ParkMP - Parking Management Platform

Juan Ignacio Tibaldi
Universidad de Palermo - Licenciatura en Tecnología de la Información

***Abstract*—En este documento se propone una solución para la gestión de un estacionamiento de vehículos, ya sea como servicio a una determinada comunidad o bien en el ámbito privado de una empresa o institución. Utilizando tecnologías de computer vision y análisis de imágenes, se establece que lugares se encuentran ocupados y cuales no, a los fines de informar a un eventual usuario de la solución donde ubicar su vehículo. Por su parte, también se aborda la utilización de protocolos de transferencia livianos en el ámbito on-premise de una compañía, y la disponibilidad de cierta información mediante un servicio cloud para su consumo masivo por aplicaciones móviles de usuarios finales. Se busca trabajar en una solución que sea adaptable y aplicable a empresas, negocios, servicios e inclusive ciudades inteligentes utilizando en algunos casos tecnologías de IoT.**

***Keywords* computer vision,análisis de imágenes,gestión de estacionamientos, power-over-ethernet, amqp, IoT, aplicaciones móviles, microcommputadoras ARM7**

I. Referencias

RabbitMQ
Librería Pika
Azure IoTHub
Device-to-Cloud: Azure IoTHub y Python
Manual de referencia OpenCV
Manual de referencia pantalla TFT for RaspberryPI
Manual de referencia RaspberryPI 3B+

II. Introducción

Un crecimiento poblacional, en ocasiones desmedido, a lo largo y ancho del planeta, impacta directamente en el incremento de medios de transporte personales como vehículos y motocicletas en las las calles de las distintas ciudaddes, esto genera un sinnúmero de problemas dentro de los cuales podemos mencionar la falta constante de sitios de estacionamiento. Ya sea en la vía pública, como en lugares específicamente diseñados para brindar este tipo de servicios, podemos observar una escasez de herramientas de gestión que faciliten el aprovechamiento de los espacios y mejoren el tiempo de espera de un usuario y por ende la calidad de su experiencia.

Si bien existen sistemas de control de estacionamiento mediante el uso de barreras y la interacción con personal específico, que gestiona la demanda y disponibilidad del servicio indicando al usuario el lugar que este debe ocupar, en este documento nos centraremos en el concepto de automatización.

En la vida diaria solemos encontrarnos con sistema electrónicos de bienvenida, e incluso cobro, de un servicio de estacionamiento, por el cual el usuario debe pasar sin realmente saber si existe o no disponibilidad de sitios una vez ingresado el vehículo, ejemplos de esto son estacionamientos de supermercados, estadios y/o lugares de eventos. A los fines de abordar una posible solución a este inconveniente se sugiere el uso de un sistema como ParkMP, el cual permite mediante el uso de una cámara infrarroja y algoritmos de deep learning para el análisis de imágenes, establecer si un lugar de estacionamiento se encuentra ocupado o disponible en un determinado momento. Esta solución hace uso de tecnologías de nube y móviles para informar al usuario final del servicio, si existe o no disponibilidad de un sitio para su vehículo antes de ingresar al estacionamiento. Por último se presta especial atención a una composición de bajo costo, utilizando dispositivos disponibles al público general.

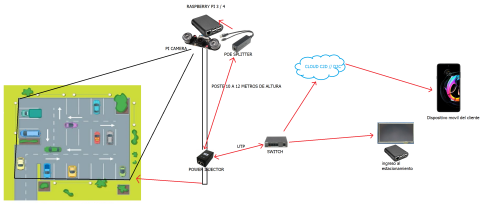


Fig. 1. Diagrama de la solución

A lo largo de esta documento se revisará la composición de la solución dentro de un marco teórico, analizando el caso de estudio y desarrollando. Por último, se concluirá con un detalle de los beneficios del uso de esta tecnología, así como también sus posibles problemáticas. Adicionalmente, se abordarán nuevas líneas de investigación que pueden ser de interés del lector en relación con la información brindada.

III. Caso de estudio

Como caso de estudio para este paper se elegió un estacionamiento perteneciente a una empresa operadora de petroleo. En yacimiento el uso de vehículos de gran porte, lugares con diferentes estacionamiento (como por ejemplo ingreso, deposito y patio), sitios de estacionamiento reservados para ambulancias, bomberos y otro tipo de maquinarias especiales, y los espacios reducidos asignados a cada uno ellos, dan como resultado un escenario ideal para la implementación de un sistema como ParkMP.

