ピッとシュ！ネットワークモデル
ご紹介資料(Ver1.3)
～ 温度測定・消毒 × IoT技術 ～

＊ピッとシュ！ネットワークモデルは開発中のため、本資料で記載された内容については、リリース時に変更することがあります。

SWIFTECHIE

シーテック株式会社

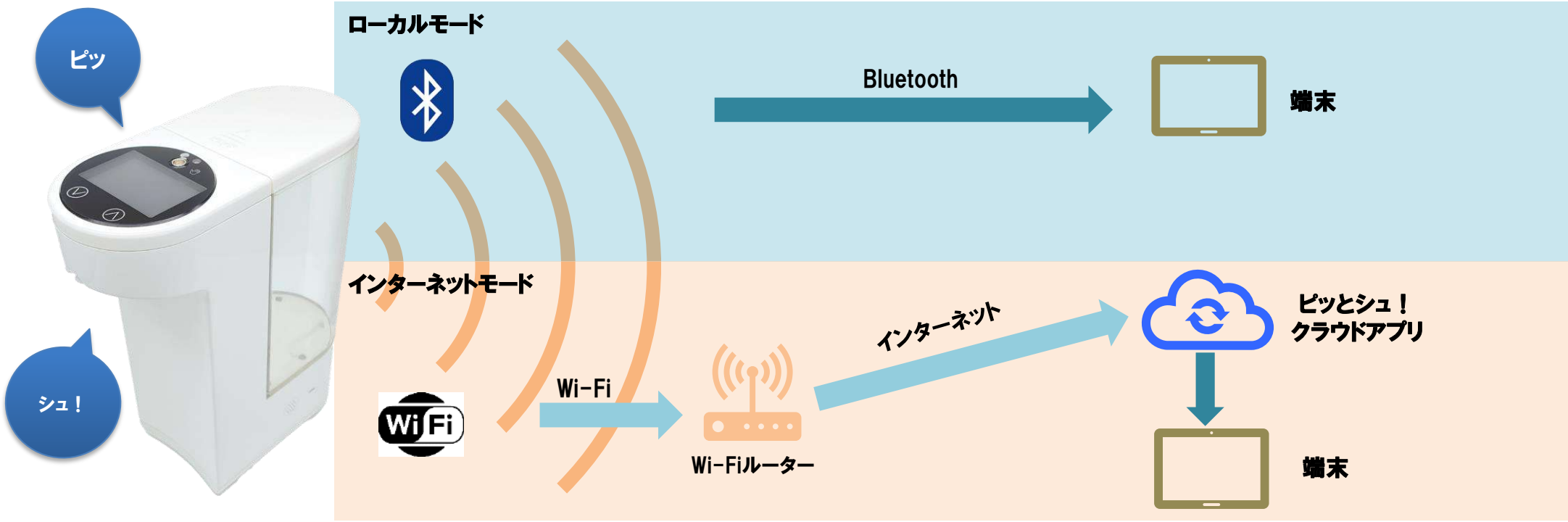
1. ピッとシュ！ネットワークモデル機能概要

1．1　共通機能

ピッと測定、シュと消毒は、スタンダードモデルと同じ機能です。

1．2　ネットワークモデルからの追加機能

Bluetooth(ローカルモード)とWi-Fi(インターネットモード)から、測定温度やICカード情報を取得します。
＊Bluetooth (ローカルモード) の場合、アプリを常時立上げておく必要があります。



1．3　取得できる情報

いつ、どこで、誰が、測定結果と消毒の有無の情報を取得できます。

いつ　→　アプリが受信した時刻、ピッとシュ！クラウドに格納された時刻を取得します
どこで　→　ピッとシュ！ネットワークモデルの初期設定で、ピッとシュの設置された場所を登録できます
誰が　→　ICカードのユニークな情報を取得し、勤怠管理等のアプリ上で個人情報とマッチングできます
測定結果　→　温度測定結果と消毒の有無の情報を取得します

