UNIVERSIDAD NACIONAL DE RÍO CUARTO

FACULTAD DE INGENIERÍA

Informe de Practica Profesional Supervisada

Despliegue IOT para plataforma educativa Edukit10

Lugar de realización: Cooperativa de Trabajo Informática y Telecomunicaciones 10 Ltda. (Belgrano 45, Río Cuarto, Córdoba)

Estudiante: Joaquín MANCHADO

Tutor por Empresa: Genaro PENNONE

Tutor por Universidad: Daniel ANUNZIATA

Período de realización: 02/01/2019 al 29/03/2019

Fecha de presentación: 27 de junio de 2019

Índice de contenidos

Resumen

El presente informe corresponde a la Práctica Profesional Supervisada (PPS) realizada en la *Cooperativa de Trabajo Informática y Telecomunicaciones 10 Ltda.* ubicada en calle Belgrano 45 de nuestra localidad. Partiendo de dos conceptos como lo es IoT (Internet of Things) y la educación en programación, surge la idea que moviliza esta PPS. El objetivo general de la misma se enfoca en diseñar e implementar una arquitectura de software en el marco de Internet de las Cosas, utilizando la plataforma educativa Edukit10, la cual está orientada a la enseñanza de la programación de sistemas embebidos facilitando el acercamiento de niños y jóvenes a las nuevas disciplinas técnicas. El hardware está desarrollado por la cooperativa, basada en Arduino. Se busca integrar una interfaz de comunicación inalámbrica que permita enviar datos de sensores a un servidor utilizando protocolos orientados a mensajes. Dichos datos se almacenarán en una base de datos para luego recuperarlos, procesarlos y visualizarlos a través de una aplicación web.

Palabras claves: Internet de las Cosas, Educación, WiFi, servidor, Raspberry Pi, Node-RED, InfluxDB, Grafana, invernadero, aplicación web

Introducción

El **internet de las cosas** (en inglés, Internet of Things, abreviado IoT) es un concepto que se refiere a una interconexión digital de objetos cotidianos con internet. Este concepto fue usado por primera vez por Kevin Ashton (creador de estándares de tecnologías como RFID).

Dar una definición de IoT no es algo sencillo. Desde un punto de vista tecnológico, The Global Standards Initiative on Internet of Things (IoT-GSI), de ITU, define IoT como: *"Una infraestructura global en el marco de la sociedad de la información que provee servicios a través de la conexión de elementos físicos o virtuales, basada en tecnologías de la información y la comunicación, tanto existentes como en desarrollo".*

Hoy por hoy, el término IoT se usa con una denotación de conexión avanzada de dispositivos, sistemas y servicios que va más allá del tradicional M2M (máquina a máquina) y abarca una amplia variedad de protocolos, dominios y aplicaciones. Es un tema emergente de importancia técnica, social y económica, y es uno de los pilares del "Internet del Futuro" junto a los conceptos involucrados a Internet de las Personas, Internet de los Contenidos y el Conocimiento e Internet de los Servicios.

En este momento se están combinando productos de consumo, bienes duraderos, automóviles, componentes industriales, componentes de servicios públicos, sensores y otros objetos de uso cotidiano con conectividad a Internet con potentes capacidades de análisis de datos que prometen transformar el modo en que trabajamos, vivimos, jugamos, enseñamos y aprendemos.

Dentro de los objetivos propuestos y alcanzados se encuentran la implementación de una arquitectura de software en la nube que permita almacenar, procesar y visualizar datos. Estos datos serán enviados agregando una interfaz de comunicación WiFi 802.11, con una placa WeMos D1 mini basada en el SoC ESP2866 a la placa Edukit10 desarrollada por la cooperativa, la cual tiene sensores integrados y la posibilidad de agregar otros externos. Para esto se seleccionó y utilizó el protocolo de comunicación orientados a mensajes MQTT. También se analizó y seleccionó el software necesario para instalar en el servidor en una Raspberry Pi, como ser base de datos de serie de tiempo, con InfluxDB; una herramienta que sirva para análisis y visualización de datos en gráficas, con Grafana. Por ultimo se desarrolló una aplicación web que se encargue de procesar y presentar la información, todo gestionado a través de la plataforma Node-RED. Los lenguajes utilizados para la programación de cada herramienta y servicio son JavaScript, C++, HTML . Se usaron metodologías ágiles (scrum) para el desarrollo y se documentó todo el proceso para respaldo de la cooperativa.

Se realizó una integración y puesta en marcha de lo antes mencionado en un prototipo de invernadero inteligente a modo de ejemplo, liberando la documentación para la implementación y posterior uso de la comunidad.

Descripción de la Empresa

- Nombre: Cooperativa de Trabajo Informática y Telecomunicaciones 10 Ltda.
- Dirección: Belgrano 45
- Tel./Fax: 3584823284
- E-mail: contacto@it10coop.com.ar
- Área de la empresa donde se desarrolla la práctica: IoT
- Permanencia: 2 de enero de 2019 al 29 de marzo de 2019.
- Jornada laboral: 5hs diarias de lunes a viernes. 6hs diarias solo los dias 2 al 8 de enero.
- Descripción: IT10 es una empresa cooperativa de Informática y Telecomunicaciones integrada por jóvenes profesionales de diversas áreas (ingeniería, informática y la comunicación) que brindan soluciones tecnológicas innovadoras de manera flexible y eficaz. Trabajan con tecnologías adaptables a cada demanda para el desarrollo de productos y servicios (software y hardware), eventos tecnológicos, capacitaciones y consultoría a profesionales, empresas e instituciones locales y regionales. En lo últimos años, la organización se ha perfilado hacia la investigación y el desarrollo de productos y servicios principalmente en el campo de la tecnología educativa y el Internet de las Cosas (IOT), siempre apostando por “la innovación con sentido social”, desde la autogestión y el trabajo en red con otros actores para potenciar juntos un ecosistema productivo en los nuevos escenarios de complejidades y entramados.
- Productos y Servicios:
    - Aplicaciones Android
        - Link: http://play.google.com/store/apps/developer?id=Cooperativa+de+Trabajo+IT10+Ltda
    - Sistemas web
    - Páginas Web
    - IoT – Internet de las cosas
    - Impresión 3d
    - Robótica Educativa
        - Edukit10
- Organigrama: Al ser una cooperativa de trabajo no dispone precisamente de un organigrama. La jerarquía dentro de la empresa es horizontal, siendo todos socios los trabajadores de la misma.

Desarrollo

Búsqueda, revisión y selección de Tecnologías

Se determinó incorporar una interface de comunicación inalámbrica 802.11 para integrar a la Edukit, de manera que el módulo provea facilidades en cuanto a portabilidad. Este es el punto de partida y primer condicionante para la selección del resto del hardware, software y herramientas a utilizar.

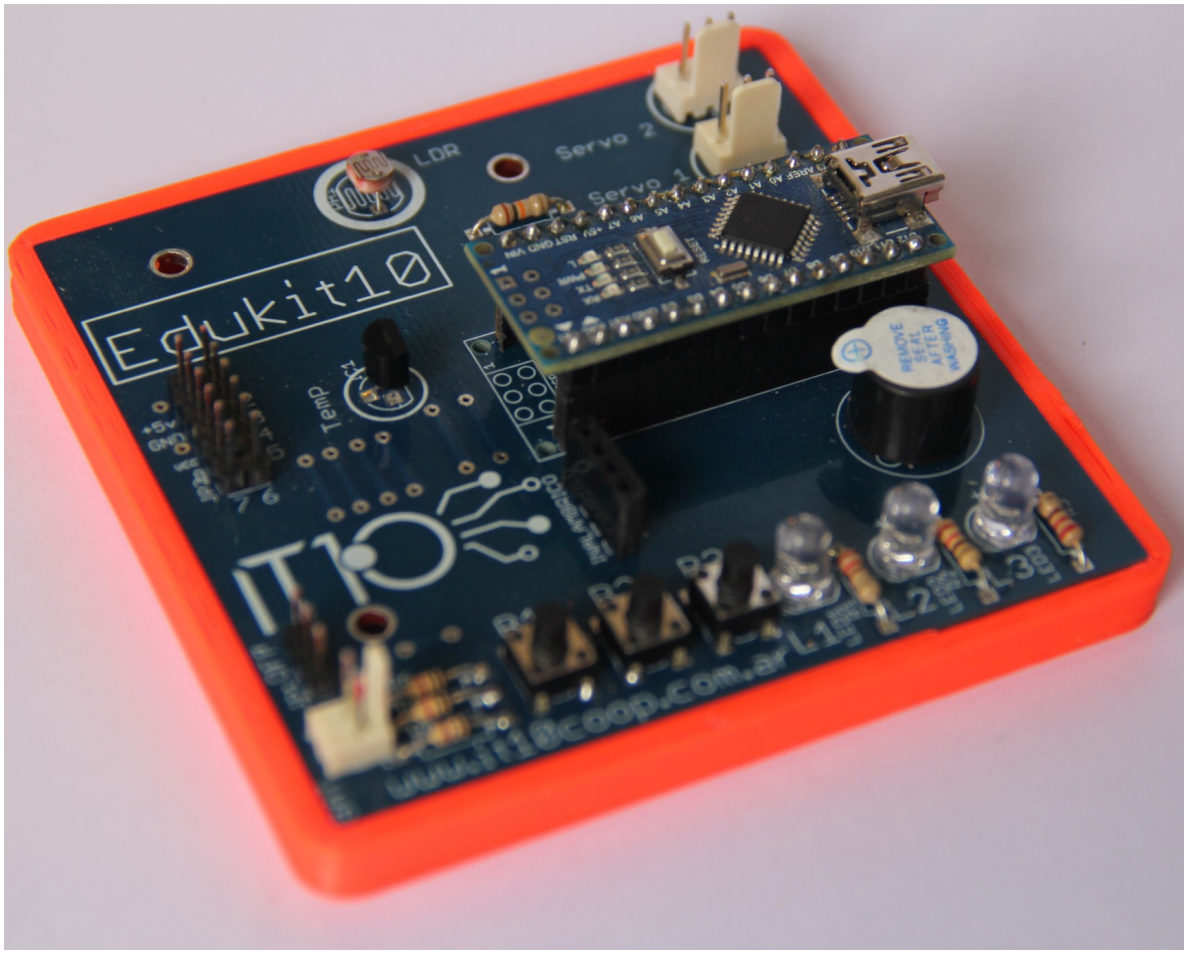
Para determinar los protocolos y selección de hardware se describe el proceso en las siguientes etapas:

- Edukit
    - Selección interfaz de comunicación
        - Hardware
        - Firmware
- Protocolo de mensajería
- Gateway / server
    - Hardware
    - Software

Se detalla a continuación cada etapa.

Edukit

Es un proyecto de hardware libre basado en Arduino, desarrollado por la Cooperativa IT10, para la enseñanza de robótica. Es una tecnología que permite acercar de una manera sencilla los fundamentos de la robótica y aquellos conceptos técnicos que se vuelven complejos para trabajarlos de la manera tradicional. Se compone de una serie de elementos electrónicos que trabajan en conjunto y permiten investigar y diseñar pequeños robots pedagógicos de forma sencilla, reciclando componentes. El software utilizado para la programación es libre y multiplataforma.

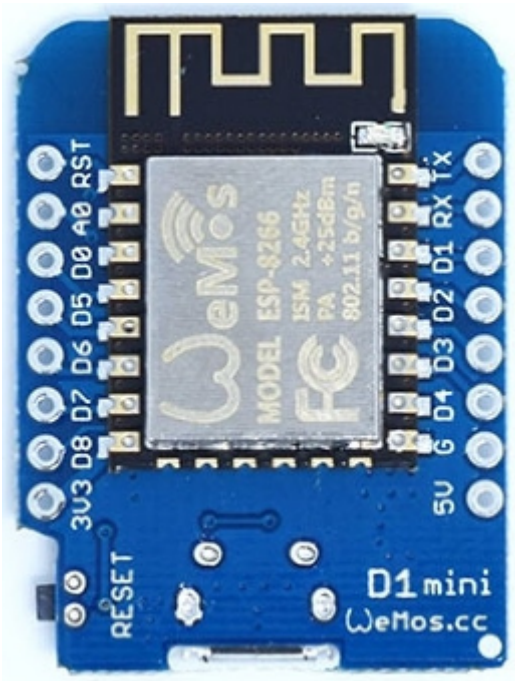


El objetivo es adicionar un módulo con comunicación inalámbrica que se vincula´ra con la Edukit a través de una conexión serial por software. No se toma en cuenta la conexión serial por hardware porque se pretende utilizarla para monitorear el funcionamiento del dispositivo. Además se requiere que el nuevo módulo sea de bajo costo y funcione con gran estabilidad por muchas horas.

En función a todo esto se elije la interfaz de comunicación. En el Anexo I se incluye el diagrama pinout y esquemático de la Edukit10.

Interfaz de comunicación inalámbrica 802.11

El nodo es el vínculo entre la Edukit y el servidor que procesa los datos. De un gran abanico de posibilidades, la cooperativa tiene a disposición alternativas que en común poseen el microchip **ESP8266**. La razón de ello es que son económicos, estables, de bajo consumo, poseen varios buses de comunicación, su procesador es sumamente rápido (en términos relativos), entre otras cosas. El elegido es el **WeMos D1 mini**, que cumple con los criterios antes mencionados.



En el Anexo II se encuentra el esquemático y características de la WeMos D1 mini. En el Anexo XII está la hoja de datos del microchip ESP8266.

Sin embargo presenta algunos problemas a la hora de su uso, más precisamente por el firmware que trae de fábrica (un firmware no es más que el software de bajo nivel, la lógica que controla los circuitos electrónicos). Los inconvenientes que se presentan son:

- Firmware AT (Ai-Thinker) muy limitado. Malas referencias de la comunidad.
- Genera fallos en la comunicación con **SoftwareSerial**. Indispensable para el proyecto. No garantiza estabilidad.
- Comandos AT complicados, que muchas veces fallan y se desperdicia tiempo en la depuración de las fallas.
- Se centra totalmente en los aspectos de conexión y desconexión, desaprovechando el potencial de la placa, como por ejemplo el uso de los pines GPIO, buses de comunicación, etc.

Es por todo esto, y gracias a experiencias previas de los integrantes de la cooperativa que se indica como primera instancia investigar sobre alternativas a este firmware.

Firmware

Indagando en la web, se da con tres firmware interesantes para analizar, gracias al trabajo de **Lazar Obradovic**, usuario de GitHub, aficionado al hacking y al mundo IoT. Estos firmware son:

- Tasmota
- ESPurna
- ESPEasy

Se detalla a continuación cada uno.

Tasmota

Mas bien llamado Sonoff-Tasmota, en representación de Theo-Arends-Sonoff-MQTT-OTA, se produjo como una evolución de proyectos de retoques anteriores de Theo Arends.

En un principio se creó para reemplazar el firmware original de los primeros dispositivos Iteads Sonoff, pero creció hasta convertirse en un proyecto que admite muchos dispositivos basados en ESP8266.

El proyecto se actualiza casi a diario, exclusivamente por el propio Theo, revisando algunas de las contribuciones de la comunidad de desarrolladores.

La comunidad es muy activa y tiene una combinación saludable de desarrolladores y usuarios "regulares" que aseguran que el firmware sea generalmente utilizable.

Se adjunta información, datos de configuración y pruebas realizadas con Tasmota en WeMos D1 mini en el Anexo III.

ESPurna

Creado por Xose Pérez en 2016, como un proyecto para proporcionar un firmware independiente del dispositivo para las placas ESP8266. Comenzó con la placa WeMos D1 Mini (siendo muy genérico) y agregó soporte para muchas otras placas fabricadas, incluyendo Sonoff.

Falta una mayor adopción social, a pesar del blog muy informativo que dirige Xose. El proyecto se actualiza pocas veces, exclusivamente por su autor, en base a las discusiones o pequeñas contribuciones de la comunidad.

La comunidad es más pequeña que la de Tasmota y está principalmente orientada a desarrolladores con el ingenio suficiente para que coincida con el conocimiento y las habilidades de Xose.

Se adjunta información, datos de configuración y pruebas realizadas con ESPurna en WeMos D1 mini en el Anexo IV.

ESPEasy

ESPEASY es el firmware alternativo más antiguo que existe. Se creó en diciembre de 2015 por un equipo holandés llamado “Lets Control It”. Muy flexible y capaz de soportar cualquier combinación de sensores / actuadores, pero requiere un poco más de configuración para comenzar.

El proyecto se actualiza muchas veces, en su mayoría por pocos desarrolladores centrales de Lets Control It, pero también combinan el código de otros colaboradores.

El tamaño de la comunidad es comparable al de Tasmota, con un balance un poco mejor entre los desarrolladores y los usuarios "regulares".

Funcionalidades en común

- Son de código abierto.
- Compatibilidad con casi todas las tarjetas y módulos de Itead Sonoff, además de las placas genéricas como NodeMCU o WeMos D1.
- Amplio soporte para muchos sensores y actuadores adicionales.
- Todos admiten varios modos de conexión WiFi, incluido el modo AP (hotspot) o STA (cliente), con múltiples configuraciones de SSID y escaneo de red Wifi.
- Compatibilidad con MQTT, tanto para sensores / actuadores, como para la configuración propia. Algunos incluso ofrecen un diseño de *topic* específico.
- Soportan actualización por aire -Over-The-Air- (OTA), copia de seguridad y restauración de la configuración.
- Todos soportan sincronización de tiempo vía NTP.

Áreas comunes de mejora

Todas las opciones de firmware tienen el mismo conjunto común de problemas y áreas potenciales de mejora:

Comunicación segura / TLS: Ninguno lo soporta. Tasmota y ESPurna empezaron a admitir TLS para MQTT, pero eso aborda solo una parte del problema y deja la interfaz de usuario web y otras interfaces sin cifrar y/o autenticar.

Documentación: Ninguno tiene una documentación bien estructurada. Tasmota es mejor que los otros, pero podría ser mucho mejor y resolver cuestiones más simples con una buena documentación.

IPv6: Todas se basan en IPv4, ninguna soporta IPv6. A futuro será un inconveniente. IPv6 es más un problema de la estructura Arduino / ESP8266 que un problema de firmware individual, pero sigue siendo una cuestión a resolver.

Comparación y decisión

Tasmota es "lo suficientemente buena" para el uso diario. Más fácil de utilizar por programadores no expertos. Permite un inicio fácil y resultados rápidos, tiene un poderoso conjunto de características y buen soporte para diferentes sensores, pero también viene con algunas cosas extrañas a las que solo hay que acostumbrarse (como sintaxis de comandos, ejecuciones de reglas, entre otros).

ESPurna es muy estructurada y ofrece una interfaz limpia y precisa, pero aún no se popularizó demasiado. Aún así, parece que tiene una mejor implementación interna del cliente MQTT y servidor HTTP. ESPurna podría carecer de algunas de las características de lujo que tiene Tasmota.

ESPEasy es demasiado desestructurado para la producción en masa (o uso diario). Si uno toma como algo positivo la libertad y capacidades de configuración, es una buena opción. Tiene opciones de configuración de run-time (tiempo de ejecución) muy abiertas y potentes. Está más avanzado en lo que respecta a la toma de decisiones locales, lo que reduce potencialmente la necesidad de un agente MQTT y una lógica de automatización externa.

Se descarta de entrada ESPEasy, a pesar de sus bondades, ya que su configuración es en tiempo de ejecución, algo inviable para el proyecto. En detalle la comparación entre Tasmota y Espurna es la siguiente:

	Tasmota	**ESPurna**
Comunidad	Bastante grande. Aportes de muchos usuarios y colaboradores.	Chica, va creciendo poco a poco, pero está muy lejos de Tasmota.
Actualizaciones	Mucho más frecuentes, debido a su extendida comunidad.	Pocas, depende casi exclusivamente de su desarrollador, el cual no dedica su tiempo 100% al mejoramiento del firmware.
Protocolos	MQTT, HTTP	MQTT, HTTP
Base de datos	No hay integración	Integración directa con InfluxDB
Encriptación/Seguridad	No	No
Arduino Nano	La vinculación se puede hacer por medio de comunicación serial por software	No hay indicios de poder usar comunicación serial aparte de la de hardware

Se concluye en que **Tasmota** es el firmware mas competente para suplantar al de fabrica de la placa WeMos D1 mini, sobre todo por su gran comunidad y posibilidad de comunicación serial por software. Además se realizaron pruebas con Tasmota y ESPurna, detalladas en los Anexos III y IV, respectivamente.

Protocolo de mensajería

Si bien existen muchos protocolos de comunicación orientados a mensajes, como XMPP. CoAP, AMQP, etc, de acuerdo a la elección del firmware no queda mas opción que optar por MQTT.

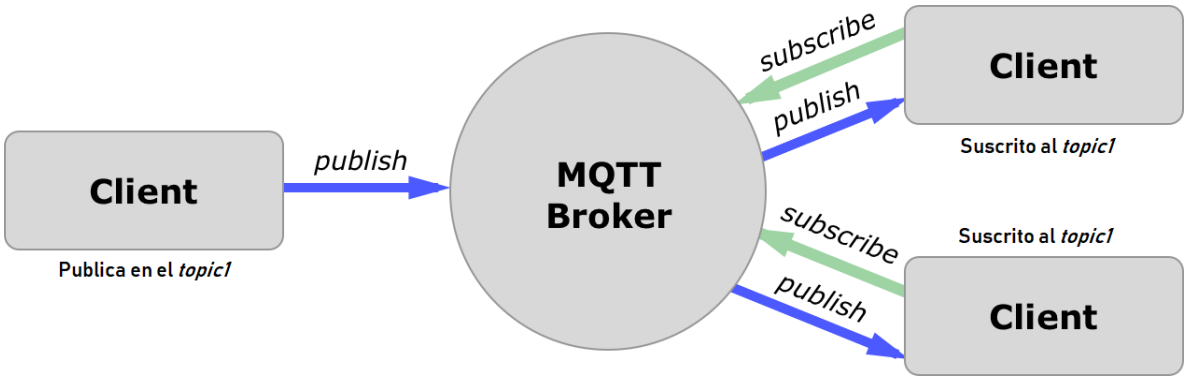
MQTT

Message Queue Telemetry Transport es un protocolo de mensajería basado en publicación-suscripción. Se fundamenta en el principio cliente/servidor. Destaca por ser ligero y sencillo de implementar. Funciona sobre el modelo TCP/IP.