IV. Marco teórico

Comenzaremos el marco teórico mencionando los detalles de funcionamiento del sistema sugerido, sin profundizar en

la propia implementación del mismo. Dando a conocer los componentes sugeridos para una implementación y aquellos que pueden oficiar de reemplazo en casos particulares.

A. *Componentes*

En primer lugar y como se observa en el diagrama de la solución (Fig.1), el sistema esta compuesto por una microcomputadora con una cámara infrarroja conectada. Esta cámara es la responsable de tomar las imágenes a demanda (una vez que un vehículo llega al puesto de control o accede a la aplicación móvil) para actualizar el estado de los distintos estacionamientos. Dicha información es entonces analizada en el propio dispositivo (microcomputadora), y el resultado es enviado a la nube mediante un paquete json y utilizando un protocolo liviano, como por ejemplo, MQTT o AMQP (asincrónico). Por último, esta información será resguardada en una base de datos cloud de tipo documental, para luego ser accedida desde cualquier dispositivo conectado a internet (fijo o movil). La cámara mencionada en este caso particular es infrarroja, dado que, en un yacimiento, se opera durante las 24 horas del día y los vehículos se movilizan y estacionan a lo largo todo este período. A fin de ampliar el alcance de este paper, cabe aclarar que en el caso de estacionamientos de, por ejemplo, un supermercado, que se encuentra debidamente iluminado y/o opera solo durante del día, podría utilizarse una cámara convencional. Adicionalmente, es importante destacar que este combo microcomputadora+cámara ofician de estación remota y pueden ser replicados varias veces dentro de una misma solución dependiendo de la cantidad de estacionamientos separados que el sitio en estudio tenga.

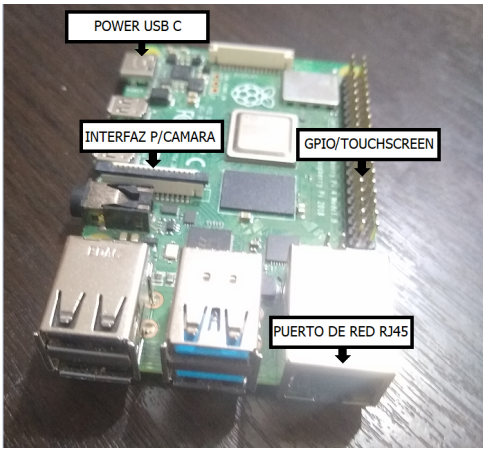


Fig. 2. Raspberry PI 4

Otros componente a los que haremos referencia son aquellos que energizan el combo microcomputadora+camaraIR, considerando que el mismo debe encontrarse en una posición de altura, desde donde pueda tomar una imágenen de todo el estacionamiento. Para ello utilizaremos puertos de red POE (Power-over-ethernet) o bien un Power Injector conectado a la red eléctrica del edificio principal, en el caso de no contar con un dispositivo de red capaz de brindar este servicio. De esta forma, desde la conexión de red, también del edificio principal, se podrá llevar un cable UTP (bajo costo), colocando

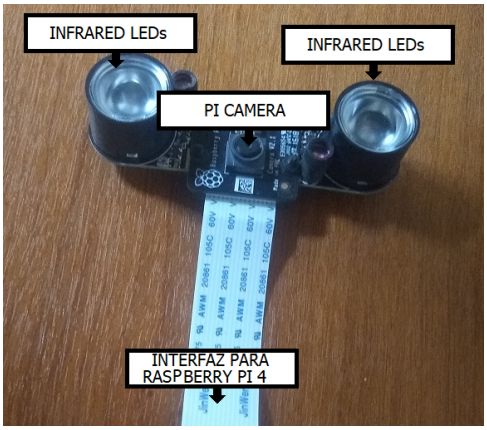


Fig. 3. Pi Camara Infrarroja con leds

un splitter cerca de la microcomputadora a fin de energizarla. Una vez energizada la microcomputadora la camara, como dispositivo perisférico de la misma, queda automáticamente energizada. Es válido recordar que el sistema sugerido esta preparado para trabajar con varios puntos remotos, uno por cada estacionamiento gestionado. Por último, con estos componentes, además de energizar la microcomputadora, se le da a dicho combo conexión de red que, una vez conectada a Internet, utilizará para enviar los paquete JSON con el resultado del análisis de la útlima imágen adquirida por la cámara.

