

Guía básica

ThingsBoard, Arduino, Esp.

Asesores institucionales:

Dr. BARRIOS DEL VALLE GUILLERMO

Dr. RAMIREZ ZUÑIGA GUILLERMO

Estudiantes:

RIOS PEREZ GABRIEL

MARTÍNEZ VILLADA OSMAR EMMANUEL

Conectar con arduino	2
Gestor de placas	2
Librerías de arduino	3
Instalar Git para poder descargar repositorios desde github	3
Conectar con python en linux y windows	4
Windows:	5
Ubuntu:	5
Comandos de la terminal para ubicar carpetas y abrir archivos	7
Ubuntu:	7
Windows:	8
Esptool en linux y windows	8
Mpfshe ll en linux	10
Librerías en python	11
Dispersión de códigos en python	12
Thingsboard	12
Dispositivos	13
Atributos	15
Telemetría	15
Registro de auditoría	16

Conectar con arduino

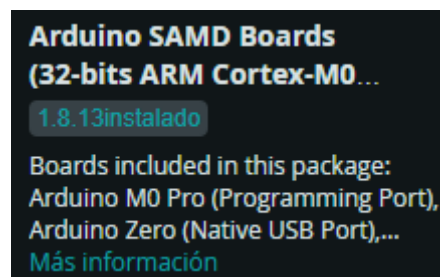
- Para poder establecer una conexión mediante el microcontrolador de Arduino, necesitamos tener los drivers correspondientes; ch340 o silicon labs cp210x (este último suele venir por defecto en los computadores) ambos permiten el uso de los puertos del computador para poder mandar datos mediante comunicación serial.
- Después realizaremos la instalación del software de IDE Arduino para poder cargar nuestros códigos a dicho microcontrolador.
- Para concluir debemos estar seguros que las librerías usadas en el código estén descargadas correctamente y en la versión requerida, esto es para que no haya fallos a la hora de subir dicho código y ejecutarlo de forma correcta.

Gestor de placas

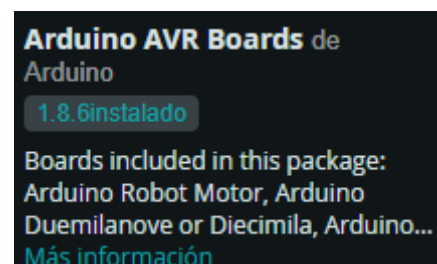
- Para poder leer diferentes microcontroladores en el software de arduino necesitamos instalar las siguientes placas; cada una para su respectivo microcontrolador.
- La forma más común de instalar placas es desde el nombre de “gestor de placas” que viene por defecto en arduino, sin embargo, al instalar versiones nuevas causa ciertos errores al momento de utilizar los microcontroladores.

Las que recomendamos utilizar son las siguientes:

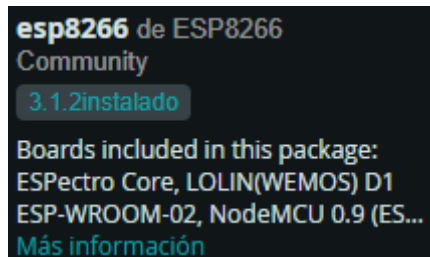
- Arduino SAMD Boards:



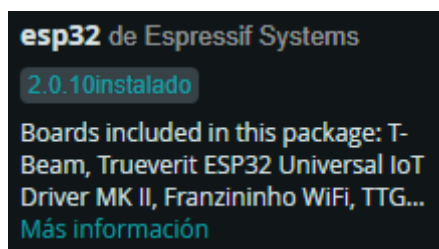
- Arduino AVR Boards:



- ESP8266: http://arduino.esp8266.com/stable/package_esp8266com_index.json



- ES32: https://dl.espressif.com/dl/package_esp32_index.json



Librerías de arduino

Para poder agregar las librerías que necesitarás o podrías ocupar a futuro te recomendamos leer la siguiente información, la cual nos permite agregar librerías externas mediante archivos .zip o desde el mismo IDE de Arduino.

