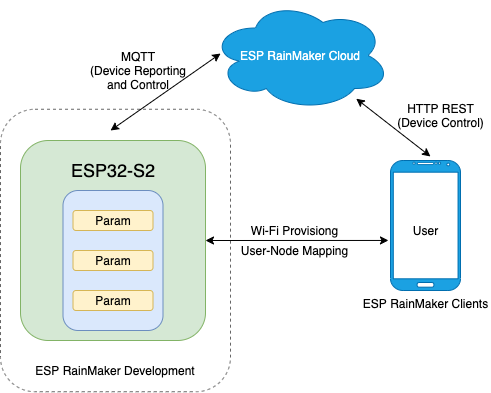
**Guía y conceptos básicos de la ESP32/ESP Rainmaker traducidos**



Como empezar:

# Introducción:

El proyecto ESP RAINMAKER debería ser usado con una placa ESP32-S2/ESP32 basada en “nodos”. Estos pueden ser configurados logueando usuarios desde un cliente (como en una aplicación de celular) y luego controlándolo desde la ESP Rainmaker Cloud. La placa que utilizaremos en este articulo será la ESP32-S2-Saola-1.



# Setup:

* Configure el entorno de host y el ESP IDF, este es el entorno de desarrollo de software desarrollado por Espressif.
* Descargue ESP Rainmaker Project, este tiene la funcionalidad central del firmware de Rainmaker, CLI y ejemplos.

# Conexión a Wi-Fi y control:

* Descargue la aplicación para celular, la usaremos para conectar la placa y controlarla.
* Inicie sesión en la aplicación, de esta manera tendrá una identificación única en la nube y le daremos una conexión segura con sus dispositivos.
* Añada el dispositivo.
* Controle el dispositivo desde la app.

# Descargar ESP Rainmaker Project:

Descargar ESP Rainmaker usando lo siguiente en git:

$ git clone --recursive <https://github.com/espressif/esp-rainmaker.git>

Nota:

La opción –recursive es necesaria para descargar también las dependencias JSON. En caso de que hayas clonado el repositorio sin esta opción ejecute lo siguiente para descargar los submódulos:

git submodule update --init –recursive

# Borrar la memoria flash:

Es recomendable borrar la memoria flash si esta usando Rainmaker por primera vez, luego deberá subir el firmware. Estos son los pasos:

$ cd /path/to/esp-rainmaker/examples/switch/

$ export ESPPORT=/dev/tty.SLAB\_USBtoUART #Set your ESP32-S2's serial port here

$ idf.py set-target esp32s2

$ idf.py build

$ idf.py erase\_flash

$ idf.py flash monitor

Cuando intente hacer esto por primera vez el monitor le enseñara lo siguiente:

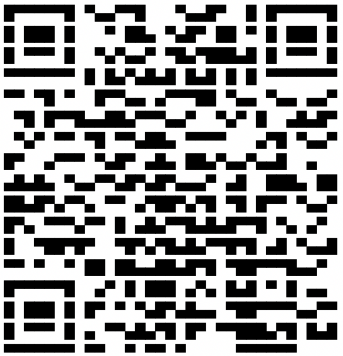
I (755) esp\_claim: Initialising Self Claiming. This may take time.

Esto significa que la placa no tiene las credenciales para conectarse a la nube de ESP Rainmaker, pero eventualmente las tendrá por el sistema de “Claiming” después de que se conecte a Wi-Fi.

Cuando todas las configuraciones básicas estén hechas el dispositivo mostrara un código QR en la terminal, este se puede usar para conectar el dispositivo y la placa por Wi-Fi.

# ¿El código QR no renderiza?

Puede ser que la terminal no renderice bien el código QR y aparezca de la siguiente forma:



En ese caso busque las siguientes líneas en la terminal y copie el URL que se le da:

I (1080) app\_wifi: If QR code is not visible, copy paste the below URL in a browser.

[https://rainmaker.espressif.com/qrcode.html?data={"ver":"v1","name":"PROV\_000317","pop":"a1000317","transport":"softap"}](https://rainmaker.espressif.com/qrcode.html?data=%7b%22ver%22:%22v1%22,%22name%22:%22PROV_000317%22,%22pop%22:%22a1000317%22,%22transport%22:%22softap%22%7d)

Tipos estándar predefinidos:

Crear un dispositivo necesita diferentes llamados. Por ejemplo para crear una bombilla con nombre, poder y brillo necesitaría lo siguiente:

**esp\_rmaker\_device\_t** \*device = esp\_rmaker\_device\_create("Light", NULL, NULL);

esp\_rmaker\_device\_add\_param(device, esp\_rmaker\_param\_create("name", NULL, esp\_rmaker\_str("Light"),

PROP\_FLAG\_READ | PROP\_FLAG\_WRITE | PROP\_FLAG\_PERSIST));

**esp\_rmaker\_param\_t** \*power\_param = esp\_rmaker\_param\_create("power", NULL, esp\_rmaker\_bool(true), PROP\_FLAG\_READ | PROP\_FLAG\_WRITE);

esp\_rmaker\_param\_add\_ui\_type(power\_param, ESP\_RMAKER\_UI\_TOGGLE);

esp\_rmaker\_device\_add\_param(device, power\_param);

esp\_rmaker\_device\_assign\_primary\_param(device, power\_param);

**esp\_rmaker\_param\_t** \*brightness\_param = esp\_rmaker\_param\_create("brightness", NULL, esp\_rmaker\_int(100), PROP\_FLAG\_READ | PROP\_FLAG\_WRITE);

esp\_rmaker\_param\_add\_ui\_type(brightness\_param, ESP\_RMAKER\_UI\_SLIDER);

esp\_rmaker\_param\_add\_bounds(brightness\_param, esp\_rmaker\_int(0), esp\_rmaker\_int(100), esp\_rmaker\_int(1));

esp\_rmaker\_device\_add\_param(device, brightness\_param);