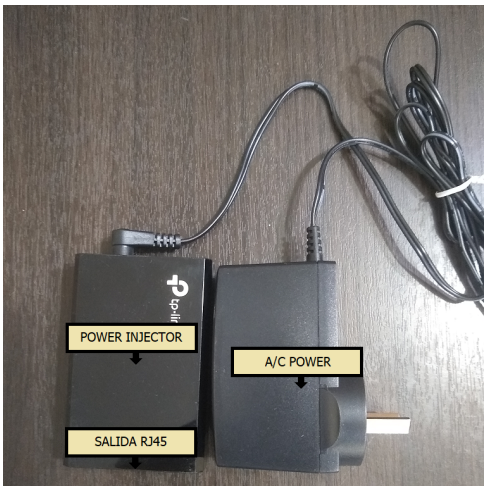


Fig. 4. Power Injector

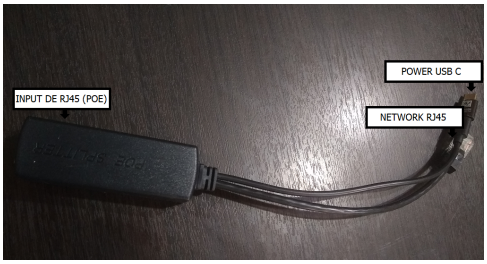


Fig. 5. POE Splitter

Por último, se mencionarán los componentes que se utilizarán para el acceso de los usuarios al sistema ParkMP. En el caso de una estación física, se requerirá de otra microcomputadora esta vez con una pantalla asociada, ya sea por sus entradas MicroHDMI y por su interfaz de entrada/salida de propositos generales (GPIO) y de un sistema energización igual al expresado anteriormente. En este caso, el sistema se conectará con la base de datos documental cloud para obtener los últimos registros de utilización del estacionamiento, a los fines de observar la disponibilidad de lugares. Es importante destacar que, al igual que en el caso de las estaciones de toma de imágenes remotas, pueden existir varios ingresos a un determinado estacionamiento, y por ende, varios estaciones físicas de usuarios. Otro de los componentes que se puede adicionar a esta solución es un chequeo de estacionamiento a través de la utilización de un dispositivo móvil, aunque esto último no sea aplicable al caso de estudio elegido.

V. Desarrollo

A. Detalle de requerimientos

A modo de delinear una arquitectura de alto nivel de los componentes de la solución propuesta, es necesario detallar los requerimientos que la misma debe cumplir, por lo tanto, la solución:

- Deberá priorizar el bajo costo de su implementación.
- Deberá ser construida de forma que sea fácilmente desplegable en sitios remotos.
- Deberá permitir la alimentación eléctrica de su componente de toma de imágenes a través de un cable UTP utilizando Power-over-Ethernet (POE) a fin de facilitar su despliegue.
- Deberá analizar la imagen en el dispositivo que lleva el periférico de toma de imágenes, para luego enviarlas en broker MQTT/AMQP on-premise y a un componente AMQT Cloud provisto por el proveedor del servicio SaaS.
- Deberá contar con una aplicación móvil descargable que permita al usuario final detectar disponibilidad de estacionamiento desde su propio dispositivo.
- Deberá contar con terminales fijas instaladas en las entradas de un estacionamiento en caso de que el usuario no cuente con un dispositivo móvil o este no cuenta con conexión a la red.
- Deberá permitir la utilización del sistema enteramente sin terminales para los casos en los que se desee acceder únicamente desde dispositivos móviles.
- Deberá contar con conexión privada LAN para las terminales y el broker AMQP on-premise y una contraparte cloud para la exposición segura de la información a Internet.
- Deberá enviar la respuesta al usuario final en un tiempo menor a un minuto de espera.
- Deberá contar tanto con dispositivos remotos como con terminales físicas robustas que puedan mantener un uptime estimado de 95
- Considerando que sus componentes pueden ser instalados a la intemperie, deberá contar tanto con dispositivos remotos como con terminales físicas robustas preparadas para inclemencias climáticas.
- Deberá alimentar eléctricamente estas terminales fijas mediante el uso de Power-over-Ethernet a los fines de favorecer el despliegue de las mismas.
- Deberá considerar la toma de imágenes solo de día o en ambientes bien iluminados.
- Deberá contar con una interfaz web de configuración en todos los dispositivos remotos, considerando que los mismo se encontrarán en lugares de difícil acceso.
- Dentro de la configuración deberá permitir marcar lugares como especiales por el administrador de la aplicación y desplegados utilizando el servidor de configuración como, por ejemplo, lugares reservados para discapacitados, bomberos, ambulancias, estacionamiento de acuerdo con el puesto que ocupan en una empresa, etc.
- Dentro del despliegue de la imagen del sistema operativo de un nuevo dispositivo de toma de imágenes, se deberán crear dos perfiles de usuario, uno que permita el acceso a operar el dispositivo (servicios, fecha y hora, salud en general) y otro que permita el despliegue de una configuración remota // usuario de la interfaz web.
- Deberá esperar a la solicitud del usuario final, en el caso de dispositivos móviles a fin de no actualizar constante la aplicación, esto no aplica para terminales donde el refresco es automático.
- En su aplicación móvil deberá permitir cargar más de un estacionamiento, dichos estacionamientos serán elegidos por el usuario antes de solicitar la información.
- El ID del estacionamiento elegido en la aplicación móvil deberá ser enviado la solicitud de información a la aplicación cloud correspondiente (tópico AMQP).
- La parte de la solución cloud se licenciará en un tenant del proveedor y se venderá mediante la modalidad SaaS.

