

JORGE HIDALGO
6 DE MAYO DE 2017

ALL YOUR FACES BELONG TO US




open
south
code

OPENSOURCE 2017


Reconocimiento de imágenes
con tecnologías abiertas

SOBRE MÍ

Jorge Hidalgo

 *@_deors*

 *deors*

 *<https://deors.wordpress.com/>*

Senior Technology Architect

Global Java Lead – Accenture Technology

Custom Distributed, Architecture & DevOps Lead –
Accenture Delivery Center in Spain



INTERNET DE LAS COSAS / INTERNET OF THINGS

- **Cosas → Objetos:**

- Sensores de presión en neumáticos
- Monitores de implantes coronarios
- GPS en taxis
- Termómetros en oficinas
- ...

- **Con un identificador único**

- **Conectados a Internet**

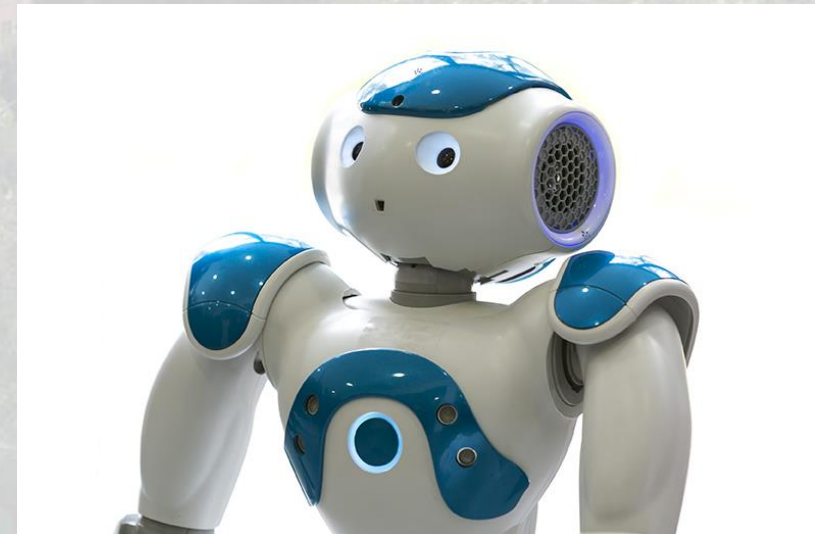
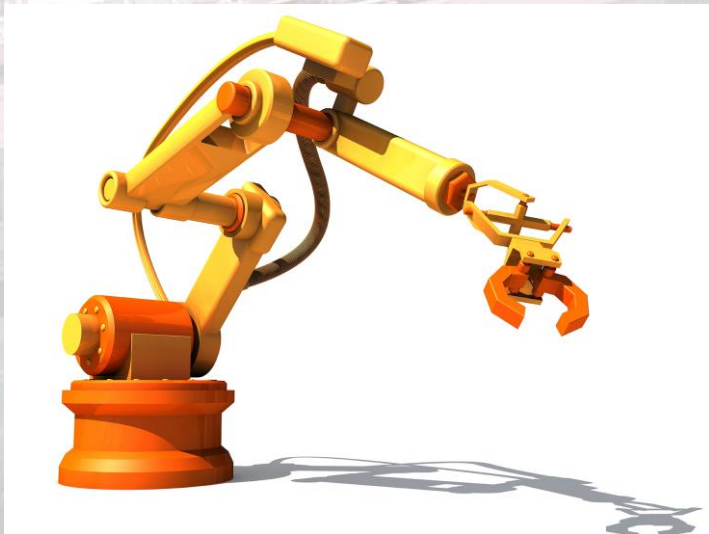
- **Transfieren datos sin intervención humana**

- **Se integran y forman parte de un sistema mayor**



NECESITAMOS INTELIGENCIA EN LAS "COSAS"

- **Respuesta autónoma**
- **Capacidad de toma de decisiones**
- **Capacidad de adaptación**



RECONOCIMIENTO DE IMÁGENES

• Soluciones cerradas:

- Microsoft Computer Vision API ([link](#))
- IBM Watson Visual Recognition ([link](#))
- Google Cloud Vision API ([link](#))
- Amazon Rekognition ([link](#))
- Clarifai ([link](#))

