IoT Engineering
11: Voice Control for Connected Products

Slides: tmb.gr/iot-11

Overview

These slides introduce *voice control for devices*.

How a voice command can control actuators.

How a voice query can read sensor values.

Prerequisites

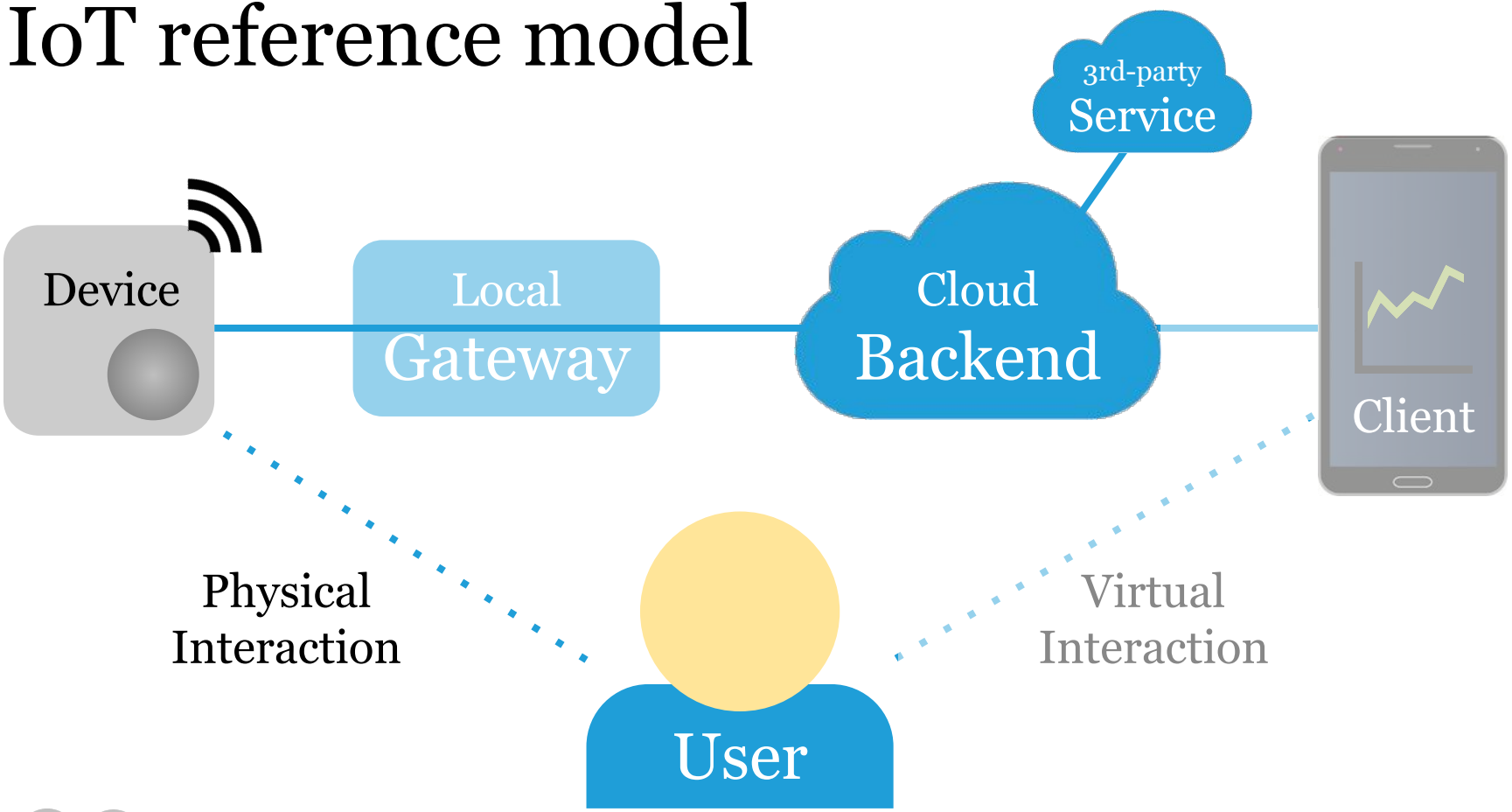
The Raspberry Pi with Node.js hosts our "backend".

For voice control we'll use an Echo or Echosim.io.