B. Arquitectura de alto nivel

Tomando en cuenta los requisitos expresados en el punto anterior podemos delinear la siguiente arquitectura de alto nivel, donde se muestran los componentes requeridos para cumplir con la demanda.

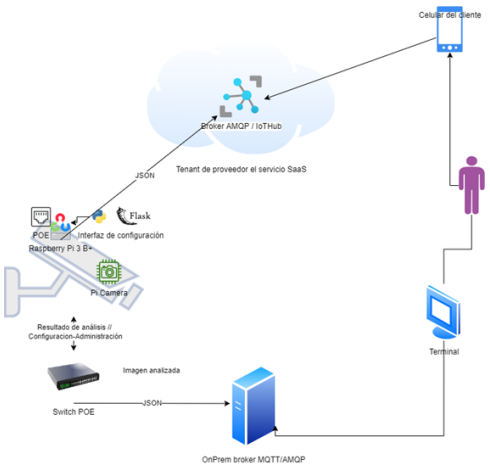


Fig. 6. Arquitectura global de la solución

C. Detalle de componentes

El sistema ParkMP, como bien se detalla en el marco teórico, este compuesto por distintos dispositivos que convergen en una base de datos documental cloud. A los fines de desarrollar una explicación acorde a distintos niveles de lectura, clasificaremos estos dispositivos las siguientes categorías:

1) Cámara NoIR 2Mpx con led infrarrojo (lectura):

a) Instalación física (conexión): Este dispositivo se encuentra conectado a la microcomputadora de análisis (ver a continuación), y se utiliza para la toma de la imagen desde un ángulo superior del estacionamiento (ej: torre o poste). La conexión se realiza, como podemos ver en la imagen, a través del puerto J3 siempre tomando en cuenta que el lado de color azul debe ir mirando hacia la entrada de audio.



Fig. 7. Microcomputadora Raspberry PI 3B+

b) Instalación lógica (interfaz): Una vez conectada la camera físicamente debemos iniciar la microcomputadora y configurar la interfaz para que esta pueda hacer uso del mencionado dispositivo. Para esto, iniciamos la herramienta de configuración embebida y elegimos las siguientes opciones. Inmediatamente después de esto podemos observar como la

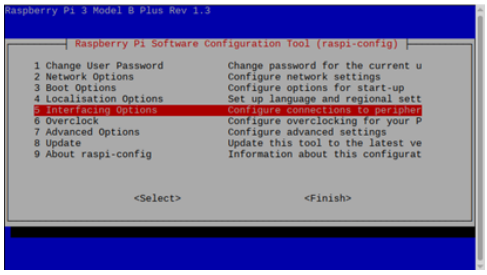


Fig. 8. Interfacing Options

herramienta de configuración nos indica que la interfaz ha sido habilitada y nos invita a reiniciar el sistema.

c) Prueba de conexión e interfaz (desde línea de comandos): Por último, podemos probar que la conexión con el dispositivo y la subsecuente configuración del mismo haya tenido el efecto deseado mediante la ejecución del siguiente comando.

```
raspistill -o home/pi/image.jpg
```

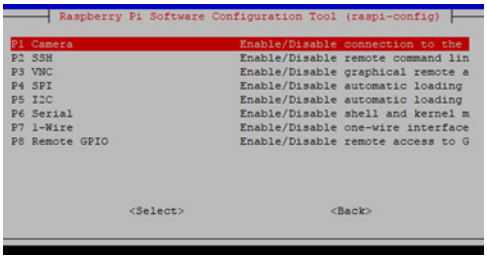


Fig. 9. P1 Camera

d) Prueba de conexión e interfaz (desde script python): Además, también es posible ejecutar esta función desde un script de Python utilizando las librerías provistas por este lenguaje, el mismo que utilizaremos para el análisis de las imágenes obtenidas.

```
from time import sleep
from picamera import PiCamera

camera = PiCamera()
camera.resolution(1024,768)
camera.start_preview()
sleep(2)
camera.capture('image.jpg')
```

