

# IoT Engineering

## 12: Raspberry Pi as an IoT Edge Device

CC BY-SA 4.0, T. Amberg, FHNW  
(unless noted otherwise)

Slides: [tmb.gr/iot-12](https://tmb.gr/iot-12)

# Overview

These slides introduce *Edge Computing on the Pi*.

A definition and use cases for edge computing.

How to read sensors and use the Pi camera.

# Prerequisites

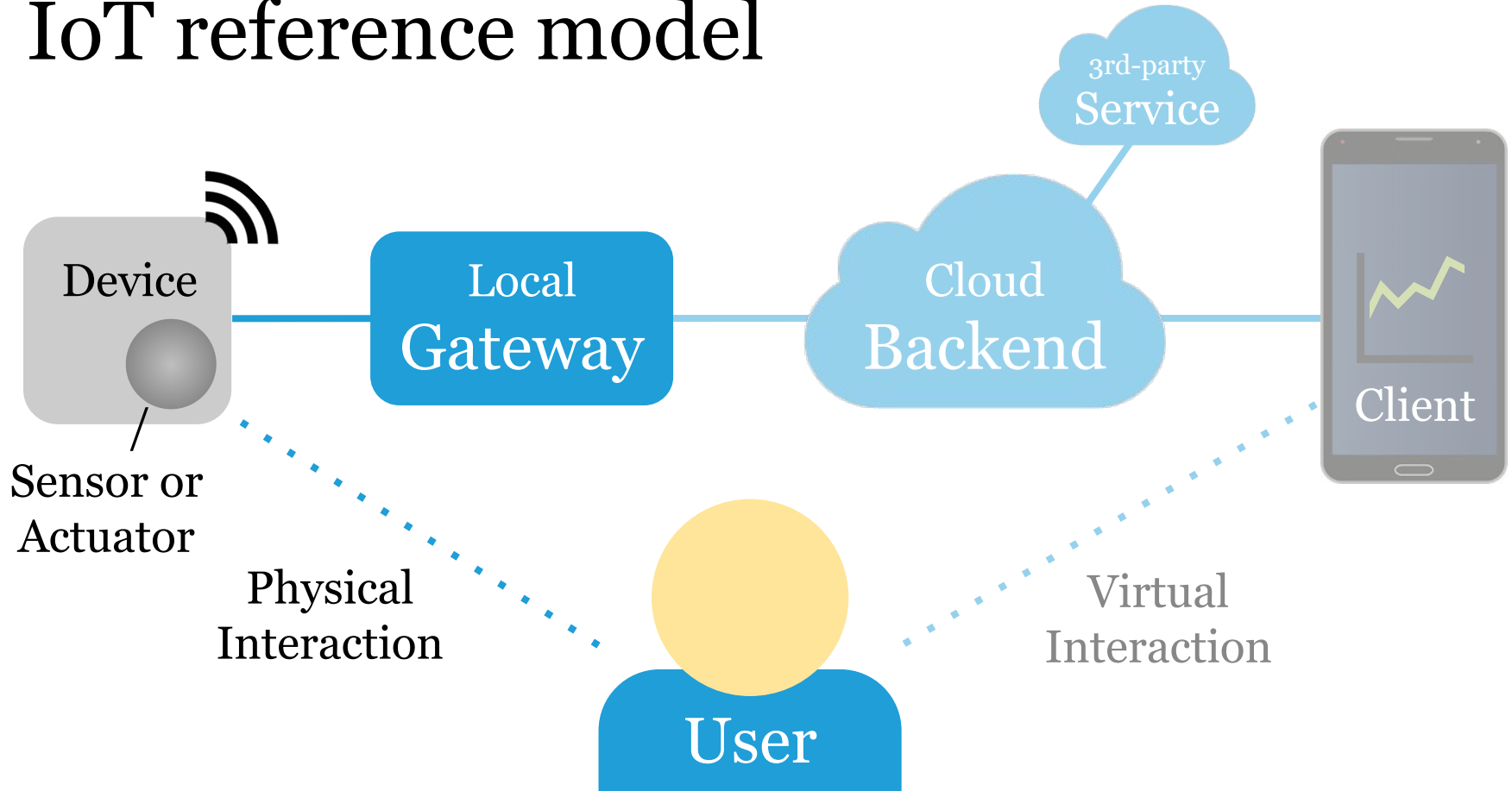
We'll use the **Raspberry Pi** with **Node.js** and **Python**.