Andy Stanford-Clark de IBM y Arlen Nipper de Cirrus Link fueron los autores de la primera versión del protocolo en 1999.

Los principios de diseño son minimizar el ancho de banda de la red y los requisitos de recursos del dispositivo, al mismo tiempo que intentan garantizar la confiabilidad y cierto grado de seguridad de la entrega.

La arquitectura de MQTT sigue una topología de estrella, con un nodo central como "broker", que es el que recopila los datos que los publishers (los objetos comunicantes) le transmiten. Un broker tiene una capacidad de hasta 10000 clientes.



La comunicación se basa en canales llamados cada uno como *topic* (tema) y puede ser de uno a uno, o de uno a muchos. Los **publishers** envían mensajes al topic y los **subscribers** pueden leerlos. Los mensajes enviados por los objetos comunicantes pueden ser de todo tipo pero no pueden superar los 256 Mb.

Un "topic" se representa mediante una cadena y tiene una estructura jerárquica. Cada jerarquía se separa con '/'. Por ejemplo, `edificio1/planta5/sala1/raspberry2/temperatura` o `/edificio3/planta0/sala3/arduino4/ruido`.

De esta forma un nodo puede subscribirse a un "topic" concreto ("edificio1/planta2/sala0/arduino0/temperatura") o a varios ("edificio1/planta2/#").

Comparativa

Haciendo una comparación con otros protocolos de mensajería existentes, se puede ver que tiene muy buenas prestaciones a fines de esta practica.

Protocol	Transport	Messaging	2G,3G,4G (1000's)	LowPower and Lossy (1000's)	Compute Resources	Security	Success Stories	Arch
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium - Optional	Utility feld area ntwks	Tree
Continua HDP	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	None	Medical	Star
DDS	UDP	Pub/Subsrb Rqst/Rspnse	Fair	Poor	100Ks/RAM Flash +++	High-Optional	Military	Bus
DPWS	TCP		Good	Fair	100Ks/RAM Flash ++	High-Optional	Web Servers	Client Server
HTTP/ REST	TCP	Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	Low-Optional	Smart Energy Phase 2	Client Server
MQTT	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium - Optional	IoT Msging	Tree
SNMP	UDP	Rqst/Response	Excellent	Fair	10Ks/RAM Flash	High-Optional	Network Monitoring	Client-Server
UPnP		Pub/Subscrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	None	Consumer	P2P Client Server
XMPP	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Fair	10Ks/RAM Flash	High-Manditory	Rmt Mgmt White Gds	Client Server
ZeroMQ	UDP	Pub/Subscrb Rqst/Rspnse	Fair	Fair	10Ks/RAM Flash	High-Optional	CERN	P2P

Gateway / server

Teniendo en cuenta el criterio de portabilidad se decide un servidor que funcione de Gateway también usando una computadora de tamaño reducido.

La opción mas interesante es utilizar una Raspberry Pi, que además de ser de bajo costo y tener potencia adecuada para los propósitos de esta practica, ya la tenían a disposición en la cooperativa.

Raspberry Pi B+

Raspberry Pi es un ordenador de placa reducida, u ordenador de placa simple (SBC) de bajo coste desarrollado en el Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de informática en las escuelas.

Aunque no se indica expresamente si es hardware libre es un producto con propiedad registrada, manteniendo el control de la plataforma, pero permitiendo su uso libre tanto a nivel educativo como particular.

El software sí es de código abierto, siendo su sistema operativo oficial una versión adaptada de Debian, denominada *Raspbian*, y es el utilizado en esta práctica, aunque permite usar otros. En todas sus versiones incluye un procesador Broadcom, una memoria RAM, una GPU, puertos USB, HDMI, Ethernet, 40 pines GPIO y un conector para cámara. Ninguna de sus ediciones incluye memoria.

En el Anexo XII se encuentra la hoja de datos de la Raspberry Pi 3B+. En el Anexo V se encuentra el proceso de instalación y configuración del sistema operativo Raspbian.

En esta mini PC se debe alojar una plataforma que pueda vincular una base de datos y un servicio que permita la visualización e interpretación de información de sensores de la Edukit.

Selección y prueba de sensores

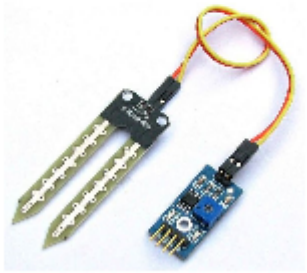
Los sensores se seleccionaron con la intención de poder crear un escenario de prueba de un invernadero inteligente. Además los mismos ya los tenían a disposición en la cooperativa.

Son cuatro sensores en total. Estos son:

- Sensor de intensidad lumínica
- Sensor de temperatura
- Sensor de gases
- Sensor de humedad de suelos

LDR

LM35



YL-69

MQ - 9

Los dos primeros están incluidos en la Edukit. Los últimos se escogieron de una variedad disponible en inventario de la cooperativa. No se repara demasiado en la calibración de cada sensor (será tarea posterior de la cooperativa), sino más bien en la correcta conversión de sus valores analógicos a digitales.

Se detallan brevemente a continuación. En el Anexo XII se encuentran las hojas de datos de cada uno.

Sensor de intensidad lumínica LDR

Un fotorresistor o fotorresistencia es un componente electrónico cuya resistencia disminuye con el aumento de intensidad de luz incidente. Sus siglas, **LDR**, se originan de su nombre en inglés *light-dependent resistor*. Su cuerpo está formado por una célula fotorreceptora y dos patillas. Este sensor viene integrado a la Edukit.

Sensor de temperatura LM35

El LM35 es un sensor de temperatura con una precisión calibrada de 1 °C. Su rango de medición abarca desde -55 °C hasta 150 °C. La salida es lineal y cada grado Celsius equivale a 10 mV, por lo tanto:

- 150 °C = 1500 mV
- -55 °C = -550 mV

Opera de 4v a 30v. Viene integrado en la Edukit.

Sensor de gases MQ-9

Este sensor de gas puede detectar monóxido de carbono (CO) y también gases inflamables, como Gas Natural y Butano. Tiene alta sensibilidad (ajustable mediante potenciómetro) y un tiempo de respuesta rápido. El monóxido de carbono se detecta a temperatura de calentamiento baja (1.4V) mediante ciclos de temperatura elevada y baja. La conductividad eléctrica incrementa con la concentración de monóxido de carbono en el aire. El sensor tiene 6 pines, 4 para la medición de la señal y 2 para el calentador, sin embargo el módulo dispone directamente de un pin analógico, otro digital, además de alimentación y masa.

Sensor de humedad de suelos YL-69

Este sensor puede medir la cantidad de humedad presente en el suelo que lo rodea empleando dos electrodos que pasan corriente a través del suelo, y lee la resistencia. Mayor presencia de agua hace que la tierra conduzca electricidad más fácil (Menor resistencia), mientras que un suelo seco es un conductor pobre de la electricidad (Mayor resistencia).

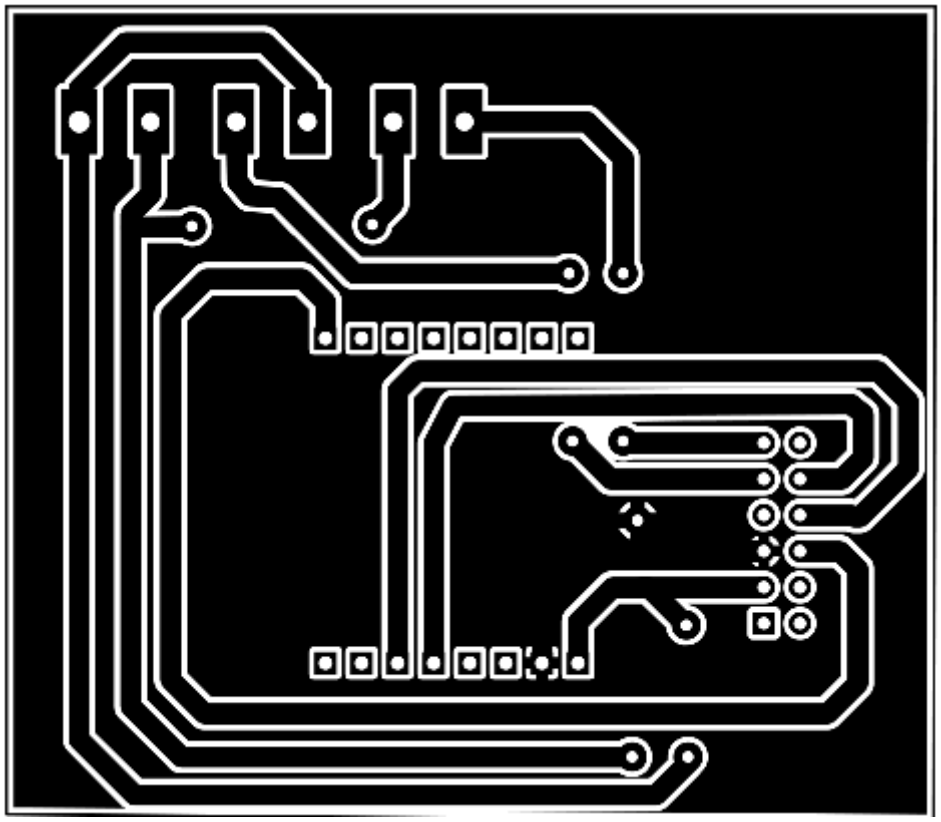
PCB Shield

Una **“placa de circuito impreso”** (del inglés: *Printed Circuit Board*, PCB), es una superficie constituida por caminos, pistas o buses de material conductor laminadas sobre una base no conductora. El circuito impreso se utiliza para conectar eléctricamente a través de las pistas conductoras, y sostener mecánicamente, por medio de la base, un conjunto de componentes electrónicos. Las pistas son generalmente de cobre, mientras que la base se fabrica generalmente de resinas de fibra de vidrio reforzada, cerámica, plástico, teflón o polímeros como la baquelita.

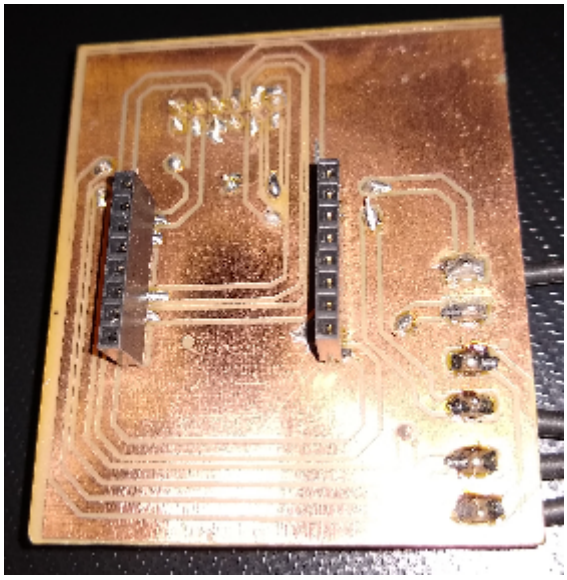
La producción de la PCB se realizo sobre una placa de simple capa de cobre, y se diseñó usando el Software online EasyEDA.

EasyEDA es un software alojado en la nube, disponible a través de una página web, totalmente gratis. Permite diseñar tanto diagramas electrónicos como PCB y simulación de circuitos. Cuenta con una enorme librería de componentes, incluyendo las placas de Arduino, sensores y Raspberry. Está disponible a través de la página web https://easyeda.com/.

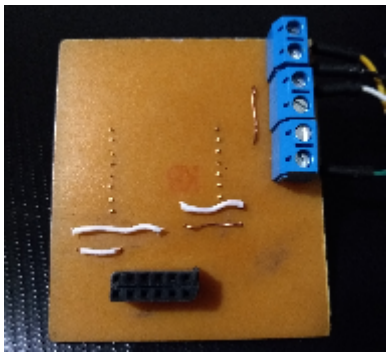
El diseño del PCB es el siguiente:



El resultado final es:



De este lado va montada la WeMos D1 mini. Del lado reverso:



Se puede ver un slot para vincular con la Edukit y borneras para conectar sensores analógicos.

El esquemático de la placa PCB se adjunta en el Anexo VI.

Servidor con base de datos y visualización

Luego de elegir el Gateway / server se seleccionó el siguiente software:

- Base de datos
- Visualización
- Plataforma

La base de datos almacena la información proveniente de los sensores y se hace de forma local. La alternativa mas interesante es usar una base de datos de serie de tiempo, muy útil para aplicaciones de IoT.

La visualización debe tomar los datos de la base de datos, interpretarlos, realizar operaciones con ellos de acuerdo al detalle de información necesario y ponerlos a disposición en un grafico que se actualice en tiempo real.

Por ultimo, la plataforma se elige de manera que pueda vincularse con una base de datos y una aplicación que interprete dichos datos para mostrarlos de forma gráfica y en tiempo real. Se pretende que ésta tenga la cualidad de poder programarse para concentrar todas las tareas de vinculación antes descriptas (recepción, inserción en base de datos y visualización) en la misma herramienta y no tener que derivarlas a otras aplicaciones.

A continuación se detalla cada parte con su criterio de selección.

Base de datos

Como se dijo anteriormente, de los diferentes tipos de base de datos se opta por usar una del tipo de series de tiempos. Una base de datos de series de tiempo (TSDB, por sus siglas en inglés) es una base de datos optimizada para datos con sello de tiempo (*timestamps*) o series temporales. Estos podrían ser métricas de un servidor, monitoreo del rendimiento de una aplicación, datos de red, datos de sensores, eventos, clics, intercambios en un mercado y muchos otros tipos de datos analíticos.

Una TSDB está optimizada para medir el cambio en el tiempo. Las propiedades que hacen que los datos de series de tiempo sean muy diferentes de otras cargas de trabajo de datos son la administración del ciclo de vida de los datos, el resumen y los análisis de gran alcance de muchos registros.

Entre las opciones disponibles en la web, se selecciona **InfluxDB**.

InfluxDB

InfluxDB, desarrollada por *influxdata*, es una base de datos basada en series de tiempo (time-series database). Maneja de forma eficiente estas series de datos, con miles de inserciones por segundo, haciendo cálculos, búsquedas, e incorporando información, como medias, máximos, etc, todo ello en tiempo real. Dentro de esta categoría, InfluxDB es una base de datos que supera a los esquemas SQL y NoSQL. Tiene una versión comunitaria de código abierto.

InfluxDB tiene un protocolo de línea para enviar datos de series de tiempo que adopta la siguiente forma: `measurement-name` `tag-set` `field-set` `timestamp`. El `measurement-name` y `tag-set` se mantienen en un índice invertido que hace que las búsquedas de series específicas sean muy rápidas.

Los timestamps en InfluxDB pueden tener una precisión de segundo, milisegundo, microsegundo o nanosegundo. En el disco, los datos se organizan en un formato de estilo encolumnado donde se establecen bloques de tiempo contiguos para la medición, el `tag-set` y el `field`. Por lo tanto, cada campo se organiza secuencialmente en el disco por bloques de tiempo, lo que hace que el cálculo de agregados en un solo campo sea una operación muy rápida. No hay límite en el número de etiquetas y campos que se pueden utilizar.

En el Anexo VII se encuentra el proceso de instalación y configuración de InfluxDB sobre Raspbian.

Criterio de selección

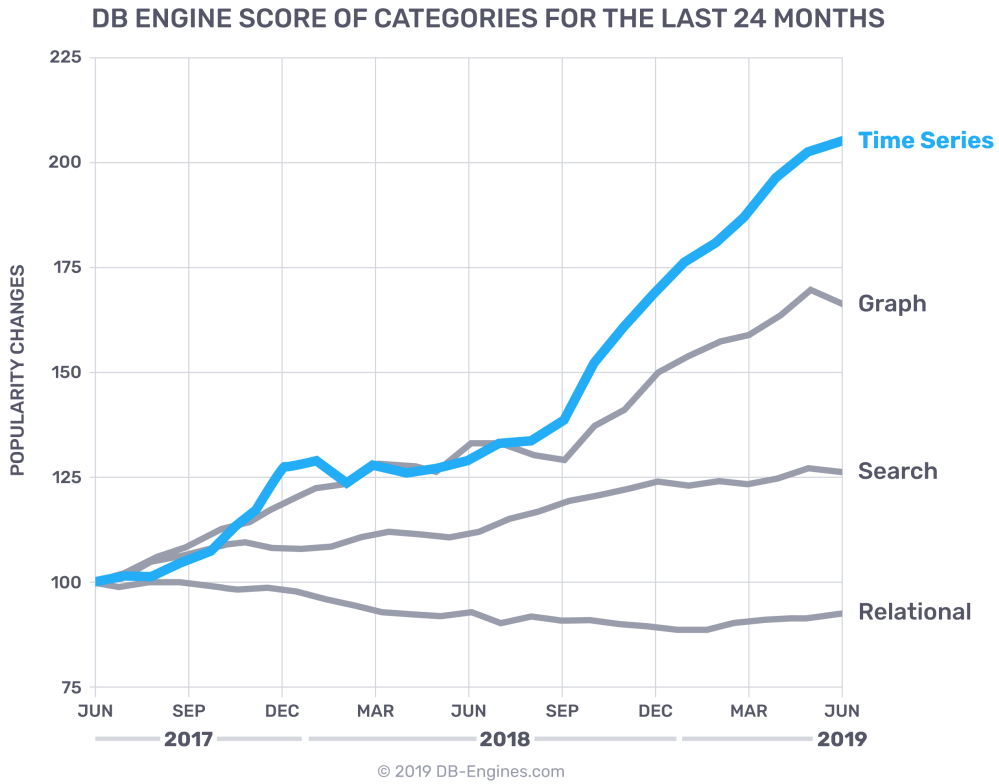
El sitio web DB-Engines, califica las bases de datos según la popularidad del motor de búsqueda, las menciones en las redes sociales, las ofertas de trabajo y el volumen de discusión técnica. Aquí están los resultados actuales:

RANK	DBMS	SCORE		
JUN 2019		JUN 2019	24 MOS ▲	12 MOS ▲
1	**InfluxDB**	**17.98**	**+9.78**	**+6.65**
2	Kdb+	5.80	+4.23	+2.78
3	Graphite	3.33	+1.33	+0.95
4	Prometheus	3.32	+2.71	+2.05
5	RRDtool	2.67	-0.35	0.00
6	OpenTSDB	2.24	+0.45	+0.68
7	Druid	1.78	+0.76	+0.65
8	TimescaleDB	1.11	+1.11	+1.06
9	KairosDB	0.50	-0.12	+0.09
10	eXtremeDB	0.41	+0.04	+0.13

Source: DB-Engines 23 Systems in Ranking, June 2019

Se puede leer toda la metodología para elaborar el ranking en https://db-engines.com/en/ranking_definition.

Además, según el mismo sitio, las bases de datos de series temporales son el segmento de más rápido crecimiento de la industria de bases de datos en el último año.



Es por todo esto, velocidad, eficiencia, popularidad y gran comunidad que se toma a **InfluxDB** como base de datos para almacenar las mediciones extraídas de los sensores en tiempo real.

Visualización

Es importante poder interpretar los datos de un proceso de manera rápida y concisa. Es por esto que se necesita de una aplicación que pueda producir graficas acordes a los datos extraídos de los sensores y en tiempo real. La herramienta elegida para dicha tarea es **Grafana**.

Grafana

Grafana es una herramienta de código abierto para el análisis y visualización de métricas. Está escrita en Lenguaje Go (creado por Google) y Node.js LTS. Además cuenta con una fuerte Interfaz de Programación de Aplicaciones (API); es una aplicación que ha venido escalando posiciones, con una comunidad entusiasta de más de 600 colaboradores bien integrados. Su código fuente está publicado en GitHub.

A partir de una serie de datos recolectados se puede obtener un panorama gráfico presentado en una forma elegante y amigable. Grafana permite consultar, visualizar, alertar y comprender métricas que pueden ser recolectadas y/o procesadas por aplicaciones de terceros. Puede recopilar de forma nativa datos de *Cloudwatch, Graphite, Elasticsearch, OpenTSDB, Prometheus, Hosted Metrics e InfluxDB*.

En el Anexo VIII se encuentra el proceso de instalación y configuración de Grafana sobre Raspbian.

Criterio de selección

Es una plataforma que permite consultar, visualizar, analizar y comprender todo tipo de métricas sin importar donde estén almacenados los datos. Todo esto desde una interfaz Web elegante y sencilla. Está respaldada por grandes organizaciones como Energy Weather, eBay o Stack Overflow, que la utilizan y es de código abierto. Tiene una gran cantidad de entradas y tiene buenos plugins de componentes para visualizar, además de que la edición de los gráficos es muy intuitiva.

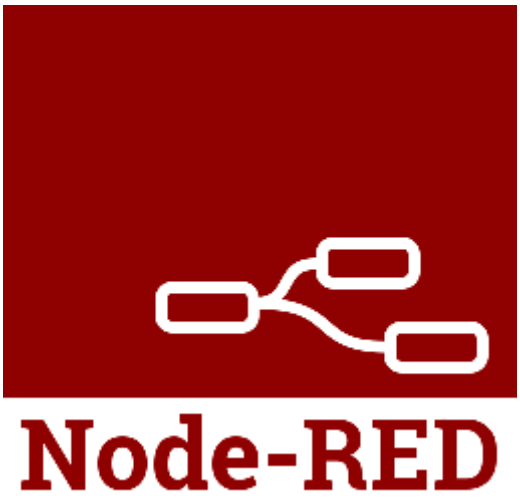
Resulta interesante poder exportar las graficas generadas, y administrar los rangos de tiempos a visualizar. Esto Grafana lo hace muy bien, con la posibilidad de que sea en tiempo real. Además se puede configurar y restringir acceso de usuarios en función de permisos, por Organizaciones.