Una vez que abordemos la herramienta OpenCV, la cual usaremos para el análisis de las imágenes, veremos la importancia de usar escala de grises por sobre colores RGB a los fines de tomar y analizarlas las imágenes de forma rápida y eficiente.

e) Instalación de librerías Python para el uso de la cámara: A fin de hacer uso de este dispositivo desde un código Python, como vimos en el punto anterior, es necesario preparar el ambiente de ejecución. Para ello debemos instalar la siguiente librería utilizando, en este caso, el comando Python pip o pip3 según corresponda:

```
python -m pip install picamera
```

f) Led infrarrojo: Por último, se recomienda para el correcto funcionamiento del sistema colocar un conjunto de leds infrarrojos a los costados de la cámara. A pesar de que la mayor parte del uso del sistema será diurno pueden llegar a ser útiles ante inclemencias climáticas y/o anocheceres tempranos.

2) Raspberry PI 3 B+ o superior (de análisis): Una vez tomada la imagen, la misma deberá pasar por una instancia de OpenCV para reconocer los vehículos estacionados, luego dicha información será guardada en una matriz de estacionamientos ocupados/disponibles y enviada, mediante un archivo JSON, al broker MQTT y al componente cloud IoTHub para su envío al dispositivo final (ya sea terminal fija o móvil del usuario final). En el caso del dispositivo móvil, si bien se detallará interfaz mas adelante, el mensaje quedará publicado para su consumo cuando el portador así lo desee.

a) Librerías necesarias: Para realizar estos envíos utilizaremos las librerías Pika/Puka o Azure IoTHub según corresponda.

Para on-prem:



Fig. 10. Leds infrarrojo



Fig. 11. Leds infrarrojo - vista trasera

```
pip install pika

import pika
```

Para cloud:

```
pip install azure-iot-hub

from azure.iot.hub import
IoTHubRegistryManager
help(IoTHubRegistryManager)
```

b) POE Splitter: Un detalle sumamente importante del uso de una microcomputadora ARM7 del tipo de una Raspberry PI 3A, 3B o 3B+ (recomendada), es que permite la utilización de un POE Splitter para alimentar el dispositivo y conectarlo a la red LAN.

c) Interfaz Web de configuración: En esta interfaz de configuración el sistema permite detallar la cantidad de "espacios" de estacionamiento que tendrá la matriz y su forma mas representativa, para luego enseñar su estado a quien lo requiera. Para realizar esto el sistema cuenta con un usuario específico del cual se deberá registrar toda actividad en un archivo de log local. Un ejemplo de interfaz es la siguiente:
Se ha diseñado una interfaz minimalista que no distraigan al usuario de la tarea principal de este componente, crear/editar



Fig. 12. Interfaz de configuración web (1)

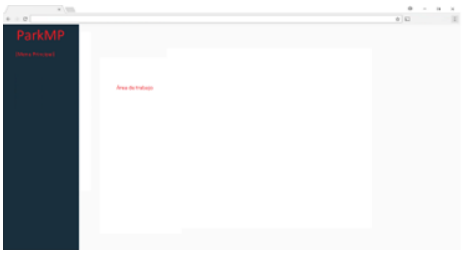


Fig. 13. Interfaz de configuración web (2)

el archivo de configuración que luego será utilizado para el funcionamiento del sistema. De acuerdo a fines prácticos de optimización del hardware, este componente se encuentra desarrollado en Flask/Python.

3) OpenCV: Para el análisis de las imágenes adquiridas por el dispositivo detallado anteriormente utilizaremos la biblioteca de visión artificial de código libre llamada OpenCV. Originalmente desarrollada por Intel, su significado etimológico viene del acrenimo Open Computer Vision y desde sus inicios (versión alfa) en el mes de enero de 1999, se ha utilizado, y es utilizada aún, en una gran cantidad de aplicaciones, algunas de sus funciones más importantes son:

- Detección de movimiento
- Reconocimiento de objetos
- Reconstrucción 3D a partir de imágenes

centrándonos en sus capacidades de reconocimiento de objectos, utilizaremos esta librería para determinar si existe o no un vehículo en la posición (región) x,y asociada a una celda de la matriz cargada por el cliente/estacionamiento. Esta matriz será autogenerada por la aplicación a partir de datos solicitados al cliente sobre una imagen del estacionamiento previamente cargada.