To use **Grove sensors** a **Grove base hat** is required.

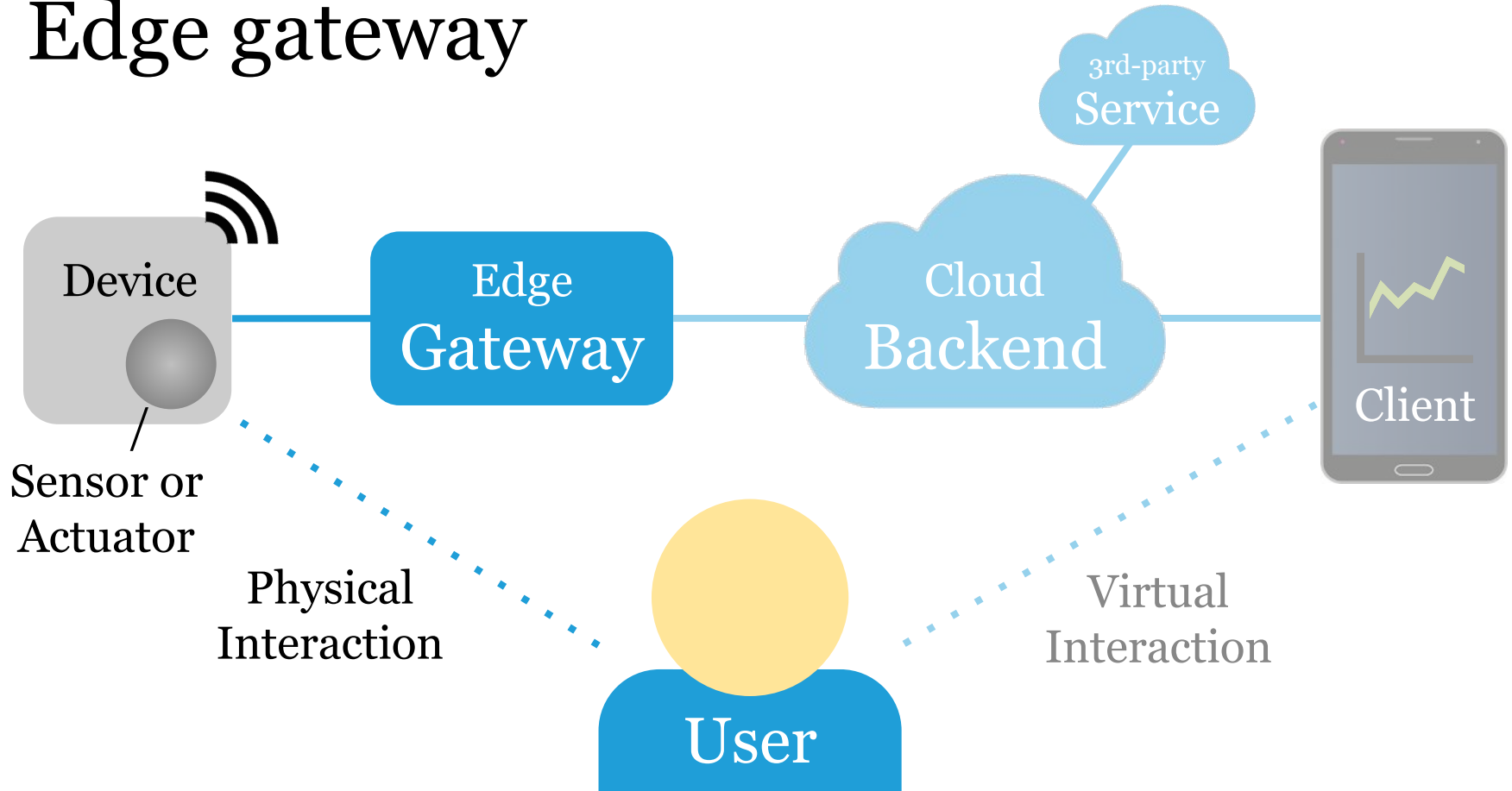
A Raspberry **Pi cam** and **Coral TPU** can be used.

Note: Slides are in beta.