No	項目	活用例	取得情報
1	ICカード情報	・個人IDカードがICカードの場合、個人の測定結果を個人毎に管理できます	ICカードUID（FeliCa、Mifare TypeA、Mifare TypeB　カードに対応）
2	測定温度情報	・高熱者を管理端末で把握でき、早期対応が可能になります ・測定した温度とICカードの情報を合わせて取得することで、ICカード利用者の情報として管理できます	測定した温度と温度状態（設定体温より高温、平熱等）、 ICカードUID（カード検知した後１０秒内温度測定した場合）
3	消毒薬液情報	・消毒液の残量を管理端末で把握でき、補充タイミングを事前に把握できます	残液量（下限フラグ）、噴霧回数

※通信混雑度や通信距離により時間がかかる場合があります。

2. ピッとシュ！ネットワークモデルハードウェア仕様

ピッとシュ！ネットワークモデルはスタンダートモデルのIoTバージョンです。

No.	名称	説明
1	スタンダートモデル機能	スタンダートモデルの機能をご利用いただけます。
2	ローカル通信モジュール	Bluetooth4.2をサポートしています。Bluetoothを使って、操作情報の発信や、デバイスの設定ができます。
3	インターネット通信モジュール	Wi-Fi 802.11 b/g/n/e/iに対応しています。インターネットに接続できるWi-Fi環境があれば、操作情報をピッとシュクラウドに送信し、接続されたスマホやタブレットにプッシュします。
4	NFCカードリーダー	サポートするカードタイプは以下の通りです。 • Mifare TypeAカード（ISO14443A） • Mifare TypeBカード（ISO14443B） • FeliCa

※通信により消費電量が大きくなたため、USB電源にてご利用ください。

3. 基本機能

3. 1　アプリの初期設定

ピッとシュ！ネットワークモデルの設定は、弊社より提供します専用アプリにて行います。
アプリは、AppleStoreまたはGooglePlayからダウンロードできます。