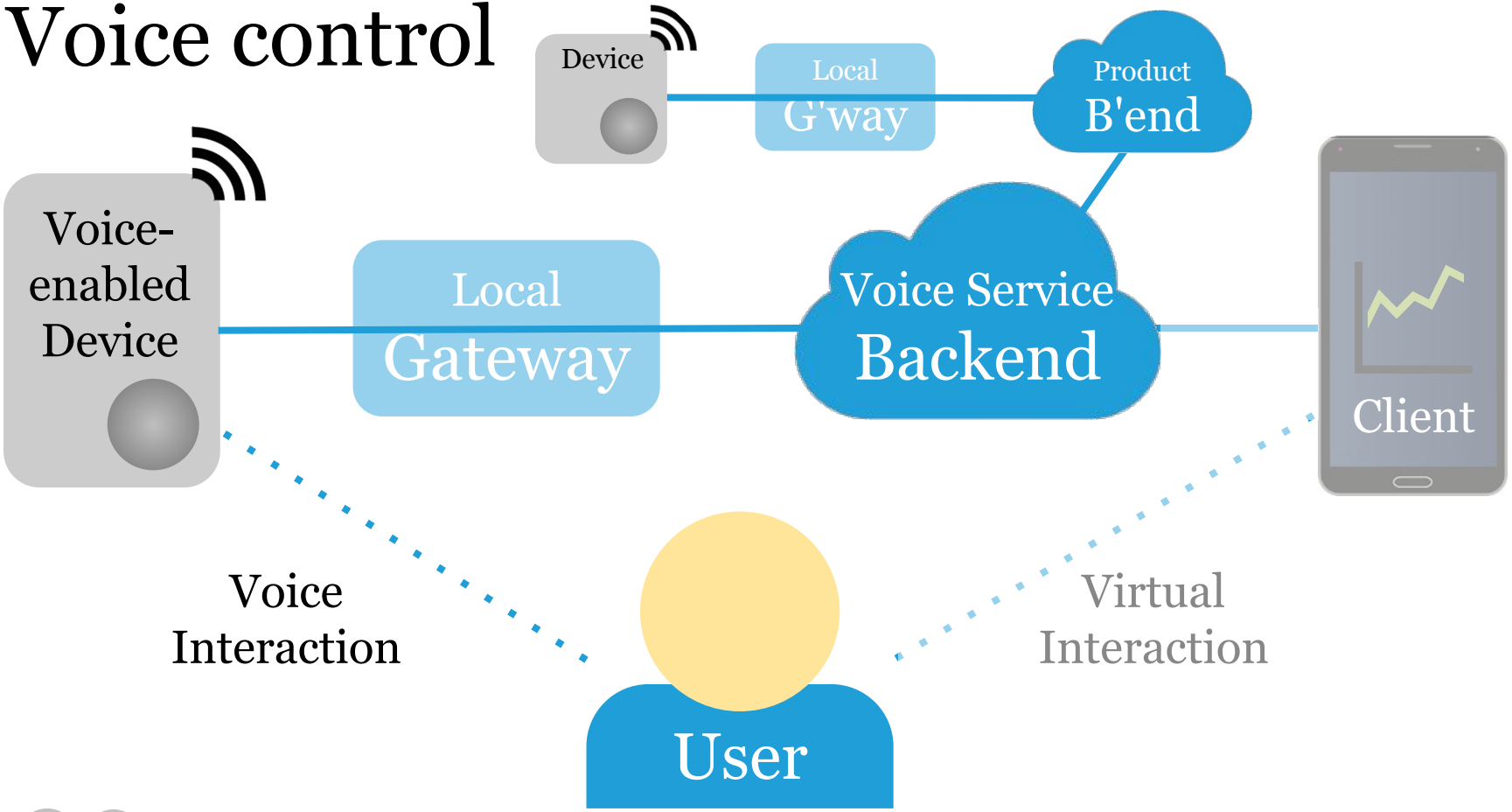
Examples are based on the Alexa voice service.

Note: Slides are in beta, examples will be updated.