Plataforma

Se hace una breve investigación sobre las plataformas en la nube, pero se termina optando por una local, ya que la intención es alojar el servidor en una mini PC Raspberry Pi. Se toma esta decisión en conjunto con la cooperativa ya que para ciertas funcionalidades las plataformas en la nube suelen tener algún costo y hubiera sido bastante desafortunado toparse con esta situación a mitad del proyecto, teniendo que migrar todo a otro servicio. Al mantener el servidor de manera local se puede programar a gusto. De todas formas, a futuro y con un buen análisis no se descarta por parte de la cooperativa hacer uso de una plataforma cloud.

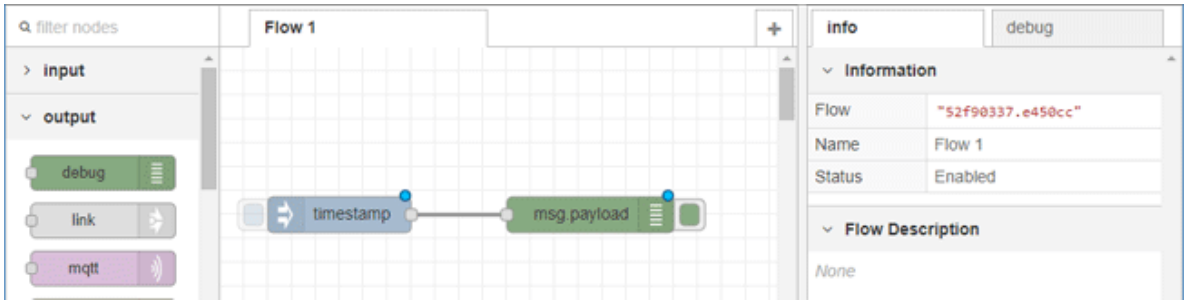
En cuanto a las plataformas locales nos encontramos con Node-RED y Home Assitant.

Node-RED es un motor de flujos con enfoque IoT, que permite definir gráficamente flujos de servicios, a través de protocolos estándares como REST, MQTT, Websocket, AMQ, etc, además de ofrecer integración con API's de terceros, tales como Twitter, Facebook, Yahoo!, etc.



Se trata de una herramienta visual muy ligera, programada en NodeJS y que puede ejecutarse tanto en dispositivos tan limitados como una Raspberry, como en plataformas complejas como IBM Bluemix, Azure IoT o Sofia2 Platform.

El editor de flujos de Node-RED consiste en una sencilla interfaz en HTML, accesible desde cualquier navegador, en la que arrastrando y conectando nodos entre sí, permite definir un flujo que ofrezca un servicio.



Viene con funcionalidades básicas pero es posible adicionarse otras descargando los paquetes necesarios. Entre ellos se encuentran:

- Dashboard: Incluye nodos que permiten realizar web dinámicas.
- Mqtt-Broker: Crea un broker local, necesario para vincular la Edukit con la base de datos. EL broker es llamado MOSCA.
- InfluxDB: Nodos para insertar, realizar consultas y eliminar datos sobre una base de datos InfluxDB.

En el Anexo IX se encuentra el proceso de instalación y configuración de Node-RED sobre Raspbian.

Criterio de selección

Ambas son atractivas para el fin del proyecto, con una buena documentación y experiencias en la red, pero por cuestiones de tiempo solo se llega a indagar más sobre Node-RED y no tanto sobre Home Assistant, otra plataforma similar a Node-RED. La decisión final es basar todo el proyecto en Node-RED.

Lenguajes de programación utilizados

Para la configuración de cada herramienta fue necesario utilizar los siguientes lenguajes de programación.

C++

Es un lenguaje de programación diseñado en 1979 por Bjarne Stroustrup. La intención de su creación fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. Además se añadieron facilidades de programación genérica, que se sumaron a los paradigmas de programación estructurada y programación orientada a objetos. Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Usado para programar la **Edukit** y el firmware **Tasmota.**

JavaScript

JavaScript (abreviado comúnmente **JS**) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y paginas web dinámicas.

Usado para programar los nodos y flujos de la plataforma **Node-RED**.

HTML

HTML es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet. Se trata de la siglas que corresponden a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto. Es el elemento de construcción más básico de una página web y se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad (como si lo haría JavaScript, por ejemplo).

Usado para programar los nodos Dashboard para la generación de la web dinámica de la plataforma **Node-RED**.

Pruebas de integración

Como prueba de integración se montó un escenario de invernadero inteligente. La idea general es tomar datos del ambiente a través de sensores con el módulo IoT y la Edukit; vía WiFi con una placa WeMos D1 mini se envían los datos usando protocolo MQTT a un broker MOSCA instalado en una Raspberry Pi. Los datos son leídos y guardados en una base de datos de serie temporal, InfluxDB. Por último, se busca obtener graficas en

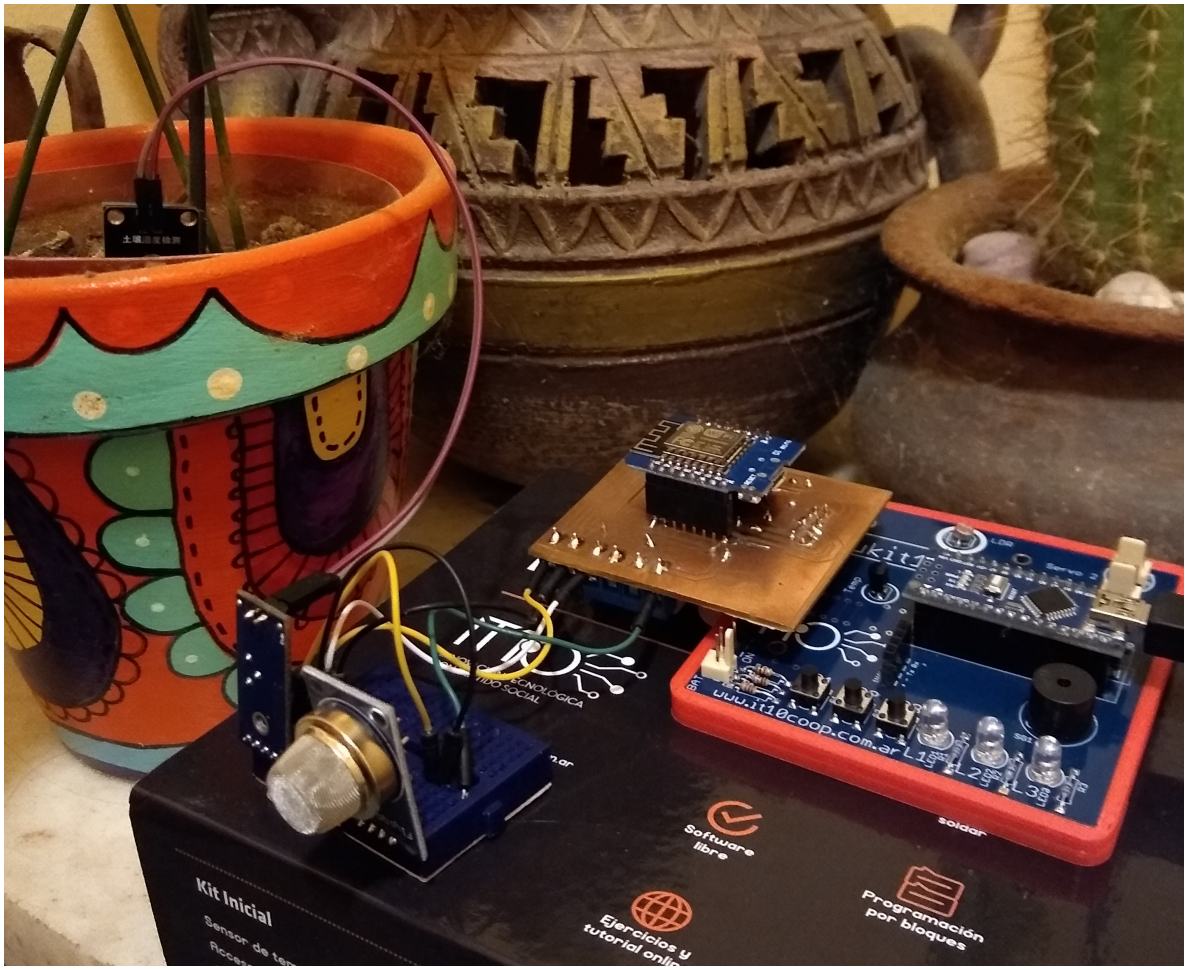
tiempo real, con Grafana, de lo que esta sucediendo en nuestro invernadero, presentando la información en una web local con Node-RED. Esto se hizo basado en un proyecto de Andreas Spiess, aficionado al mundo IoT.

La integración y vinculación de herramientas se hace con la plataforma Node-RED, que además de crear el broker MQTT e insertar y consultar la base de datos InfluxDB provee una web dinámica donde se mostraran los gráficas generados con Grafana.

El proceso de instalación, configuración de las herramientas y resultados de pruebas con estas tecnologías se puede encontrar en el Anexo X.

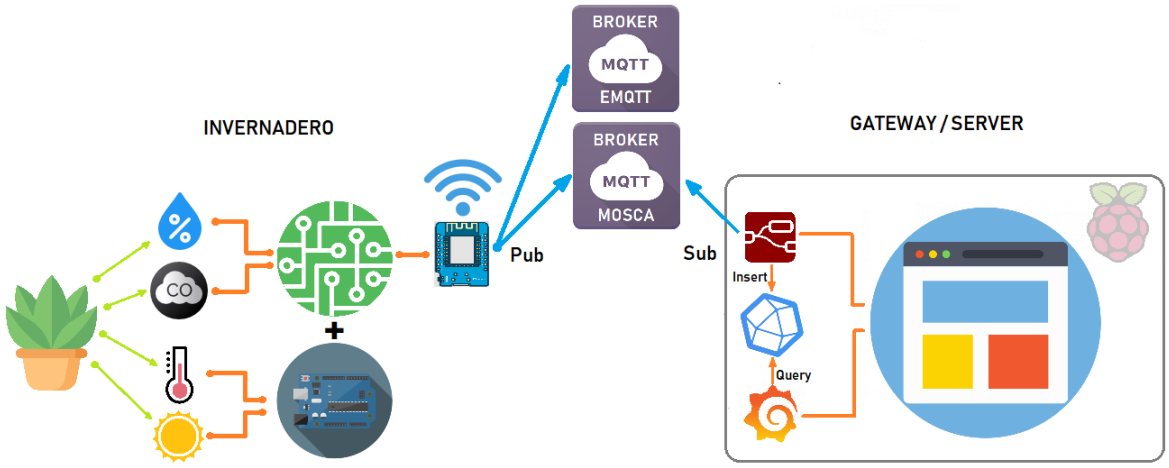
La documentación para realizar la implementación del invernadero inteligente se puede encontrar en el Anexo XI, Además está en un repositorio en Github, liberado a la comunidad.

El invernadero para pruebas se ve montado de esta manera:



Se pueden observar que el YL-69 y el MQ-9 esta conectados a dos borneras de la placa PCB, por encima de esta se encuentra la WeMos D1 mini.

Esquema general y funcionamiento del prototipo



Edukit y módulo IoT

Dentro del entorno del invernadero se encuentra la planta y la Edukit con su módulo IoT en una PCB. La Edukit concentra los sensores de intensidad lumínica y de temperatura (LDR y LM35 respectivamente). La PCB dispone de un sensor de humedad de suelos, otro de monóxido de carbono (YL-69 y MQ-9 respectivamente) y la placa WeMos D1 mini. Si bien el conexionado de los sensores adicionales y la WeMos D1 mini se hace sobre la PCB, eléctricamente están unidos a la Edukit. Viene a cumplir la función de un shield para Arduino.

La comunicación entre la Edukit y la interfaz inalámbrica es serial por software. El vinculo serial por hardware se reserva para monitorizar el proceso desde el Arduino.

Los datos de los sensores se envían en forma de un string del estilo:

```
edukit,mq9= 0.52,hum= 2.00,ldr=828.00,temp=34.21
```

Este es un mensaje y el mismo se publica a dos broker MQTT. Uno es un broker EMQTT, que lo dispone la cooperativa para sus propósitos. En este caso se destinó un topic para que los miembros puedan hacer un seguimiento de los mensajes que publican los módulos IoT (que tendrán usarán a futuro). El otro broker se llama MOSCA y es creado como un nodo dentro de la plataforma Node-RED.

Los datos se envían con cierta periodicidad que puede ser cambiada en el código de la Edukit. Al ser strings relativamente largos, y como se hacen dos publicaciones a brokers distintos, estas se deben hacer con intervalos de al menos 3 segundos. Esto sucede porque los mensajes primero viajan por serial a la WeMos, y luego se publican. La comunicación serial es muy lenta, y esos tiempos deben respetarse, de lo contrario los strings llegarían incompletos. Por fortuna, a fines de esta práctica y para uso en un invernadero, las muestras se toman en períodos de como mínimo 30 minutos, por lo que ese lapso entre publicaciones a distintos brokers pasa a ser despreciable. Los datos del entorno varían muy lentamente con el tiempo.

En cuanto al firmware de la WeMos D1 mini, pese a seleccionarse Tasmota, este no cumplía con las necesidades para el proyecto. No interpretaba los mensajes que llegaban por serial. Era una funcionalidad en la que los desarrolladores estaban trabajando pero en la última versión (hasta ese momento) no estaba disponible aún. Por lo tanto se modificó el código fuente (sin alterar el normal funcionamiento) para que fuera posible recibir un string por software serial, interpretarlo como un comando y luego publicarlo a un topic como mensaje MQTT a un broker.

Gateway / Server

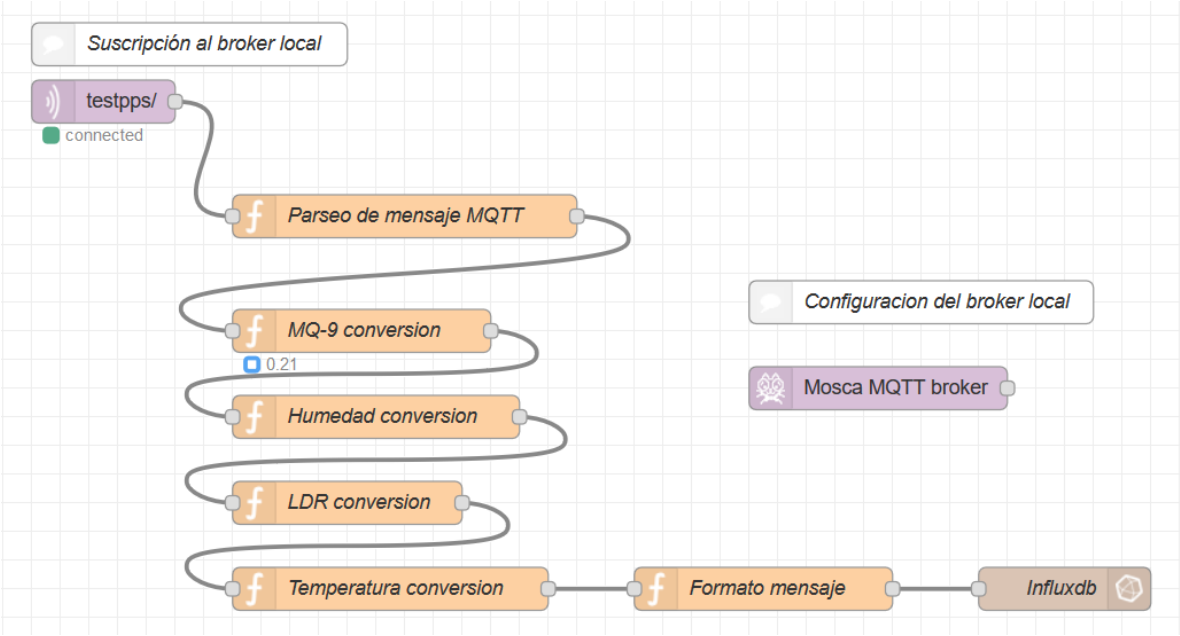
Dentro de la Raspberry Pi se aloja la plataforma Node-RED, la base de datos InfluxDB y la aplicación Grafana. Todas las herramientas se relacionan entre si gracias a Node-RED.

La plataforma tiene 3 funciones bien marcadas:

- Uso del protocolo de mensajería, haciendo las veces de Broker por un lado, y cliente por otro
- Insertar datos y realizar consultas a la base de datos
- Generar una web presentando gráficas en función a los datos de los sensores

El funcionamiento de Node-RED es en base a flujos y nodos. Los nodos se conectan entre sí generando un flujo.

En cuanto al protocolo de mensajería hay dos nodos importantes, uno que genera el broker MOSCA y otro que se suscribe al topic donde la Edukit publica los datos de los sensores.



Los datos que llegan se parsean y se insertan en la base de datos. Existen nodos que provienen del mismo paquete instalado para vincularse con InfluxDB que sirven para otras operaciones, como gestionar consultas y devolver resultados sobre una base.

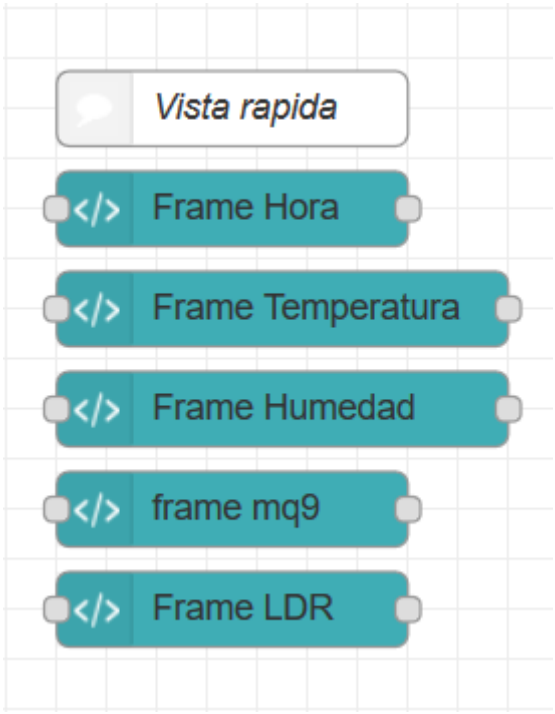
La herramienta de visualización Grafana realiza consultas a la base de datos, y una vez obtenidos puede operar con ellos, como calcular máximos, mínimos, medias, etc. Se generan 5 graficas, las cuales son una para cada sensor y la última para mostrar la fecha y la hora. Dichas graficas son embebidas en la web que se desarrolla con Node-RED.

Se crea un sitio web usando los nodos *Dashboard* y combinando los antes mencionados, permitiendo tener en un mismo lugar toda la información necesaria del invernadero en tiempo real y con posibilidad de exportar para posteriores análisis. Los archivos exportados se guardan en un directorio de la Raspberry Pi en formato csv.

Visualización

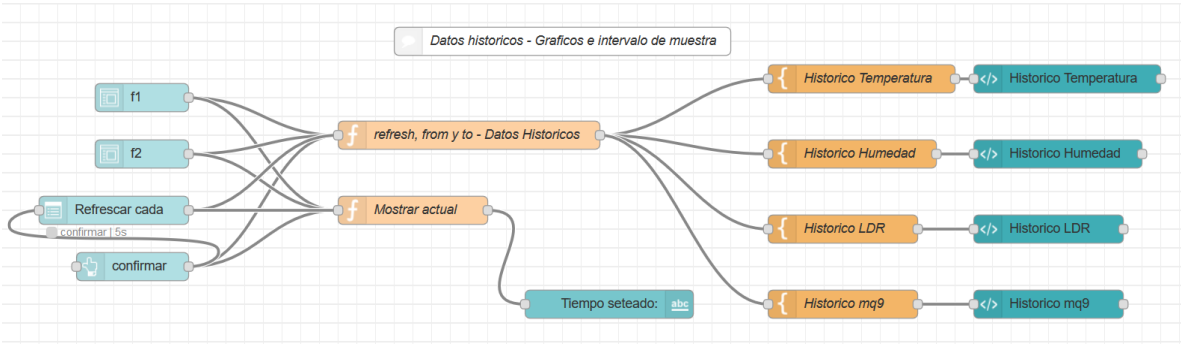
La aplicación web es la síntesis de todo el trabajo hecho. Primeramente esta Node-RED, que con distintos nodos levantan el servidor localmente y administran los sectores de la web por medio de bloques modificables en muchos aspectos.

La información esta divida en dos pestañas, una con valores instantáneos y la otra con valores históricos. El flujo de Node-RED para generar la pestaña de valores instantáneos es la siguiente:

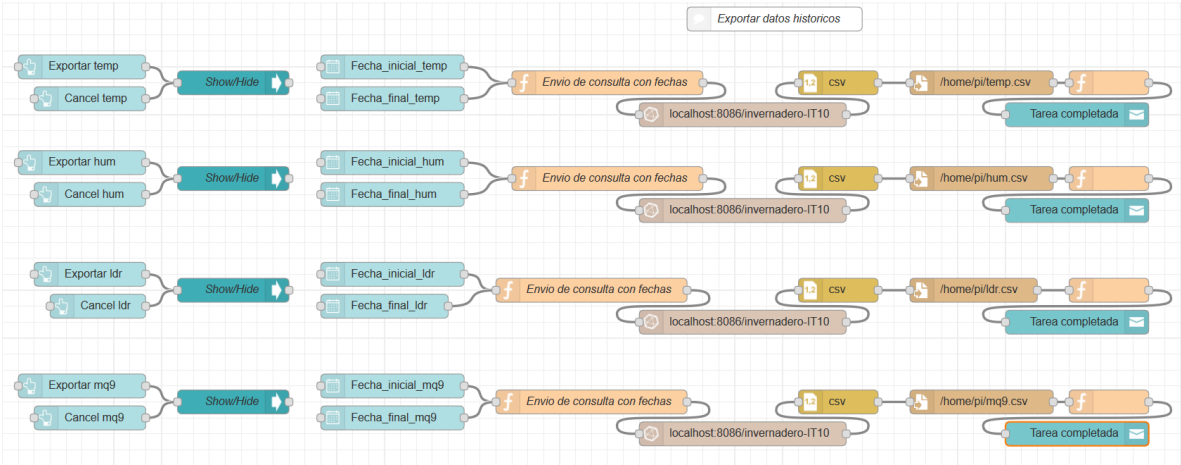


No hay conexiones entre nodos porque son solo bloques aislados dentro de la pagina principal que presentan la información.