4) Broker AMQP (On-prem): Para esta solución se ha optado por un broker AMQP on-premise llamado RabbitMQ tanto por funcionalidad como por afinidad con la herramienta por parte del proveedor del servicio. La funcionalidad principal buscada en esta herramienta es la cola asincrónica de mensajes que permite asegurar la recepción de la información correcta en caso de corte menores de enlace de comunicaciones.
Además, es un sistema liviano que puede ser alojado en un contenedor de Docker y desplegado sobre esa solución. Este sistema cuenta con numerosas ventajas por sobre sus competidores, por ejemplo:

- Confiabilidad:
  Este sistema proporciona almacenamiento de mensajes en caso de que no existan consumidores disponibles, es

decir, brinda la posibilidad de que el consumidor acepte la entrega del mensaje para asegurarse de que lo procesó correctamente y, en caso de que haya fallado su procesamiento, permite que el mensaje se pueda reencolar para ser consumido por una instancia diferente del consumidor o que sea procesado de nuevo por el mismo consumidor que inicialmente falló.

- Creación de clusters:
  Si bien RabbitMQ proporciona gran rendimiento procesando miles de mensajes por segundo, en ocasiones debe ser capaz de procesar una mayor cantidad de mensajes sin impactar el rendimiento de las a aplicaciones. Para esto RabbitMQ permite la creación de clústeres para escalar horizontalmente la solución.
- Colas con High Availability
  Este crecimiento horizontal, de ser necesario, permite "repartir" el contenido de las colas de mensajes en distintos nodos del cluster.
- Permite la escalabilidad de las aplicaciones:
  Cuando un consumidor se subscribe a una cola, y existen mensajes para este consumidor que superan la capacidad que el mismo tiene para procesarlos, es posible crear nuevas instancias de ese consumidor para hacer frente al mencionado flujo de mensajes.
- Soporte a múltiples protocolos
  Además de AMQP, este sistema soporta STOMP, MQTT y HTTP a través de la utilización de plugins.
- Mecanismos de autenticación
  Este sistema de suscripción de mensajes incorpora mecanismos de autenticación y control de acceso a cada uno de sus componentes.
- Soporte de lenguajes Por último, el sistema elegido soporta una gran cantidad de lenguajes de programación con los que es posible construir productores y consumidores de mensajes. Entre estos tenemos Java, Scala, PHP, Python, Ruby, entre otros.

5) Broker IoTHub (Microsoft Azure): Como contraparte del punto anterior, y al momento de enviar el mensaje a la nube a fin de disponibilizarlo para el consumo de dispositivos móviles, se utilizará el componente nativo PaaS de Microsoft Azure llamado IoTHub.
Azure IoT Hub es un servicio administrado, hospedado en la nube, que actúa como centro de mensajes para la comunicación entre una aplicación de IoT y los dispositivos conectados. Puede conectar millones de dispositivos y sus soluciones de back-end con confianza y de forma segura. Al comunicarse utilizando un protocolo liviano como MQTT/AMQP este sistema (o sus distintos terminales) pueden ser vistos como dispositivos IoT, principalmente su apartado cloud. En este caso no se hará uso de otros componentes de proveedor de nube que permitirían trabajar con estos mensajes o resguardarlos, pero siempre se podría ampliar el concepto para realizar tareas de DataScience y monitoreo a través de Dashboards (ej: Grafana).

6) Terminal / Celular del usuario final: Como últimos componentes de la arquitectura diseñada para responder a los requerimientos establecidos anteriormente, están tanto las terminales fijas como los celulares del usuario final. En el caso de las terminales también compuestas por una microcomputadora del tipo Raspberry Pi que pueda suscribirse al bróker MQTT con su ID de terminal y tomar los últimos estados de un determinado estacionamiento y mostrarlo en pantalla, indicándole de esta forma al usuario final los lugares disponibles que tiene. De acuerdo a los requerimientos este dispositivo también debe contar con la posibilidad de ser alimentado eléctricamente utilizando Power-over-ethernet.

En el caso de los dispositivos móviles se contará con una aplicación descargable de los principales markets (Ej: AppStore, Google Play) donde se podrán cargar los distintos estacionamientos, ya sea con perfil de cliente/estacionamiento o perfil de usuario final, para su gestión. El usuario final podrá escanear un QR en un determinado estacionamiento a fin de agregarlo a su aplicación y así gestionarlo de ahí en adelante, por su parte, el cliente/estacionamiento podrá cargar su matriz dentro de la aplicación creando un ID único para su servicio. En todos los casos, se busca un interfaz de usuario sencilla y amigable que permita el uso simple de la solución prestada.