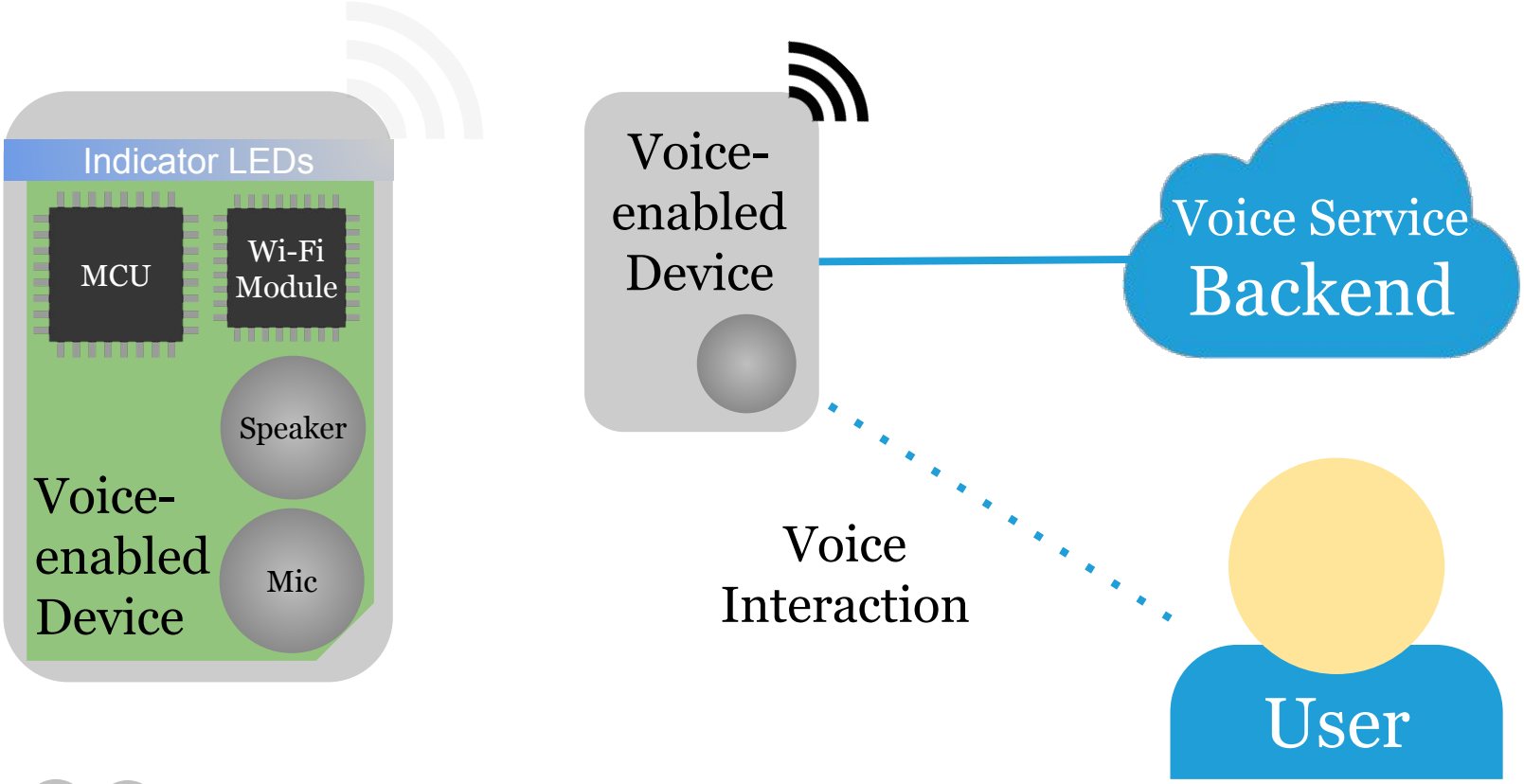
IoT reference model



Voice control



Voice-enabled device



Voice-enabled device

A voice interface can be a separate device, e.g. Echo.

Or a connected product that is (also) voice-enabled*.

In any case there is a mic, an indicator and a speaker.

Processing audio is done at a voice service backend**.

*Example dev kits. **Or locally w/ edge computing.

Voice services

A voice service provides natural language processing.

Voice services include Alexa, Siri & Google Assistant.

Dialogflow is a meta service to use multiple services.

Snips.ai is a local alternative to cloud-based solutions.

Why care? Amazon sold 100 M devices with Alexa.

Voice interaction

Voice interaction provides a "natural" user interface.

Interaction patterns include *commands* and *queries*.

A command usually changes device or database state.

A query can query live or historical data of a device.