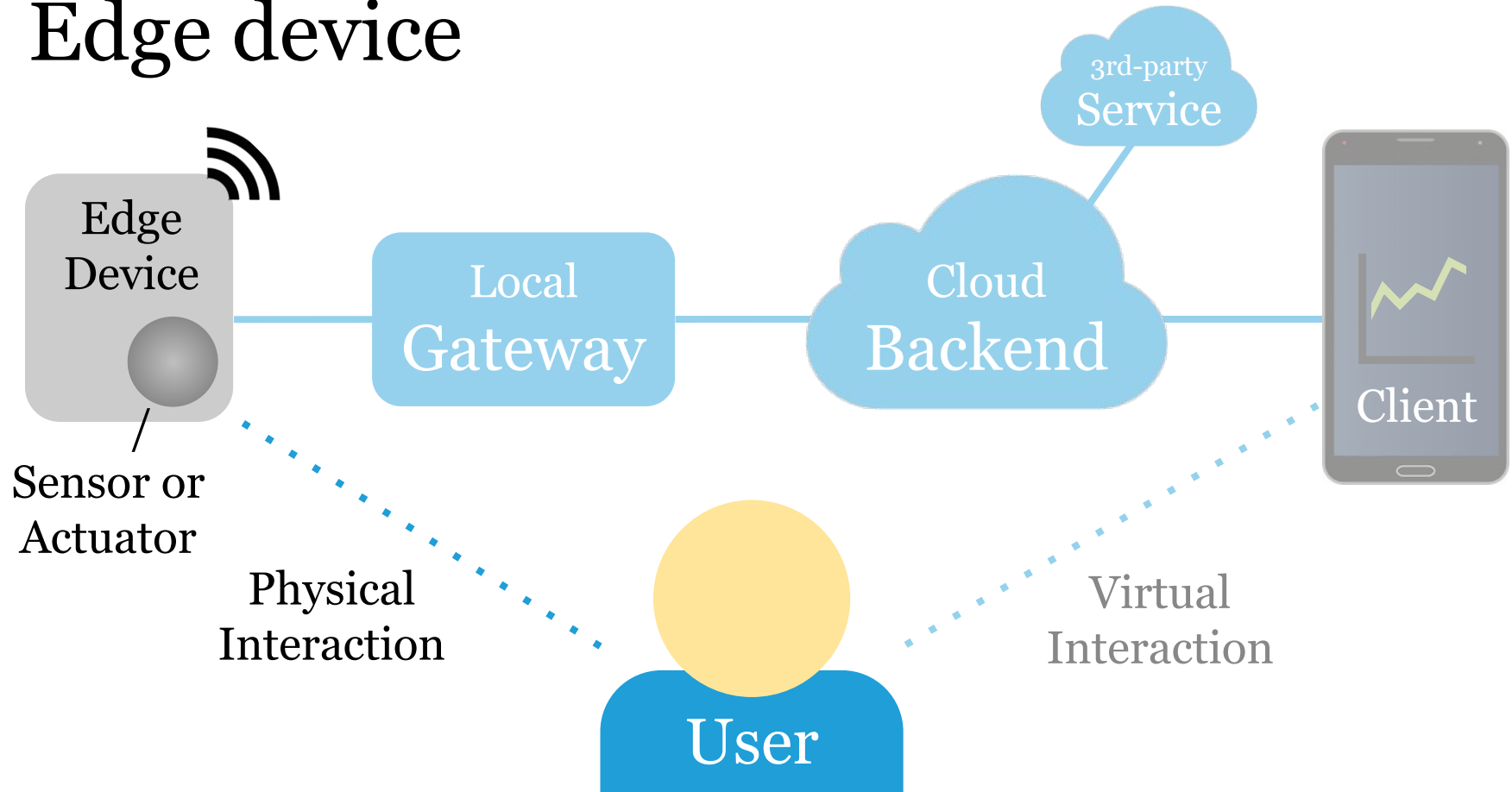
# IoT reference model



# Edge gateway



# Edge device



# Edge computing

"*Edge computing* or fog computing [...] represents a shift in architecture in which intelligence is pushed from the cloud to the edge, localizing certain kinds of analysis and decision-making." — [The edge of the IoT](#)

"[It] enables quicker response times, unencumbered by network latency, as well as reduced traffic, selectively relaying [...] data to the cloud." — (as above)

# Edge computing patterns

Sensor  $\rightarrow$  Device  $\rightarrow$  Edge GW  $\rightarrow$  Device  $\rightarrow$  Actuator.