<https://programarfacil.com/blog/arduino-blog/instalar-una-libreria-de-arduino/>

la librería que debemos comprender es la de arduinoJson, ya que con esta librería es la que nos permite enviar datos mediante el protocolo mqtt, en la siguiente pag, se encuentra una explicación algo detallada de dicha librería:

<https://www.luisllamas.es/enviar-y-recibir-mensajes-por-mqtt-con-arduino-y-la-libreria-pubsubclient/>

Instalar Git para poder descargar repositorios desde github

Windows:

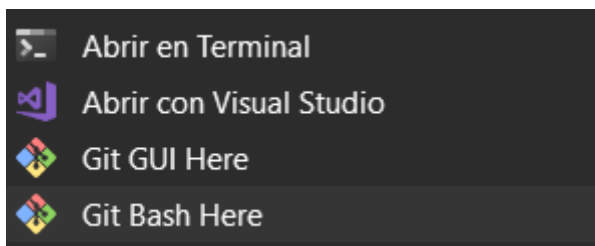
Primero debemos descargar Git SCM para el sistema operativo que necesitemos, podemos obtener el link de descarga del link que se encuentra a continuación:

<https://gitforwindows.org/>

Una vez descargado el ejecutable, se debe realizar la instalación siguiendo las instrucciones que te muestre en pantalla.

Es muy similar a instalar algún otro programa simplemente aceptar los términos y condiciones y dar en siguiente hasta llegar al final.

Una vez terminado el proceso de instalación anterior, se debe abrir el símbolo del sistema, aparecerá con un dibujo muy peculiar del lado izquierdo con el nombre de git bash, como el que se muestra a continuación:



Una vez abierto se debe configurar el nombre de usuario y su correo con los siguientes comandos:

```
git config --global user.name "Tu nombre de usuario"
```

```
git config --global user.email "tuemail@gmail.com"
```

Ubuntu:

Para descargar ocuparemos abrir la terminal del sistema, y dentro simplemente pondremos los siguientes comandos:

```
sudo apt-get update
```

```
sudo apt-get install git
```

Ya terminada la instalación verificamos con el comando: git-version

A continuación, ejecuta los siguientes comandos en la terminal para poder configurar tu correo y nombre de usuario que están asociados a tu cuenta GIT:

```
git config --global user.name "Tu nombre de usuario"
```

```
git config --global user.email "tuemail@gmail.com"
```

Recuerda cambiar **tu nombre** y **ejemplo@email.com** por los datos de tu cuenta GIT.

Conectar con python en linux y windows

Windows:

Para esto, primero necesitamos instalar lo que es python, este puede ser descargado como un archivo .exe .

Para verificar su instalación solo abrimos la terminal y se colocará el siguiente comando:

```
$ python3 --version
```

Ubuntu:

Python puede ser instalado desde la terminal / cmd.

Para esto simplemente aplicamos el siguiente comando:

```
$ sudo apt install python3
```

Pedirá la contraseña de SuperUsuario, no se mostrará que escriba dicha contraseña así que simplemente denle enter cuando ya la hayan escrito y listo, para verificar si se instaló de forma correcta en ubuntu se puede usar el comando mencionado anteriormente

```
gbv@gbv-Vostro-15-3510:~$ sudo chmod 777 /dev/ttyUSB0
gbv@gbv-Vostro-15-3510:~$ mpfshell

** Micropython File Shell v0.9.1, sw@kaltpost.de **
-- Running on Python 3.10 using PySerial 3.5 --

mpfs [/]> open ttyUSB0

Failed to open: ser:/dev/ttyUSB0

mpfs [/]>
gbv@gbv-Vostro-15-3510:~$ sudo chmod 777 /dev/ttyUSB0
gbv@gbv-Vostro-15-3510:~$ esptool.py --port /dev/ttyUSB0 erase_flash
esptool.py v4.6
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... Unsupported detection protocol, switching and trying again...
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 48:3f:da:36:2a:3a
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 1.9s
Hard resetting via RTS pin...
```

Este es un ejemplo de cuando conectamos por primera vez nuestra placa y sin haber hecho nada con ella abrimos mpfshell.

Al inicio debemos abrir los puertos con el código “sudo chmod 777 /dev/ttyUSB0”, en el ejemplo tenemos nos está mandando un error después de abrir mpfshell porque no detecta el puerto, pero esto es porque la placa es nueva y tenemos que subirle una nueva flash para que la pueda trabajar con ella.