En cuanto al flujo que genera la pagina para los datos históricos es el siguiente:



Y para poder lograr exportar la información de estos valores históricos se necesita otro flujo de Node-RED, y es este:



Para hacer esto es necesario determinar los intervalos de tiempo en los que se quiere extraer información, el formato, el tipo de consulta que se debe hacer a la base de datos e indicar en donde se guardará el archivo.

Todo el despliegue, configuraciones y proceso de creación de flujos y nodos se encuentra la documentación liberada en Github y también en el Anexo XI.

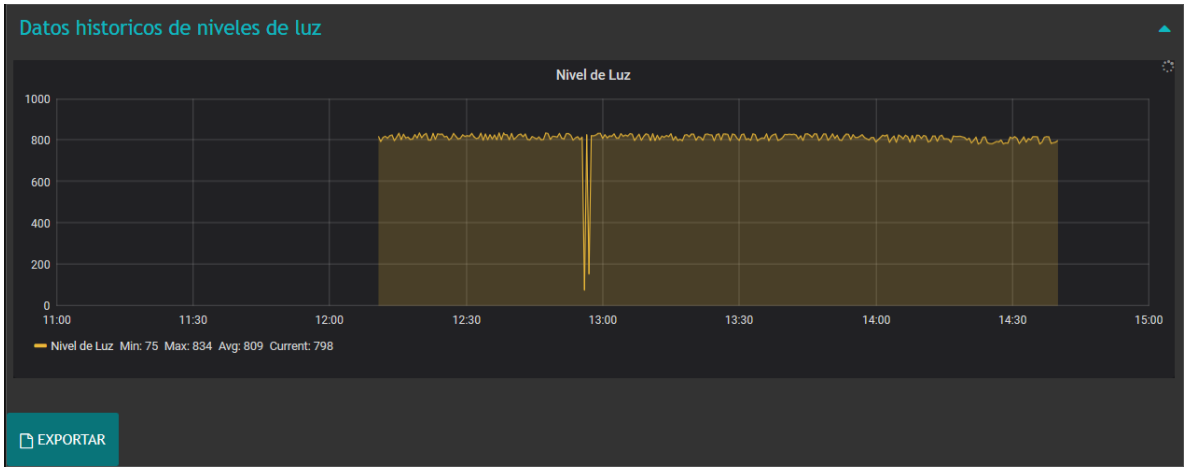
Resultado final

Todo se muestra en un sitio web creado por Node-RED. Se puede acceder al mismo de forma local en un explorador cualquiera con la URL http://192.168.20.129/1880/Invernadero. A continuación se muestran partes del sitio web creado y sus menues.

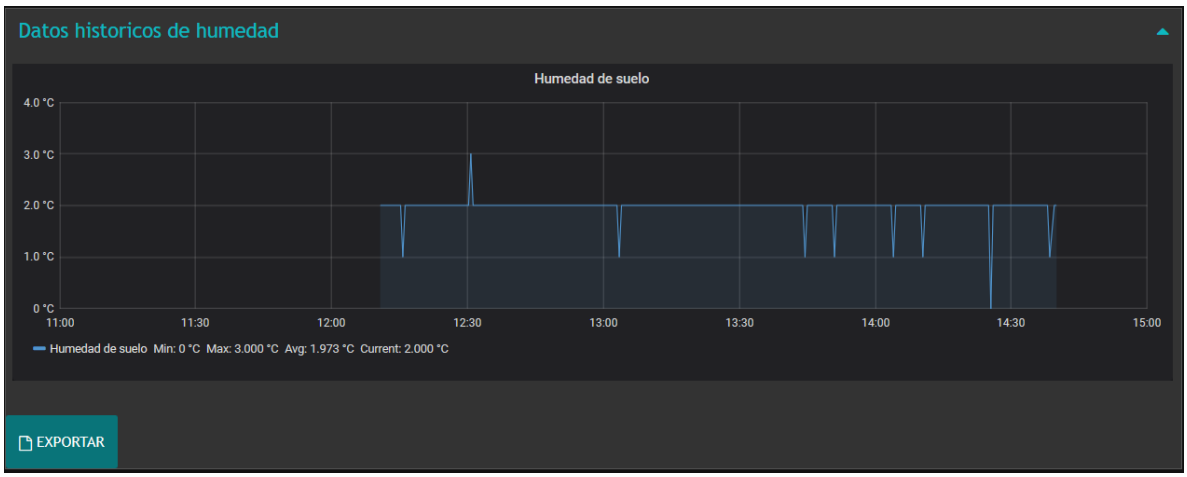
Pestaña valores instantáneos - *Invernadero*

Pestaña valores históricos - *Registros*

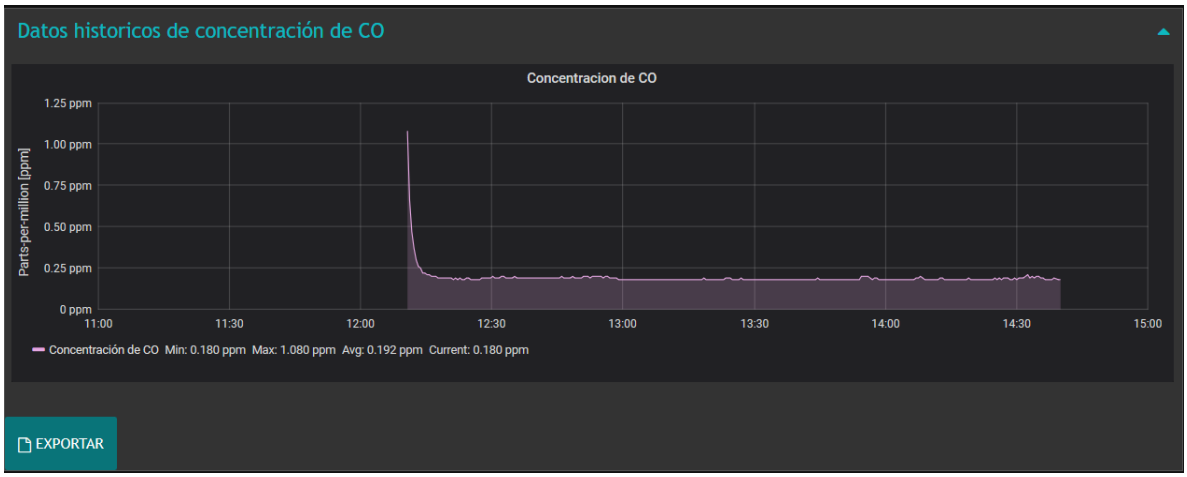
Intensidad de luz



Humedad en suelo



Concentración de monóxido de carbono



Temperatura ambiente

Se encuentra desplegado el menú para exportar.

Conclusiones

El principal objetivo de la práctica profesional es insertar al estudiante al ámbito laboral para que pueda desarrollar los conocimientos adquiridos durante el cursado de la carrera. En este aspecto debo decir que al haber tomado contacto con un entorno distinto al académico fue muy productivo ya que me dejó muchas enseñanzas en lo que se refiere a la carrera que hoy estoy culminando. Me tocó experimentar distintos tipos de actividades, algunas mas conocidas por haberlas desarrollado en alguna asignatura del plan de estudios, como programar un Arduino, hasta otras no tan habituales, como tomar decisiones para la selección de un software o hardware.

Luego de haber investigado las mejores opciones para desarrollar el objetivo de la practica era necesario tener que comunicarlo al equipo de trabajo y ponerlos al tanto de cada decisión. No era algo a lo que estuviera acostumbrado pero todos los aportes que me dieron fueron para mejor, y me respaldaron en cada decisión tomada. Me sentí muy a gusto en cada paso que daba.

El entorno laboral fue de lo mejor, son personas que ya conocía de otros ámbitos, por lo que la relación con ellos es muy buena. Muchos de mis compañeros son programadores, entonces ante cualquier duda podía consultarles ya que están mucho mas involucrados en la temática. Si bien la programación es una herramienta importante de la carrera, los ingenieros en telecomunicaciones no somos programadores.

En contraparte a lo antes dicho, al haber estado en un ámbito de una cooperativa, donde su lógica de organización es distinta a la de una empresa y a su vez los miembros ya me eran conocidos, no sentí haber tenido grandes responsabilidades, más de las propias de la practica, como respetar los tiempos, comunicar avances, etc. No se si me ha preparado para en un futuro trabajar en un empresa en la que la lógica de trabajo y de organización sea mas vertical, que haya que responder por otras personas o rendir cuentas a un superior, entre otras cosas.

En cuanto a lo profesional y técnico, tomé herramientas que en un futuro podría implementar para un emprendimiento propio. La temática de IoT es muy amplia, pero creo haber allanado el camino lo suficiente como para poder seguir aprendiendo por mi cuenta. Mas allá de eso, entiendo que no estoy preparado para lanzarme con un emprendimiento. Me gustaría poder aprender más, ganar experiencia, y luego si hacer algo propio.

Respecto a las tareas realizadas, quedaron algunas cosas pendientes que fueron surgiendo en el desarrollo de la practica, y otras que se descartaron. Entre las pendientes tenemos:

- Bridge entre brokers MQTT
- Aprovisionamiento - Instalación y variables de entorno
- Web responsive

En el caso del bridge entre brokers surgía como opción para no tener que hacer publicaciones distintas con la misma información. La idea era que un broker reenvíe lo publicado a otro automáticamente cuando llega un mensaje a determinado topic. Esto no fue posible ya que no hubo muchas opciones de modificación al broker propio de Node-RED. Una solución seria configurar otro broker por fuera de la plataforma en la que se pueda programar para desempeñar esta tarea.

Una cosa que parecía interesante era la opción de poder configurar el modulo y la Edukit a través de la conexión inalámbrica, lo que se conoce como configuración OTA (Over-The-Air). Por falta de tiempo no se pudo indagar mucho en el aprovisionamiento y la modificación de las variables de entorno.

Por ultimo, no fue posible hacer un sitio web responsive, la programación en Node-RED solo permite hacer webs que puedan visualizarse en el explorador de una PC, por cuestiones de resolución de pantalla.

Referencias y bibliografía

Obradovic L. (Enero 2018). Tasmota vs ESPurna vs ESPEasy - overview. Recuperador el 2 de enero de 2019. https://lobradov.github.io/FOSS-Firmware-comparison-overview/

Sonoff/Tasmota - https://github.com/arendst/Sonoff-Tasmota

Espurna - https://github.com/xoseperez/espurna

ESPEasy - https://www.letscontrolit.com/wiki/index.php/ESPEasy

MQTT - https://mqtt.org/documentation

Raspberry Pi - https://www.raspberrypi.org/documentation/

EasyEDA - https://docs.easyeda.com/en/FAQ/Editor/index.html

InfluxDB - https://github.com/influxdata/influxdb

Influxdata - Time series database (TSDB) explained. https://www.influxdata.com/time-series-database/

Grafana - https://grafana.com/docs/

MQTT Mosca broker - https://github.com/mcollina/mosca/wiki

Mueller J. (Dec 2017). Understanding the Less Popular Push/Streaming Protocols (XMPP, CoAP, MQTT, etc.). Recuperado 8 de enero de 2019. https://www.programmableweb.com/news/understanding-less-popular-pushstreaming-protocols-xmpp-coap-mqtt-etc/analysis/2017/12/11

14core - The IOT Protocols The Base of Internet of Things Ecosystem - https://www.14core.com/the-iot-protocols-the-base-of-internet-of-things-ecosystem/

DB-Engines - https://db-engines.com/en/

DB-Engines - DB-Engines Ranking of Time Series DBMS - https://db-engines.com/en/ranking/time+series+dbms

Spiess A. - #255 Node-Red, InfluxDB, and Grafana Tutorial on a Raspberry Pi - https://www.youtube.com/watch?v=JdV4x925au0

Anexos

ANEXO I: Pinout y Esquemático Edukit10

ANEXO II: Pinout, especificaciones y Esquemático WeMos D1 Mini

ANEXO III: Instalación, configuración y pruebas realizadas con Tasmota

ANEXO IV: Instalación, configuración y pruebas realizadas con ESPurna

ANEXO V: Instalación y configuración de Raspbian sobre Raspberry Pi 3 B+

ANEXO VI: Esquemático PCB

ANEXO VII: Instalación y configuración de InfluxDB sobre Raspbian

ANEXO VIII: Instalación y configuración de Grafana sobre Raspbian

ANEXO IX: Instalación y configuración de Node-RED sobre Raspbian

ANEXO X: Integración de tecnologías Node-Red, InfluxDB y Grafana

ANEXO XI: Documentación Proyecto prototipo IoT Invernadero Inteligente

ANEXO XII: Hojas de datos de todos los componentes de Hardware

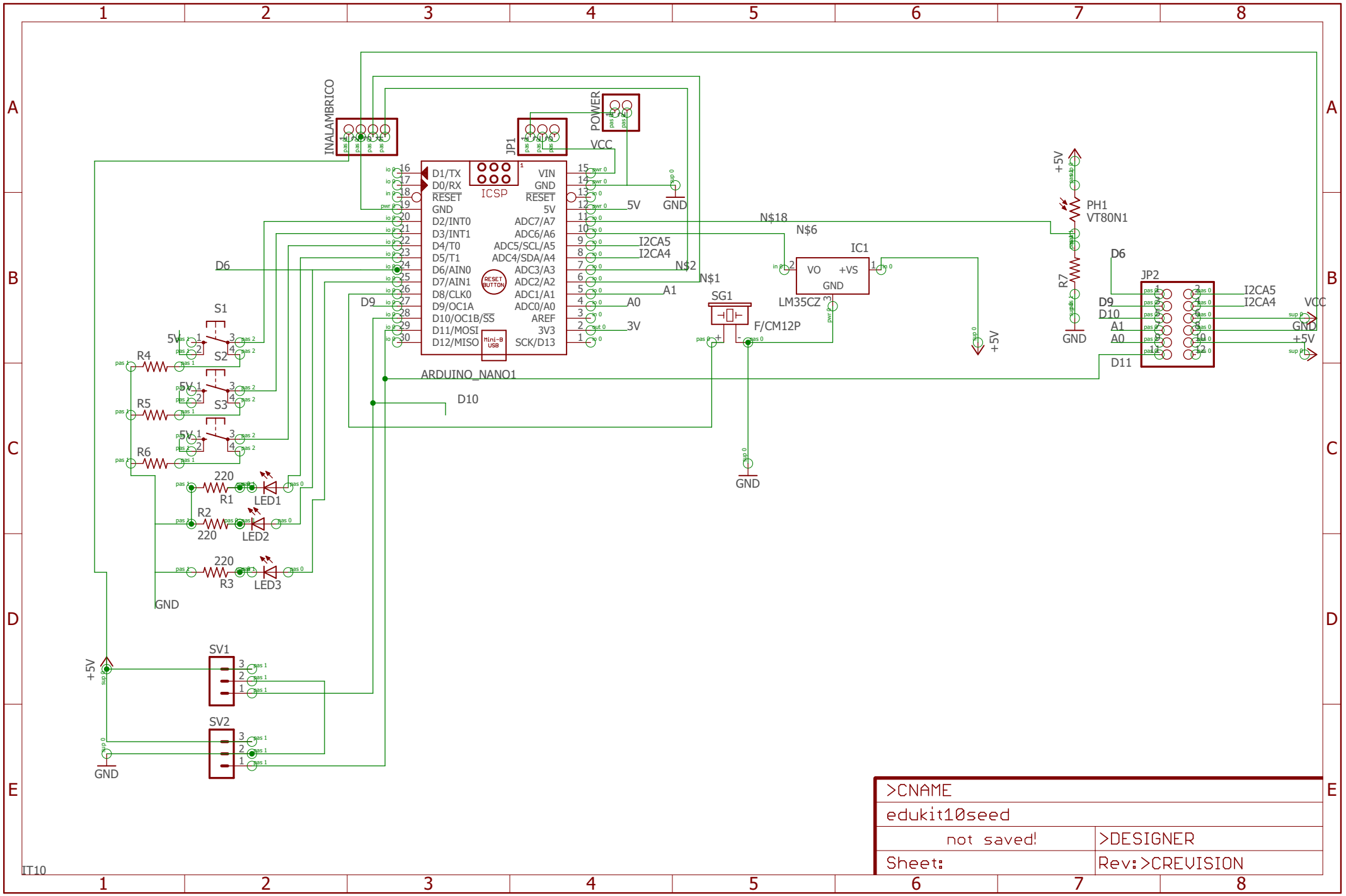
- ESP8266
- Sensores:
    - LM35
    - LDR
    - MQ-9
    - YL-69
- Raspberry Pi 3 B+

Anexo I

Pinout Multiconector Edukit10

```
Pinout                  | Pin      Función           Arduino
                        |
     ---    ---         |
N/A | o |  | o | 1      |  1-     Digital/pwm            D11
     ---    ---         |
+5v | o |  | o | 2      |  2-     Analógico, Digital     A0
     ---    ---         |
GND | o |  | o | 3      |  3-     Analógico, Digital     A1
     ---    ---         |
Vcc | o |  | o | 4      |  4-     Digital/pwm            D10
     ---    ---         |
8   | o |  | o | 5      |  5-     Digital/pwm            D9
     ---    ---         |
7   | o |  | o | 6      |  6-     Digital/pwm            D6
     ---    ---         |
                        |  7-     Analógico/i2c SDA      A4
* VCC(9)                |
  GND(10)               |  8-     Analógico/i2c SCL      A5
  N/A(12)               |
  +5v(11)               |  9-     VCC, Entrada Reg       Vin
                        |
                        |  10-    GND                    GND
                        |
                        |  11-    Salida Reg +5v         5v
                        |
                        |  12-    Pin ciego              NA
```

Esquemático Edukit10


INALAMBRICO
POWER
JP1
VCC
ICSP
D1/TX
D0/RX
RESET
GND
D2/INT0
D3/INT1
D4/T0
D5/T1
D6/AIN0
D7/AIN1
D8/CLK0
D9/OC1A
D10/OC1B/SS
D11/MOSI
D12/MISO
VIN
GND
RESET
5V
ADC7/A7
ADC6/A6
ADC5/SCL/A5
ADC4/SDA/A4
ADC3/A3
ADC2/A2
ADC1/A1
ADC0/A0
AREF
3V3
SCK/D13
RESET BUTTON
Mini-B USB
I2CA5
I2CA4
N$2
N$1
A1
A0
3V
D6
D9
D10
D11
N$18
N$6
IC1
VO
+VS
GND
LM35CZ
SG1
F/CM12P
PH1
VT80N1
R7
JP2
S1
S2
S3
R4
R5
R6
R1
R2
R3
220
LED1
LED2
LED3
SV1
SV2
ARDUINO_NANO1
+5V
GND
>CNAME
edukit10seed
not saved!
>DESIGNER
Sheet:
Rev:>CREVISION
IT10

Anexo II

Pinout WeMos D1 mini

Pin

Pin	Function	ESP-8266 Pin
TX	TXD	TXD
RX	RXD	RXD
A0	Analog input, max 3.3V input	A0
D0	IO	GPIO16
D1	IO, SCL	GPIO5
D2	IO, SDA	GPIO4
D3	IO, 10k Pull-up	GPIO0
D4	IO, 10k Pull-up, BUILTIN_LED	GPIO2
D5	IO, SCK	GPIO14
D6	IO, MISO	GPIO12
D7	IO, MOSI	GPIO13
D8	IO, 10k Pull-down, SS	GPIO15
G	Ground	GND
5V	5V	-
3V3	3.3V	3.3V
RST	Reset	RST

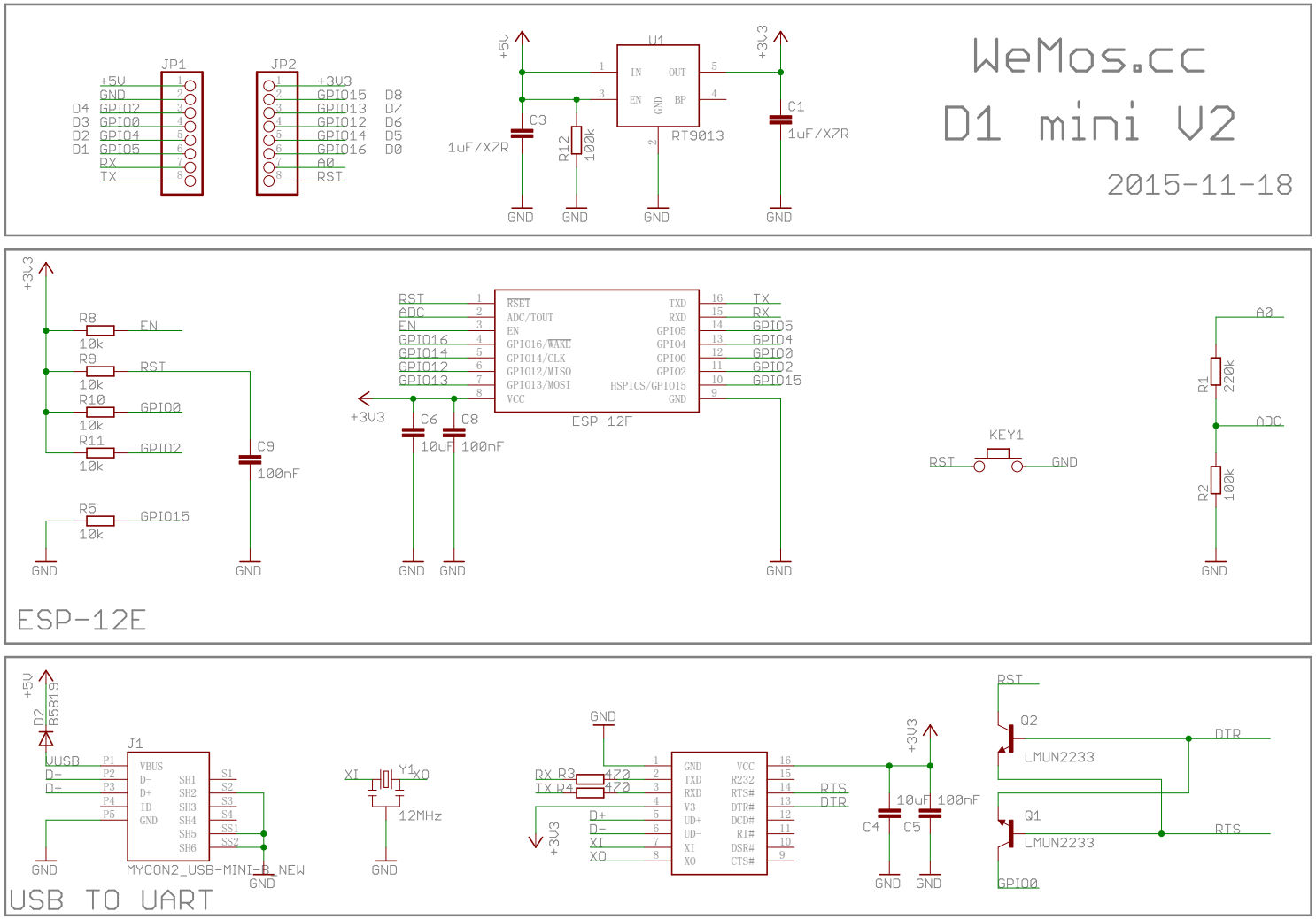
Especificaciones

ESPECIFICACIONES TÉCNICAS

- Voltaje de Alimentación: 5V DC
- Voltaje Entradas/Salidas: 3.3V DC
- SoC: ESP8266 (Módulo ESP-12E)
- CPU: Tensilica Xtensa LX3 (32 bit)
- Frecuencia de Reloj: 80MHz/160MHz
- Instruction RAM: 32KB
- Data RAM: 96KB
- Dimensiones: 34.2mm x 25.6mm
- Peso: 10g