Sensor  $\rightarrow$  Edge Device  $\rightarrow$  Actuator.

Use cases: Low latency decisions, or lots of data.

E.g. cloudless voice recognition unlocks a door.

Or FFT over local machine data triggers alerts.



# Edge device/gateway

A device/gateway with *substantial* computing power.

Supports high bandwidth sensors, e.g. audio, video.

Enables *local* data analysis and decision-making.

Reduces decision latency and network traffic.

Historically, the term has been used in [networking](#).

# Raspberry Pi as an edge device

The Raspberry can be used as a edge device/gateway.

It has (digital) GPIO pins to read data from sensors.

A camera connector enables taking pictures & video.

Add-on *hats* enable additional hardware capabilities.

# Raspberry Pi GPIO with Node.js

Check the [Raspberry Pi pinout](#), pins are 3.3V, *not* 5V.

To read/write [GPIO](#) pins, install [rpi-gpio](#):

```
$ npm install rpi-gpio
```

Or [johnny-five](#) with [raspi-io](#):

```
$ npm install johnny-five raspi-io
```

Or [gpio-stream](#):

```
$ npm install gpio-stream
```

# Digital output with Node.js

.js

```
const gpio = require("rpi-gpio");
const pin = 29; // BCM pin 5, D5
let state = false;
gpio.setup(pin, gpio.DIR_OUT, () => {
  setInterval(() => {
    state = !state;
    gpio.write(pin, state, (err) => {
      console.log(err);
    });
  }, 500); // ms
});
```

# Digital input with Node.js

.js

```
const gpio = require("rpi-gpio");
const pin = 16; // BCM pin 16, D16

gpio.setMode(gpio.MODE_BCM);
gpio.setup(pin, gpio.DIR_IN, gpio.EDGE_BOTH);
gpio.on("change", (pin, value) => {
    console.log("pin " + pin + ": " + value);
});
console.log("watching pin " + pin + "...");
```

# Raspberry Pi Grove GPIO with Python

To use Grove and read analog input, use a [Grove hat](#).

Seeed provides a [library](#) and [examples](#)\* for Grove.

```
$ sudo apt-get install python-pip python3-pip  
$ curl -sL https://github.com/Seeed-Studio/\  
grove.py/raw/master/install.sh | sudo bash -s -
```

\*Both are available in [Python](#) only.

# Raspberry Pi Grove GPIO w/ Node-RED

[Node-RED](#) supports GPIO if it runs on a Raspberry Pi.

See [Running on Raspberry Pi](#) and [Accessing GPIO](#).

There are several [nodes based on grove.py](#), e.g.

- [node-red-grovepi-nodes](#)
- [node-red-contrib-grovepi](#)
- [node-red-contrib-grove](#)

# Hands-on, 15': Raspberry Pi GPIO

Use the [Grove hat](#) to connect [Grove sensors](#) to the Pi.

Install the grove.py library and use it with Node-RED.

Check the [Grove hat pinout](#), use 3.3V modules only.

No hat? Use [wires](#), check the [Raspberry Pi pinout](#).



# Photo/video with Pi cam in Node.js .js

```
// $ sudo raspi-config > Ifc. options > Camera
// $ npm install raspicam
const RaspiCam = require("raspicam");

const cam = new RaspiCam({
  mode: "photo", // or "video" by default 5s
  output: "./photo.jpg" // or "./video.h264"
});
cam.start();
```

# Edge computing use cases

Categories\* of use cases enabled by edge computing:

- Analytics
- Sensor fusion
- Embedded vision
- Embedded machine learning

\*There might be some overlap.

# AWS IoT GreenGrass

[AWS IoT GreenGrass](#) is Amazon's edge gateway offer.

"AWS IoT Greengrass seamlessly extends AWS onto physical devices so they can act locally on the data they generate, while still using the cloud for management, analytics, and durable storage."

See [AWS IoT GreenGrass documentation](#) for details, including a [Raspberry Pi \(3B+\) GPIO Connector](#).

# Azure IoT Edge

[Azure IoT Edge](#) is Microsoft's edge gateway project.

"Cloud intelligence deployed [...] on IoT edge devices"

The gateway code, used as a template, is [open source](#).

It runs [on a Raspberry Pi](#), but [not on the Pi Zero W](#).

See the [Azure IoT Edge documentation](#) for details.