Esta flexibilidad es muy útil ya que te deja definir cualquier tipo de dispositivo. Para simplificar algunos casos de uso comunes, hemos definido algunos tipos estándar y añadimos APIs de ayuda para ellos. Utilizando tipos estándar el código de arriba se transformara en lo siguiente:

light\_device = esp\_rmaker\_lightbulb\_device\_create("Light", NULL, true);

esp\_rmaker\_device\_add\_cb(light\_device, write\_cb, NULL);

esp\_rmaker\_device\_add\_param(light\_device, esp\_rmaker\_brightness\_param\_create("brightness", 100));

Estos tipos estándar son útiles para definir distintos manejos especiales en apps móviles u otras aplicaciones de terceros como Alexa o el asistente de Google. Abajo hay una tabla de dispositivos estándar, servicios, parámetro y elementos UI.

# Dispositivos:

| **Nombre** | **Tipo** | **Parámetros** | **Primario** | **Imagen** |
| --- | --- | --- | --- | --- |
| Switch | esp.device.switch | **Name**, **Power\*** | Power |  |
| Bombilla de luz | esp.device.lightbulb | **Name**, **Power\***, Brightness, Color Temperature, Hue, Saturation, Intensity | Power |  |
| Ventilador | esp.device.fan | **Name**, **Power\***, Speed, Direction | Power |  |
| Sensor de temperatura | esp.device.temperature-sensor | **Name**, **Temperature\*** | Temperature |  |

# Servicios:

| **Nombre** | **Tipo** | **Parámetros** |
| --- | --- | --- |
| OTA | esp.service.ota | **OTA URL**, **OTA Status**, **OTA Info** |

# Parámetros:

| **Nombre** | **Tipo** | **Tipo de dato** | **UI Type** | **propiedades** | **Min, Max, Step** |
| --- | --- | --- | --- | --- | --- |
| Name | esp.param.name | String |  | Read, Write | 1, 32, - |
| Power | esp.param.power | Bool | esp.ui.toggle | Read, Write |  |
| Brightness | esp.param.brightness | Int | esp.ui.slider | Read, Write | 0, 100, 1 |
| Color Temperature | esp.param.cct | Int | esp.ui.slider | Read, Write | 2700, 6500, 100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hue | esp.param.hue | Int | esp.ui.slider | Read, Write | 0, 360, 1 |
| Saturation | esp.param.saturation | Int | esp.ui.slider | Read, Write | 0, 100, 1 |
| Intensity | esp.param.intensity | Int | esp.ui.slider | Read, Write | 0, 100, 1 |
| Speed | esp.param.speed | Int | esp.ui.slider | Read, Write | 0, 5, 1 |
| Direction | esp.param.direction | Int | esp.ui.dropdown | Read, Write | 0, 1, 1 |
| Temperature | esp.param.temperature | Float |  | Read |  |
| OTA URL | esp.param.ota\_url | String |  | Write |  |
| OTA Status | esp.param.ota\_status | String |  | Read |  |
| OTA Info | esp.param.ota\_info | String |  | Read |  |

# 

# Elementos UI:

| **Nombre** | **Tipo** | **Tipos de datos** | **Requerimientos** | **Muestra** |
| --- | --- | --- | --- | --- |
| Text (Default) | esp.ui.text | All | N/A |  |
| Toggle Switch | esp.ui.toggle | Bool | N/A |  |
| Slider | esp.ui.slider | Int, Float | Bounds (min, max) |  |

Conceptos:

# Nodos:

Los nodos son productos basados únicamente en ESP32-S2/ESP32. Estos tienen un identificador asociado (node\_id) y un conjunto de credenciales dadas por el servicio de “claiming”. Los usuarios le pueden dar un nombre amigable al nodo. Un único nodo puede soportar muchos dispositivos.

## Atributos:

Son opcionales y se pueden personalizar los meta datos para cierto nodo, algo así como un numero de serie.

# Dispositivos:

Un dispositivo es una entidad-controlable lógica como un switch, bombilla de luz, termostato, sensor de temperatura, etc. Un nodo generalmente tiene un solo dispositivo. Aunque una ESP32-S2 puede tener múltiples dispositivos como dos switch en la misma placa, o puede actuar como un puente para otros dispositivos como sensores BLE. Cada dispositivo debe tener un nombre único entre nodos y opcionalmente un tipo.

## Atributos:

Son parecidos a los atributos de un nodo. Estos son adicionales, definidos por el usuario, tienen parejas de nombres y valores de solo lectura, estos actúan como meta datos para el dispositivo. Estos pueden ser como la versión de firmware, numero serial, etc. Serán útiles especialmente cuando haya varios dispositivos en puente con un nodo.

# Parámetros:

Estos son parámetros de control y monitoreo de un dispositivo. Estos deben tener un único nombre en todos los dispositivos. Pueden tener campos adicionales como valor, tipo, uniones, información UI, etc. Los valores de los parámetros pueden ser booleanos, int, float o String.

# Servicios:

Un servicio es una entidad muy parecida a un dispositivo en términos de estructura, con la principal diferencia de que se usa para operaciones que no son necesarias que el usuario las vea.

# Claiming:

La comunicación MQTT entre un nodo y el servicio de ESP Rainmaker usa x.509 certificados basados mutuamente en autenticación de la conexión TLS. En el contexto de ESP Rainmaker, “Claiming” es el proceso en el cual la ESP32-S2/ESP32 obtiene la certificación/autenticación del ESP Rainmaker Claiming Service.

La comunicación con el servicio de claiming ocurre a través del HTTP REST APIs