ESPECIFICACIONES DE NETWORKING

- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Stack de Protocolo TCP/IP integrado
- Procesador MAC/Baseband integrado
- Módulos WEP, TKIP, AES y WAPI integrados
- Potencia de salida de +19.5dBm en modo 802.11b


WeMos.cc
D1 mini V2
2015-11-18
JP1
JP2
+5V
GND
GPIO2
GPIO0
GPIO4
GPIO5
RX
TX
D4
D3
D2
D1
+3V3
GPIO15
GPIO13
GPIO12
GPIO14
GPIO16
A0
RST
D8
D7
D6
D5
D0
U1
IN
OUT
EN
GND
BP
RT9013
C3
1uF/X7R
R12
100k
C1
1uF/X7R
ESP-12E
R8
R9
R10
R11
R5
10k
EN
RST
GPIO0
GPIO2
GPIO15
C9
100nF
RSET
ADC/TOUT
EN
GPIO16/WAKE
GPIO14/CLK
GPIO12/MISO
GPIO13/MOSI
VCC
TXD
RXD
GPIO5
GPIO4
GPIO0
GPIO2
HSPICS/GPIO15
GND
TX
RX
GPIO5
GPIO4
GPIO0
GPIO2
GPIO15
ESP-12F
C6
C8
10uF
100nF
KEY1
R1
220k
R2
100k
ADC
A0
USB TO UART
D2
B5819
J1
MYCON2_USB-MINI-B_NEW
VBUS
D-
D+
ID
GND
SH1
SH2
SH3
SH4
SH5
SH6
S1
S2
S3
S4
SS1
SS2
UUSB
Y1
12MHz
XI
XO
RX R3
TX R4
470
470
TXD
RXD
V3
UD+
UD-
XI
XO
VCC
R232
RTS#
DTR#
DCD#
RI#
DSR#
CTS#
RTS
DTR
C4
C5
10uF
100nF
Q2
Q1
LMUN2233
LMUN2233
GPIO0

ANEXO III

Sonoff-Tasmota

Firmware alternativo para dispositivos basados en ESP8266 con web, temporizadores, actualizaciones de firmware 'Over The Air' (OTA) y compatibilidad con sensores, lo que permite el control bajo Serial, HTTP, MQTT y KNX, para su uso en Smart Home Systems. Escrito para Arduino IDE y PlatformIO.

Dispositivos soportados

Los siguientes dispositivos son compatibles con el control Serial, Web y MQTT:

- iTead Sonoff Basic (R2)
- iTead Sonoff RF
- iTead Sonoff SV
- iTead Sonoff TH10/TH16 con sensor temperature
- iTead Sonoff Dual (R2)
- iTead Sonoff Pow con Energy Monitoring
- iTead Sonoff Pow R2 con Energy Monitoring
- iTead Sonoff 4CH (R2)
- iTead Sonoff 4CH Pro (R2)
- iTead S20 Smart Socket
- Sonoff S22 Smart Socket
- iTead Sonoff S26 Smart Socket
- iTead Sonoff S31 Smart Socket con Energy Monitoring
- iTead Slampher
- iTead Sonoff Touch
- iTead Sonoff T1
- iTead Sonoff SC
- iTead Sonoff Led
- iTead Sonoff BN-SZ01 Ceiling Led
- iTead Sonoff B1 (R2)
- iTead Sonoff iFan02
- iTead Sonoff RF Bridge 433
- iTead Sonoff Dev
- iTead Sonoff PSA Smart Switch Module
- iTead 1 Channel Switch 5V / 12V
- iTead Motor Clockwise/Anticlockwise
- Electrodragon IoT Relay Board
- AI Light o cualquier my9291 compatible RGBW LED bulb
- H801 PWM LED controller
- MagicHome PWM LED controller (aka Flux-Light LED module)

- AriLux AL-LC01, AL-LC06 y AL-LC11 PWM LED controller
- Supla device - Espablo-inCan mod. for electrical Installation box
- BlitzWolf BW-SHP2 Smart Socket con Energy Monitoring
- BlitzWolf BW-SHP4 Smart Socket con Energy Monitoring (UK version)
- BlitzWolf BW-SHP6 Smart Socket with Energy Monitoring
- Luani HVIO board
- Wemos D1 mini
- HuaFan Smart Socket
- Hyleton-313 Smart Plug
- Allterco Shelly 1
- Allterco Shelly 2 con Energy Monitoring
- NodeMcu y Ledunia
- Switches basados en KS-602, como GresaTek, Jesiya, NewRice, Lyasi, etc
- OBI Wifi Stecker Schuko
- OBI Wifi Stecker Schuko weiß (Rev.2)
- SmartPlug AISIRER, AVATAR con POW
- SmartPlug NEO COOLCAM NAS WR01W

El que usaremos para desarrollar algunas pruebas es el **Wemos D1 mini**.

Características

Las siguientes características están disponibles:

- MQTT `GroupTopic` puede abordar múltiples dispositivos
- Carga de firmware por OTA o mediante carga de página web
- Mensajes de estado expandidos
- Los mensajes de Syslog UDP se pueden filtrar por nombre de programa que comienza con `ESP-`
- El botón puede enviar un mensaje MQTT diferente definido con `ButtonTopic`
- Configuración inicial de Wifi por `user_config.h`, Serial, Smartconfig, Wifi manager o configuración WPS
- Un servidor web proporciona el control de Sonoff y contiene una facilidad de carga de firmware
- Compatibilidad con los sensores de temperatura DHTxx, AM2301 o DS18B20 que se utilizan en Sonoff TH10 / TH16
- Soporte para sensores I2C BH1750, BME280, BMP280, HTU21, SI70xx, LM75AD y SHT1x
- Los datos de telemetría se pueden enviar usando un prefijo diferente opcional de los mensajes de estado
- Soporte nativo de Domoticz MQTT
- Fácil integración en soluciones de automatización del hogar como openHAB, HomeAssistant, ...
- Emulación de Wemo y Hue para el soporte de Amazon Echo (Alexa)
- Compatibilidad con el protocolo IP KNX para la comunicación a redes KNX y también de dispositivo a dispositivo.

Primeros pasos

Pre-requisitos

Hardware necesario

- Uno de los módulos ESP8266 soportados
- Un convertidor de USB a serie de muy buena calidad que puede suministrar 3.3V suficientes para alimentar el dispositivo (o una fuente de alimentación de 3.3V por separado).

Software necesario

Hay binarios disponibles para descargar e instalar. Instalar un binario es mucho más fácil que configurar **PlatformIO** o **Arduino** para construir el proyecto. Se puede usar el software Esptool (https://github.com/arendst/Sonoff-Tasmota/wiki/Esptool) para actualizar un binario preconfigurado en su dispositivo. Los archivos binarios están en *releases section*(https://github.com/arendst/Sonoff-Tasmota/releases).

Si se necesita o se desea modificar el código o la configuración predeterminada, se puede utilizar uno de los siguientes:

1. PlatformIO - https://platformio.org/ (todas las bibliotecas y configuraciones necesarias están preconfiguradas en platformio.ini- https://github.com/arendst/Sonoff-Tasmota/blob/master/platformio.ini)
2. Arduino IDE- https://www.arduino.cc/en/Main/Software

Otros requerimientos

- El código fuente del firmware
- Un broker MQTT
- Un cliente MQTT para interactuar con el módulo

Preparación del hardware

Conexión serial

La siguiente tabla muestra la conexión entre los conectores del programador y los módulos. Preste atención a las líneas cruzadas RX y TX.

Programador	Módulo Sonoff
3V3	3V3/VCC
TX	RX
RX	TX
GND	GND

Algunos módulos de Sonoff exponen un encabezado de cinco pines. El quinto pin es irrelevante para la conexión serial. Por favor, consulte las páginas del módulo específico para obtener más información.

Poniendo el módulo en modo flash

Particularmente en la Wemos D1 mini, no es necesario realizar nada, solo conectarlo a la interfaz serial.

Upload

PlatformIO

1. Descargar y extraer el último "Código fuente (zip) de Sonoff-Tasmota" (o clonar el repositorio).
2. Cargue la carpeta base de Sonoff-Tasmota en PlatformIO
3. Conectar el módulo en modo Flash
4. Seleccionar una variante de firmware para instalar en su módulo sin comentar una de las líneas `env_default` en `platformio.ini` (a continuación los detalles de la variante). Es posible que también deba cambiar el `upload_port` para que coincida con el puerto de comunicaciones de la interfaz serial USB.
5. Abrir `my_user_config.h` y configure sus ajustes de WiFi y, opcionalmente, un MQTT, Syslog, WebServer, NTP, etc. NOTA: Si se carga varias veces es necesario modificar el parametro `CFG_HOLDER`. Cada vez que se intente cargar el firmware se debe cambiar el numero. Este número se almacena en la

memoria flash y si el nuevo código tiene el mismo número, se conservan las configuraciones actuales en la memoria flash. Por ende, no sucederan cambios.
6. Seleccionar "Upload" en el menú para actualizar el firmware.
7. Después de la transferencia exitosa del firmware, desconectar el módulo.

Continúe en "Primeros pasos" y asegúrese de revisar las instrucciones para conectar sensores adicionales.

Variantes de Firmware

- `sonoff.bin`: el firmware predeterminado para todos los dispositivos
- `sonoff-minimal.bin`: es un firmware provisional que se utilizará cuando las imágenes de firmware anteriores sean demasiado grandes para que quepan como OTA o carga web; la instalación de este primero y luego la carga del sonoff.bin deseado permite un crecimiento futuro del tamaño del firmware por encima del límite OTA del archivo de 1/2 flash.
- `Sonoff-sensores`: es una versión con la mayoría de los sensores usados habilitados.

Herramientas para la carga del firmware

Visual Studio Code

Cómo preparar y configurar Visual Studio Code con PlatformIO para la compilación y carga de Tasmota.

Descargar e instalar Visual Studio Code

Descargar Visual Studio Code (VSC) desde su pagina web (https://code.visualstudio.com/)

Instalar la extensión PlatformIO

Instalar la extensión *PlatformIO IDE* en VSC.

Seleccionar `Ver` - `Extensiones` y escribir *PlatformIO* en el cuadro de búsqueda.

Asegurarse de seleccionar la extensión oficial de PlatformIO.org *PlatformIO IDE* y seleccionar Instalar. Aceptar para instalar dependencias.

Descargar Tasmota

Descargar la última versión de Tasmota de https://github.com/arendst/Sonoff-Tasmota/releases y descomprimir en una carpeta conocida.

Copiar archivos

Copiar todos los archivos del código fuente de la versión Tasmota en la carpeta de trabajo de VSC.

Compilar tasmota

Iniciar VSC y seleccionar `File` - `Open folder...` para apuntar a la carpeta de trabajo.

Nota: Presionar `Ctrl + Shift + P` y escribir **PlatformIO** para ver todas las opciones.

Seleccionar el firmware deseado editando el archivo `platformio.ini` según sea necesario.

Cargar Tasmota

Habilitar las opciones deseadas en `platformio.ini` para la carga serial como:

```
; *** Subir el método de restablecimiento de serie para Wemos y NodeMCU
upload_port = COM5
; upload_speed = 512000
upload_speed = 115200
; upload_resetmethod = nodemcu
```

Para el caso de la Wemos D1 mini y el S.O. Lubuntu 18.04 usado, el `upload_port` en vez de ser **COM5** es **/dev/ttyUSB0**. Además hay que dar permisos para que platformIO pueda acceder al dispositivo.

Habilitar las opciones deseadas en `platformio.ini` para cargarlas en su servidor OTA local como:

```
; *** Subir archivo al servidor OTA utilizando HTTP
upload_port = domus1: 80 / api / upload-arduino.php
extra_scripts = pio / http-uploader.py
```

Luego **Build**(si es necesario) y **Upload**.

Consejo:

En caso de que VSC muestre una gran cantidad de errores usando `PlatformIO - Intellisense`, una posible "solución" es cambiar el cpp Intelli Sense a "TAG PARSER"

Esta configuración se puede cambiar en la configuración del área de trabajo mediante: Use `Ctrl` + `Shift` + `P` y escribir `Preferences: Open Workspace Settings` y escribir `intelli Sense` en el cuadro de búsqueda. Ahora cambiar el valor de `Intelli Sense Engine` a `Tag Parser`.

Configuración del dispositivo

Antes de compilar, considere la modificación de los valores `STA_SSID1` y `STA_PASS1` dentro de `user_config.h` para que coincidan con su **SSID** de WiFi y su **contraseña** de WiFi.

```
    #define STA_SSID1 "indebuurt1" // [Ssid1] Wifi SSID
    #define STA_PASS1 "VnsqrtnrsddbrN" // [Password1] Contraseña de Wifi
```

Esta operación simplifica los primeros pasos, ya que Sonoff reconocerá y se unirá automáticamente a su WiFi.

Establecer las reglas de horario de verano/horario estándar en `user_config.h` también simplificará la configuración futura

```
    #define TIME_DST  North, Last, Sun, Mar, 2, +120  // Northern Hemisphere, Last sunday in march at 02:00 +120 minutes
    #define TIME_STD  North, Last, Sun, Oct, 3, +60  // Northern Hemisphere, Last sunday in october 03:00 +60 minutes
```

Si la nueva configuración no parece aplicarse después de la actualización, puede hacer lo siguiente para solucionarlo:

- Cambie el valor de CFG_HOLDER] en `user_config.h` y volver a flashear
- Mantener presionado el botón del dispositivo durante al menos 4 segundos para iniciar un comando RESET
- Restablecer el dispositivo a través de la consola web.

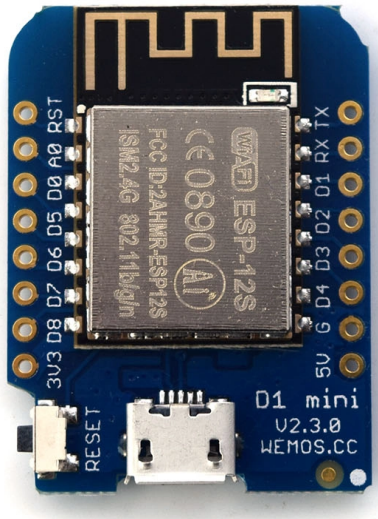
Upgrade

Configuración inicial

Se ha cargado correctamente el firmware de Sonoff-Tasmota en su módulo.

- Conectar a la alimentación (CA) (serial NO conectado). SOLO PARA SONOFF
- Si el WiFi no se configuró antes de compilar como se describe en la sección Upload, configurarlo usando uno de los modos de configuración provistos: Uso del botón o usar WPS si el router lo admite.
- Conectarse a la página web del dispositivo (http://[sonoff-ip]/cn) NOTA: Reemplazar `[sonoff-ip]` con la dirección IP real del dispositivo.
- Seleccionar el tipo de módulo correcto desde la interfaz de configuración
- Configurar la conexión del broker MQTT, elegir un topic único
- Probar la comunicación con el cliente MQTT
- Integrarlo con la interfaz de control (Home Assistant, Domoticz, etc) de su elección.

Flashando Wemos D1 mini con Tasmota



Seguir pasos mencionados anteriormente para la carga de Tasmota, pero añadir las siguientes líneas al final de `platformio.ini`:

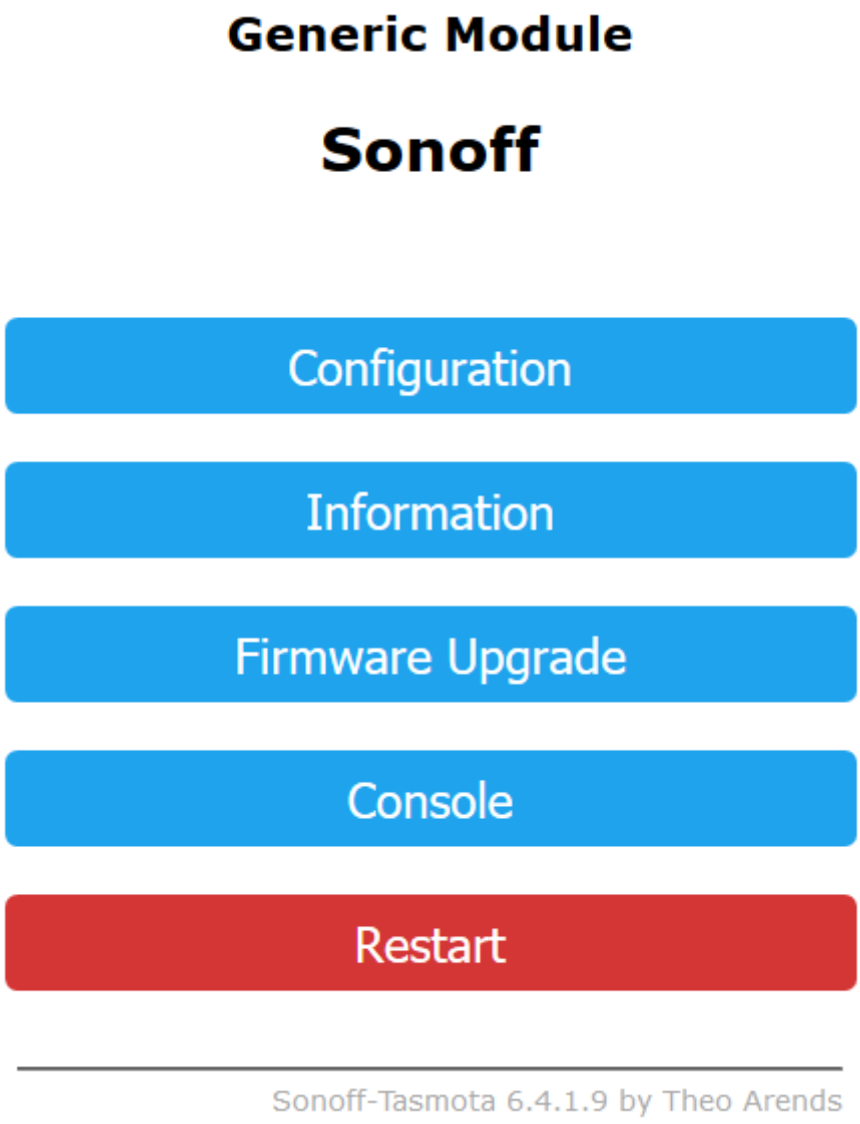
```
[env:wemos-d1-mini]
platform = espressif8266
framework = arduino
board = esp01_1m
board_flash_mode = dout
build_flags = -Wl,-Tesp8266.flash.1m0.ld -DMQTT_MAX_PACKET_SIZE=1000
lib_deps = PubSubClient, NeoPixelBus, IRremoteESP8266, ArduinoJSON
extra_scripts = pio/strip-floats.py

; *** Serial Monitor options
monitor_baud = 115200

; *** Upload Serial reset method for Wemos and NodeMCU
upload_resetmethod = nodemcu
upload_speed = 115200
;upload_port = COM6

; *** Upload file to OTA server using SCP
;upload_port = user@host:/path
;extra_scripts = pio/strip-floats.py, pio/sftp-uploader.py
```

```
21
22 ; *** Upload file to OTA server using HTTP
23 ;upload_port = domus1:80/api/upload-arduino.php
24 ;extra_scripts = pio/strip-floats.py, pio/http-uploader.py
```

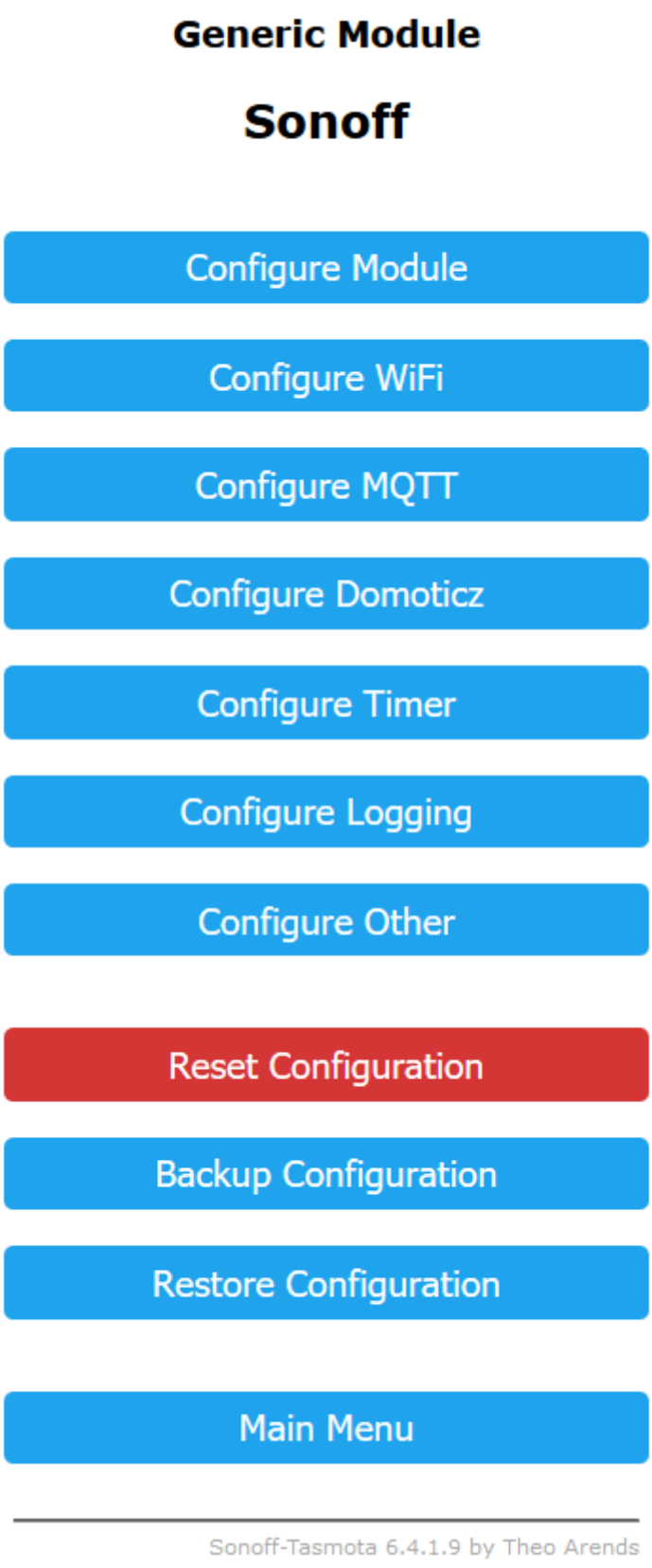
Luego **Build** y **Upload**.