# Baetyl

[Baetyl](#) by Baidu is an edge gateway to "extend cloud computing, data & service seamlessly to edge devices"

It integrates with the Baidu cloud management suite.

The gateway code is [open source](#) and runs on the Pi.

See the [Baetyl docs](#) and [LFEEdge.org docs](#) for details.

# Eclipse Kura

Eclipse Kura is an [open source](#) IoT Edge Framework.

It's based on Java/OSGi, runs on Raspberry Pi (2/3), and offers API access to Serial ports, GPIOs, I2C, etc.

Local *field protocols* include Modbus, OPC-UA and S7, transformed to MQTT with flow programming.

See the [Eclipse Kura documentation](#) for details.

# EdgeX Foundry

[EdgeX Foundry](#) is a "open platform for the IoT edge"

The edge gateway microservice code is [open source](#).

Communication is based on ZeroMQ, services in Go.

See the [EdgeX Foundry documentation](#) for details.

# Hands-on, 10': Edge gateways

Chose one of the edge gateway projects and analyse it.

Which protocol is used to transmit data to the cloud?

How are updates deployed to the gateway, by whom?

How can a proprietary local protocol be integrated?

Be prepared to present your results.

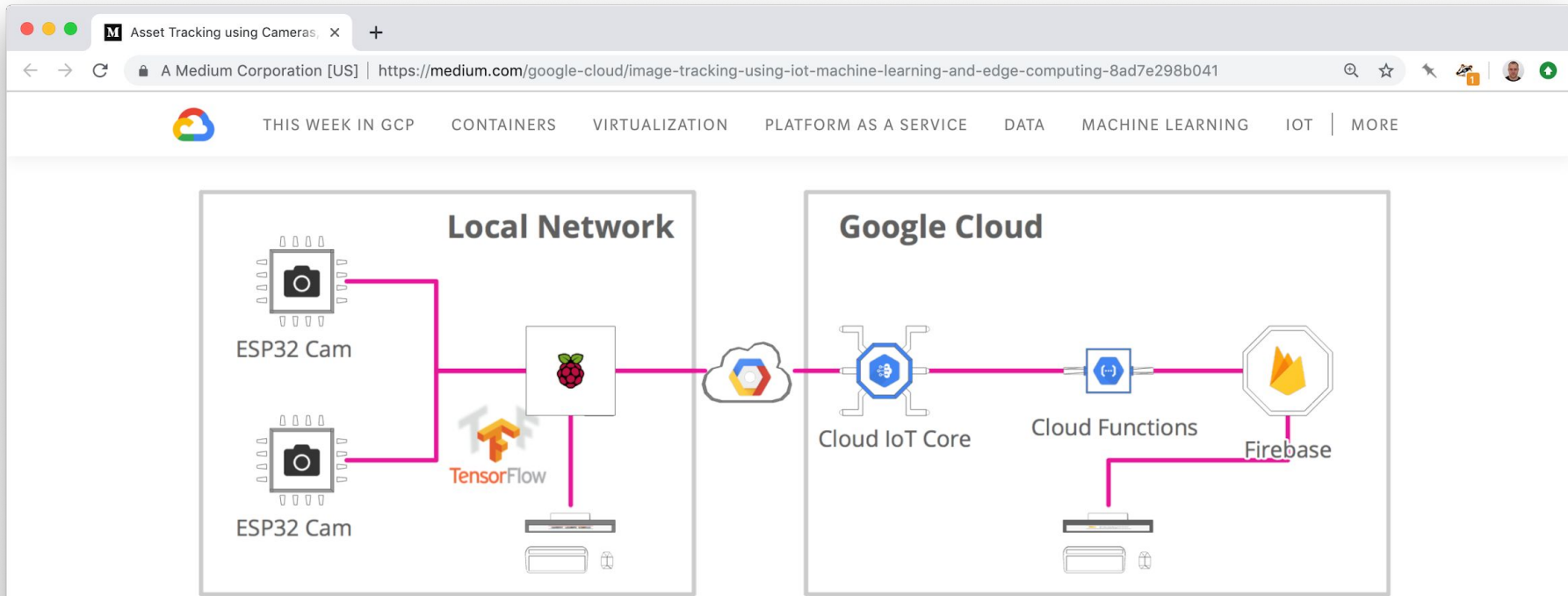


# Embedded vision/machine learning

Devices like [Nvidia Jetson Nano](#), [Coral Dev Board](#) and Raspberry Pi add-ons like the Intel Movidius or the Coral USB accelerator bring ML to "the edge".