Para esto primero le borramos la flash que tenga con el código “esptool.py --port /dev/ttyUSB0 erase_flash”

```
gbv@gbv-Vostro-15-3510:~$ cd Descargas/
gbv@gbv-Vostro-15-3510:~/Descargas$ esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash
--flash_size=detect 0 esp8266-20230426-v1.20.0.bin
esptool.py v4.6
Serial port /dev/ttyUSB0
Connecting....
Detecting chip type... Unsupported detection protocol, switching and trying again...
Connecting....
Detecting chip type... ESP8266
Chip is ESP8266EX
Features: WiFi
Crystal is 26MHz
MAC: 48:3f:da:36:2a:3a
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Flash will be erased from 0x00000000 to 0x0009afff...
Flash params set to 0x0040
Compressed 634016 bytes to 420365...
Wrote 634016 bytes (420365 compressed) at 0x00000000 in 10.1 seconds (effective 503.5 kbit/s)
...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
gbv@gbv-Vostro-15-3510:~/Descargas$ cd ..
gbv@gbv-Vostro-15-3510:~$ cd Documentos/
gbv@gbv-Vostro-15-3510:~/Documentos$ cd prueb1_dht22/
gbv@gbv-Vostro-15-3510:~/Documentos/prueb1_dht22$ ls
main.py
```

En esta parte deberíamos de tener descargada la flash que le vamos a colocar en .bin así que nos colocamos en la carpeta donde la tenemos descargada (en este caso está en la carpeta Descargas) y con el código “esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_size=detect 0 **esp8266-20230426-v1.20.0.bin**” esta última parte resaltada es el nombre del archivo (por si se llegara a descargar otra version).

Solo deberíamos ir a la carpeta donde tenemos nuestro archivo .py para subirlo a nuestra placa.

```
gbv@gbv-Vostro-15-3510:~/Descargas$ cd ..
gbv@gbv-Vostro-15-3510:~$ cd Documentos/
gbv@gbv-Vostro-15-3510:~/Documentos$ cd prueb1_dht22/
gbv@gbv-Vostro-15-3510:~/Documentos/prueb1_dht22$ ls
main.py
gbv@gbv-Vostro-15-3510:~/Documentos/prueb1_dht22$ mpfshell

** Micropython File Shell v0.9.1, sw@kaltpost.de **
-- Running on Python 3.10 using PySerial 3.5 --

mpfs [/]> open ttyUSB0
Connected to esp8266
mpfs [/]> ls

Remote files in '/':

    boot.py

mpfs [/]> rm boot.py
mpfs [/]> put main.py
mpfs [/]> ls

Remote files in '/':

    main.py

mpfs [/]> repl
>
*** Exit REPL with Ctrl+] ***

MicroPython v1.20.0 on 2023-04-26; ESP module with ESP8266
Type "help()" for more information.
>>>
```

Al colocarnos en la carpeta iniciamos mpfshell y vemos que ya tiene la placa un boot.py, lo podemos retirar con “rm” o lo podemos dejar y no causa ningún problema. ya que le cargamos el main.py de nuestro código entramos a la repl y no va a empezar el código de repente sino que tenemos que dar un reset con “ctrl + D” y empezará a correr.

Comandos de la terminal para ubicar carpetas y abrir archivos

Ubuntu:

Entrar en una carpeta	cd “nombre de la carpeta”
Regresar atrás una carpeta	cd ..
Regresar al escritorio u “home”	cd
Ver los archivos que hay en la carpeta donde estas ubicado	ls
crear una o varios directorios a la vez.	mkdir “nombre del directorio”

Elimina permanentemente un directorio vacío

rmdir "nombre del directorio"

mover archivos

mv "nombre del archivo" "dirección de destino"

copiar archivos

cp "nombre del archivo" "dirección de destino"

Windows:

Entrar en una carpeta

cd "nombre de la carpeta"

Regresar atrás una carpeta

cd ..