Configurar Tasmota para Wemos

En primera instancia se puede conectarse a la página web del dispositivo (http://[sonoff-ip]) NOTA: Reemplazar `[sonoff-ip]` con la dirección IP real del dispositivo.



Ingresando a `Configuration` se pueden modificar todos los parametros necesarios para poder trabajar. En nuestro caso necesitamos configurar el tipo de modulo, información de red y el broker MQTT.



Hardware ESP8266

Pines disponibles

El ESP8266 tiene 17 pines GPIO (0-16) pero varios están reservados o tienen restricciones. No se debe utilizar ninguno de los pines reservados. Si no, podría bloquear el programa. En el ESP8266, se usan seis pines (GPIO 6 - 11) para interconectar la memoria flash

GPIO 1 y 3 se utilizan como TX y RX del puerto serial del hardware (UART), por lo que en la mayoría de los casos, no puede usarlos como I / O normales mientras se envían / reciben datos en serie.

Boot modes

Algunos pines de I/O tienen una función especial durante el arranque: seleccionan 1 de los 3 modos de arranque:

GPIO15	GPIO0	GPIO2	Mode
0V	0V	3.3V	Uart Bootloader
0V	3.3V	3.3V	Boot sketch (SPI flash)
3.3V	x	x	SDIO mode (not used for Arduino)

Nota: no es necesario agregar una resistencia de extracción externa a GPIO2, la interna está habilitada en el arranque.

Tenemos que asegurarnos de que se cumplan estas condiciones agregando resistencias externas, o el fabricante de la placa de su placa las ha agregado para usted. Esto tiene algunas implicaciones, sin embargo:

GPIO15 siempre está bajo, por lo que no puede usar la resistencia interna de pull-up. Debe tener esto en cuenta cuando utilice GPIO15 como entrada para leer un interruptor o conectarlo a un dispositivo con una salida de colector abierto (o de drenaje abierto), como I²C. GPIO0 se coloca alto durante el funcionamiento normal, por lo que no puede usarlo como una entrada Hi-Z. GPIO2 no puede ser bajo en el arranque, por lo que no se puede conectarle un interruptor. Las resistencias internas pull-up / -down GPIO 0-15 tienen una resistencia pull-up incorporada, como en un Arduino. GPIO16 tiene una resistencia desplegable incorporada.

PWM

El ESP8266 no admite hardware PWM, sin embargo, el software PWM es compatible con todos los pines digitales. El rango predeterminado de PWM es de 10 bits a 1 kHz, pero se puede cambiar (hasta > 14-bit@1 kHz).

Entrada analógica

El ESP8266 tiene una sola entrada analógica, con un rango de entrada de 0 - 1.0V. Si suministra 3.3V, por ejemplo, dañará el chip. Puede usarse un potenciómetro como divisor de voltaje.

El ADC (convertidor analógico a digital) tiene una resolución de 10 bits.

Digital I/O

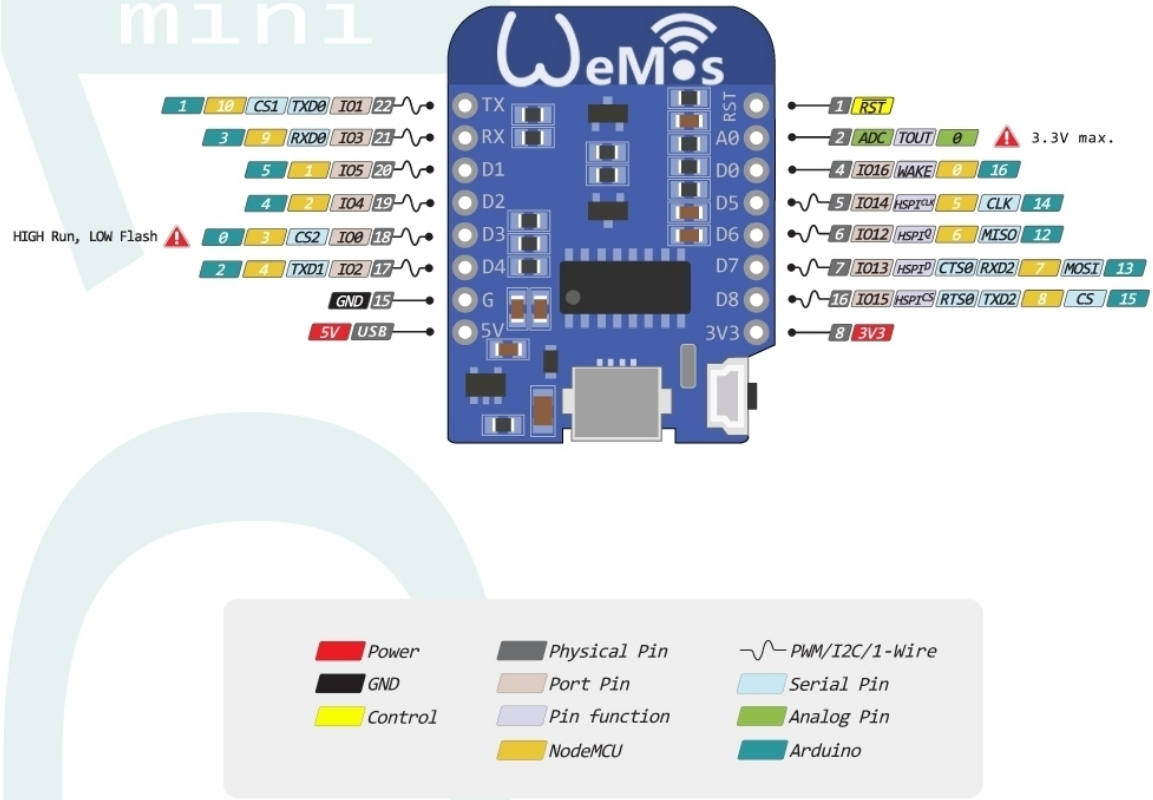
Al igual que un Arduino normal, el ESP8266 tiene pines de entrada / salida digital (I/O o GPIO, pines de entrada / salida de uso general). Como su nombre lo indica, se pueden usar como entradas digitales para leer un voltaje digital, o como salidas digitales para generar 0V (corriente de sumidero) o 3.3V (corriente de fuente).

GPIO overview

NodeMCU Labelled Pin	GPIO	Function	State	Restrictions
D3	0	Boot mode select	3.3V	No Hi-Z
D10	1	TX0	-	Not usable during Serial transmission
D4	2	Boot mode select TX1	3.3V (boot only)	Don’t connect to ground at boot time Sends debug data at boot time
D9	3	RX0	-	Not usable during Serial transmission
D2	4	SDA (I²C)	-	-
D1	5	SCL (I²C)	-	-
x	6 - 8	Flash connection	x	Not usable, and not broken out
x	9, 10	Flash connection *		* Only available on the ESP8285
x	11	Flash connection	x	Not usable, and not broken out
D6	12	MISO (SPI)	-	-
D7	13	MOSI (SPI)	-	-
D5	14	SCK (SPI)	-	-
D8	15	SS (SPI)	0V	Pull-up resistor not usable (extern pull down resistor)
D0	16	Wake up from sleep	-	No pull-up resistor, but pull-down instead Should be connected to RST to wake up

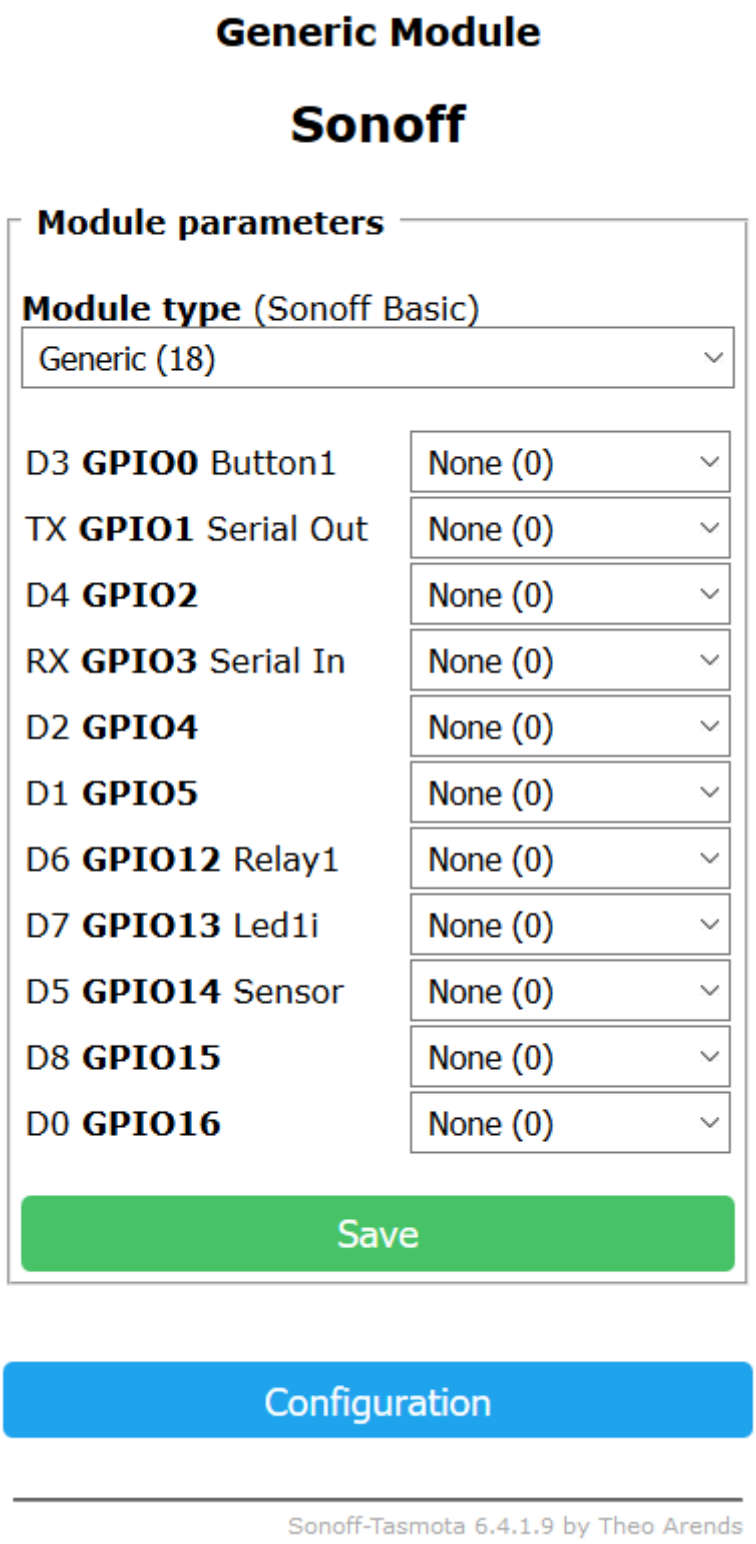
Disposición de pines

La disposicion de pines de la Wemos D1 mini es la siguiente:



Generic module

En la página `Configuration -> Configure Module`, seleccionar **Module Type: "18 Generic"** (para versiones anteriores a la 5.12.0, esto se llamó "Wemos D1 Mini"). Después de guardar la configuración, WEMOS se reinicia con la configuración genérica.

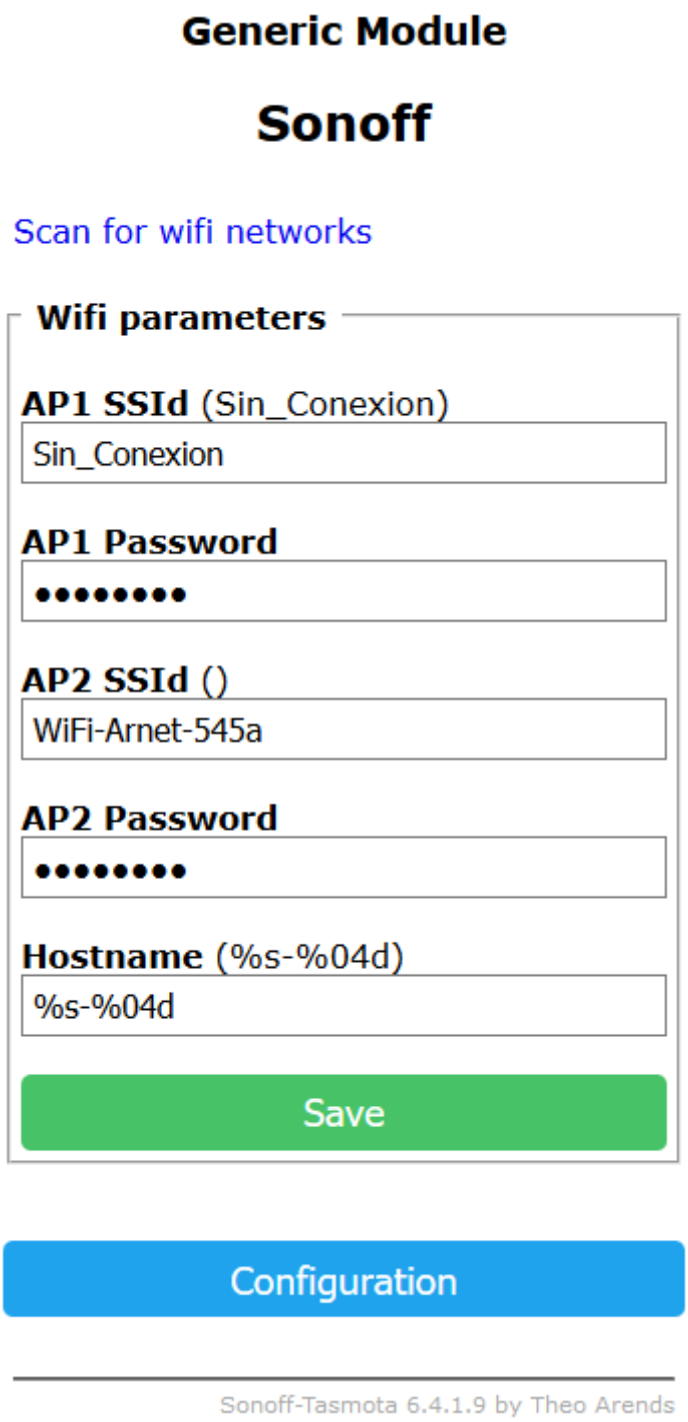


Tambien se puede modificar desde el archivo `user_config.h`. Se define el modulo con:

```
1    #define MODULE          GENERIC
```

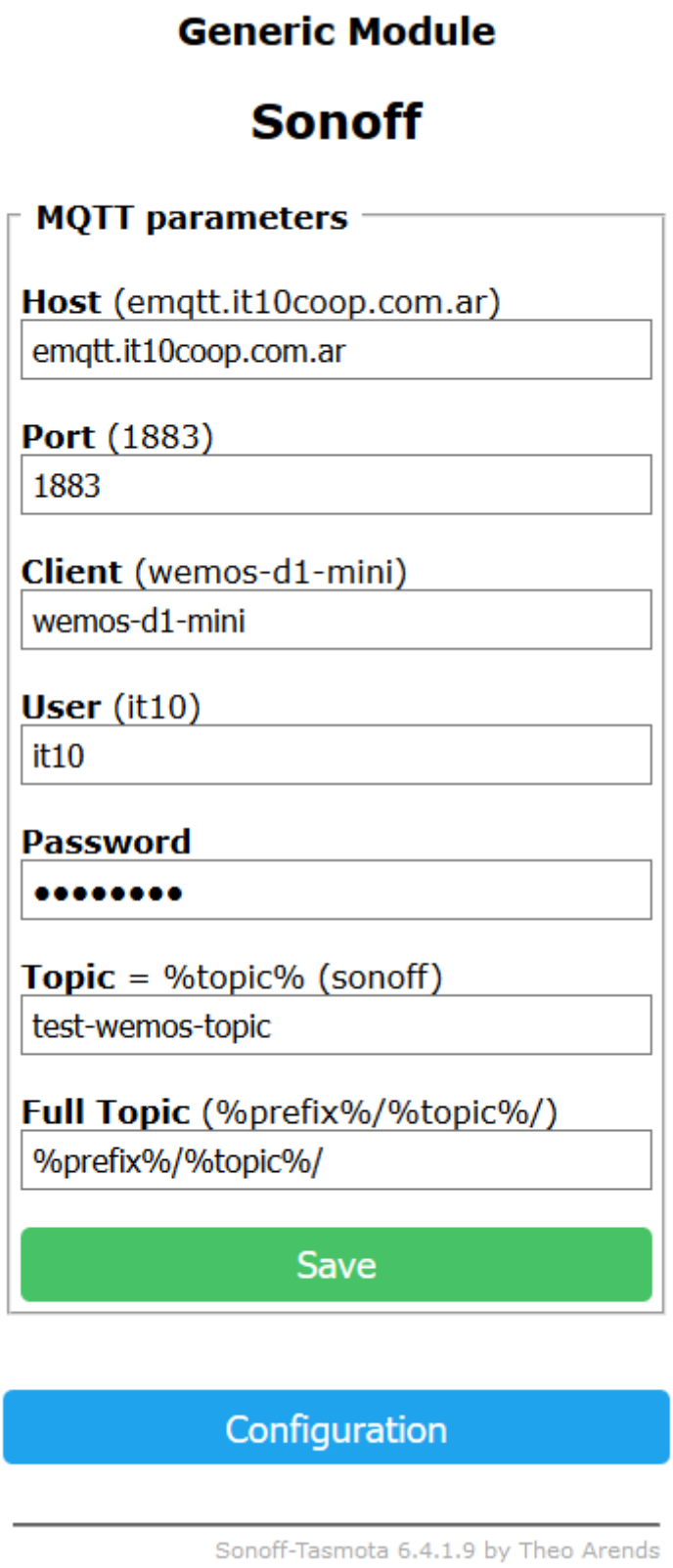
Conexión a una red

Ingresando a `Configure Wifi` se pueden configurar dos redes con sus SSID y contraseñas respectivas. Esto es porque por defecto Tasmota intenta conectarse a la primer red, si falla, prueba con la segunda, para garantizar la conexión wireless en todo momento.



Broker MQTT

Se procede a configurar los parametros para poder interactuar con el dispositivo a tras del protocolo de mensajería MQTT.



Configuración inicial desde `user_config.h`

Tambien se puede modificar el archivo `my_user_config.h` para tener una configuración inicial personalizada sin necesidad de operar el dispositivo desde la WebUI. Se define el modulo para trabajar con Wemos D1 Mini con:

```
1    #define MODULE          GENERIC
```

En cuanto a la información de red:

```
    #define WIFI_IP_ADDRESS        "0.0.0.0"         // [IpAddress1] Set to 0.0.0.0 for using DHCP or enter a static IP address
    #define WIFI_GATEWAY           "192.168.1.1"     // [IpAddress2] If not using DHCP set Gateway IP address
    #define WIFI_SUBNETMASK        "255.255.255.0"   // [IpAddress3] If not using DHCP set Network mask
    #define WIFI_DNS               "192.168.1.1"     // [IpAddress4] If not using DHCP set DNS IP address (might be equal to WIFI_GATEWAY)

    #define STA_SSID1              "[Ssid1]"               // [Ssid1] Wifi SSID
    #define STA_PASS1              "[Password1]"               // [Password1] Wifi password
    #define STA_SSID2              "[Ssid2]"               // [Ssid2] Optional alternate AP Wifi SSID
    #define STA_PASS2              "[Password2]"               // [Password2] Optional alternate AP Wifi password
    #define WIFI_CONFIG_TOOL       WIFI_RETRY        // [WifiConfig] Default tool if wifi fails to connect

```

Para configurar el broker MQTT:

```
    #define MQTT_HOST              "direccion-MqttHost"               // [MqttHost]
    [...]
    #define MQTT_PORT              1883              // [MqttPort] MQTT port (10123 on CloudMQTT)
    #define MQTT_USER              "[MqttUser]"       // [MqttUser] MQTT user
    #define MQTT_PASS              "[MqttPassword]"       // [MqttPassword] MQTT password
    [...]
    #define MQTT_TOPIC             "Nombre-topic"           // [Topic] (unique) MQTT device topic, set to 'PROJECT "_%06X"' for unique topic including device MAC address
```

Luego puede haber mas o menos parametros configurables, pero que es posible hacerlo desde un primer momento antes de compilar y flashear la placa, y de esta forma evitar usar la WebUI.