Smaller devices with integrated cameras, like [JeVois](#), [Pixy](#), [OpenMV cam](#), or [Sipeed MAix](#) let "things" see.

*Machine learning* here means *inference*, but *training* models gets easier, too — e.g. with [EdgeImpulse](#).



Asset Tracking using Cameras, IoT,  
Machine Learning and Edge  
Computing.

# Synthetic Sensors

[Synthetic Sensors](#) is a nice edge-computing example.

It uses a mic and machine learning to "hear" events.

And also [sensor fusion](#) to further reduce uncertainty.

The setup and features are described in [this paper](#).

Consider watching [the video](#).

# JeVois (2017)

The **JeVois** is an **open source** machine vision camera.  
It is self-contained, with a 4 core CPU, USB & UART.  
It runs OpenCV, TensorFlow, Caffe, Darknet < 50\$.  
It processes video and outputs Serial ASCII strings.  
It can detect faces, barcodes, and "salient" events.

# Google AIY (2017)

Google AIY is a Do-it-Yourself machine learning kit.

It's based on a Raspberry Pi Zero W with a "bonnet".

A Intel Movidius VPU processes video from a Pi cam.

It detects "joy".

Or cat/dog/etc.

See ML models.

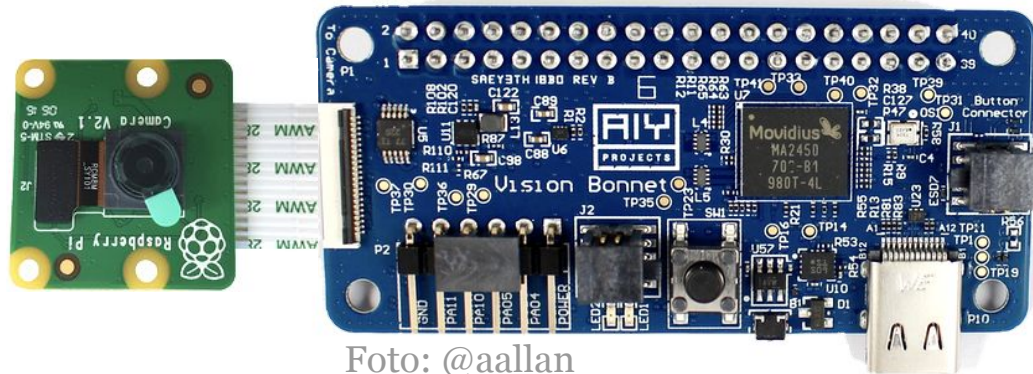


Foto: @aallan

# Snips.ai (2018)

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

The hardware kit is a Raspberry Pi with a mic array.

Intents are formatted similar to AWS Alexa intents.

Inference happens on the device, not in the cloud.

Federated learning allows gradual improvement.

Google explains federated learning in this post.

# Coral Edge TPU (2019)

The **Coral Edge TPU** is available as a *USB accelerator*.

It speeds up ML **inference** on Linux / Raspberry Pi.

The USB "coprocessor" runs **TensorFlow Lite** models.