Fig. 14. Microcomputadora para terminal fija



Fig. 15. Pantalla TFT para terminal fija

7) Funcionamiento esperado del sistema: A fin de resumir cual será el funcionamiento del sistema propuesto, se realizaron varios casos de uso típicos utilizando UML. En estos casos de uso se puede evidenciar como se espera que el sistema responda en cada caso.

a) Caso de uso n°1: En el primer caso de uso podemos observar un usuario (actor) que revisa la aplicación del estacionamiento X, para saber si existe disponibilidad antes

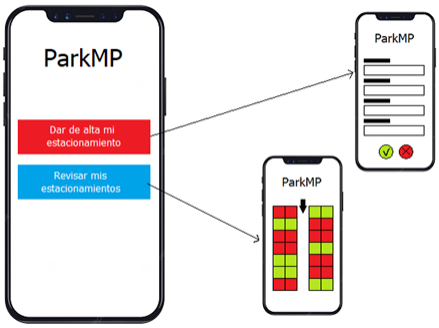


Fig. 16. Interfaz de aplicación móvil

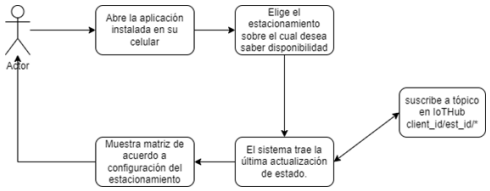


Fig. 17. Caso de uso n°1

de ingresar al mismo. El caso finaliza cuando la app entrega un diagrama representativo del estacionamiento (matriz) con los lugares ocupados/libres.

b) Caso de uso n°2: En el segundo caso de uso se observará el comportamiento del sistema cuando un usuario ingresa por una terminal fija. En este caso observamos además

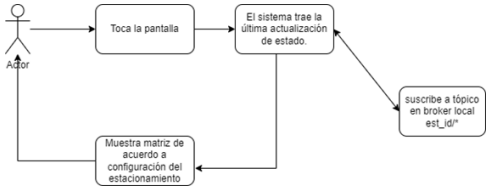


Fig. 18. Caso de uso n°2

como el sistema se comporta a nivel suscripción al broker AMQP, ya que no es necesario especificar de que cliente o a que estacionamiento se hace referencia.

c) Caso de uso n°3: En el siguiente caso de uso podemos observar lo que ocurre de manera casi constante en el sistema, es decir, cada X segundos se toma una fotografía del estacionamiento, o parte del estacionamiento que se este observando con ese componente remoto en particular, y se analiza utilizando OpenCV. Una vez reconocidos los objectos "vehículo" se compara con la matriz cargada por el cliente de la solución (estacionamiento) y se marca cada sitio como ocupado/libre. Esta última actualización se publica a los suscriptores para su consumo.

d) Caso de uso n°4: Una vez abordados los casos de uso correspondientes al uso normal/diario de la solución, es momento de atender a aquello casos especiales como por ejemplo la carga de un nuevo estacionamiento.

En este caso vemos como la carga manual del

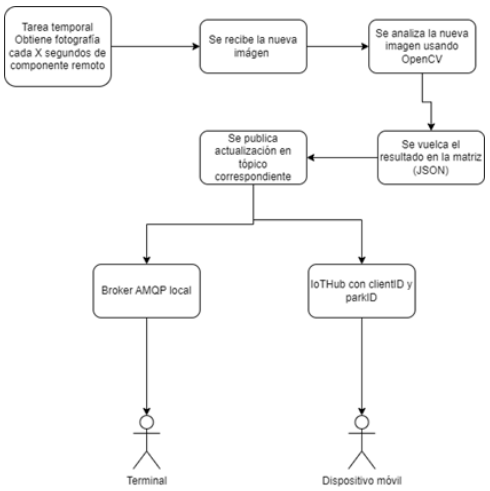


Fig. 19. Caso de uso n°3

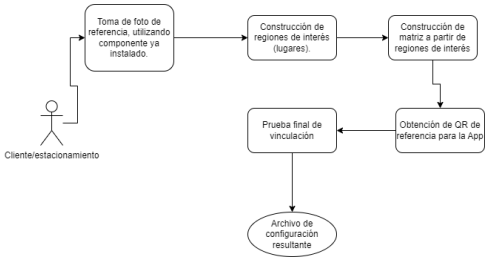


Fig. 20. Caso de uso n°4

estacionamiento, si bien es la parte más tediosa de la solución, conlleva una gran importancia y requerirá de acompañamiento por parte del proveedor de la solución.

e) Caso de uso n°5: Otro caso de uso especial es el alta de un estacionamiento nuevo en la aplicación móvil del usuario final de la solución.

