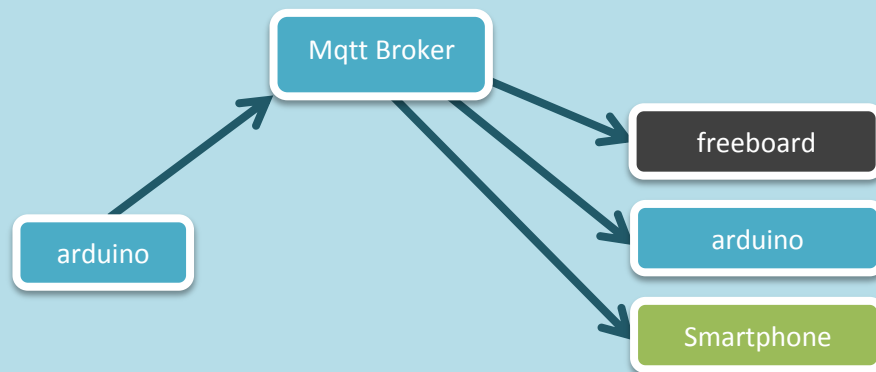


Arduino Manual para IoT

(openFramework)

Arduino+Mosquitto+NodeRED +Freeboard = Visualizacion Data en tiempo real



 **Flatbox**

Guía de instalación de un framework abierto para IoT framework/dashboard con NodeRed, Mosquitto y Freeboard.io dashboard

Basado en el tutorial de primalcortex (febrero 25, 2015)

Conocimientos básicos Requeridos:

Arduino IDE

Programación básica

Programación con sensores.

Programación con librería `#include <ArduinoJson.h> //`

<https://github.com/bblanchon/ArduinoJson/releases/tag/v5.0.7>

Haber realizado los ejemplos por defecto para Comunicaciones por medio de clientes TCP (wifi/GSM/ETH)

Haber realizado los ejemplos de las librerías de `#include <PubSubClient.h> //`

<https://github.com/knolleary/pubsubclient/releases/tag/v2.3>

Conocimiento básico de Linux

Conocimiento básico de Git

Conocimientos básicos de nodejs

Conocimiento básico de Apache2

Symplinks

Debido a su bajo costo se utilizara un chip esp8266 para el desarrollo del tutorial (cualquiera de las versiones de hardware disponibles esta bien)

Diagrama 1.0 (Modulo de ESP8266, NodeMCU v3) (\$6 USD)