Most voice interactions are short and specific.

Voice command

A users starts with a wake word, e.g. "Computer, ...".

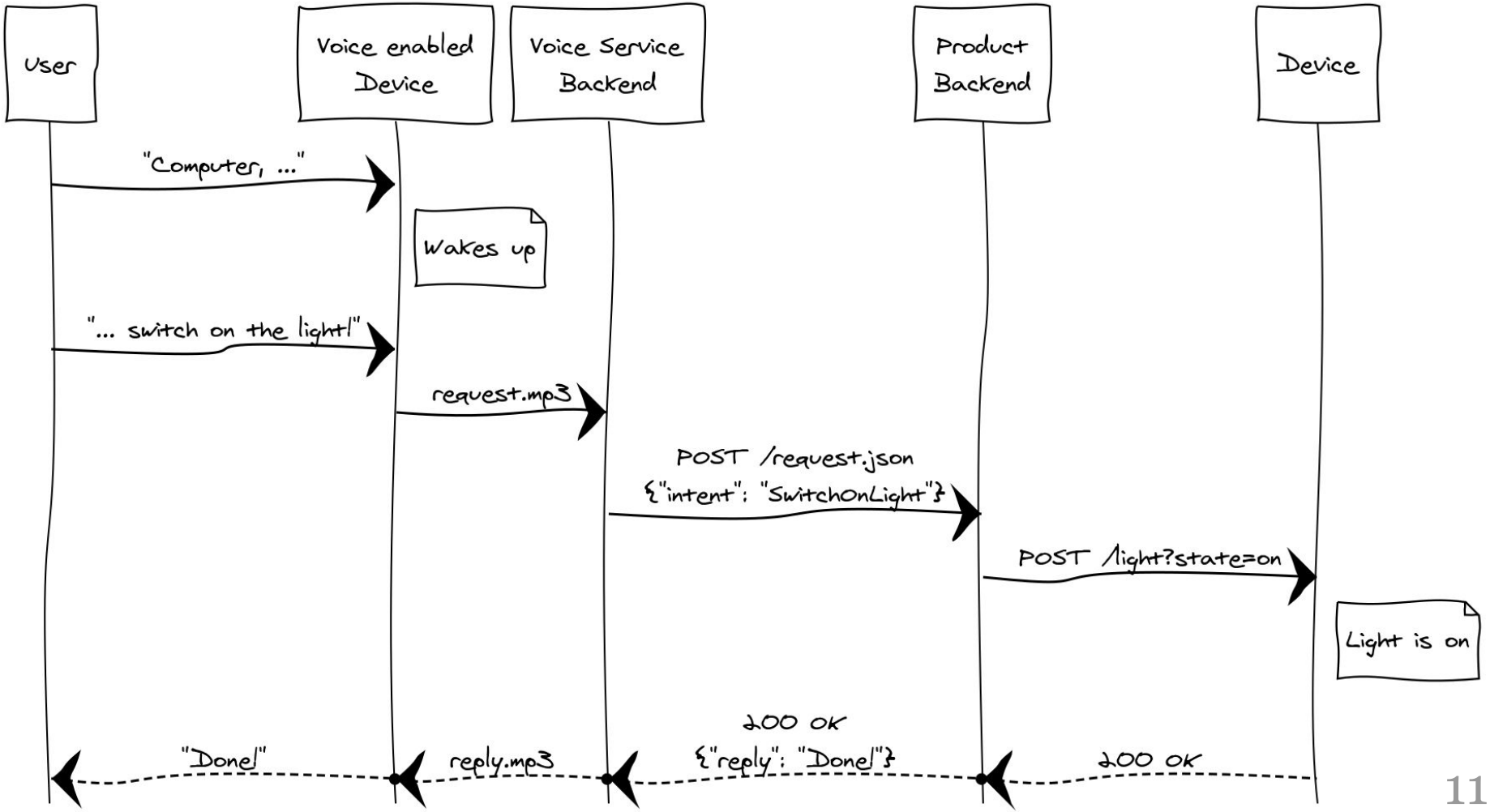
The device records each request as an audio stream.

The voice service backend infers the speaker's intent.

A Webhook call posts the intent, e.g. in JSON format.

The product backend transforms it into actions.

E.g. "Computer, switch on the light!"