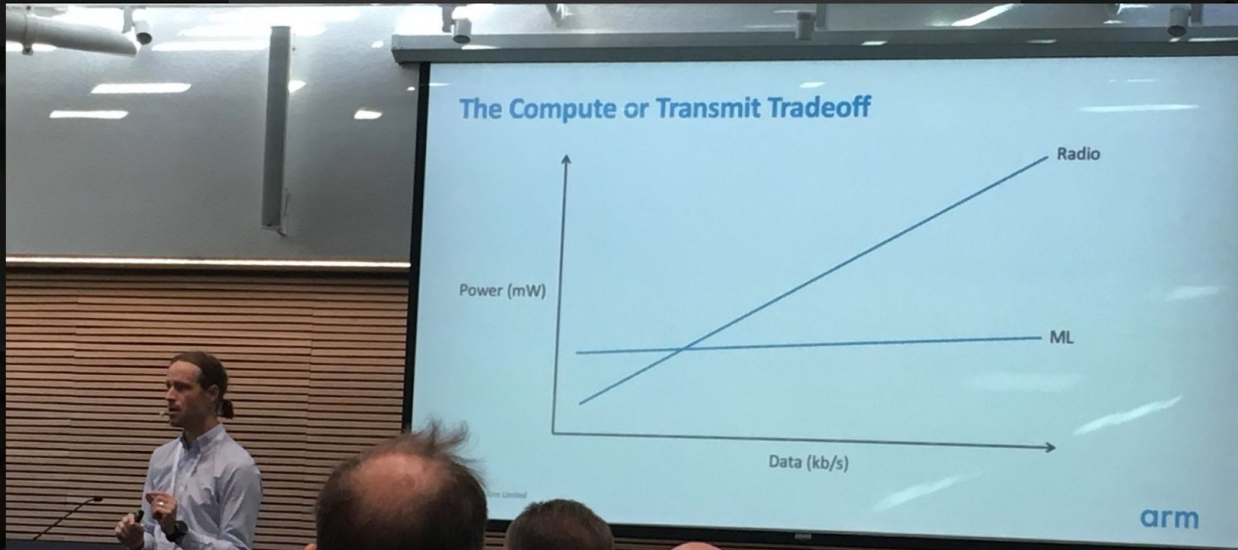
A **Python API** allows performing image classification, object detection, and transfer-learning on your device.

There is **Coral documentation** to **get started**.

Home

Tweet

✕



Jaakko Ala-P

@japikas

Technology director,  
Industrial Internet, Embedded  
Mobile, and stuff.

📍 Finland

🔗 embeddedexperts

📅 Joined May 2014



Jaakko Ala-Paavola @japikas · 4h

Why #embedded #MachineLearning? @zach\_shelby illustrates that very clearly by power efficiency at #embeddedconference in #HEL. With increased data, your radio power budget increases accordingly, whereas your #ML power consumption stay more flat.





SHOP

LEARN

BLOG

SUPPORT

Find a Retailer

Need Help? ▾



LOG IN

REGISTER

PRODUCT MENU

find products, tutorials, etc...



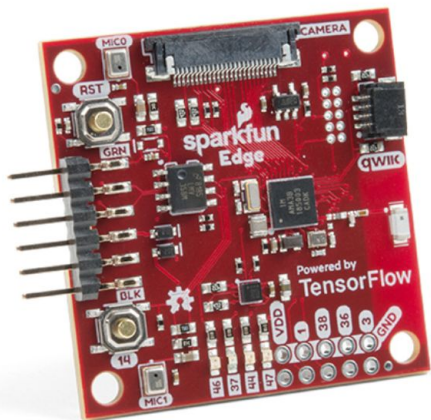
SPARK

EDUCATION

AVC

FORUM

HOME / PRODUCT CATEGORIES / MACHINE LEARNING AND AI / SPARKFUN EDGE DEVELOPMENT BOARD - APOLLO3 BLUE



## SparkFun Edge Development Board - Apollo3 Blue

DEV-15170 ✨

★★★★☆ 6

\$14.95



Shipping outside of the US?

[Click here for info](#)



**Notify Me** We do not currently have an estimate of when this product will be back in stock.

- 1 +

BACKORDER

Stock availability

DESCRIPTION

FEATURES

DOCUMENTS

Edge computing is here! You've probably heard of this latest entry to the long lineage of tech buzzwords like "IoT," "LoRa," and "cloud" before it, but what is "the edge" and why does it matter? The cloud is impressively powerful but all-the-time connection requires power and connectivity that may not be available. Edge computing handles discrete tasks such as determining if someone said "yes" and responds accordingly. The audio analysis is done at the edge rather than on the web. This dramatically reduces costs and complexity while limiting potential data privacy leaks.

In collaboration with Google and Ambiq, SparkFun's Edge Development Board is based around the

# Hands-on, 10': Embedded ML use cases

Assuming a devices can know your face, voice, mood.

Which new use cases become possible with edge ML?

Try to take a "**thing centered**" perspective, as a device.

Be prepared to present your results.

# Summary

We defined edge-computing as "local intelligence".

We learned to access GPIO pins on a Raspberry Pi.

We looked at and compared edge gateway projects.

We understand main use cases of edge computing.

Next: From prototype to connected product.

# Feedback or questions?

Write me on Teams or email

[thomas.amberg@fhnw.ch](mailto:thomas.amberg@fhnw.ch)

Thanks for your time.