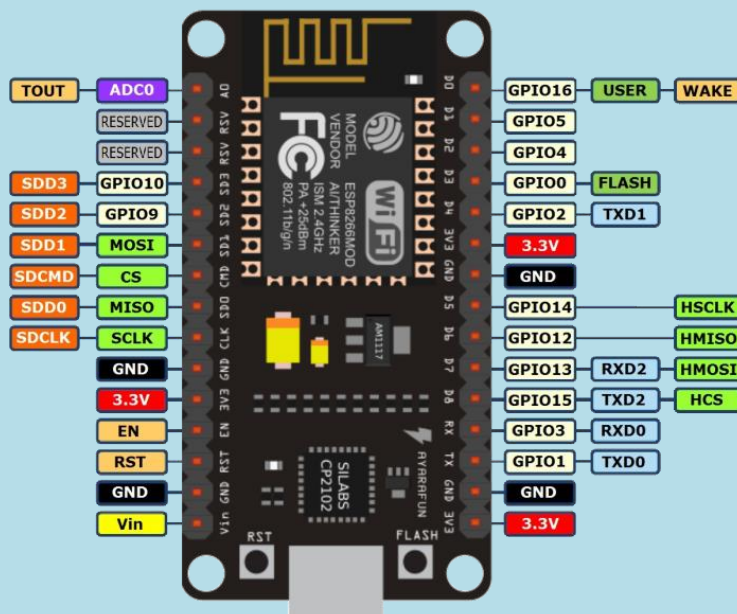
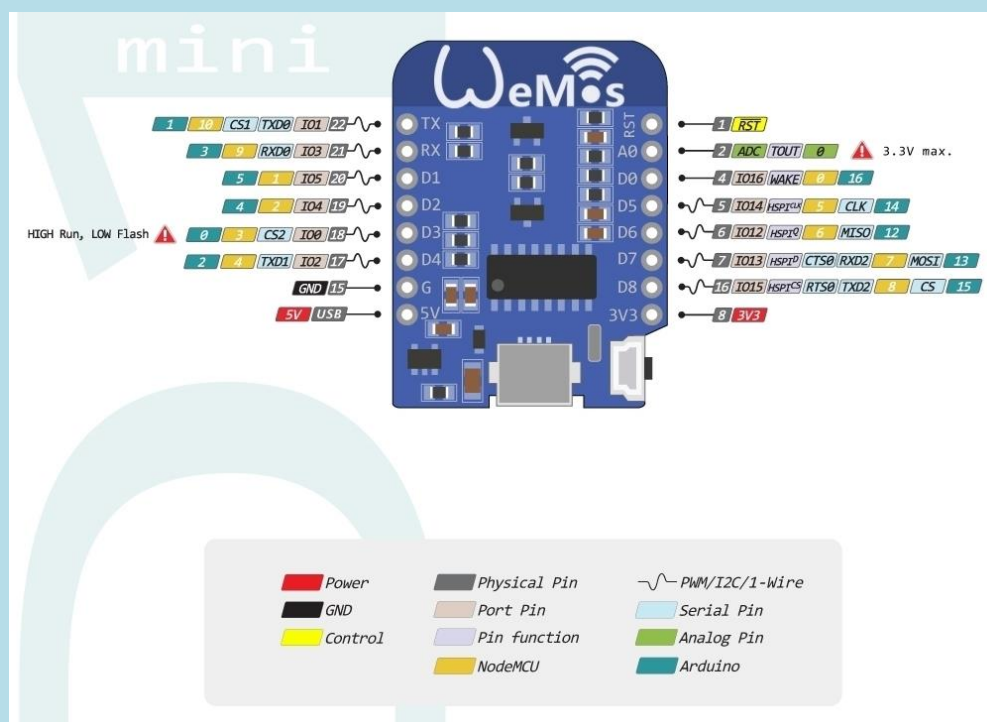
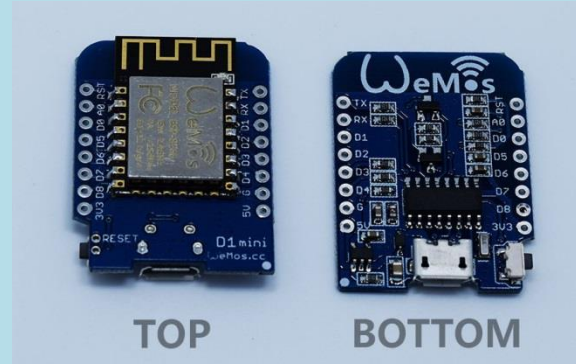


Diagrama 1.1 (Modulo de ESP8266, wemos D1 mini) (\$4 USD)



Software:

Git:

```
$ sudo apt-get update  
$ sudo apt-get install git
```

Configurado con tu cuenta de Git.

```
$ git config --global user.name "Your Name"  
$ git config --global user.email "youremail@domain.com"
```

libwebsockets-dev:

```
apt-get update  
apt-get install uuid-dev libwebsockets-dev
```

Node.js:

```
curl -sL https://deb.nodesource.com/setup | sudo bash -
```

```
sudo apt-get install nodejs
```

```
sudo apt-get install build-essential
```

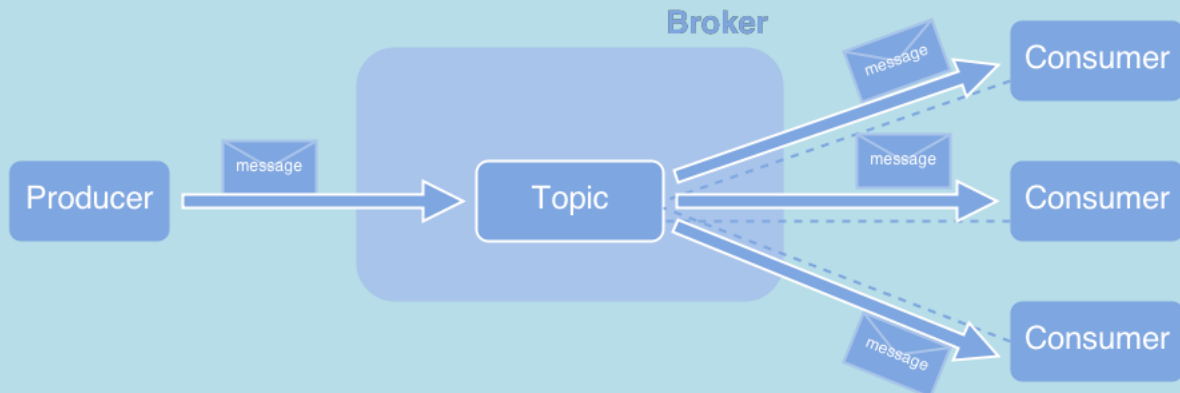
En la muchos de los casos también deberás instalar npm:

```
sudo apt-get install npm
```