デバイスの検索

アプリがデバイスを検索します。ピッとシュ！を発見すると、温度測定を促すメッセージが表示されます。

デバイスの確認

設定する方の体温を実際に測定させる事で、ピッとシュ！のデータを受信出来ていることを確認します。

デバイス名入力（ローカルモード）

ピッとシュ！を識別するための名前を入力します。

測定リスト画面

測定リストにデバイス名が表示されていたら設定完了です。

3．2　アプリの基本機能

アプリ上で、測定データを表示したり、検索したり、データを変更したりすることが出来ます。

高温検知機能

異常温度を検知した場合、アラートが鳴り異常を知らせます。計測された温度によって測定結果の色が変わります。

設定画面

アプリの設定画面です。デバイスやアカウントの情報が確認できます。

測定履歴検索

期間を指定し、その期間内の測定結果を表示します。

測定履歴

これまでに測定された温度履歴が確認できます。

測定結果の編集

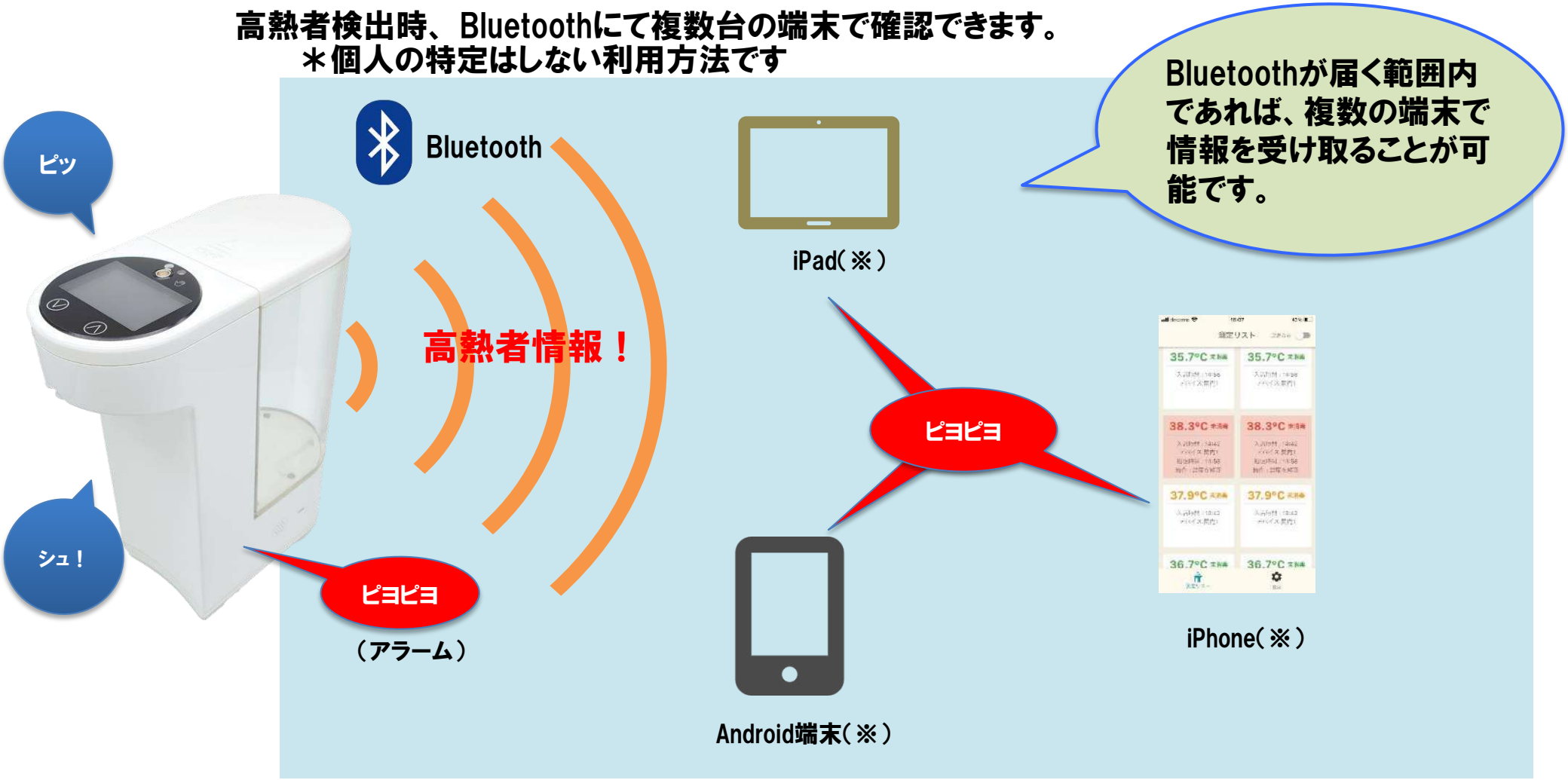
測定結果をタップすることで、高熱者への対応の可否や医療用体温計での測定結果への変更が行えます。

