Comentario técnico: CTC-101
Componente: **RPC de Mongoose-OS en GCP con ESP32**
Autor: Sergio R. Caprile, Senior R&D Engineer

Revisiones	Fecha	Comentarios
0	30/06/20	

En el CTC-099 analizamos la Google Cloud Platform (GCP) y desarrollamos la utilización de Cloud IoT Core. En el CTC-100 nos conectamos utilizando MQTT a dicha plataforma IoT con un ESP32 y Mongoose-OS. Este entorno incorpora una API del tipo RPC (Remote Procedural Call) que podemos utilizar a través de MQTT, como analizamos en el CTC-096. Analizaremos en esta oportunidad la utilización de esta API RPC para control y monitoreo de dispositivos conectados a la Google Cloud Platform.

Breve descripción del funcionamiento

La operación de RPC se realiza sobre el mismo esquema de conexión que analizamos en el CTC-100, es decir, sobre el bridge MQTT de Cloud IoT Core. Como tal, está limitada en principio a una operación muy similar a la descripta en el CTC-096. Dadas las limitaciones adicionales que impone la plataforma, no es posible utilizar cualquier tópico, sino que la operación se realiza mediante los tópicos de eventos y comandos, utilizando subtópicos.

- El dispositivo escucha mensajes que transportan pedidos de RPC dentro de un subtópico (*subfolder*) específico de *commands* (por defecto '*rpc*').
  - Dentro de cada mensaje, un parámetro especifica el subtópico de *events* donde el dispositivo publicará la respuesta, de modo que el interesado pueda obtenerla.
- El controlador se subscribe a un tópico de Cloud Pub/Sub, el cual debe configurarse para re-publicar allí los mensajes que hayan sido publicados en el subtópico (*subfolder*) específico de *events* (por defecto '*rpc*') que se utilice.

Adicionalmente, dado que por la forma de operación en general existe una cierta demora entre la publicación de un mensaje en el broker y la re-publicación de éste en Cloud Pub/Sub, el sistema tiene una latencia que desalienta la utilización con finalidad interactiva.

Configuración

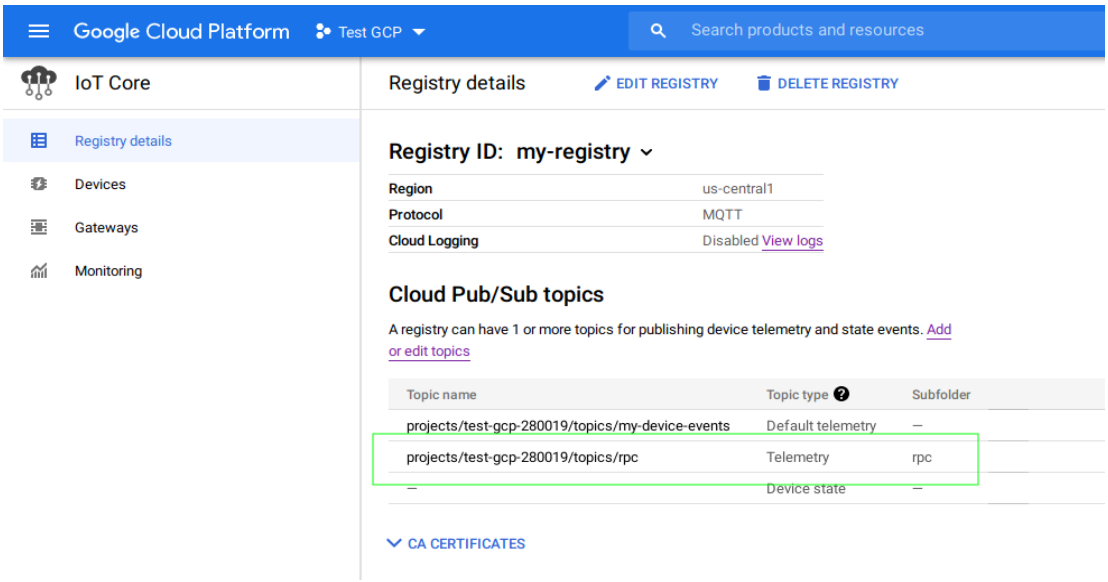
Deberemos primero modificar la configuración de Cloud IoT Core antes de operar sobre el dispositivo.