HTTPD - Apache2 Web Server:

```
sudo apt-get install apache2
```

Que es un open framwork para IoT, es un conjunto de programas con licencias de código abierto los cuales nos ayudaran a capturar, estructurar y presentar los datos que transmitamos al internet desde nuestros módulos de Arduino. Existen diversos protocolos de IoT como AMQP, MQTT, STOMP, para este caso en particular estaremos utilizando MQTT para conocer a profundidad sobre el mismo pueden leer más en : <http://mqtt.org/> este es un simple servicio de publicar suscripción:



El software que estaremos usando dentro de Linux será:

IBM NodeRed: (<http://nodered.org/>) Node-RED nos provee de una interfaz basada en un navegador web que nos permite crear flujos de eventos e interconectarlos todos ellos a través de un ligero entorno de trabajo y desarrollo. Está construido en node.js, lo que le permite funcionar justo al borde de la red o en la nube, dotándole de una notable flexibilidad. El ecosistema del gestor de paquetes de node (npm) puede ser usado para extender de forma sencilla la paleta de nodos disponibles, permitiendo conexiones entre nuevos dispositivos y servicios. Node-RED ha sido desarrollado como un proyecto open-source en GitHub y está bajo una licencia Apache 2

.MQTT Broker Mosquitto: (<http://mosquitto.org/>) Mosquitto es un corredor de Mqtt para linux el cual es liviano y facil de instalar

Freeboard.io Dashboard: (<http://freeboard.io/>) es un motor dinámico de tableros opensource project GitHub page (<https://github.com/Freeboard/freeboard>). El cual nos permite diseñar y administrar los datos capturados por el corredor de MQTT por medio de WEBSockets

Paso 1: instalando NodeRed:

Primero verificamos que se encuentre instalado node.js y npm con el comandos **node -v** and **npm -v** .

Luego creamos nuestra estructura de archivos llamado IoTServer en el cual mantendremos todas nuestras configuraciones de software.

```
cd ~  
mkdir IoTServer  
cd IoTServer
```

Luego corremos desde el bash:

```
sudo npm install -g --unsafe-perm node-red
```

Ahora podemos correr el servidor de nodered con el comando: **node red.js -v**. utilizamos el **-v** al final para saber si existe algún error o si algún modulo se encuentra mal instalado:

```

$ node-red

Welcome to Node-RED
=====

25 Feb 22:51:09 - [info] Node-RED version: v0.13.1
25 Feb 22:51:09 - [info] Node.js version: v4.2.6
25 Feb 22:51:09 - [info] Loading palette nodes
25 Feb 22:51:10 - [warn] -----
25 Feb 22:51:10 - [warn] Failed to register 1 node types
25 Feb 22:51:10 - [warn] Run with -v for details
25 Feb 22:51:10 - [warn] -----
25 Feb 22:51:10 - [info] User Directory : /home/nol/.node-red
25 Feb 22:51:10 - [info] Server now running at http://127.0.0.1:1880/
25 Feb 22:51:10 - [info] Creating new flows file : flows_noltop.json
25 Feb 22:51:10 - [info] Starting flows
25 Feb 22:51:10 - [info] Started flows

```

Podremos verificar el funcionamiento del nuestro sitio de nodered en la dirección :

<http://localhost:1880>

Podemos encontrar más información sobre como configurar NodeRed en <http://nodered.org/docs/configuration.html> ahora podemos empezar a usar nodered con: **node red.js -v -s settings.js**

En node red podemos diseñar el flujo de información para la data del ESP8266 recibida por el MQTT e incluso elaborar un servicio simple REST basado en HTTP protocolo, procesar la data y almacenarla.