User
Voice enabled Device
Voice Service Backend
Product Backend
Device
"Computer, ..."
Wakes up
"... switch on the light!"
request.mp3
POST /request.json
{"intent": "SwitchOnLight"}
POST /light?state=on
Light is on
200 OK
"Done!"
reply.mp3
{"reply": "Done!"}
200 OK

Voice query

A users starts with a wake word, e.g. "Computer, ...".

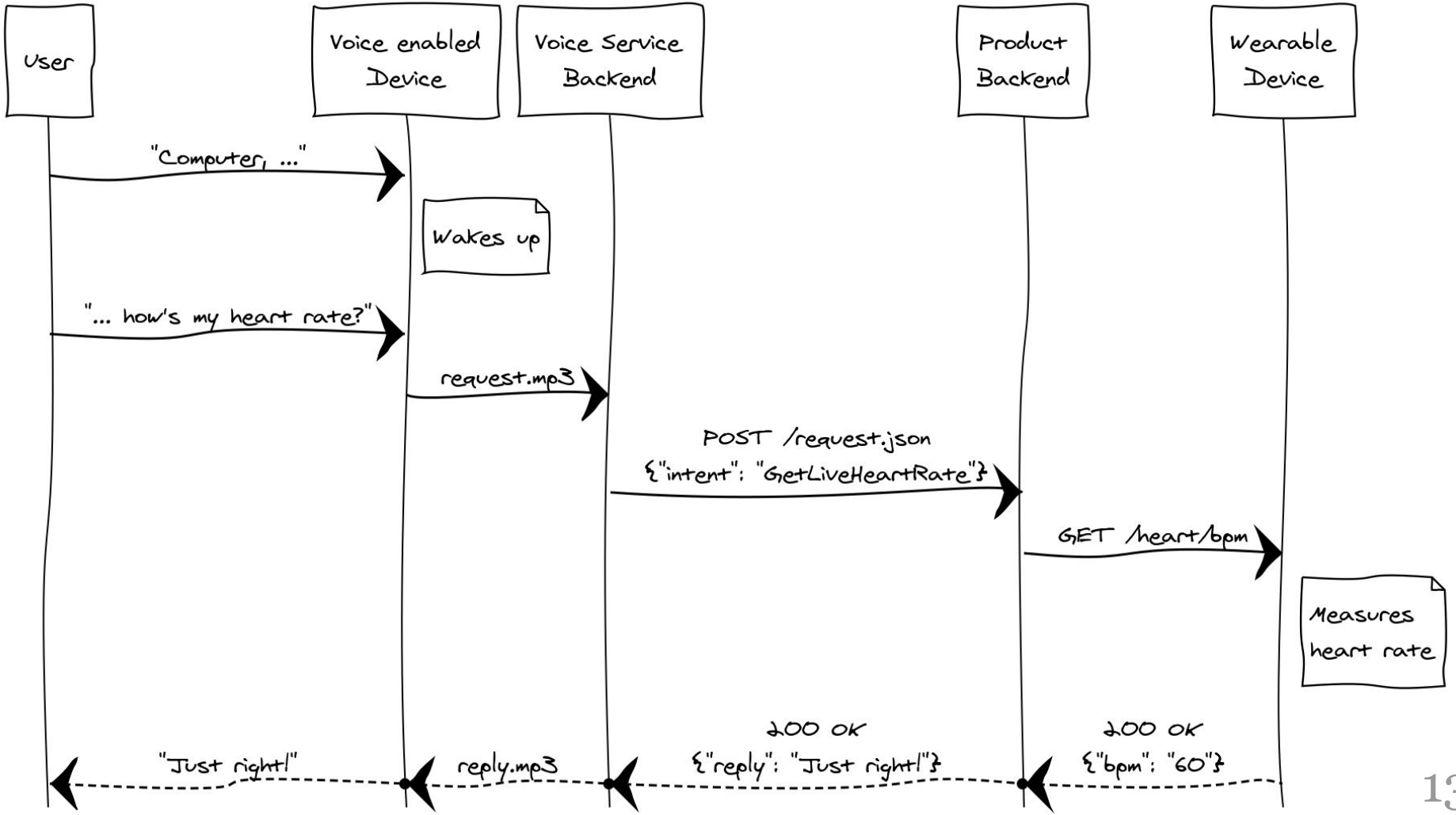
The device records, backend infers the speech intent.