Pruebas con Tasmota

Para probar el funcionamiento se han desarrollado una serie de escenarios. Estos se eligieron en base a posibles implementaciones a futuro, y capacidad de escalabilidad. Se ven a continuación.

Probando comando Timer por MQTT

Usando un timer se puede ejecutar acciones cuando estos acaben. Para que sea posible utilizarlos se deben programar reglas. Cada regla debe seguir la siguiente sintaxis:

```
on <trigger> do <command> endon
```

on inicia una nueva regla.

es lo que necesita que ocurra para ejecutar el `<command>`

do separador entre `<trigger>` y `<command>`

endon marca el final de la regla. Puede ser seguida por otra regla.

Listado de reglas y comandos en https://github.com/arendst/Sonoff-Tasmota/wiki/Rules.

Es util para generar timers en el que se pueda activar algo y repetirse en todos los dias (con la parte de days, que cada cifra es un dia de la semana, arrancando del domingo) a la misma hora, con mas o menos opciones.

Usando el cliente `mosquitto`, publicando al broker emqtt de la cooperativa al topic cmnd/.../.../ se puede enviar comandos a la WeMos D1 mini. Por ejemplo

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/timer1 -m
'{"Arm":1,"Time":"17:56","Days":"0010000","Output":1,"Action":3}'
```

Muestra direccion IP cada 10 segundos, usando un solo timer

1- Hacer mosquitto_pub a:

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/rule1 -m 'on rules#timer=1 do
IPAddress1;ruletimer1 3 endon on rules#timer=1 do backlog ruletimer1
3;IPAddress1 endon'
```

2- Despues activar regla con on:

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/rule1 -m 'on'
```

3- Luego setear el timer haciendo:

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/ruletimer1 -m "10"
```

El numero 10 es el tiempo con el que arranca el timer, para luego ejecutar la regla. Despues de esto, el timer se renueva de acuerdo a la regla que generamos, que puede ser el mismo tiempo u otro.

NOTA: Si se quiere usar otro timer, se hace lo mismo, publicando a */rule2* la regla (con un *rules#timer=2*, para indicar que finaliza el timer 2 y no otro), activando la regla con "on", y seteando el nuevo timer.

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/rule2 -m 'on rules#timer=2 do
status 6;ruletimer2 30 endon on rules#timer=2 do backlog ruletimer2 30;status
6 endon'
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/rule2 -m "on"
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/ruletimer2 -m "30"
```

Conectando un led a la WeMos d1 mini

Luego de conectar el led, ir a configuracion desde la WebUI, en el **D1 GPIO5** seleccionamos **Led(52)**.

Para enviar el comando por mqtt:

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/LedPower -m '1'
```

El mensaje es "1" u "ON" para encender el LED, o "0" u "OFF" para apagarlo.

Encedido/apagado periodico de LED

Ahora probemos usar reglas para hacer un encedido/apagado periódico del LED: 1- Crear regla

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/rule3 -m 'on rules#timer=1 do
LedPower 1 endon on rules#timer=1 do backlog ruletimer1 3;LedPower 0 endon'
```

2- Activar regla

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/rule3 -m 'on'
```

3- Lanzar timer

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/ruletimer1 -m '3'
```

Para parar los performs timer, que sucederan de acuerdo a las veces que ponga en una regla:

```
"on rules#timer1 do [comando]"
```

Para borrar completamente la accion de un ruletimer se debe hacer con ingresando la linea anterior con los comandos que hagan falta.

Por MQTT esto se puede hacer, por ejemplo para desactivar el timer de esta prueba:

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/ruletimer1 -m "LedPower 0"
```

O sino:

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/backlog -m "ruletimer1 3;LedPower
0"
```

Modulando la intensidad de brillo de un LED con PWM

Teniendo conectado el LED, se debe cambiar el dispositivo que esta esta seteado para el pin **D1 GPIO5**, que actualmente es **Led (52)**. Ahora lo modificamos poniendo **PWM1 (37)**. De no hacer esto, todos los comandos utilizados para modificar parametros de PWM no funcionarán.

Una vez cambiado el dispositivo ya sea via web o por mqtt se procede a habilitar el uso de PWM. Con MQTT se hace:

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/SetOption15 -m "0"
```

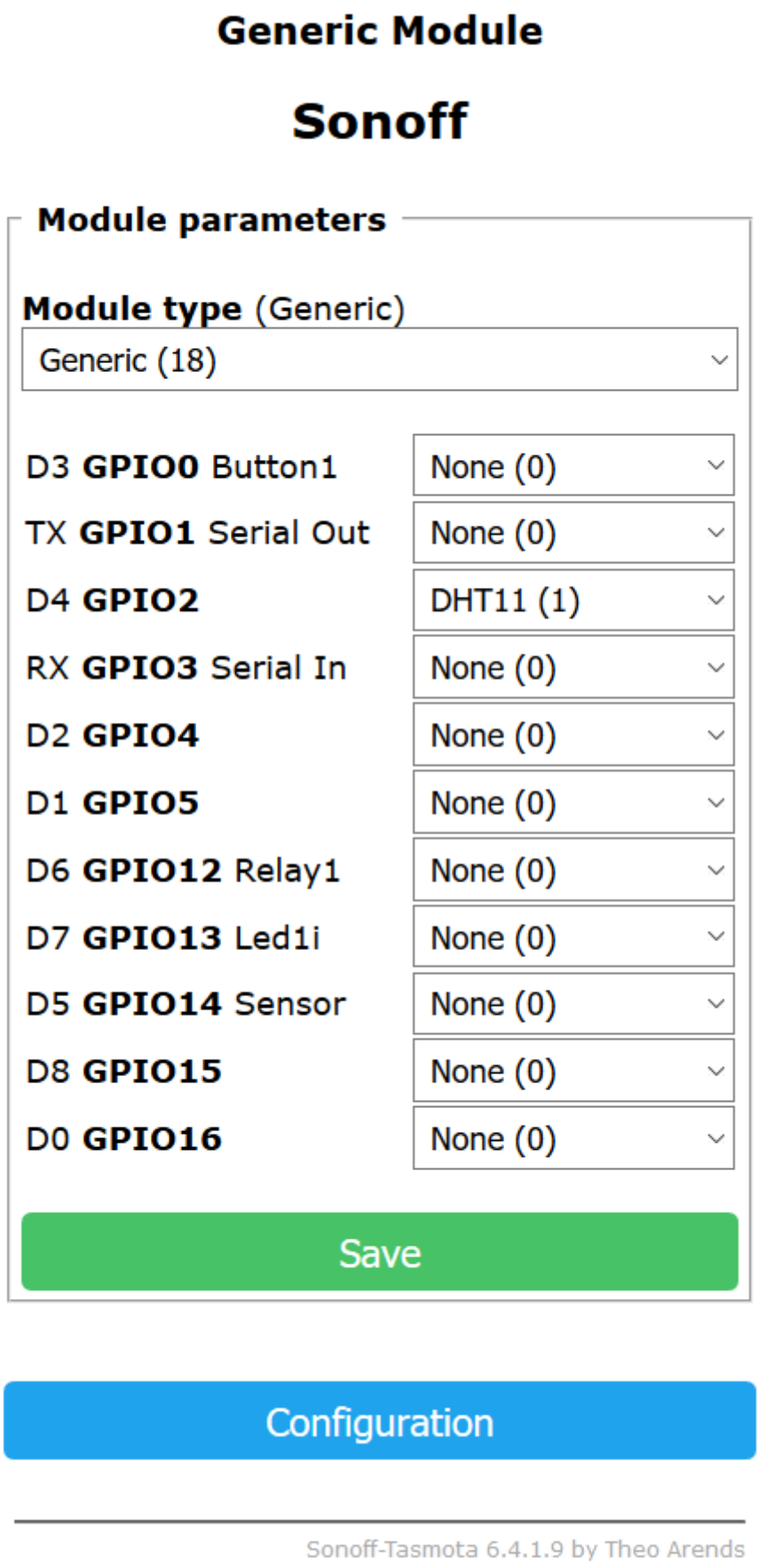
Luego se puede modificar el valor digital que sale por el pin. Este es un valor de 10 bits, entonces el rango va desde "0" a "1023". De todas formas este rango se puede modificar. Con MQTT es:

```
mosquitto_pub -h emqtt.it10coop.com.ar -p 1883 -t
cmnd/sensar/ppstest/tasmota-wemos-d1-mini/PwmRange -m "[1-1023]"
```

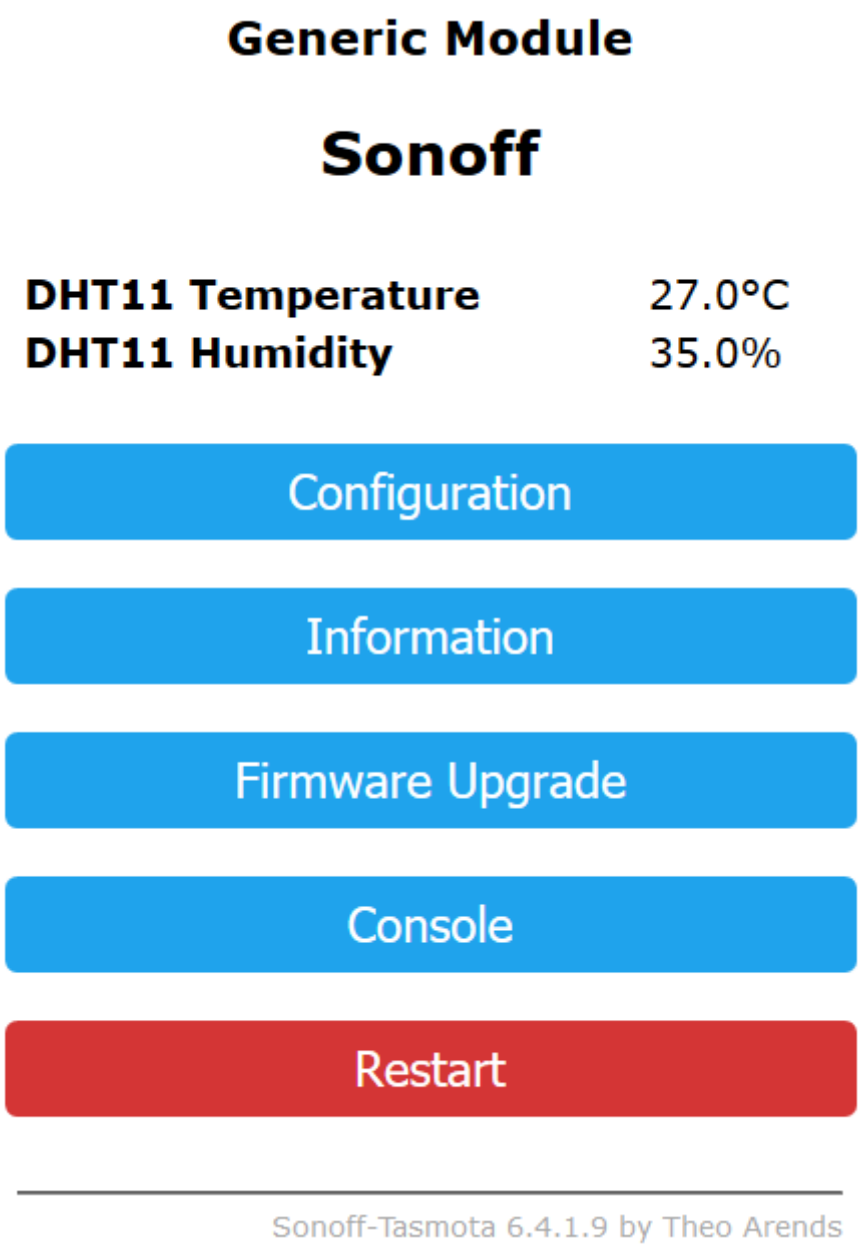
Los valores posibles van de 255 a 1023. Modificar el rango maximo hace que los valores que vaya tomando se van mapeando al rango total de 0-1023. Esto quiere decir que si pongo como valor maximo 255, la intensidad del LED será máxima, como si fuera 1023, y si el valor que se le da es de 75, estara dentro del rango, y brillará con una intensidad media, como si estuviera seteado en un valor de 512 para el rango de 0-1023.

Conectando sensores a Wemos D1 Mini

Es tan simple como ir a la configuración del modulo, y seleccionar **DHT11 (1)** en el pin **D4 GPIO2**.



Guardamos y ya podemos verlo funcionando.



Conexion Serial con Edukit10

Edukit es un proyecto de hardware libre basado en Arduino, desarrollado por la Cooperativa IT10, para la enseñanza de robótica. Es una tecnología que permite acercar de una manera sencilla los fundamentos de la robótica y aquellos conceptos técnicos que se vuelven complejos para trabajarlos de la manera tradicional. Se compone de una serie de elementos electrónicos que trabajan en conjunto y permiten investigar y diseñar pequeños robots pedagógicos de forma sencilla, reciclando componentes. El software utilizado para la programación es libre y multiplataforma.

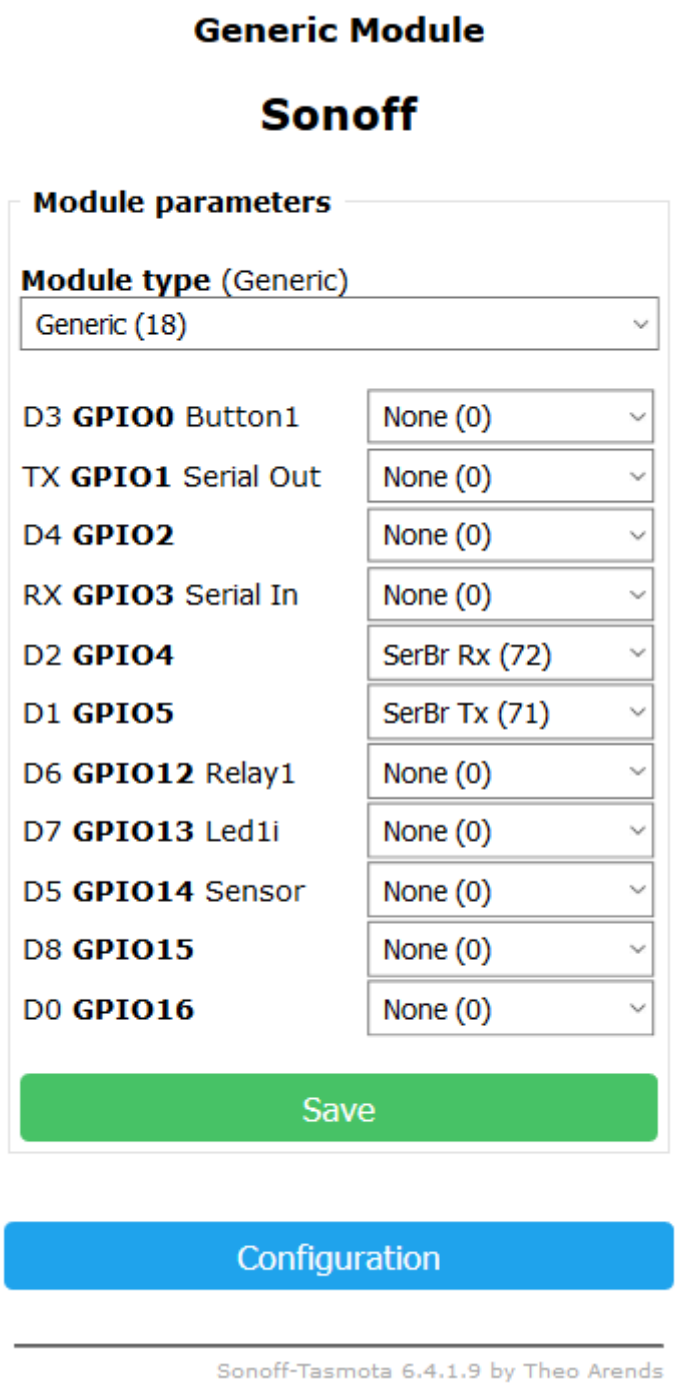
Se pretende lograr una conexión serial entre la Wemos D1 mini y la Edukit. Las configuraciones correspondientes se detallan a continuación.

Configurando Wemos D1 mini

El ESP8266 tiene dos UARTS (puertos serie): UART0 en los pines 1 y 3 (TX0 y RX0 respectivamente), Y UART1 en los pines 2 y 8 (TX1 y RX1 respectivamente). Sin embargo, GPIO8 se utiliza para conectar el chip flash . Esto significa que UART1 solo puede transmitir datos.

UART0 también tiene control de flujo de hardware en los pines 15 y 13 (RTS0 y CTS0 respectivamente). Estos dos pines también pueden usarse como pines alternativos TX0 y RX0.

Se realiza una conexión serial por software, y para esto se utilizan pines digitales. En este caso GPIO5(D1) y GPIO4(D2) (SerBr Tx y SerBr Rx respectivamente).



Solo es posible hacerlo a traves de web y luego guardando la configuracion como back up. https://github.com/arendst/Sonoff-Tasmota/issues/5068

Configurando Edukit

De acuerdo al esquematico de Edukit, se usan los pines digitales 4 y 5, que se corresponden con los pines D10 y D9 respectivamente. Con estos pines, y usando la librería `SoftwareSerial.h` creamos un enlace serial por software entre la Wemos D1 mini y Edukit.

Envío de comandos y mensajes por conexión serial

Tasmota no tiene integrada una funcionalidad que permita la recepción de comandos por serial. Esto es algo que nos es muy util, ya que al conectar sensores a la Edukit, esta podría enviar datos obtenidos, estado de los mismos, etc. Además que se sería provechoso poder modificar ciertos parámetros de Tasmota desde Edukit.

Dicho esto y viendo que el desarrollador del firmware no ha tenido en mente agregar esta caracterísa se optó por modificar el codigo fuente agregándola. A continuación se especifica el proceso.

SSerialReceived

Tasmota muestra por consola los mensajes por software serial de la siguiente forma:

```
MQT: tele/sensar/ppstest/tasmota-wemos-d1-/RESULT =
{"SSerialReceived":"MENSAJE"}
```

Solamente se puede mostrar pero resulta imposible generar alguna accion por medio de la conexión serial.

MqttPublishSerial

De acuerdo a la librería **PubSubClient**, que es la utilizada por Tasmota para publicar y suscribirse, existe método llamado `publish`:

```
boolean PubSubClient::publish(const char* topic, const char* payload) {
    return publish(topic,(const uint8_t*)payload,strlen(payload),false);
    }
```

Basicamente lo que hace es recibir como argumento un **topic** y el **payload** correspondiente para publicar. Valiendonos de esto, podriamos crear una funcion que de acuerdo a un mensaje serial se pueda parsear en dos partes, **topic;payload**, y publicar al broker.

En el archivo `sonoff/xdrv_02_mqtt.ino` estan todas las funciones que se utilizan para ejecutar el protocolo MQTT. Allí se especifica que utiliza la libreria `PubSubClient.h`. Se incluye en el codigo la siguiente funcion:

```
//################# FUNCION PARA ENVIAR MENSAJES DEL SERIAL POR MQTT
##################
void MqttPublishSerial(const char* carga_serial)
{
  #define STRSIZE 80
  const char * serial = carga_serial;
  char topic[STRSIZE];
  char payload[STRSIZE];
  int i=0;
  int j=0;
  while (serial[i] != ';')
  {
      topic[i]=serial[i];
      i++;
  }
  topic[i]='\0';
  i++;
  while (serial[i] != '\0')
  {
      payload[j]=serial[i];
      i++;
      j++;
  }
  payload[j]='\0';

  MqttClient.publish(topic, payload);
  //MqttClient.publish("tele/sensar/ppstest/tasmota-wemos-d1-",serial);
  Serial.printf("Mensaje enviado!, %s , %s",topic, payload);
```

```
}
//#########################################################################
#############
```

Esa una función que no devuelve ningun valor. Recibe como argumento el mensaje serial que llega desde Edukit a traves del Bridge Serial de Tasmota. La idea aqui es parsear el String que llega y asignarlo a 2 nuevos strings como topic y payload.

El string `carga_serial` tiene un delimitador que es un punto y coma (;). Este marca el fin del topic y el comienzo del payload correspondiente. Se recorre el string `serial` con un *while* hasta que encuentra ese delimitador, a medida que va guardando caracter por caracter en **topic**. Luego se agrega un **NULL** al final de la cadena. Luego se procede a hacer lo mismo con **payload**. Finalmente se utiliza el metodo `publish` antes descripto sobre el objeto **MqttClient**, inicializado anteriormente en el codigo. Además se envia un mensaje por el Hardware Serial de Tasmota, indicando el mensaje que se ha enviado.

MqttPublishTeleSerial

Esta función se crea con el objeto de que cuando se recibe un comando o un mensaje para publicarse con MQTT que proviene del Software Serial este pueda ser visualizado en la consola de Tasmota. Tambien se declara en `sonoff/xdrv_02_mqtt.ino`. A continuación describimos su codigo:

```
//################ FUNCION PARA ENVIAR MENSAJES A TELE POR MQTT CUANDO
RECIBO SERIAL ##################
void MqttPublishTeleSerial(uint8_t prefix, const char* subtopic, boolean
retained)
{
/* prefix 0 = cmnd using subtopic
 * prefix 1 = stat using subtopic
 * prefix 2 = tele using subtopic
 * prefix 4 = cmnd using subtopic or RESULT
 * prefix 5 = stat using subtopic or RESULT
 * prefix 6 = tele using subtopic or RESULT
 */
  char romram[33];
  char stopic[TOPSZ];

  snprintf_P(romram, sizeof(romram), ((prefix > 3) &&
!Settings.flag.mqtt_response) ? S_RSLT_RESULT : subtopic);
  for (byte i = 0; i < strlen(romram); i++) {
    romram[i] = toupper(romram[i]);
  }
  prefix &= 3;
  GetTopic_P(stopic, prefix, mqtt_topic, romram);

  char mensaje[] = "Mensaje enviado!: ";

  char combined[100] = {0};

  strcat(combined, mensaje);
  strcat(combined, stopic);

  MqttPublish(combined, false);
}
//#########################################################################
############
```

Es muy parecida a la función `void MqttPublishPrefixTopic_P(uint8_t prefix, const char* subtopic, boolean retained)`. Lo que se agrega efectivamente es:

```
  char mensaje[] = "Mensaje enviado!: ";

  char combined[100] = {0};

  strcat(combined, mensaje);
  strcat(combined, stopic);

  MqttPublish(combined, false);
```

Una leyenda que aparezca inicialmente indicando el mensaje enviado. Se combina con la cadena que determina el topic a donde se publica para visualizar mensajes por consola. Se basa en uno de los 3 prefijos utilizados en Tasmota, **tele**, **cmnd** y **stat**. En este caso se usa tele, generalmente utilizado para enviar mensaje periodicos de estado de sensores, información y estado de la placa, características activadas, etc.