Para hacer que el servicio corra desde el boot:

```
sudo npm install -g pm2
```

```
pm2 start /usr/bin/node-red -- -v
```

```
pm2 info node-red
pm2 logs node-red
```

```
pm2 save
pm2 startup
```

Reboot!!!

Para almacenar la data en una base de datos podemos usar MongoDB instalando los módulos

Que hacen falta de la siguiente manera:

En el directorio ~/ .node-red

```
npm install node-red-node-mongodb
```

Paso2: Inatalar Mosquitto MQTT con websocket

Bajar la última imagen del repositorio oficial:

```
cd ~/IoTServer
wget http://mosquitto.org/files/source/mosquitto-1.4.5.tar.gz
tar xvzf mosquitto-1.4.5.tar.gz
cd mosquitto-1.4.5/
```

NOTA Cambiar wget al ultimo reléase del mosquitto actualmente mosquitto-1.4.8.tar.gz

Editar el archivo config.mk para habilitar el soporte a websockets:

```
# Build with websockets support on the broker.
WITH_WEBSOCKETS=yes
```

Y completar la instalación:

```
make
make install
```

Al terminar deberán crear y modificar el archivo de configuración por defecto **mosquitto.conf**
Desde ~/IoTServer/mosquitto-1.4.8/

```
cp mosquitto.conf /usr/local/etc
cd /usr/local/etc
```

Y cambien la configuración por defecto a:

```
# Port to use for the default listener.  
#port 1883  
listener 1883  
listener 9001  
protocol websockets
```

NOTA Importante: después de la versión 1.4.5 el usuario por defecto para la ejecución del servicio es **mosquitto**. **Así que deberá asignarle permisos para que ejecute el servicio:**

```
useradd -lm mosquitto  
cd /usr/local/etc  
chown mosquitto mosquitto.conf
```

Adicionalmente se recomienda agregar las siguientes líneas de configuración a modo de obtener la mayor visibilidad sobre el servicio:

En el archivo `cd /usr/local/etc/mosquitto.conf`

Ver imagen abajo.


```

# Note that if the broker is running as a Windows service it will default to
# "log_dest none" and neither stdout nor stderr logging is available.
# Use "log_dest none" if you wish to disable logging.
log_dest file /var/log/mosquitto.log

# If using syslog logging (not on Windows), messages will be logged to the
# "daemon" facility by default. Use the log_facility option to choose which of
# local0 to local7 to log to instead. The option value should be an integer
# value, e.g. "log_facility 5" to use local5.
#log_facility

# Types of messages to log. Use multiple log_type lines for logging
# multiple types of messages.
# Possible types are: debug, error, warning, notice, information,
# none, subscribe, unsubscribe, websockets, all.
# Note that debug type messages are for decoding the incoming/outgoing
# network packets. They are not logged in "topics".
log_type error
log_type warning
log_type notice
log_type information

# Change the websockets logging level. This is a global option, it is not
# possible to set per listener. This is an integer that is interpreted by
# libwebsockets as a bit mask for its lws_log_levels enum. See the
# libwebsockets documentation for more details. "log_type websockets" must also
# be enabled.
websockets_log_level 0

# If set to true, client connection and disconnection messages will be included
# in the log.
connection_messages true

# If set to true, add a timestamp value to each log message.
log_timestamp true

```

Configurando el inicio automático del servicio.

La forma más rápida es agregando un archivo al directorio init.d, entonces creamos un archivo llamado mosquitto en /etc/init.d

```
#!/bin/bash

case "$1" in
    start)
        echo "Starting Mosquitto MQTT Broker"
        touch /var/log/mosquitto.log
        chown mosquitto /var/log/mosquitto.log
        su - mosquitto -c "/usr/local/sbin/mosquitto -c /usr/local/etc/mosquitto.conf" & > /dev/null 2>/dev/null
        ;;
    stop)
        echo "Stopping Mosquitto MQTT Broker"
        killall mosquitto
        ;;
    *)
        echo "Usage: /etc/init.d/mosquitto start|stop"
        exit 1
        ;;
esac

exit 0
```