En este caso de uso podemos ver como el usuario escanea

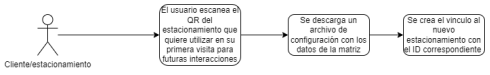


Fig. 21. Caso de uso n°5

el código QR del estacionamiento a fin de descargar todos los archivos de configuración requeridos para realizar la interpretación del payload de AMQP y plasmarlo en la matriz que luego se le mostrará a dicho usuario.

D. Recomendaciones

A modo de completar este apartado se listarán ciertas recomendaciones derivadas de la experiencia en implementaciones similares, así como también de la construcción de POCs sobre lo expresado anteriormente.

- A pesar de que se establece en los requisitos de que la franja horaria de utilización de ParkMP será de día o en ambientes bien iluminados, se recomienda el uso de una

cámara NoIR con lámparas IR leds para casos puntuales donde las inclemencias climáticas o eléctricas impidan el normal uso del sistema, se deja esto como recomendación ya que el cliente del sistema puede optar por interrumpir el servicio en estos casos.
- Se recomienda la alimentación eléctrica mediante Power-over-Ethernet para reducción de costos tanto de instalación como de mantenimiento.
- Se recomienda la utilización de ajustes robustos para las cámaras a los fines de evitar movimientos involuntarios provocados por viento.
- Se recomienda mantener un resguardo de los archivos de configuración de cada una de los componentes remotos identificados con ID propio.
- Se recomienda la habilitación del protocolo SSH para la administración de los componentes remotos.
- Se recomienda la habilitación del protocolo SSH para las tareas de resguardo de archivos de configuración.
- Se recomienda especial atención en el mantenimiento del componente on-premise para resguardo temporal de imágenes. Se recomienda el uso de scripts de rotación que borren imágenes históricas cada X tiempo. El componente cloud será de mantenimiento por el proveedor del servicio SaaS.
- En caso de estacionamientos de empresas, se recomienda la ubicación de las terminales a dos alturas o en postes de altura variable, para que puedan utilizas por vehículos de diferente porte
- En el caso de requerir dashboard de monitoreo y/o estadísticos se recomienda la inclusión, dentro del script (de código abierto) que se encuentra en los componentes remotos, de un suscriptor MQTT que pueda ser vinculado con un broker en el proveedor de nube que el cliente desee.

VI. Conclusión

En este trabajo se intenta aborda una solución de bajo costo que, mediante el uso de software libre y hardware de commodity, permita optimizar los tiempos de espera y por ende, en muchos casos, las ganancias de un servicio de estacionamiento. En los casos en los que la solución aplica a estacionamientos de empresas o locales comerciales, se busca optimizar la disponibilidad y minimizar la tiempo de busqueda de los vehículos dentro del propio sitio.
Si bien se trata de una solución ampliamente mejorable y escalable de acuerdo a las necesidades puntuales de cada cliente, el servicio básico se encuentra cubierto y haciendo enfasís en la facilidad de instalación y configuración.

VII. Líneas futuras de investigación

En cuanto a líneas futuras de investigación se observan:

- Utilizar DataScience para determinar mejoras en el servicio prestado, por ejemplo, detectando posibles ventas de servicio de estacionamiento mensualizado.
- Implementar un modo de configuración remota de cada componente de toma de imágenes centralizado en una sola interfaz web.
- Bajar aún mas los costos utilizando microcontroladores como el Nodemcu esp8266 para la toma de imágenes centralizando el análisis en equipos on-prem/cloud según corresponda.
- Implementar el análisis de imágenes en la nube, elimnando el procesamiento y envío de mensajes on-prem, convirtiendo, de esta manera, toda la solución en una solución cloud-native de IoT.
- Implementar frameworks de desarrollo html-mobile como pueden ser Flutter o Ionic para favorecer la actualización de la solución.

VIII. CV

JUAN IGNACIO TIBALDI
UNIVERSIDAD DE PALERMO - CICLO 2022

Email: jtibal@palermo.edu
Teléfono: 2974319117
Dirección: Tomas Turkovic 1235 - Rada Tilly (Chubut)]