Regresar al escritorio u "home"

cd

Enlista los archivos donde te encuentres ubicado

dir

limpia la terminal

cls

Te muestra el árbol de directorios de una carpeta concreta que le digas

tree "nombre de la carpeta"

este comando te dará todos los comando disponibles en tu computador

help

Esptool en linux y windows

como instalarlo para linux:

dentro de la terminal de ubuntu pondremos el siguiente comando: **sudo apt-get install esptool**

en el caso de windows podemos instalarlo de igual manera desde la terminal con los siguientes comandos: «**python -m pip install esptool**», «**py -m pip install esptool**» o «**pip2 install esptool**».

de igual forma les pedirá la contraseña de superusuario.

comandos:

load_ram Descargar una imagen en la RAM y ejecutarla

dump_mem	Volcar memoria arbitraria a disco
read_mem	Leer ubicación arbitraria de memoria
write_mem	Leer-modificar-escribir en ubicación arbitraria de memoria
write_flash	Escribir un bloque binario en la memoria flash
run	Ejecutar código de aplicación en la memoria flash
image_info	Volcar encabezados de una imagen de aplicación
make_image	Crear una imagen de aplicación a partir de archivos binarios
elf2image	Crear una imagen de aplicación a partir de un archivo ELF
read_mac	Leer dirección MAC desde ROM OTP
chip_id	Leer ID de chip desde ROM OTP
flash_id	Leer el fabricante y el ID del dispositivo de la memoria flash SPI
read_flash_status	Leer registro de estado de la memoria flash SPI
write_flash_status	Escribir registro de estado de la memoria flash SPI
read_flash	Leer contenido de la memoria flash SPI
verify_flash	Verificar un bloque binario en comparación con la memoria flash
erase_flash	Realizar un Borrado de Chip en la memoria flash SPI
erase_region	Borrar una región de la memoria flash
merge_bin	Combinar múltiples archivos binarios en bruto en un solo archivo para flasheo posterior
get_security_info	Obtener algunos datos relacionados con la seguridad
version	Imprimir la versión de esptool
opciones:	
-h, --ayuda	mostrar este mensaje de ayuda y salir
--chip	
{auto,esp8266,esp32,esp32s2,esp32s3beta2,esp32s3,esp32c3,esp32c6beta,esp32h2beta1,esp32h2beta2,esp32c2,esp32c6,esp32h2}, -c	

{auto,esp8266,esp32,esp32s2,esp32s3beta2,esp32s3,esp32c3,esp32c6beta,esp32h2beta1,esp32h2beta2,esp32c2,esp32c6,esp32h2}

Tipo de chip objetivo

--port PUERTO, -p PUERTO Dispositivo de puerto serie

--baud VELOCIDAD, -b VELOCIDAD Velocidad de baudios del puerto serie utilizada al flashear/leer

--before {default_reset,usb_reset,no_reset,no_reset_no_sync}

Qué hacer antes de conectarse al chip

--after {hard_reset,soft_reset,no_reset,no_reset_stub}, -a
{hard_reset,soft_reset,no_reset,no_reset_stub}

Qué hacer después de que esptool.py haya terminado

--no-stub Deshabilitar el lanzamiento del fragmento de arranque (stub) del flasher, solo comunicarse con el gestor de arranque ROM. Algunas características no estarán disponibles.

--trace, -t Habilitar la salida de traza de nivel de interacciones de esptool.py.

--override-vddsdio [{1.8V,1.9V,OFF}]

Anular el regulador de voltaje interno VDDSDIO de ESP32 (usar con precaución)

--connect-attempts INTENTOS_CONEXION

Número de intentos de conexión, negativo o 0 para infinito. Por defecto: 7.

Mpfshell en linux

Para su instalación primero debemos tener en cuenta que necesitamos el comando “pip” para poder realizar dicha instalación para esto, debemos tener ya instalado python3, si ya se cuenta con el python3 simplemente debemos ingresar el siguiente comando en la terminal:

```
sudo pip install mpfshell
```

Ya instalado simplemente ponemos “mpfshell” en la terminal y nos indicará que ya nos encontramos dentro de la aplicación con su versión en color verde como se muestra a continuación:

```
** Micropython File Shell v0.9.0, sw@kaltpost.de **
-- Running on Python 3.11 using PySerial 3.5 --
```

Para poder conectar con una esp es necesario poner el siguiente comando:

open "puerto donde esta tu esp"

Ejemplo: open COM3

Hay ocasiones donde el computador no reconoce los microcontroladores, es necesario revisar que los drivers se encuentren actualizados o si es compatible con tu computador, los más recomendados para usar con el ch340 y el silicon labs 320x.