`su - mosquitto -c "/usr/local/sbin/mosquitto -c /usr/local/etc/mosquitto.conf" & > /dev/null 2>/dev/null`

Encuentra el nivel de permisos con `# runlevel`
Crea el link automático de inicio con:

```
cd /etc/rc2.d
ln -s /etc/init.d/mosquitto S97mosquitto
```

Paso 3: instala freeboard.io dashboard

Ahora solo necesitamos un medio de ver la data en la web... ya casi terminamos..

```
cd ~/IoTServer
git clone https://github.com/Freeboard/freeboard.git
```

Se crea el symlink de apache apuntado hacia donde se instala el freeboard.

```
sudo -s
cd /var/www
ln -s /home/pcortex/IoTServer/freeboard iot
```

Antes de encontrar el link en <http://myserveraddres/iot>.

Debes otorgar permisos editando el archivo en /etc/apache2/sites-enabled/000-default.conf

Y cambiar la configuración por:

```
ServerAdmin webmaster@localhost
DocumentRoot /var/www
<Directory /var/www/>
Options FollowSymLinks MultiViews Includes
AllowOverride None
Order allow,deny
allow from all
</Directory>
```

Y reiniciamos el apache!

```
sudo service apache2 restart
```

Primero bajamos la versión de desarrollo de Mqtt.js

<http://git.eclipse.org/c/paho/org.eclipse.paho.mqtt/javascript.git/plain/src/mqttws31.js?h=development>)

```
cd ~/IoTServer/freeboard/plugins
mkdir mqtt
cd mqtt
wget --output-document mqttws31.js http://git.eclipse.org/c/paho/org.eclipse.paho.mqtt/javascript.git/plain/src/mqttws31.js
```

Y luego bajamos el plug in par freeboard de MQTT:

```
cd ~/IoTServer
git clone https://github.com/alsm/freeboard-mqtt
cd freeboard/plugins/mqtt
cp ~/IoTServer/freeboard-mqtt/paho.mqtt.plugin.js .
```

Ahora solo editamos el archivo paho.mqtt.plugin.js y cambiamos la línea:

De:

```
"external_scripts" : [  
    "<full address of the paho mqtt javascript client>"  
],
```

A:

```
"external_scripts" : [  
    "plugins/mqtt/mqttws31.js"  
],
```

Por ultimo modificamos el archive **index.html** en el directorio donde instalamos freeboard de:

```
<scripttype="text/javascript">  
    head.js("js/freeboard/plugins.min.js",  
        // *** Load more plugins here ***  
        function(){
```

A:

```
<scripttype="text/javascript">  
    head.js("js/freeboard/plugins.min.js",  
        "plugins/mqtt/paho.mqtt.plugin.js",  
        // *** Load more plugins here ***  
        function(){
```

Por ultimo comprobamos que mosquito este corriendo los WEBSOCKETS con el commando

nohup mosquitto -c /etc/mosquitto/mosquitto.conf &

y verificamos que el Protocolo este corriendo:

```
netstat -nap | grep 9001
```

tcp	0	0 0.0.0.0:9001	0.0.0.0:*	LISTEN	18973/mosquitto
-----	---	----------------	-----------	--------	-----------------

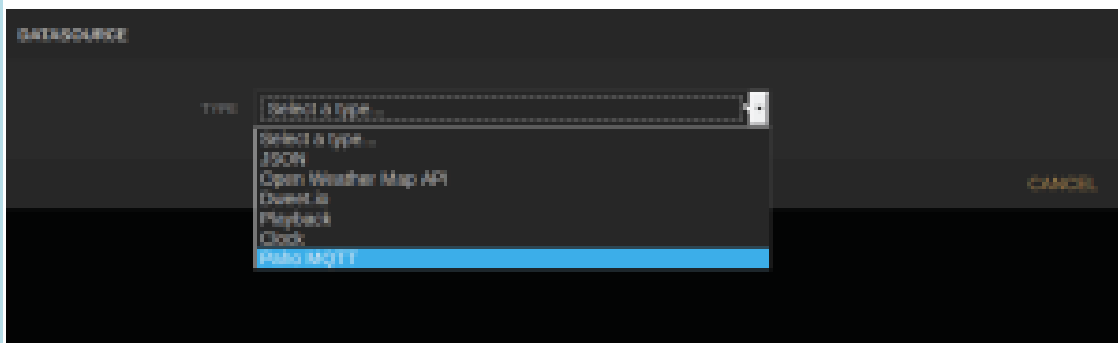
Con eso habremos logrado ya configurar nuestro servidor : así que solo queda programar nuestro Arduino y probar el envío de datos.