Configuración de Cloud IoT Core

Debemos modificar la *device registry* agregando un nuevo tópico para que Cloud Pub/Sub replique allí los mensajes que llegan por el subtópico utilizado para RPC. Si no lo hacemos, los mensajes serán publicados en el mismo tópico que los demás (telemetría, por ejemplo), sólo que se indicará el *subfolder* correspondiente entre los atributos. Si bien esto puede procesarse en nuestra aplicación de control, la herramienta *mos tool* lo requiere. Si fallamos en configurarlo, nos encontraremos con este mensaje:

```
$ mos --port gcp://test-gcp-280019/us-central1/my-registry/esp32_807A98 call Sys.GetInfo
failed to subscribe to response topic: topic "rpc" does not exist and --gcp-rpc-create-topic is
not set
$
```

Si bien la herramienta muy gentilmente nos ofrece ocuparse de todo por nosotros, preferimos ingresar manualmente y configurar el tópico *projects/<nombre de nuestro proyecto>/topics/rpc* de Cloud Pub/Sub para replicar lo que ingresa por el subtópico (*subfolder*) *rpc* de telemetría, como podemos observar:



Configuración del dispositivo

A lo analizado en el CTC-100 agregamos:

```
libs:
  - origin: https://github.com/mongoose-os-libs/rpc-gcp     # provee soporte RPC sobre GCP
```

Adicionalmente podemos modificar el subtópico en el que el dispositivo recibe los pedidos de RPC, configurando `rpc.gcp.subfolder` .

Operación

Recordemos que además de lo anterior, debemos configurar las credenciales para conectarnos por WiFi a nuestra red (SSID y clave). Luego de compilado el código (`mos build`) y grabado el microcontrolador (`mos flash`) mediante *mos tool*, observamos el log hasta observar que el dispositivo se ha conectado.
Como bien pudimos adivinar al observar la configuración, para utilizar *mos tool* le especificamos los parámetros necesarios para la operación dentro de un pseudo URL en la opción `--port` :

```
$ mos --port gcp://PROJECT/REGION/REGISTRY/DISPOSITIVO
```

donde PROJECT, REGION y REGISTRY corresponden a los datos que ingresamos al hacer el *quickstart* de Cloud IoT Core; mientras que DISPOSITIVO es el id de nuestro dispositivo, como lo ingresamos en el CTC-100.
Por ejemplo, pedimos el estado del dispositivo esp32_807A98 mediante `Sys.GetInfo`:

```
$ mos --port gcp://test-gcp-280019/us-central1/my-registry/esp32_807A98 call Sys.GetInfo
```

La operatoria fuera de *mos tool*, es decir, escribir nuestra propia aplicación, es muy similar a lo descripto en el CTC-096, sólo que en vez de utilizar MQTT lo haremos a través de la API de Cloud Pub/Sub.

Un detalle importante es que todas las respuestas de todos los dispositivos siendo interrogados arribarán en el mismo tópico. Si necesitamos operar sobre muchos dispositivos simultáneamente tal vez sea preferible configurar varios tópicos de Cloud Pub/Sub relacionados con sus respectivos subfolders. En el caso de *mos tool*, indicamos estas modificaciones como datos dentro del pseudo URL:[1]

```
$ mos --port gcp://PROJECT/REGION/REGISTRY/DISPOSITIVO?sub=topico&respsf=subfolder
```

Finalmente, aclaramos que la extensión a la library desarrollada en Cika y propuesta en el CTC-100 se encarga de filtrar los mensajes de RPC de modo que no disparen el handler del usuario. De no utilizarse dicha library sino la original del fabricante, al momento de escribir este texto el handler que se registre para recibir comandos recibirá también los mensajes de RPC (pues se envían en un subfolder). Para mayor información, remitimos al repositorio de la library en github.[2]

1 https://mongoose-os.com/docs/mongoose-os/api/misc/rpc-gcp.md
2 https://github.com/CikaElectronica/gcp2/