A Webhook call posts the intent, e.g. in JSON format.

The product backend transforms intent into queries.

It gets data from the device and formulates a reply.

E.g. "Computer, how's my heart rate?"


User
Voice enabled Device
Voice Service Backend
Product Backend
Wearable Device
"Computer, ..."
Wakes up
"... how's my heart rate?"
request.mp3
POST /request.json
{"intent": "GetLiveHeartRate"}
GET /heart/bpm
Measures heart rate
200 OK
{"bpm": "60"}
200 OK
{"reply": "Just right!"}
reply.mp3
"Just right!"

Wake word detection

If the voice device detects the *wake word* it wakes up.

E.g. "Alexa", "Computer", "Siri", or "OK Google".

Devices should not record audio before waking up.

A wake word protects privacy and preserves power.

It also provides a clear conversation starting point.

Wake word detection can be done in hardware.

Amazon Alexa

Alexa is Amazon's cloud-based voice service.

"It converts spoken words to text using automatic speech recognition, deduces the speaker's meaning using natural language understanding, and provides the underlying customer intent to your skill."

— from the Alexa Skills Kit documentation.

Alexa skills

Skills are apps for the Amazon Alexa voice service.

A skill runs on the product backend or as glue code.

Here are some examples of smart home Alexa skills.

Amazon provides blueprints of skills to developers.

Custom Alexa skills can be published in the store.

Alexa Skills Kit SDK

The Alexa Skills Kit has a Node.js SDK to build skills.

It includes a simple Node.js Hello World Alexa skill.

The examples use AWS Lambda to run glue code.

Here's how to enable logging on AWS Lambda.

Consider using the Smart Home Skill API.

Hands-on, 15': Hello World Alexa skill

Build and deploy the Hello World Alexa skill tutorial.

Use *Test* or Echosim.io to simulate an Alexa device.

Read the docs on intents, utterances and slots.

How would you integrate a product backend?

Deploying requires an AWS account.

Intents

```
{ "intents": [ // what a user intends
  { "intent": "GetLastFeeding" },
  { "intent": "GetFedToday" },
  { "slots": [
    { "name": "Date",
      "type": "AMAZON.DATE" }],
      "intent": "GetFedAtDate"
    }
] }
```

Utterances

How a user expresses an intent.

```
GetLastFeeding when I last fed the fish
GetLastFeeding when I gave food to the fish

GetFedToday if I fed the fish
GetFedToday did I give the fish any food

GetFedAtDate if I fed the fish {Date}
GetFedAtDate did I feed the fish {Date}
```

Slots

A *slot* is a placeholder for a class of variable input.

E.g. here, *Date* can be today, yesterday, May 4th, …

```
GetFedAtDate did I feed the fish {Date}
```

The parsed date is transmitted in the Webhook call.

Every slot has a slot type, either built-in or custom.

Naming a skill

Amazon asks publishers to choose a unique name.

Ideally, the name is a generic word*, e.g. *fish tank*.

See also the Amazon naming guidelines for skills.

*These will be gone quickly, like short DNS names.

Using the skill

"Alexa, ask *fish tank* to feed the fish!"

"Alexa, ask *fish tank* if I did feed the fish?"

"Alexa, ask *fish tank* did I feed the fish today?"

"Alexa, ask *fish tank* when did I last feed the fish?"

Voice interaction design

Let users speak in their own words, adapt to them.

Individualize your entire interaction, be personal.

Collapse your menus, make all options top-level.

Talk with them, not at them — be relatable.

See how to shift screen-first to voice-first design.