Paso 4. Cargar el código al arduino.

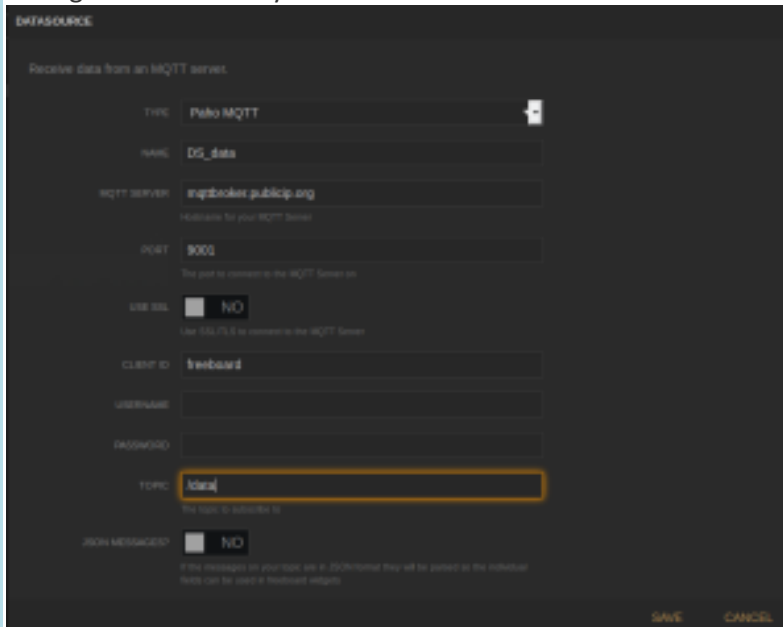
Bajar el Zip con el Código para arduino de: <https://github.com/EdwinKestler/IoTarduinoDayGT/>
Compilarlo y cargarlo al ESP desde el IDE de Arduino.

Paso 5. Configurar nuestros paneles de Freeboard.

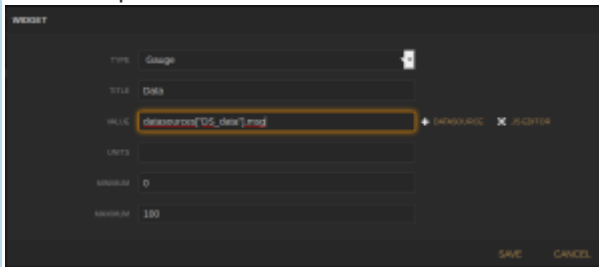
Seleccione la Fuente de datos (PahoMQTT)



Configurar el servidor y la fuente de dato:

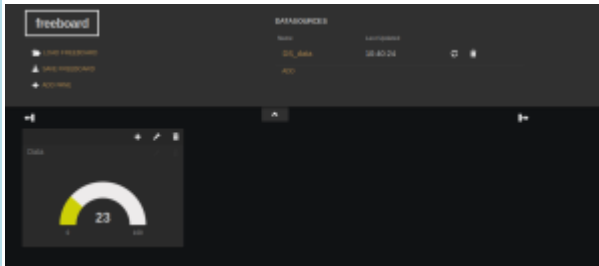


Crear los paneles.



The screenshot shows the 'WIDGET' configuration window in a dark-themed interface. The 'TYPE' is set to 'Gauge'. The 'TITLE' is 'Data'. The 'URL' field contains the text 'dataources["DS_data"].msg' and is highlighted with a yellow border. To the right of the URL field are two buttons: '+ ADD SOURCE' and 'X REMOVE'. Below the URL field are fields for 'UNITS', 'MINIMUM', and 'MAXIMUM'. The 'MINIMUM' field is set to '0' and the 'MAXIMUM' field is set to '100'. At the bottom right are 'SAVE' and 'CANCEL' buttons.

Y VER La Data Fluir.....



Felicitades han creado su primer openframework de IoT

IOT Arduino Day Guatemala 2016!