 規約上の注意

利用規約で禁止されている使い方をすることは出来ません。

11.2.1. Pepper の商標・キャラクターについての制限

Pepper の商標・キャラクターの権利はソフトバンクロボティクス株式会社に帰属します。Pepper を使用した著作物、コンテンツ、Pepper そのものの使用には制限があります。当然、Pepper のキャラクターイメージを損なうことは禁止されています。

また、Pepper を使って他者の名誉を毀損する行為も同様に禁止されています。詳細については、以下の資料をご確認ください。

[Pepper のメディア掲載、公共での稼働をお考えの方に]

<http://www.softbank.jp/robot/biz/support/character/>

[商標・著作物・Pepper キャラクターに関するガイドライン]

http://cdn.softbank.jp/mobile/set/data/static/robot/legal/pepper_character_guideline.pdf

11.2.2. メディア掲載・公共環境での稼働について

一部のメディア掲載、公共での稼働については、事前にソフトバンクロボティクス株式会社の許諾が必要です。

11.2.2.1. 禁止事項

以下は禁止事項です。

- TV 番組へのタレント出演
- CM 出演
- ウェブ広告
- 交通広告
- 屋外広告

11.2.2.2. 事前申請が必要

以下は事前申請が必要な項目です。

- TV 番組・新聞・ウェブ・ラジオなどの取材
- プレスリリース
- 発表会への登壇
- 新聞・チラシ・雑誌・ラジオやこれらに準じる各メディアでの広告
- 広告以外のウェブ掲載
- イベント・店頭・各種施設での展示・稼働（メディア露出がある場合のみ要申請）

11.2.3. 商標について

Pepper 商標等のソフトバンクロボティクス社のロゴは使用できません。文字表記の Pepper 商標に関してはガイドライン掲載の条件を満たすことにより使用が許可されています。ただし、表示方法については以下の通り制限があります。

11.2.3.1. 名称 (Pepper) に関する制限

開発したソリューション・製品の名前に、商標 Pepper を連続して使用することを禁止致します。但し、Pepper との間に「for ~」や「Pepper 用~」の文言を挿入することは制限しません。

- Pepper 用○○販売促進アプリ（用が Pepper との間にあるため○）
- ○○販売促進 Pepper（→Pepper と商品名が連続して使用されているため×）
- ○○販売促進 for Pepper（→for が Pepper との間にあるため○）

また、Pepper の商標は P（大文字）epper（小文字）の表記になります。以下に掲載する例は全て誤りです。

- pepper（小文字表記）
- PEPPER（大文字表記）
- pEPPER（大文字と小文字の箇所が誤っている）
- ペッパー（カタカナ表記）
- Pepper's（Pepper から何らかの変形を伴った形式）

Pepper の名前を含んだ、商標・企業名・インターネットドメイン・SNS アカウント等の登録は禁止しております。

11.2.4. Pepper の画像について

広告、告知物に Pepper の画像を使用する場合はご自身で撮影された画像をご使用ください。無断でソフトバンクロボティクス社およびソフトバンクグループ会社のホームページ等から写真画像、Pepper ロゴ、イラストなどのコンテンツを使用することは禁止されています。Pepper for Biz をご使用で、公式画像が必要な場合は営業担当もしくは Pepper for Biz お問い合わせ窓口へ連絡する必要があります。Pepper パートナープログラムへの加入後は Pepper パートナー会員事務局へご相談ください。

11.3. [参考] qicli コマンド

Pepper 本体と SSH 接続すると、CLI で Pepper を操作することが出来ます。CLI で Pepper を操作したほうが便利な場合もあります。

11.3.1. SSH 接続の方法

SSH 接続は、安全にリモートコンピュータに接続する通信プロトコルです。ロボアプリの開発は基本的に Android Studio で行いますが、Android Studio ではできないことを SSH 接続して命令することが出来ます。SSH 接続の方法は、Mac の場合はターミナルから以下のコマンドを入力します。

```
ssh nao@[IP address]
```

図 11-1 SSH 接続のコマンド

環境やクライアントアプリの設定によって操作手順が異なりますので、それに応じた方法で接続してください。Windows の場合は、SSH 接続クライアントアプリを用意する必要があります。Tera Term を使用する場合、「チャレンジレスポンス認証を使う(キーボードインタラクティブ)」設定で接続してください。

11.3.2. NAOqi 側のログファイル

SSH 接続していれば、Linux の tail コマンドを使って、直接ログファイルを参照することも可能です。

```
tail -f /var/log/naoqi/tail-naoqi.log
```

図 11-2 Linux の tail コマンド

tail-naoqi.log 以外にも“/var/log/naoqi/servicemanager/”ディレクトリ以下には様々なログファイルがあります。ログの様子は公開されておりませんが、logcat に出力されていないエラーログなどがデバッグに役立つこともあります。

11.3.3. セーフティ機能解除

Pepper にはセーフティ機能が働いています。安全が確保できない方向に手を出すときは、ぶつかっても Pepper が壊れたり、人にけがをさせたりしないようにゆっくりと動作します。移動のときは障害物があれば自動で停止します。以下のコマンドで腕のセーフティ機能を解除することが出来ます。

```
qicli call ALMotion.setExternalCollisionProtectionEnabled "Arms" 0
```

図 11-3 腕のセーフティ機能を解除するコマンド

11.3.4. ログを確認する

ログを確認する qicli コマンドがあります。

```
qicli log-view
```

図 11-4 ログを確認する qicli コマンド

発生した問題によっては Android Studio 上で確認できない状態であることも考えられます。その問題解決のために必要になることもあります。

発行 ソフトバンクロボティクス株式会社

本書の一部または全部を、ソフトバンクロボティクス株式会社から正式な許諾を得ずに、いかなる方法(転載・転用・送信・上映)においても無断で複写、複製することを禁ずる。

◇免責事項◇

本書において記載されている会社名、製品名は、それぞれの会社の商標もしくは登録商標の場合がある。本書では®、TM マークを明記しない。