Hands-on, 10': Voice interaction design

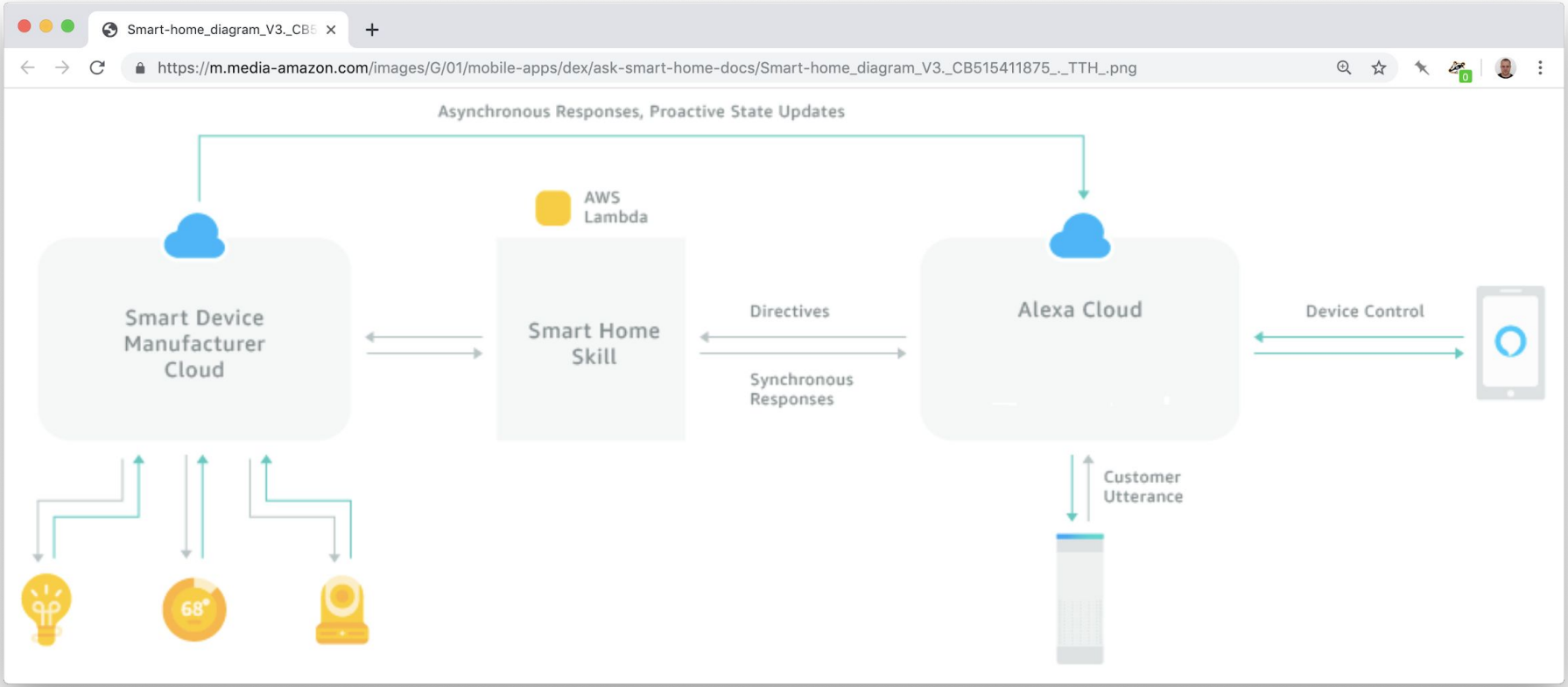
Imagine a simple voice controlled connected device.

Find three use-cases and the corresponding *intents*.