O bien utilizar el siguiente comando para poder dar acceso a los puertos:

```
sudo chmod 777 "puerto donde se encuentra tu microcontrolador"
```

Ejemplo:

```
sudo chmod 777 dev/tty/USB0
```

Ya realizada la conexión con el microcontrolador deseado podemos subir los archivos con el comando *put "nombre del archivo"*

Puedes tener más información acerca de mpfshell con el comando help o en el siguiente link:

<https://www.prometec.net/subir-programas-micropython/>

Librerías en python

Para poder ocupar alguna librería en tu código de python debemos instalarla previamente con algunos comandos desde la terminal. Los comandos son sencillos y el número de librerías que se pueden instalar es demasiado amplio.

A continuación les dejaremos un link con información más detallada de como instalar estas librerías o módulos en lo que sería python.

<https://programminghistorian.org/es/lecciones/instalar-modulos-python-pip>

También se podría ocupar un programa diferente para programar en lenguaje de python como sería Thonny y seguir ocupando las mismas librerías y dentro de la interfaz es posible instalar más librerías desde Tools >> Manage packages...

<https://mrchunckuee.blogspot.com/2021/06/raspberry-pi-pico-instalar-o-agregar.html?m=1>

Los comandos en la terminal para saber que librerías o módulos tienes instalados es el siguiente: "pip freeze". Esto listará todas las librerías Python instaladas en tu sistema.

Dispersión de códigos en python

A la hora de tener diferentes códigos haciendo diferentes funciones podemos hacerlos llamar en nuestro código principal para que realicen las funciones conforme queramos que se realicen. Esto es para que se ejecute el código inmediatamente en cuanto se encienda la placa aunque no esté conectada a una interfaz.

por ejemplo:

```
7 import machine
8 import gc
9 import network, utime, ntptime, time
10 from wifi import activate_wifi
11 from funcion import *
12 from funciones import *
13 from machine import Pin
14 from umqtt.simple import MQTTClient
15 import math
```

Estas son las primeras líneas de código de un main.py en las cuales ocupo “from” para decir de que código voy a sacar las funciones y “import” para saber cuáles funciones voy a utilizar en este código, si después del import tengo un asterisco significa que voy a importar todas las funciones de este código.

Normalmente cuando necesitamos que un código se inicie desde el momento en el que se alimenta la placa hay una manera fácil de hacerlo llamando a nuestro código boot.py porque nuestra placa automáticamente reconocerá este nombre y lo iniciará lo antes posible y normalmente en este se guardan las constantes que se ocuparán en el main.py

Para más información referente a esto dejaré un link con la explicación más detallada y un ejercicio de ejemplo:

<https://techexplorations.com/guides/esp32/micropython-with-the-esp32/16-how-to-run-a-program-at-boot/>

Thingsboard

ThingsBoard es una plataforma de código abierto diseñada para la administración y visualización de dispositivos de Internet de las Cosas (IoT, por sus siglas en inglés). Proporciona una variedad de herramientas y características que permiten a los desarrolladores y empresas crear aplicaciones y soluciones de IoT eficientes y escalables.

Las principales funciones y beneficios de ThingsBoard en proyectos de IoT son:

Gestión de dispositivos: ThingsBoard permite registrar, conectar y administrar dispositivos IoT de manera centralizada. Puedes monitorear el estado de los dispositivos, realizar actualizaciones de firmware y gestionar su ciclo de vida.

Recopilación de datos: La plataforma es capaz de recopilar y almacenar datos generados por los dispositivos IoT en tiempo real. Estos datos pueden ser sensoriales, de medición u otra información relevante para el proyecto.

Visualización y análisis: ThingsBoard ofrece herramientas para visualizar los datos recopilados en forma de gráficos, tablas, mapas y otros tipos de representaciones visuales. Esto facilita el análisis de los datos y la toma de decisiones informadas.

Reglas y notificaciones: Puedes configurar reglas y condiciones basadas en los datos recopilados para activar acciones automáticas o enviar notificaciones en caso de eventos específicos. Por ejemplo, puedes configurar alertas cuando un sensor registra una lectura fuera de un rango establecido.

Integraciones: ThingsBoard es compatible con diversas tecnologías y protocolos de comunicación utilizados en IoT, lo que facilita la integración de diferentes tipos de dispositivos y sistemas.