4. 利用モデル

4. 1　無人の店頭に設置したピッとシュ！にて高熱者を検知し事務所内にて確認

4. 1. 1　ローカルモード

高熱者検出時、Bluetoothにて複数台の端末で確認できます。
＊個人の特定はしない利用方法です



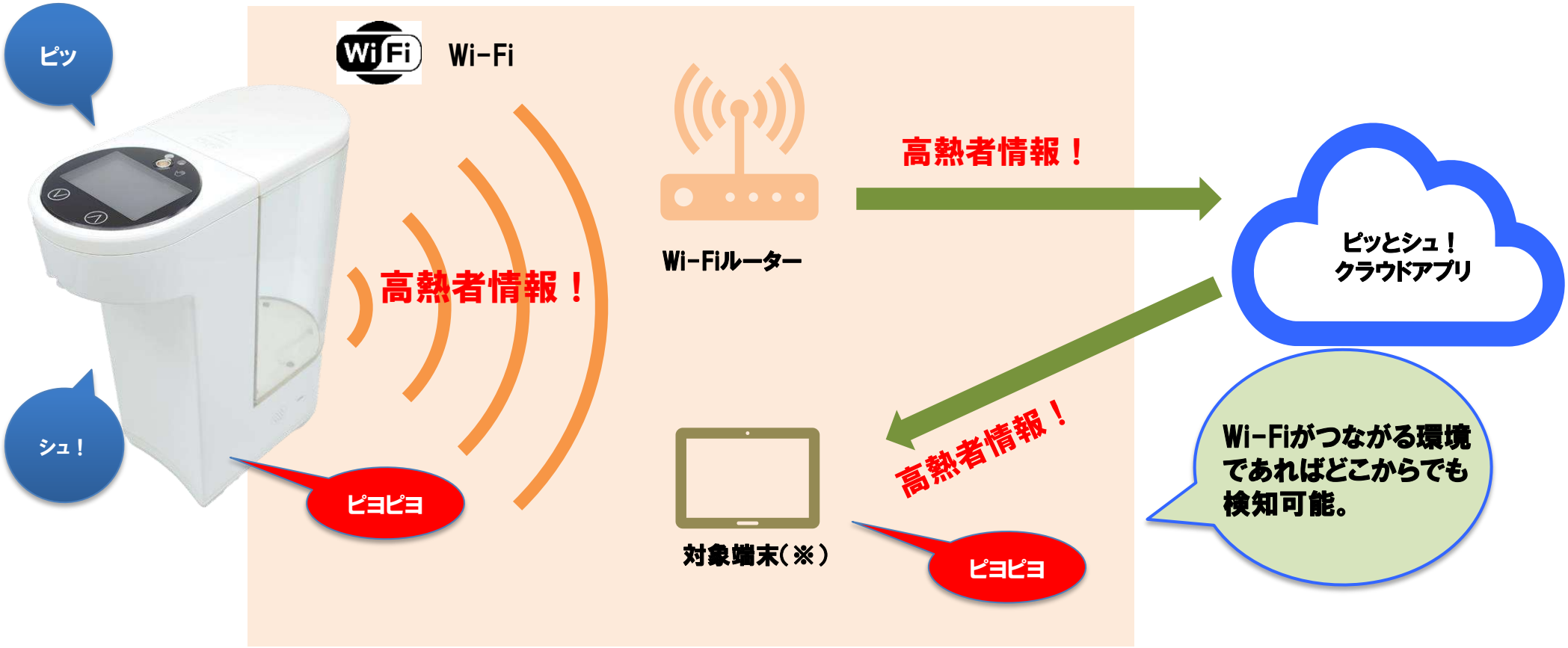
（※）アプリの初期設定が終わった端末

4. 利用モデル

4．1．2　インターネットモード

高熱者検出時、Wi-Fi経由で対象の端末に連携します。

＊個人の特定はしない利用方法です

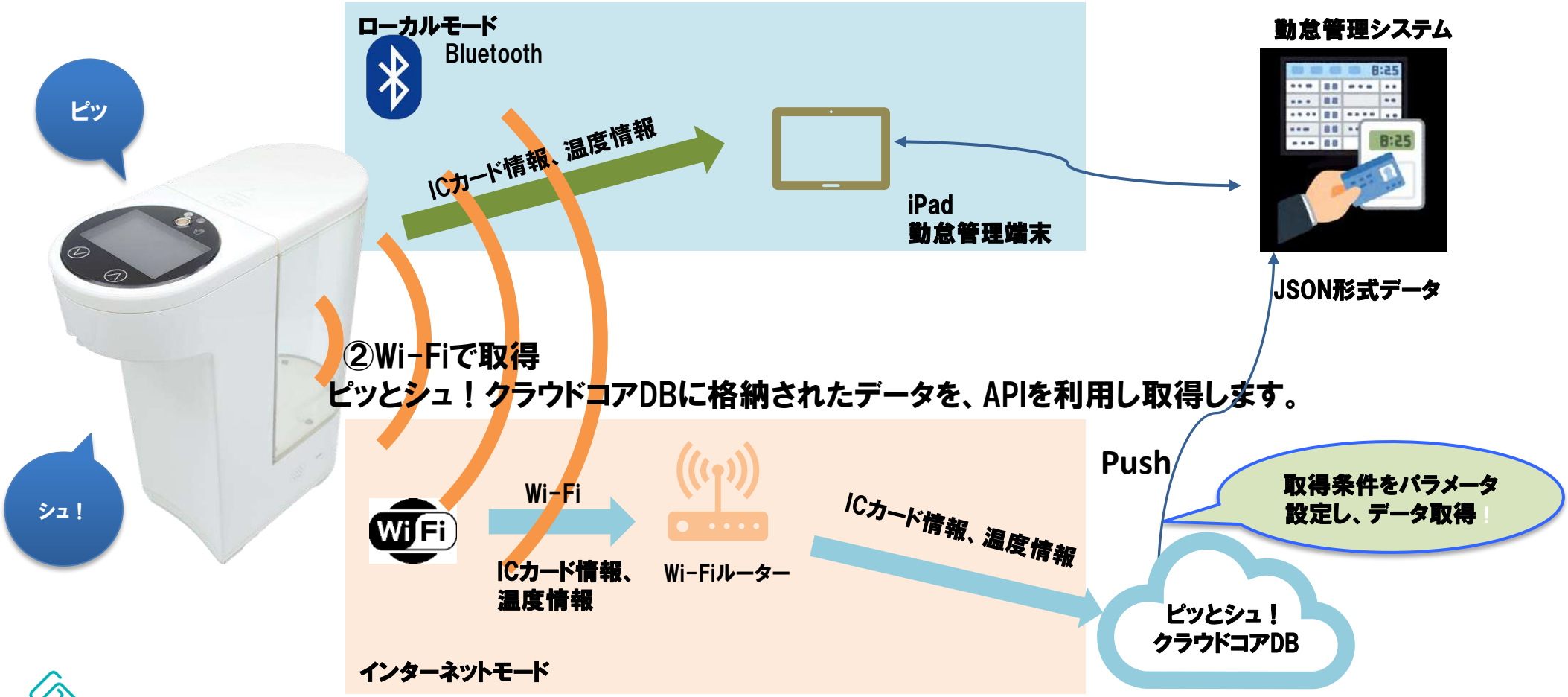


(※)アプリの初期設定が終わった端末

4．2　取得データの活用
4．2．1　勤怠管理システムとの連携

①Bluetoothでデータ取得

各社が提供するiPadなどを利用した勤怠管理用の端末で、出退勤時のICカード情報と温度測定結果を取得します。



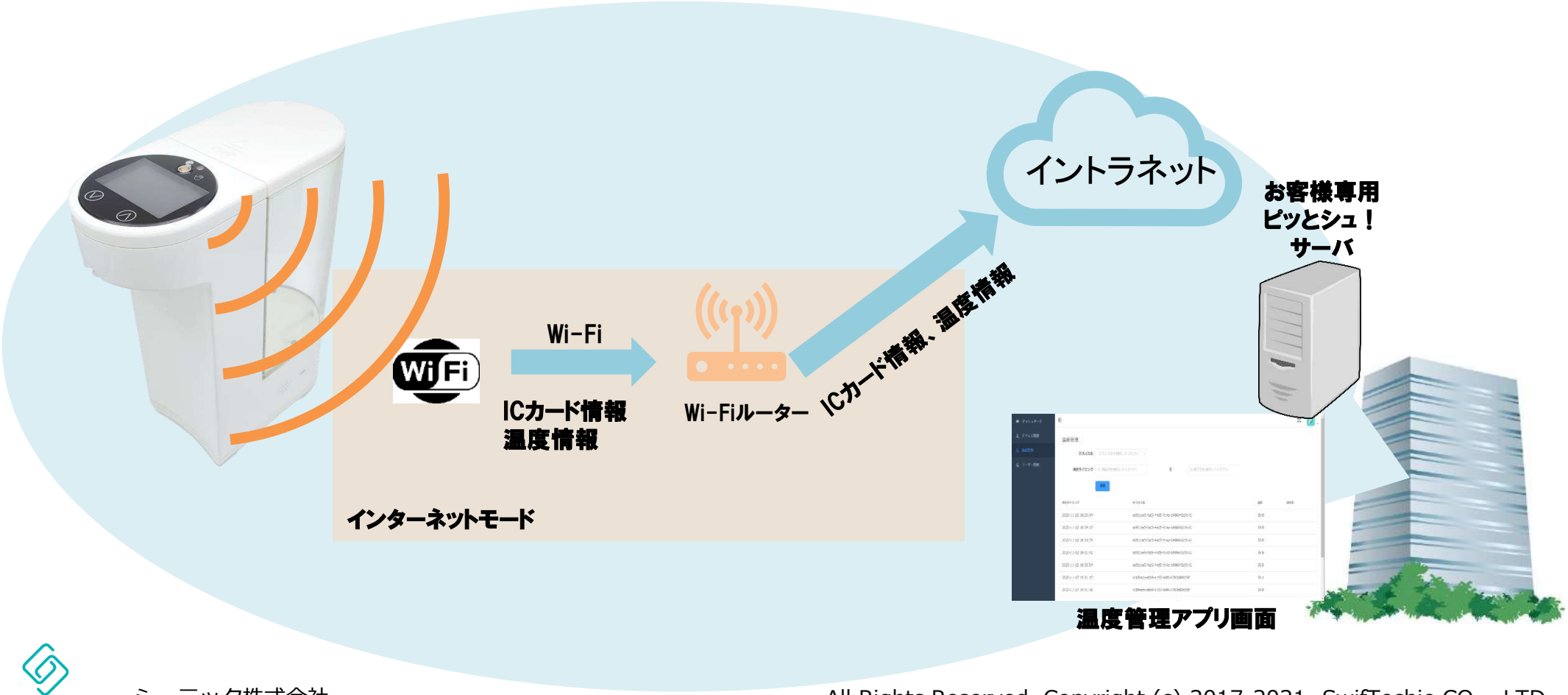
4．2．2　各システムとのデータ連携

ピッとシュ！クラウドコアDBに格納された、測定温度、ICカード、登録時間等は、弊社からアカウントを払い出し、APIに従ってパラメータ設定することで、必要なデータを抽出する事ができます。

人事情報システムや、健康管理システムなどに、個人の日々の温度測定結果をデータ連携することができます。

4．3　お客様専用ピッとシュ！クラウドアプリ（コアDB）のご提供

ピッとシュ！クラウドアプリ（コアDB）をお客様向にに個別のご提供いたします。
お客様の環境（お客様のクラウドやサーバ）の上にピッとシュクラウド！アプリ（コアDB）を構築します。
ピッとシュ！アプリ（コアDB）をお客様環境内でご利用いただけます。