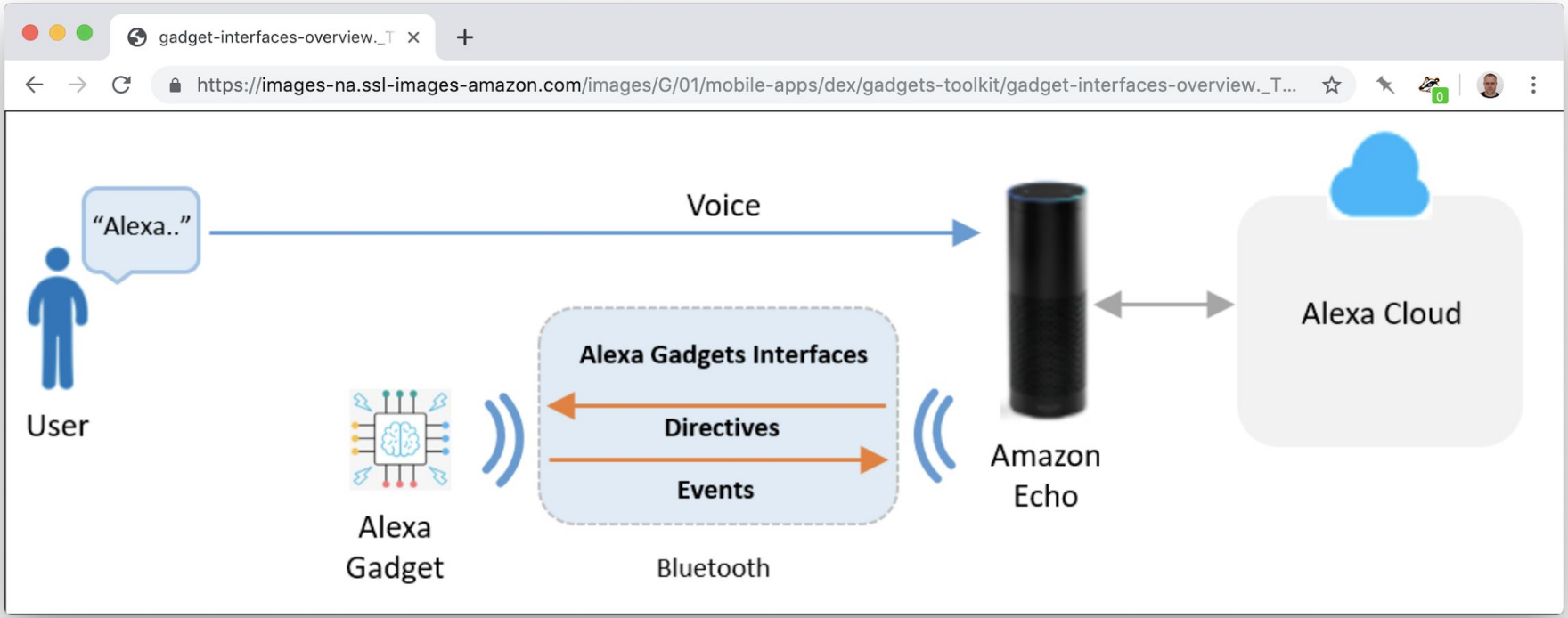
Write down two realistic *utterances* for each intent.

Be ready to present your results.

Smart Home Skill API reference model



Alexa Gadgets Toolkit reference model



How gadgets interact with Alexa via Bluetooth.

Alexa Voice Service API

Alexa Voice Service allows to "voice-enable" devices.

Amazon lists hardware dev kits for manufacturers.

The functional requirements are defined in detail.

And a user experience design guide is provided.

Dialogflow Natural Language Service

Dialogflow provides a natural language "meta" API.

The service works with Google, Amazon, Apple, etc.

It abstracts dialogs for chat bots and voice control.

You define intents and entities in different contexts.

Your fulfillment server consumes Webhook calls.

Privacy considerations

"Voice prints", like fingerprints, can identify people, e.g. the Alexa voice service supports voice profiles.

Voice recordings are personal data* under the GDPR, so for EU citizens it's possible to get their recordings.

Sending personal data to a cloud backend trades user privacy for use cases that are not possible on-device.

*Got an Alexa? Check your personal archive.

Unintended consequences

A Southpark episode spams people's shopping lists.

Normal sentences trigger recording of conversations.

Judges issue warrants to hand over Alexa recordings.

Amazon complies with GDPR, but sends wrong data.

And Amazon employees listen to users recordings.

Edge-device based solutions

Edge-based solutions often work without a backend.

Snips.ai is a simple, private-by-design voice assistant.

Project Alias is a privacy add-on for voice assistants*.

Snowboy is an on-device hotword detection engine.

*Here's a video of how it works.

Hands-on, 10': Use cases in context

Come up with a use case for a *home*, *hotel* & *hospital*.

What changes with the context, what stays the same?

Who is the user? What does the system (not) know?

Who can see the data? How private is the data?

See Alexa for business & hospitality and read this.

Summary

We saw how a voice interface is connected to a device.

Backends are integrated via a simple Webhook call.

Voice interaction includes commands and queries.

Voice services provide intents, and values for slots.

Training a voice app means collecting utterances.

Feedback or questions?

Write me on https://fhnw-iot.slack.com/

Or email thomas.amberg@fhnw.ch

Thanks for your time.

Moritz Wittmann on Twitter: "S

https://twitter.com/MoritzWittmann/status/908319633660416001

Home

Tweet

Moritz Wittmann
@MoritzWittmann

Follow

South Park messing with Alexa

3:20 PM - 14 Sep 2017

1,026 Retweets **1,334** Likes

Moritz Witt
@MoritzWittmann

Software Enginee

Stuttgart, Gerr

Joined Januar