Plataforma IoT

http://174.138.53.126/#/console/device/esp8266

http://docs.thinger.io/arduino/#supported-hardware-esp8266-nodemcu

pio lib install thinger.io o en platformio.ini agregar línea lib_deps = thinger.io

SKETCH OVERVIEW

http://docs.thinger.io/arduino/#coding-sketch-overview

“The loop() is the place to always call to the thing.handle() method, so the thinger libraries can handle the connection with the platform. This is the place also for calling your endpoints, or streaming real-time data to an open WebSocket. Please, take into account to do not add any delay inside the loop() except if you know what you are doing, like working with deep sleep modes or so in your device. Any other delay will condition the proper functioning of Thinger in your device. Also it can be bad to read a sensor value in every loop if the sensor takes too much time to complete a read. This will result in a device with a noticeable lag while attending to our commands.”

ADDING RESOURCES

In the Thinger.io platform, each device can define several resources. You can think that a resource is anything you can sense or actuate. For example, a typical resource will be a sensor value like temperature or humidity, or a relay that turns on and off a light. This way, you should define the resources you need to expose over the Internet.

All resources must be defined inside the setup() method of the Arduino sketch. This way the resources are configured at the beginning, but can be accessed later as necessary.

There are three different types of resources, which are explained in the following sections.

- Input Resources
- Output Resources
- Input/Output Resources

EASIER RESOURCES

COMMUNICATION BETWEEN DEVICES

In Thinger.io, it is possible that devices can communicate between them. There are two possibilities here. One is the communication between devices from the same account, and the other is the communication between devices from different accounts. Here we describe the two different approaches: Same account communication, Communication between different accounts

USING ENDPOINTS

In Thinger.io, an endpoint is defined as some kind of external resource that can be accessed by the device. With the endpoints feature, devices can easily send emails, SMS, push data to external WebServices, interact with IFTTT, and any general action that can be made by using WebHooks (Calling HTTP/HTTPS URLs).

STREAMING RESOURCES

In Thinger.io you can open WebSockets connections against your devices, so you can receive sensor values, events, or any other information in real-time. The WebSockets are mainly used in the Dashboard feature of the Console, and are normally used for streaming resources at a fixed configurable interval. This functionality is available right out of the box when you define an output resource. However, if you want to transmit the information right when it is required, like when your device detects a movement, presence, etc., you must program some code, that is quite similar to calling an endpoint.

In this case, you must detect when you want to stream the event, like the accelerometer value is over some threshold, your presence sensor is making a detection, or the compass heading is changing. This is up to you when it is necessary to stream new data. Streaming resources also requires that another endpoint is connected listening for them (i.e., from a WebSocket connection), so if there is no one listening for this data, the data is not sent. This is handled automatically by the client library and the server, therefore it is safe to stream data always, as the device will transmit the information only when there is a destination.

ENABLING DEBUG OUTPUT

Thinger.io library provides extensive logging of its activities, which is especially useful when one needs to troubleshoot authentication and Wi-Fi connectivity issues. Include the following definition in your sketch, but make sure it comes first, before any other includes (it was reported to cause crashes on some boards otherwise).

#define _DEBUG_ *the rest of your sketch goes here It is also necessary to enable Serial communication, as all the debugging information is displayed over Serial. So enable it in your sketch in the setup method. <code> void setup() { Serial.begin(115200); } </code>* ***ESP8266 DEEP SLEEP AND SMARTCONFIG*** *SmartConfig allows one to configure board's WiFi credentials via an external device on the same network (e.g. smartphone or another wifi client). This means no sensitive information goes into a sketch nor in a config file on a device. Deep Sleep is a special mode of ESP8266 which allows it to shut down most of the circuits and wake up after some configurable time. For deep sleep (and wake up) to work properly, one has to connect GPIO16 (usually a D0 on dev boards) and RST pins. However, some boards chose to wire a built-in LED to the same D0 pin, and will go into a crash loop when using ThingerSmartConfig class, which uses the LED as a debugging aid at runtime. The solution is to use an overloaded constructor and disable its use of the LED. <code> ThingerSmartConfig thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL, false);* required for deep sleep </code>

INTERACTING

It is possible to easily interact with your devices within minutes once you have defined your resources and the device is connected to the platform. There are several ways of interaction, like creating dashboards, accessing with a mobile application (currently only in Android), accessing from the API explorer, or calling the cloud API directly.

CLOUD CONSOLE

The Cloud Console add some support for out of the box device interaction. You can interact with your devices from the API explorer, or also create real-time dashboards to display the device information. It is not necessary to write additional code to support this kind of interaction, just only declare your

resources in your device and you will be able to access them directly in the console.

So this section will review different interaction approaches from the cloud console.

- Dashboards
- API Explorer

From:
https://wiki.unloquer.org/ -

Permanent link:
https://wiki.unloquer.org/personas/brolin/proyectos/plataforma_iot?rev=1496269270

Last update: **2017/05/31 22:21**