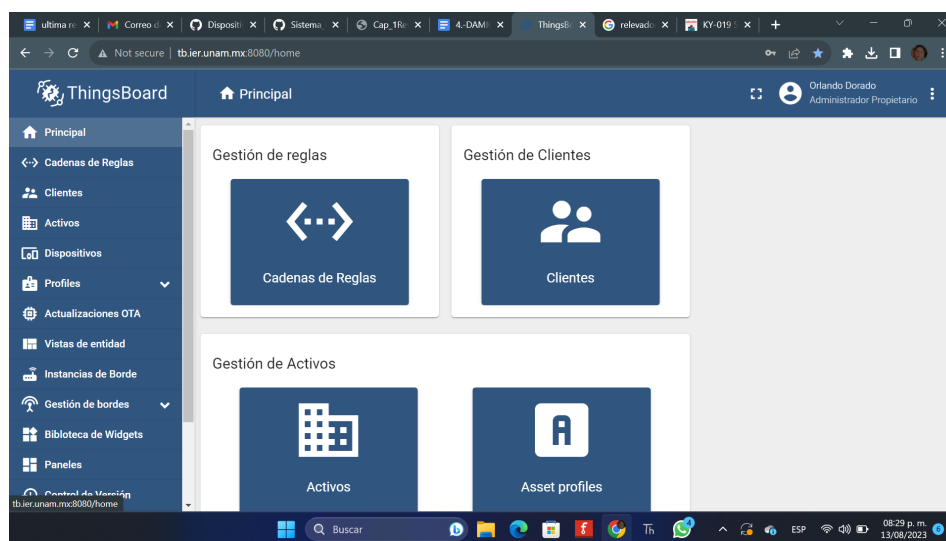
Seguridad: La plataforma ofrece capacidades de seguridad para proteger los datos y la comunicación entre los dispositivos y el sistema.

Personalización y extensibilidad: ThingsBoard es altamente personalizable y extensible. Puedes adaptar la plataforma según las necesidades específicas de tu proyecto y desarrollar módulos adicionales para agregar nuevas funcionalidades.

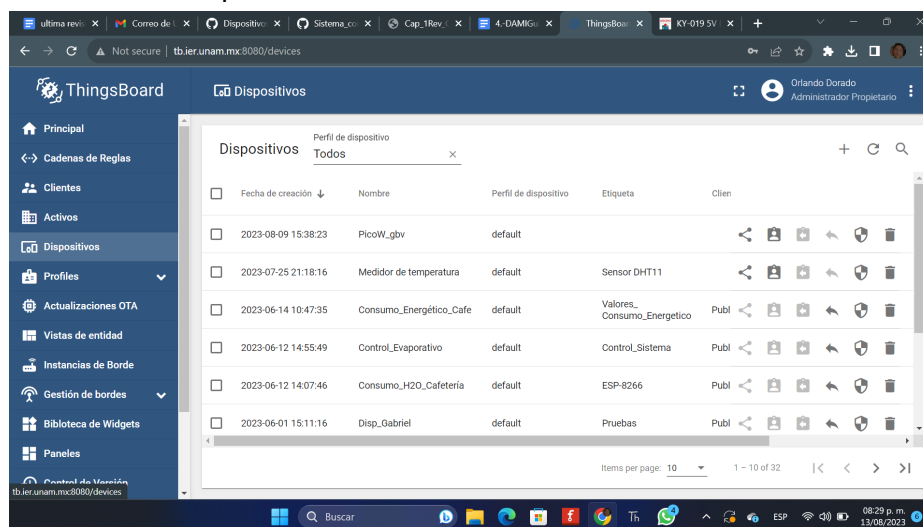
En resumen, ThingsBoard es una herramienta poderosa para la gestión y visualización de dispositivos IoT, lo que puede ser de gran utilidad en proyectos relacionados con la monitorización remota, automatización industrial, agricultura inteligente, gestión de activos, ciudades inteligentes y muchas otras aplicaciones donde se requiera administrar y analizar datos de dispositivos conectados. Al ser de código abierto, también brinda flexibilidad para personalizar y expandir sus capacidades según las necesidades de tu proyecto.