• Soluciones abiertas:

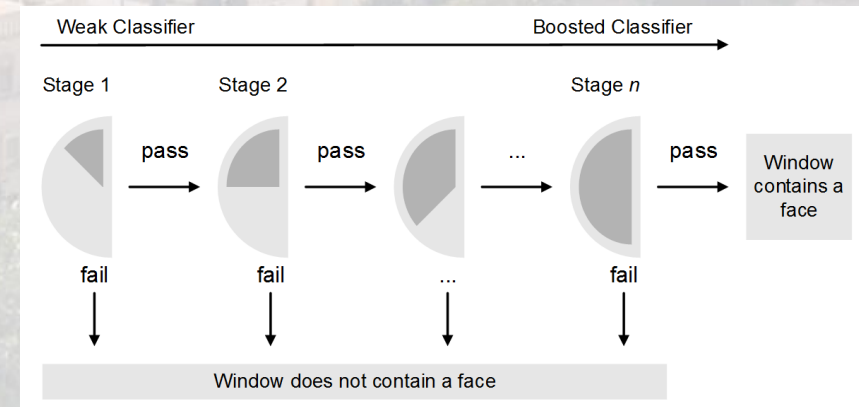
- OpenCV (www.opencv.org)
- SimpleCV (simplecv.org)
- TensorFlow (www.tensorflow.org)
- AForge.Net (www.aforogenet.com)
- VLFeat.org (www.vlfeat.org)
- Point Cloud Library (PCL) (pointclouds.org)
- Gesture Recognition Toolkit (GRT) (www.nickgillian.com/grt)

OPENCV LIBRARY

- **Desarrollada en C++**
- **Con interfaces para Python y Java**
- **Libre (licencia BSD)**
- **Soportada en Windows, Linux (x86 y ARM), Mac OS, iOS y Android**
- **Foco importante en aplicaciones de tiempo real:**
 - Optimizada para uso de hardware de aceleración
 - Optimizada para uso de múltiples núcleos
- **Última versión: 3.2.0 (diciembre de 2016)**
- **En GitHub: <https://github.com/opencv/opencv>**

CASCADE CLASSIFIER

- **Un conjunto de reglas (en formato XML)**
- **Se aplica a cada *frame* de un *stream* de vídeo (o a imágenes individuales)**
- **Detecta rasgos (*features*)**
 - Una cara humana
 - Una cara de gato
 - Un coche
 - La forma de un árbol
 - ...
- **OpenCV contiene un conjunto de clasificadores listos para usar**
- **Se pueden crear nuevos clasificadores mediante entrenamiento**



CASCADE CLASSIFIER - TIPOS

- **LBP**

- Más rápido
- Cálculos sobre enteros
- La mejor opción para entornos móviles/embebidos

- **HAAR**

- Más preciso
- Cálculos sobre punto flotante
- La mejor opción cuando el % de acierto es fundamental

ENTRENAMIENTO EN OPENCV

- **Es un proceso lento**
- **Requiere un conjunto numeroso de imágenes con el rasgo a detectar**
 - Para formas rígidas (un logo), basta con unas pocas
 - Para formas variables (un tipo de objeto), se necesitan cientos
 - Para rasgos de seres vivos (cara, pelo, forma corporal...), se necesitan miles
- **Cada imagen vendrá acompañada de las coordenadas de los rasgos**

```
img1.jpg 1 140 100 45 45  
img2.jpg 2 100 200 50 50 50 30 25 25
```

- **Se enriquece la muestra de 'positivos' con variaciones (*samples*)**
- **Requiere un conjunto aún más numeroso de imágenes 'negativas'**

ENTRENAMIENTO EN OPENCV

- **La herramienta de entrenamiento actual es *opencv_traincascade***
- **Recibe como entradas:**
 - Fichero de vectores generado a partir de las imágenes 'positivas'
 - La colección de imágenes 'negativas'
 - Parámetros de configuración del algoritmo de *machine learning*
- **Usa los positivos para formular las hipótesis y mejorarlas**
- **Usa los negativos para confirmar las hipótesis y mejorarlas**
- **El resultado final es el fichero con los datos del *cascade classifier* (XML)**