＊ご利用に際し、ピッとシュクラウド！アプリ利用料（年払い）と導入支援費が別途必要です。（個別見積対応）



（参考　API例　MQTT_API概要）

①【資料概要】

デバイスからの検温・消毒情報のプッシュ通知情報を受信する際の内容を定義します。API詳細については、ディベロッパーズガイドを弊社から入手ください。

②【基本情報】　受信に用いる基本情報は、下記となります。

連番	項目	内容	備考
1	MQTT バージョン	3.1.1	
2	通信方式	TCP(ポート1883)	
3	プロトコル	mqtt://	
4	ホストURL	pitosyu-mqtt.swiftechie.com	

③【機能一覧】

連番	機能	内容	備考
1	connect	MQTTサーバーへ接続する	
2	subscribe	トピックを(subscribe)購読する	
3	unsubscribe	トピックを解除(unsubscribe)する	
4	message push	メッセージが受信した時のコールバック	
5	ping	接続を維持するため、pingメッセージを定時に送る	推薦ping間隔は20分。
6	disconnect	接続を切断する	

④【subsucribe機能概要】

デバイスからの検温・消毒情報のプッシュ通知情報を受信する際の内容を定義します。

サービス名	トピック購読
役割	トピックを購読する際の内容を定義。

・パラメータ

連番	パラメータ論理名	パラメータ物理名	バリデーション											フォーマット	備考
			必須	最大文字数	文字種										
					半角						全角				
					制限なし	数字	アルファベット大文字	アルファベット小文字	記号	カナ	制限なし	カナ			