Dispositivos

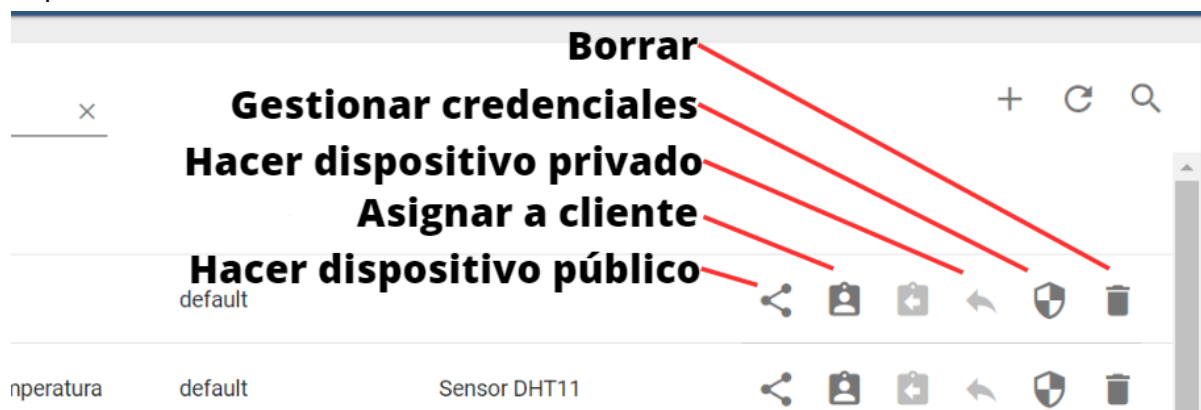
La primera vez que entremos a ThingsBoard encontraremos una interfaz de la siguiente manera:



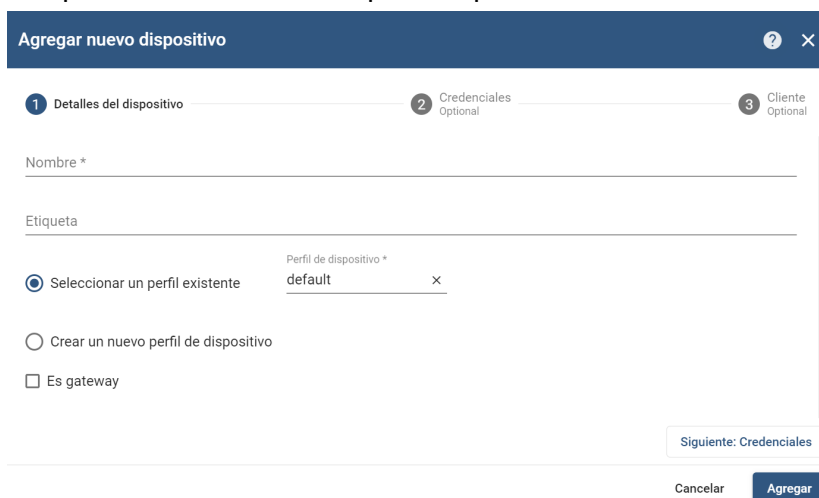
En el panel de la izquierda encontraremos una sección para los dispositivos en la cual podremos ver todos los dispositivos creados.



En esta parte tendremos algunas configuraciones u opciones de cada uno de los dispositivos



Pero a la hora de querer crear un nuevo dispositivo que trabaja como puente de información para la plataforma podemos hacerlo en la parte superior derecha en el símbolo “+”



Los siguientes pasos son simples porque solo colocaremos un nombre y una etiqueta y esta puede ser que dispositivo estaremos ocupando para mandar datos

Atributos

En ThingsBoard, los atributos son características o propiedades asociadas a los dispositivos o entidades conectadas que se utilizan para describir y representar información adicional. Estos atributos contienen datos que describen el estado, las características o cualquier otra información relevante de los dispositivos o entidades monitoreados por ThingsBoard.

Los atributos en ThingsBoard se pueden utilizar para varios propósitos, como:

Monitoreo y seguimiento: Los atributos se utilizan para almacenar y representar datos relacionados con el estado actual de un dispositivo, como temperatura, humedad, ubicación, estado de conexión, nivel de batería, etc. Estos atributos se actualizan periódicamente y se pueden utilizar para monitorear y realizar un seguimiento en tiempo real de los dispositivos.

Configuración y control: Los atributos también se pueden utilizar para almacenar configuraciones y opciones de control para los dispositivos conectados. Por ejemplo, se pueden utilizar atributos para establecer la frecuencia de actualización de datos, activar o desactivar ciertas funcionalidades, configurar umbrales de alarma, etc.

Analítica y reglas: Los atributos pueden ser utilizados por las reglas y el motor de análisis de ThingsBoard para realizar acciones o generar alertas en función de ciertos valores o condiciones de los atributos. Esto permite implementar lógica personalizada y automatización basada en datos en la plataforma.