OPENCV HELLO CAM! – PYTHON VERSION

```
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2

with PiCamera() as camera:
    camera.resolution = (1920, 1080)
    camera.framerate = 30
    rawCapture = PiRGBArray(camera, size = (1920, 1080))

    time.sleep(1)

    for frame in camera.capture_continuous(rawCapture, format = "bgr", use_video_port = True):
        image = frame.array

        cv2.imshow("frame", image)
        key = cv2.waitKey(1) & 0xFF

        rawCapture.truncate(0)

        if key == ord('q'):
            break
```

OPENCV JAVA API – PRINCIPALES CLASES

org.opencv.core

- Mat

org.opencv.videoio

- VideoCapture

org.opencv.objdetect

- CascadeClassifier

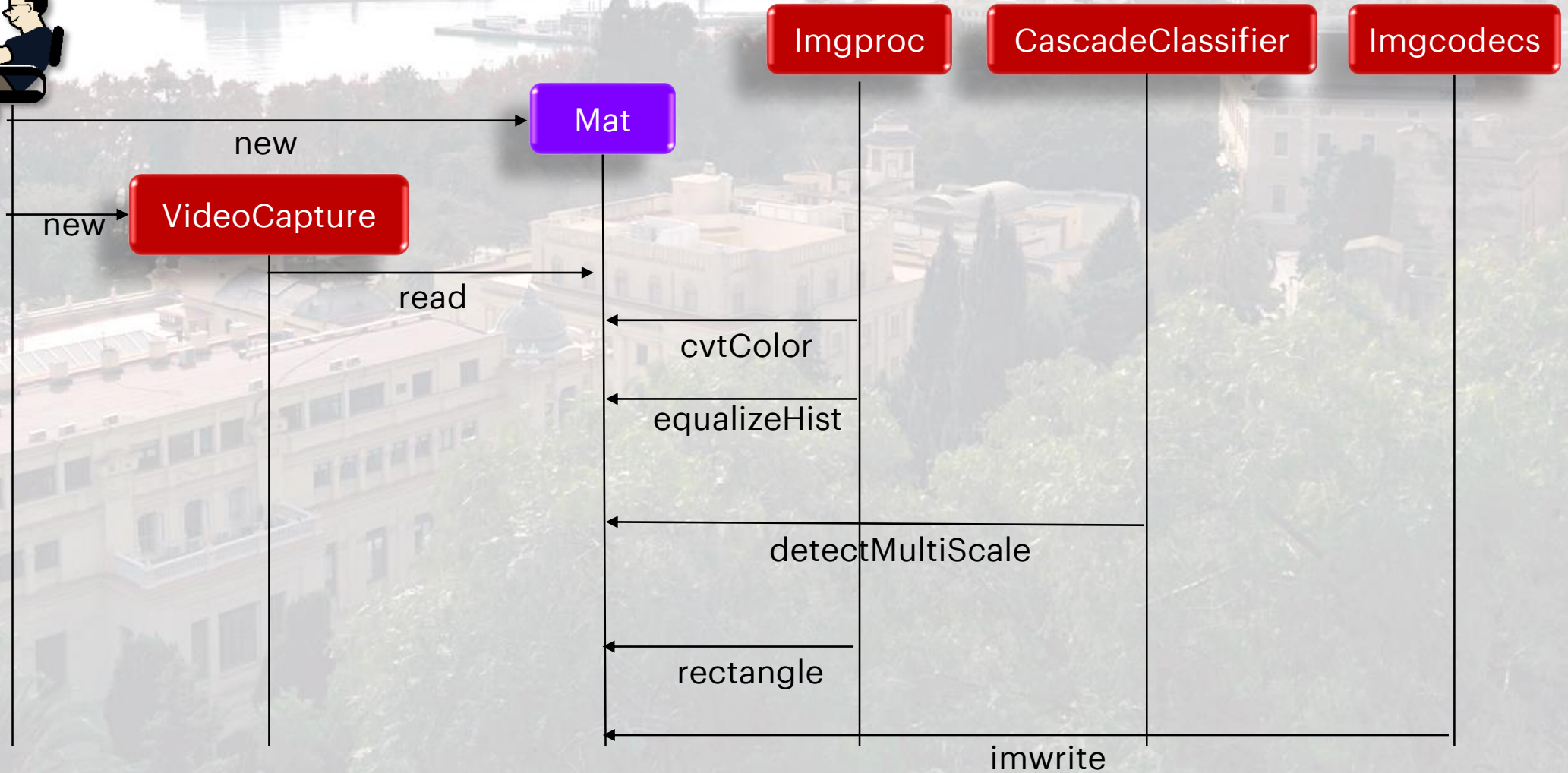
org.opencv.imgproc

- Imgproc

org.opencv.imgcodecs

- Imgcodecs

DIAGRAMA DE SECUENCIA

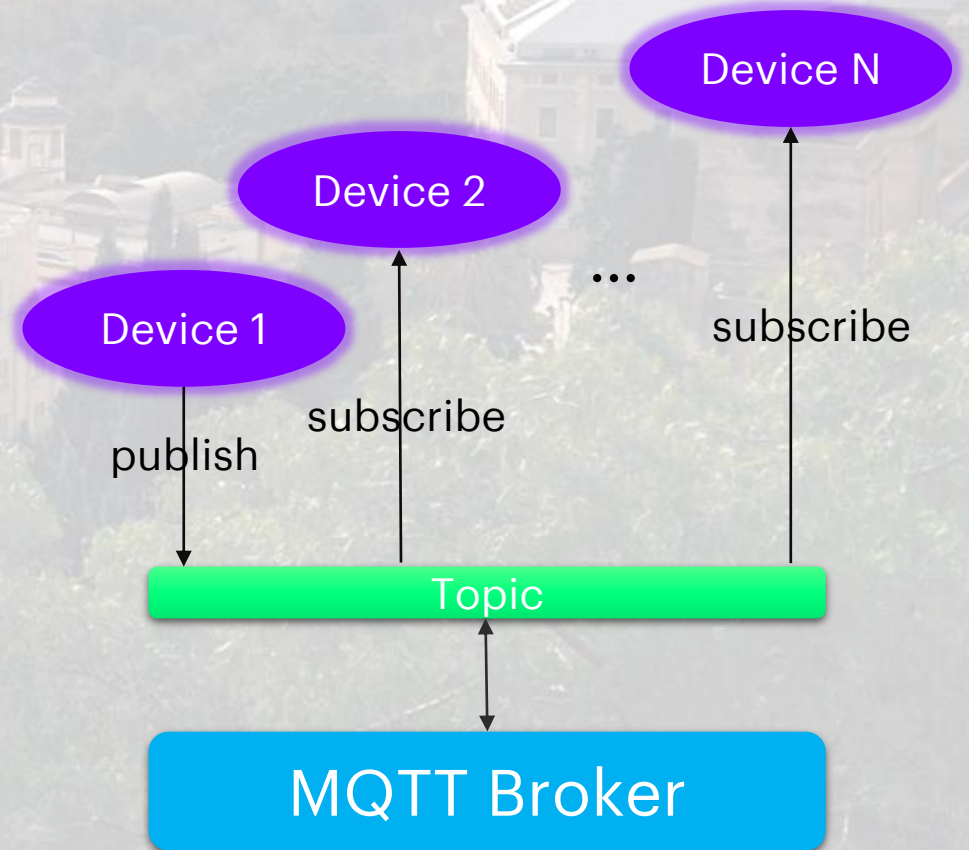


¿HECHOS UN LÍO?