0	パケットID	packet id			-	○	-	-	○	-	-	-	2 bytes	クライアントよりデフォルト値	
1	QoS 1	QoS1		1	-	○	-	-	○	-	-	-	0,1のみ	2をサポートしない	
2	トピック 1	topic1		-	-	○	○	○	○	-	-	-	固定格式	release/d2/v1/{org_id} 購読するトピックは一つのみ	

・戻り値(SUBACK)

連番	戻り値論理名	戻り値物理名	戻り値内容												
			必須	最大文字数	文字種								フォーマット	内容	備考
					半角						全角				
					制限なし	数字	アルファベット大文字	アルファベット小文字	記号	カナ	制限なし	カナ			
1	パケットID	packet id	○	1	-	-	-	-	-	-	-	-		パケット番号	subscribeリクエストのpacket idと同じ
2	エラーコード	returnCode 1	○	1	-	-	-	-	-	-	-	-		topic1の処理結果	0: success QoS 0 1:success QoS 1 2:success QoS 2 128: failure

⑤【subsucribe機能リクエストのサンプルソース(kotlin)】

```
client.subscribe("release/d2/v1/$orgId", 0)
client.subscribeCompletionHandler {
  val id = it.messageId()
  val q = it.grantedQoSLevels()
  log.debug("message id: $id")
  log.debug("QoS granted: $q")
}
```