Visualización y presentación: Los atributos se pueden utilizar para mostrar información relevante en los paneles y tableros de ThingsBoard. Los datos de los atributos se pueden representar en gráficos, tablas, medidores y otros widgets para brindar una visualización clara del estado y rendimiento de los dispositivos.

enseguida dejamos el link de la documentación oficial de ThingsBoard con respecto a los atributos:

<https://thingsboard.io/docs/user-guide/attributes/>

Telemetría

En el contexto de ThingsBoard, la telemetría se utiliza para recopilar datos en tiempo real o periódicamente desde los dispositivos IoT, como sensores, actuadores u otros dispositivos conectados, y transmitir estos datos a la plataforma para su análisis y visualización. Los datos recopilados pueden incluir información como lecturas de sensores, estados de dispositivos, ubicación geográfica y más.

La telemetría en ThingsBoard tiene varios usos y beneficios:

Monitoreo en tiempo real: Permite a los usuarios ver y supervisar en tiempo real los datos de sus dispositivos IoT. Esto es útil para realizar un seguimiento de los cambios y las tendencias a medida que ocurren.

Análisis y reportes: La plataforma recopila y almacena los datos de telemetría, lo que permite a los usuarios generar informes y realizar análisis retrospectivos sobre el comportamiento de los dispositivos con el tiempo.

Alertas y notificaciones: Los usuarios pueden configurar alertas basadas en umbrales o condiciones específicas en los datos de telemetría. Cuando se detecta una situación anómala, la plataforma puede enviar notificaciones a través de diversos canales, como correos electrónicos o mensajes SMS.

Automatización: Los datos de telemetría también pueden utilizarse para automatizar acciones. Por ejemplo, si un sensor de temperatura registra una lectura fuera del rango deseado, la plataforma podría activar automáticamente un dispositivo de enfriamiento.

Optimización de recursos: La telemetría permite a los usuarios realizar un seguimiento del rendimiento de sus dispositivos y recursos, lo que puede llevar a una mejor gestión y eficiencia en el uso de esos recursos.

Visualización de datos: ThingsBoard proporciona herramientas de visualización que permiten a los usuarios crear paneles personalizados para mostrar gráficos, tablas y otros elementos visuales que representen los datos de telemetría de manera comprensible.

Registro de auditoría

El registro de auditoría en ThingsBoard se refiere a la función que permite rastrear y registrar las acciones realizadas por los usuarios y los eventos que ocurren dentro de la plataforma. Esta característica es esencial para mantener un registro detallado de las actividades y cambios que se producen en la plataforma, lo que facilita la supervisión, el cumplimiento normativo, la resolución de problemas y la seguridad.

El registro de auditoría registra eventos relevantes, como:

Acciones de usuarios: Cualquier acción realizada por los usuarios en la plataforma, como iniciar sesión, crear, editar o eliminar dispositivos, modificar reglas, configurar paneles de control, etc.

Cambios en configuraciones: Cualquier cambio en las configuraciones de la plataforma, como ajustes de seguridad, cambios en las reglas de alerta, ajustes de acceso, etc.

Acceso a datos: Cualquier acceso a los datos de telemetría, configuraciones o elementos de la plataforma por parte de los usuarios autorizados.

Eventos de sistema: Eventos generados por el propio sistema, como notificaciones de mantenimiento, cambios en la versión de software, cambios en el estado del servidor, etc.

Errores y problemas: Registro de errores, problemas técnicos o cualquier comportamiento inusual que se haya registrado en la plataforma.

Los registros de auditoría son esenciales por varias razones:

Seguridad: Ayudan a mantener un control sobre las acciones realizadas en la plataforma, lo que contribuye a detectar actividades maliciosas o no autorizadas.

Cumplimiento normativo: En muchos sectores, es obligatorio mantener registros de auditoría para cumplir con regulaciones de seguridad y privacidad.

Resolución de problemas: Los registros de auditoría pueden ser útiles para diagnosticar problemas, rastrear causas y encontrar soluciones.

Responsabilidad: Permiten asignar responsabilidades a usuarios específicos en caso de acciones incorrectas o problemas.

Mejora continua: Los registros de auditoría proporcionan información valiosa sobre cómo se utiliza la plataforma, lo que puede llevar a mejoras en la experiencia del usuario y en la eficiencia operativa.