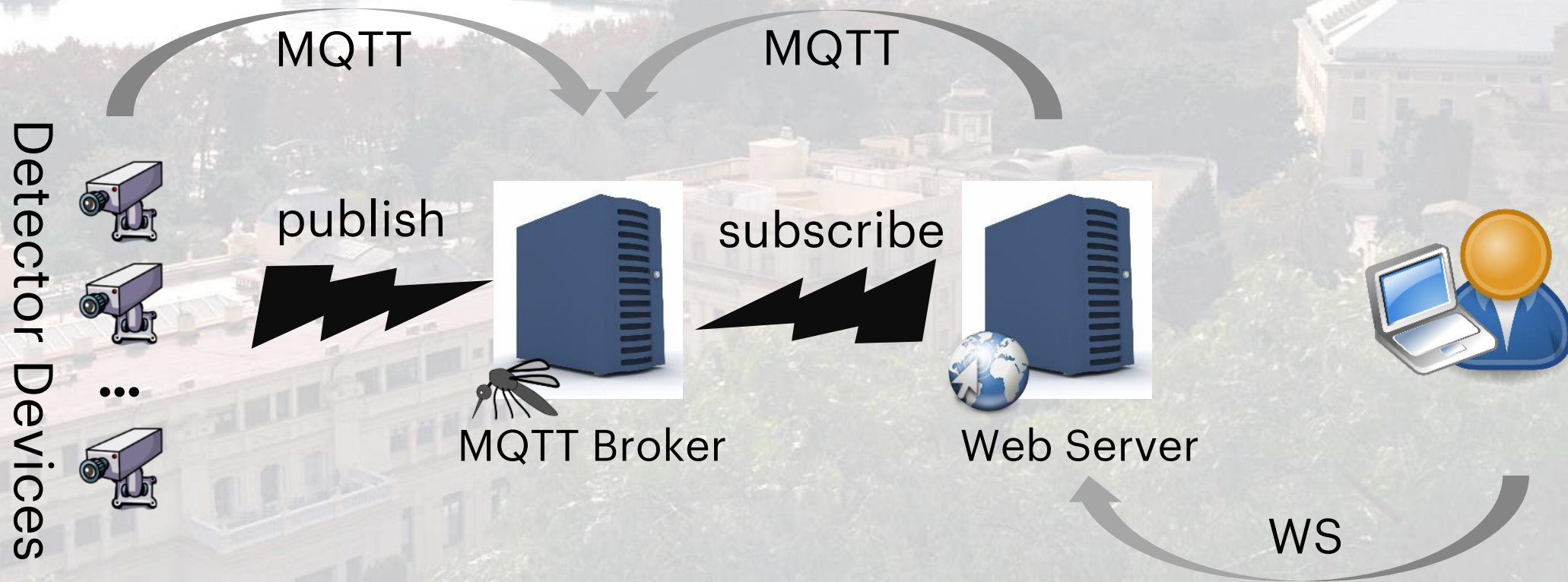


PROTOCOLO MQTT

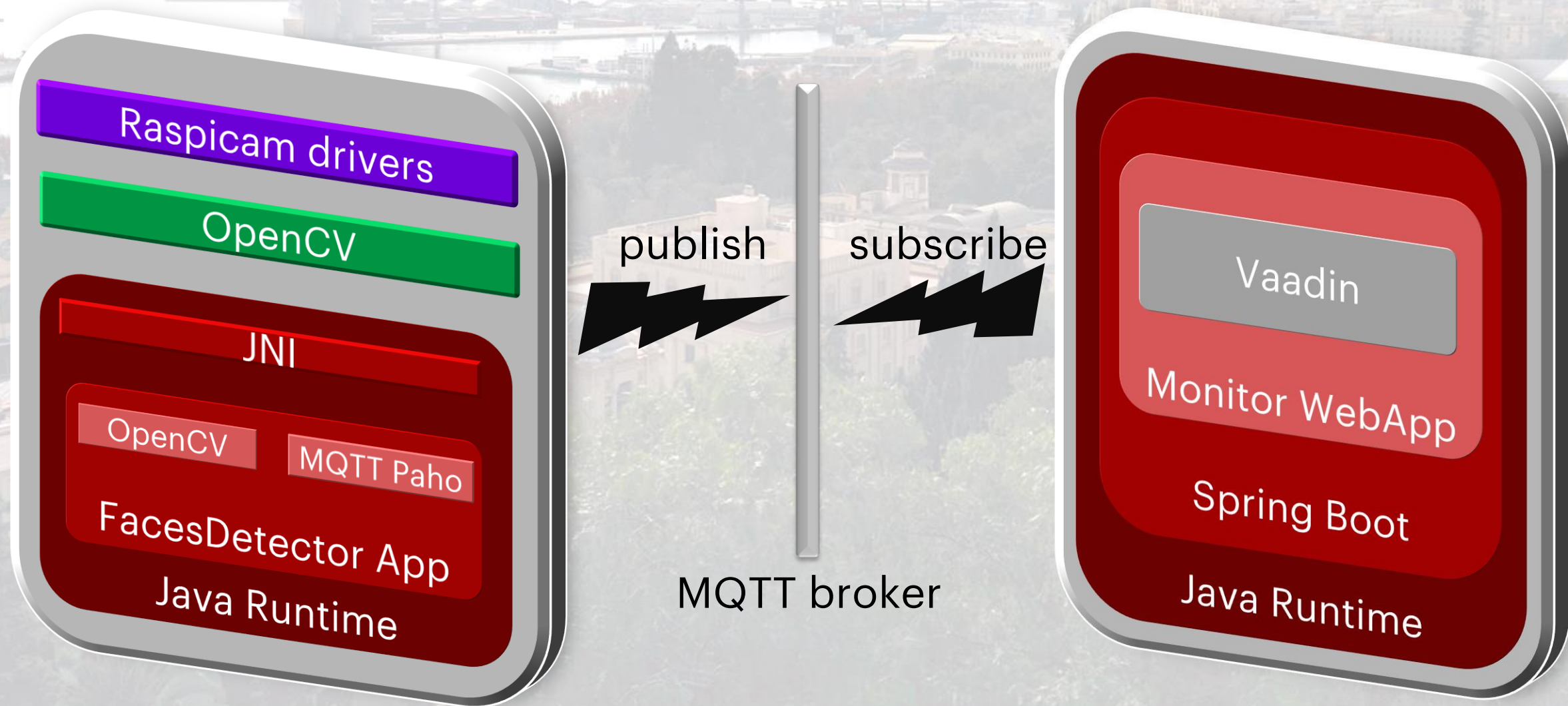
- Bajo ancho de banda
- Bajo uso de energía
- Perfecto para IoT
- Patrón Publish / Subscribe
- Puede transportarse mediante WebSocket
- Librerías – Eclipse IoT ([link](#)):
 - ✓ Cliente – Eclipse Paho: <http://www.eclipse.org/paho/>
 - ✓ Broker – Eclipse Mosquitto: <http://mosquitto.org>



ESCENARIO DE LA DEMO



ESCENARIO DE LA DEMO



DEMO TIME!



¿AUN HECHOS UN LÍO?



CÓDIGO FUENTE

(Kudos para Mariano Rodríguez @locoforf1)

Faces Detector:

<https://github.com/locoforf1/faces-detector>

WebApp Monitor:

<https://github.com/locoforf1/faces-detector-server>

Instrucciones detalladas:

<https://static.rainfocus.com/oracle/oow16/sess/14633954100810018DAr/ppt/JavaOne%20-%20FacesCounter.pptx>

OTROS SITIOS INTERESANTES PARA EMPEZAR CON OPENCV

Scala y OpenCV con Akka Streams

<https://beachape.com/blog/2016/03/08/scala-and-opencv-ep-1-akka-webcam/>

<https://beachape.com/blog/2016/03/14/scala-and-opencv-ep-2-akka-face-detector/>

Ejemplos del libro “Mastering OpenCV”

<https://github.com/MasteringOpenCV/code>

PARA LOS VALIENTES QUE QUIERAN ENTRENAR ~~POKÉMON~~ CLASIFICADORES

Documentación oficial

http://docs.opencv.org/3.2.0/dc/d88/tutorial_traincascade.html

Blog de Coding Robin

<http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html>

Gran pregunta y mejor respuesta en el foro de OpenCV

http://answers.opencv.org/question/39160/opencv_traincascade-parameters-explanation-image-sizes-etc/

¿PREGUNTAS?



An aerial photograph of a city, likely Valencia, Spain, showing a harbor with cranes, a river, and various buildings. A large, bright yellow arrow points from the left towards the right, passing through the center of the image. The text '¡GRACIAS!' is written in a bold, black, sans-serif font across the middle of the arrow.

¡GRACIAS!