Proceso de ejecución

En el archivo `sonoff/xdrv_08_serial_bridge.ino`, bajo la función `void SerialBridgeInput(void)`, que es la que declara que hacer con las entradas por serial, es donde modificamos para utilizar las funciones creadas anteriormente. A continuación se describe una parte del codigo:

```
[...]
  if (serial_bridge_in_byte_counter && (millis() > (serial_bridge_polling_window + SERIAL_POLLING))) {
    serial_bridge_buffer[serial_bridge_in_byte_counter] = 0;  // serial data completed
    snprintf_P(mqtt_data, sizeof(mqtt_data), PSTR("{\"" D_JSON_SSERIALRECEIVED "\":\"%s\"}"), serial_bridge_buffer);
    MqttPublishPrefixTopic_P(RESULT_OR_TELE, PSTR(D_JSON_SSERIALRECEIVED));
//    XdrvRulesProcess();

//######################## RECEPCION Y PUBLICACION MQTT DE MENSAJES POR SERIAL ##########################################

    if (serial_bridge_buffer != 0){
      MqttPublishSerial(serial_bridge_buffer);
      snprintf_P(mqtt_data, sizeof(mqtt_data), PSTR("%s"), serial_bridge_buffer);
      MqttPublishTeleSerial(RESULT_OR_TELE, PSTR(D_JSON_SSERIALRECEIVED), false);
    }

//#############################################################################################################
    serial_bridge_in_byte_counter = 0;
  }
```

Cuando ya esta disponible el Serial Bridge, si hay datos que estan llenando un buffer de serial se ejecutan las funciones correspondientes para indicar por consola que es lo que se envio. Debajo de eso, de manera secuencial cargamos nuestras funciones.

```
//####################### RECEPCION Y PUBLICACION MQTT DE MENSAJES POR SERIAL ##########################################

    if (serial_bridge_buffer != 0){
      MqttPublishSerial(serial_bridge_buffer);
      snprintf_P(mqtt_data, sizeof(mqtt_data), PSTR("%s"), serial_bridge_buffer);
      MqttPublishTeleSerial(RESULT_OR_TELE, PSTR(D_JSON_SSERIALRECEIVED), false);
    }

//#########################################################################################################
```

Si el buffer se esta llenando, ejecutamos `MqttPublishSerial` con lo que haya en ese buffer. Luego utilizamos `MqttPublishTeleSerial` al igual que `MqttPublishPrefixTopic_P`, pero en este caso para indicar por consola que ha sido enviando un mensaje serial al broker MQTT.

Probando las nuevas funciones

Enviar dato de temperatura

Se intentará enviar datos de temperatura periodicamente por serial y publicando al broker en un topic distinto al que usa Tasmota. El dato es tomado por la Edukit, que dispone de un sensor **LM35CZ**.

Sketch Arduino de Edukit

```
    #include <SoftwareSerial.h>

    SoftwareSerial mySerial(10, 9); // RX, TX

    int lecturaADC = 0;
    double voltajeLM35 = 0.0;
    double TemperaturaLM35 = 0.0;
    unsigned long tiempo1 = 0;
    unsigned long tiempo2 = 0;

    void setup() {
      // Open serial communications and wait for port to open:
      Serial.begin(9600);
      while (!Serial) {
        ; // wait for serial port to connect. Needed for native USB port only
      }
      Serial.println("Inicio!");
      // set the data rate for the SoftwareSerial port
      mySerial.begin(9600);

      tiempo1=millis();

    }

    void loop() {

      //#1 Lectura del canal 0 del ADC, recordando que el ADC de Arduino es de 10 Bits, el resultado se guardara en una variable int (16 bits).
```

```
      lecturaADC = analogRead(A6);

      /*
       * Convertir el valor binario a un voltaje que se guardara en una variable del programa, dado que la variable lecturaADC es entera, la operación lecturaADC/1023 regresará un valor entero.

       tiempo2=millis();
       if(tiempo2 > (tiempo1+(10000))){
        tiempo1 = millis(); //Actualiza el tiempo actual

        //Opcion (b)
        voltajeLM35 = ((double)lecturaADC/1023)*5;

        TemperaturaLM35 = voltajeLM35/0.01;
        Serial.print("El valor en temperatura de salida en el sensor es: ");
        Serial.print(TemperaturaLM35,2); //Imprime la variable double con 2 decimales
        Serial.println(" °C");

        char temp[5];
        dtostrf(TemperaturaLM35, 5, 2, temp);
        //(variable float, ancho, cantidad de decimales, variable char)

        String leyenda= "El valor de temperatura es: ";
        String grados= " °C";

        String mensaje = leyenda + temp + grados;
        String topic = "sensar/ppstest/;";

        pub_mqtt(topic, mensaje);
       }

      if (mySerial.available()) {
        Serial.write(mySerial.read());
      }
      if (Serial.available()) {
        ;
      }
    }
    //########################----------------FUNCIONES--------------------########################

    void pub_mqtt(String topic, String payload) //que sea recurrente
    {
      String a;
      a = topic + payload;
      Serial.println(a);
      mySerial.println(a);
    }
```

Uso de memoria de Tasmota

Segun Visual Studio Code:

```
DATA:    [======    ]  58.9% (used 48268 bytes from 81920 bytes)
PROGRAM: [=====     ]  51.7% (used 529464 bytes from 1023984 bytes)
```

Segun Tasmota:

```
ESP Chip Id 7934606
Flash Chip Id   0x164020
Flash Size  4096kB
Program Flash Size  1024kB
Program Size    521kB
Free Program Space  480kB
Free Memory 27kB
```

ANEXO IV

ESPurna

Instalando Espurna

1. Borrar la flash: Para esto hay que tener esptool, que esta en python. Si lo hacemos desde linux seguramente hay que instalar Pyserial, pero primero instalar pip. En la terminal:

```
curl -O https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
```

Después instalar Pyserial:

```
pip install Pyserial
```

Luego cambiar permisos a /dev/ttyUSB0

Por ultimo:

```
esptool.py --port /dev/ttyUSB0 erase_flash
```

2. Cargar nuevo firmware: Vamos a usar Visual Studio Code, se descarga, se instala. También hace falta Node.js y npm. Seguir pasos de la web: https://nodejs.org/en/download/

Ya en VSC, abrir terminal, haciendo `crtl+shift+P` y escribir Terminal: Create New Integrated Terminal. Ya en la terminal poner:

```
npm install --global gulp-cli
```

3. Instalar extensiones PlatformIO IDE, opcional python y gitlens.
4. Clone https://github.com/xoseperez/espurna.git Cargar la carpeta espurna como proyecto, pararse en espurna/code. Así arranca PlatformIO automáticamente
5. ir a platformio.ini, y modificar el env_default por este: `env_default = wemos-d1mini` El anterior lo comentamos por las dudas

Conectarse a la WeMos D1 mini

6. Conectarse al red de ESPURNA-XXXXX. La pass es fibonacci.
7. Ir al explorador e ingresar a 192.168.4.1. Ingresar con admin:fibonacci. Pedirá cambiar la contraseña, le ponemos **It10it10**. La guardamos. Se va a desconectar de la red Wifi. Borramos la información de la red para poder ingresar con la nueva contraseña.
8. Ingresar nuevamente a 192.168.4.1 con la pass nueva.
9. Ir a WIFI, agregar las redes que hagan falta, por ejemplo: `Sin_Conexion/IT1Ocoop.com.ar/192.168.20.200/192.168.20.1/8.8.8.8`
10. Guardar, reconectar y conectarse a esta red ingresada, ya que el dispositivo va a estar ahí.
11. Ingresar a la IP del dispositivo que ingresamos antes, por ejemplo 192.168.20.200

12. Para modificar mqtt una cosa distinta a TASMOTA es que pide QoS, retain y keep alive. Ademas hay que decir si se usa JSON como payload.

El root topic es: sensar/ppstest/espurna

13. Instalar node-red para interactuar con dispositivo. Seguir el tutorial hasta el paso 4(no incluido)(Despues revisar el resto para securizar la conexión): https://www.digitalocean.com/community/tutorials/how-to-connect-your-internet-of-things-with-node-red-on-ubuntu-16-04

Modificando los parámetros para una configuración directa

Podemos modificar los parámetros iniciales para no tener que realizarlos luego via web. Para esto hay que ir a `code/espurna/config/general.h`, donde modificaremos los aspectos:

- Contraseña de administrador Cambiamos la contraseña:

```
#define ADMIN_PASS              "fibonacci"     // Default password (WEB, OTA, WIFI SoftAP)
```

A la siguiente:

```
#define ADMIN_PASS              "It10it10"     // Default password (WEB, OTA, WIFI SoftAP)
```

- Paramtetros de red WiFi En la seccion de wifi, modificamos donde corresponde con:

```
// Optional hardcoded configuration (up to 2 networks)
#ifndef WIFI1_SSID
#define WIFI1_SSID                  "Sin_Conexion"
#endif

#ifndef WIFI1_PASS
#define WIFI1_PASS                  "IT10coop.com.ar"
#endif

#ifndef WIFI1_IP
#define WIFI1_IP                    "192.168.20.200"
#endif

#ifndef WIFI1_GW
#define WIFI1_GW                    "192.168.20.1"
#endif

#ifndef WIFI1_MASK
#define WIFI1_MASK                  "255.255.255.0"
#endif

#ifndef WIFI1_DNS
#define WIFI1_DNS                   "8.8.8.8"
#endif
```

- Web Para que no nos solicite cambiar la contraseña de administrador nuevamente, ponemos un cero en la linea:

```
#define WEB_FORCE_PASS_CHANGE       1         // Force the user to change
the password if default one
```

- MQTT

Debemos habilitar la posibilidad de comunicarnos por MQTT. Ponemos un 1 en la línea:

```
#define MQTT_ENABLED                0             // Do not enable MQTT
connection by default
```

Luego hay que brindar información del broker, usuario, contraseña y root topic. Se hace como sigue:

```
#ifndef MQTT_SERVER
#define MQTT_SERVER                 "emqtt.it10coop.com.ar"             //
Default MQTT broker address

#endif

#ifndef MQTT_USER
#define MQTT_USER                   "it10"              // Default MQTT
broker usename
#endif

#ifndef MQTT_PASS
#define MQTT_PASS                   "it10"              // Default MQTT
broker password
#endif

#ifndef MQTT_PORT
#define MQTT_PORT                   1883            // MQTT broker port
#endif

#ifndef MQTT_TOPIC
#define MQTT_TOPIC                  "sensar/ppstest/espurna"    // Default
MQTT base topic
#endif
```

- Hostname En el archivo `defaults.h` en la misma carpeta modificamos la línea con el nombre que queremos. Si queda vacío por defecto el nombre que tiene es ESPURNA-XXXXXX (con los 3 últimos octetos del chipID). Cambiamos el nombre de esta manera:

```
#define HOSTNAME                "wemos-d1-mini"
```

MQTT

Otra información interesante del archivo `general.h`, son las partículas que se pueden concatenar al topic para interactuar con la WeMos d1 mini:

```
// These particles will be concatenated to the MQTT_TOPIC base to form the
actual topic
#define MQTT_TOPIC_JSON             "data"
#define MQTT_TOPIC_ACTION           "action"
#define MQTT_TOPIC_RELAY            "relay"
#define MQTT_TOPIC_LED              "led"
```

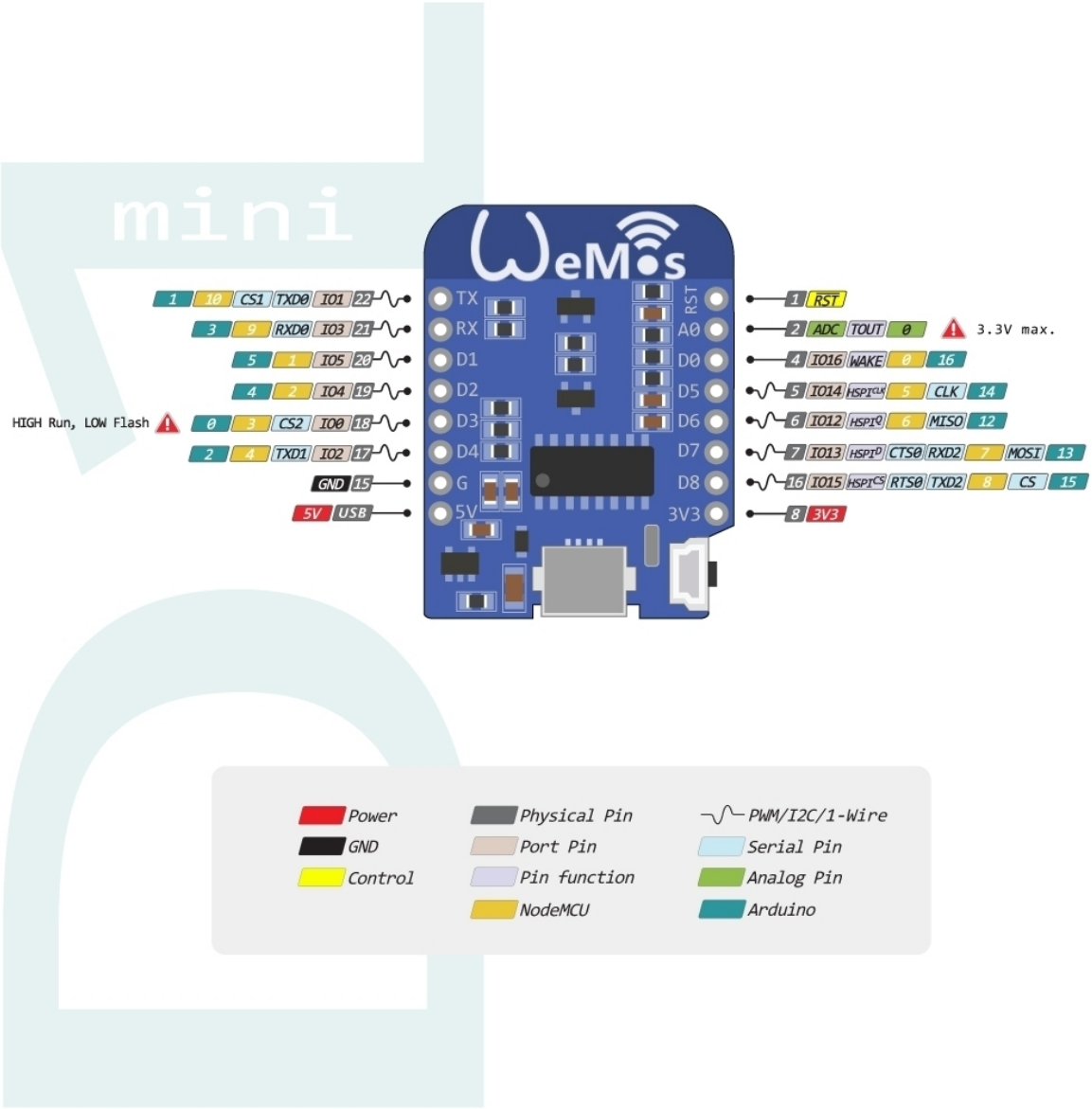
```
#define MQTT_TOPIC_BUTTON           "button"
#define MQTT_TOPIC_IP               "ip"
#define MQTT_TOPIC_SSID             "ssid"
#define MQTT_TOPIC_VERSION          "version"
#define MQTT_TOPIC_UPTIME           "uptime"
#define MQTT_TOPIC_DATETIME         "datetime"
#define MQTT_TOPIC_FREEHEAP         "freeheap"
#define MQTT_TOPIC_VCC              "vcc"
#define MQTT_TOPIC_STATUS           "status"
#define MQTT_TOPIC_MAC              "mac"
#define MQTT_TOPIC_RSSI             "rssi"
#define MQTT_TOPIC_MESSAGE_ID       "id"
#define MQTT_TOPIC_APP              "app"
#define MQTT_TOPIC_INTERVAL         "interval"
#define MQTT_TOPIC_HOSTNAME         "host"
#define MQTT_TOPIC_TIME             "time"
#define MQTT_TOPIC_RFOUT            "rfout"
#define MQTT_TOPIC_RFIN             "rfin"
#define MQTT_TOPIC_RFLEARN          "rflearn"
#define MQTT_TOPIC_RFRAW            "rfraw"
#define MQTT_TOPIC_UARTIN           "uartin"
#define MQTT_TOPIC_UARTOUT          "uartout"
#define MQTT_TOPIC_LOADAVG          "loadavg"
#define MQTT_TOPIC_BOARD            "board"
#define MQTT_TOPIC_PULSE            "pulse"
#define MQTT_TOPIC_SPEED            "speed"
#define MQTT_TOPIC_IRIN             "irin"
#define MQTT_TOPIC_IROUT            "irout"
#define MQTT_TOPIC_OTA              "ota"

// Light module
#define MQTT_TOPIC_CHANNEL          "channel"
#define MQTT_TOPIC_COLOR_RGB        "rgb"
#define MQTT_TOPIC_COLOR_HSV        "hsv"
#define MQTT_TOPIC_ANIM_MODE        "anim_mode"
#define MQTT_TOPIC_ANIM_SPEED       "anim_speed"
#define MQTT_TOPIC_BRIGHTNESS       "brightness"
#define MQTT_TOPIC_MIRED            "mired"
#define MQTT_TOPIC_KELVIN           "kelvin"
#define MQTT_TOPIC_TRANSITION       "transition"

#define MQTT_STATUS_ONLINE          "1"         // Value for the device ON
message
#define MQTT_STATUS_OFFLINE         "0"         // Value for the device OFF
message (will)

#define MQTT_ACTION_RESET           "reboot"    // RESET MQTT topic particle
```

Pruebas con ESPurna y WeMos D1 mini

No hay suficiente documentación como para realizar pruebas similares a las que se hacen con Tasmota, esto es, programar temporizadores o triggers para realizar tareas, por ende realizaremos algo similar usando **Node-Red** y aportando estas funcionalidades desde ahí y no desde el firmware. La imposibilidad de usar la opción `Schedule` se puede deber a que el firmware instalado es solo para WeMos d1 mini, y no el relayshield WeMos d1 mini.

Disposición de pines

La disposición de pines de la WeMos D1 mini es la siguiente:



Probando conectar un LED

Cambiando el archivo `hardware.h` en la parte de la wemos-d1-mini. Agregamos lineas:

```
#define LED2_PIN            5
#define LED2_MODE           LED_MODE_MQTT
#define LED2_PIN_INVERSE    0
```

Luego configurar como antes despues de cargado el firmware. Para tener en cuenta, poner en la parte de `General`, en `LED Mode` seleccionar **"MQTT Managed"**, para poder modificar el estado con mensajes MQTT. Si enviamos un 0, se apaga el led; con 1 se enciende; con 2 alterna el estado y cambia de acuerdo al que tiene actualmente (si esta on, se pone off, y viceversa).

Enviando mensajes con Node-RED

Una vez abierta la interfaz se colocan dos bloques:

- input/inject
- output/mqtt

El primero es para enviar un string con el valor que quiero para modificar el estado. En nuestro caso enviaremos un "2", asi el led alterna entre on/off. Esto se envia al topic: **sensar/ppstest/espurna/led/0/set**, y pedimos que se repita con intervalo de 1 segundo.

Luego, con el bloque MQTT configuramos el broker, con el mismo topic y por ultimos hacemos DEPLOY. El led alternara entre on/off a traves de mensajes MQTT.

Conectando un DHT11

Primero se debe habilitar el soporte para el sensor en `code/espurna/config/sensor.h`. Se debe modificar estas lineas en la seccion `DHTXX temperature/humidity sensor`:

```
#define DHT_SUPPORT                     0
```

por

```
#define DHT_SUPPORT                     1
```

y

```
#define DHT_TYPE                        DHT_CHIP_DHT22
```

por

```
#define DHT_TYPE                        DHT_CHIP_DHT11
```

El pin no lo modificamos, es el GPIO 14, que en la WeMos D1 mini la identificamos como D5.

Además en el archivo `platformio.ini` hay que agregar los flags donde se refiere a nuestra placa WeMos D1 mini los siguientes flags

```
build_flags = ${common.build_flags} -DWEMOS_D1_MINI -DDEBUG_FAUXMO=Serial -DNOWSAUTH -DDHT_SUPPORT=1 -DDHT_PIN=14 -DDHT_TYPE="DHT_CHIP_DHT11"
```

Los últimos tres indican que se usará un sensor DHT, el pin 14 y el sensor DHT11.

Luego, se puede modificar el el intervalo de tiempo y la cantidad de mediciones que se desean obtener cambiando los parámetros siguientes de `sensor.h`:

```
//
================================================================================
// SENSORS - General data
//
================================================================================

#define SENSOR_DEBUG                    0               // Debug sensors

#define SENSOR_READ_INTERVAL            6               // Read data
from sensors every 6 seconds

#define SENSOR_READ_MIN_INTERVAL        1               // Minimum read
interval
```

```
#define SENSOR_READ_MAX_INTERVAL        3600        // Maximum read interval
#define SENSOR_INIT_INTERVAL            10000       // Try to re-init non-ready sensors every 10s

#define SENSOR_REPORT_EVERY             10          // Report every this many readings

#define SENSOR_REPORT_MIN_EVERY         1           // Minimum every value
#define SENSOR_REPORT_MAX_EVERY         60          // Maximum
```

Uso de memoria

DATA: [=====] 53.5% (used 43800 bytes from 81920 bytes) PROGRAM: [=] 11.4% (used 478092 bytes from 4194304 bytes)

ANEXO V

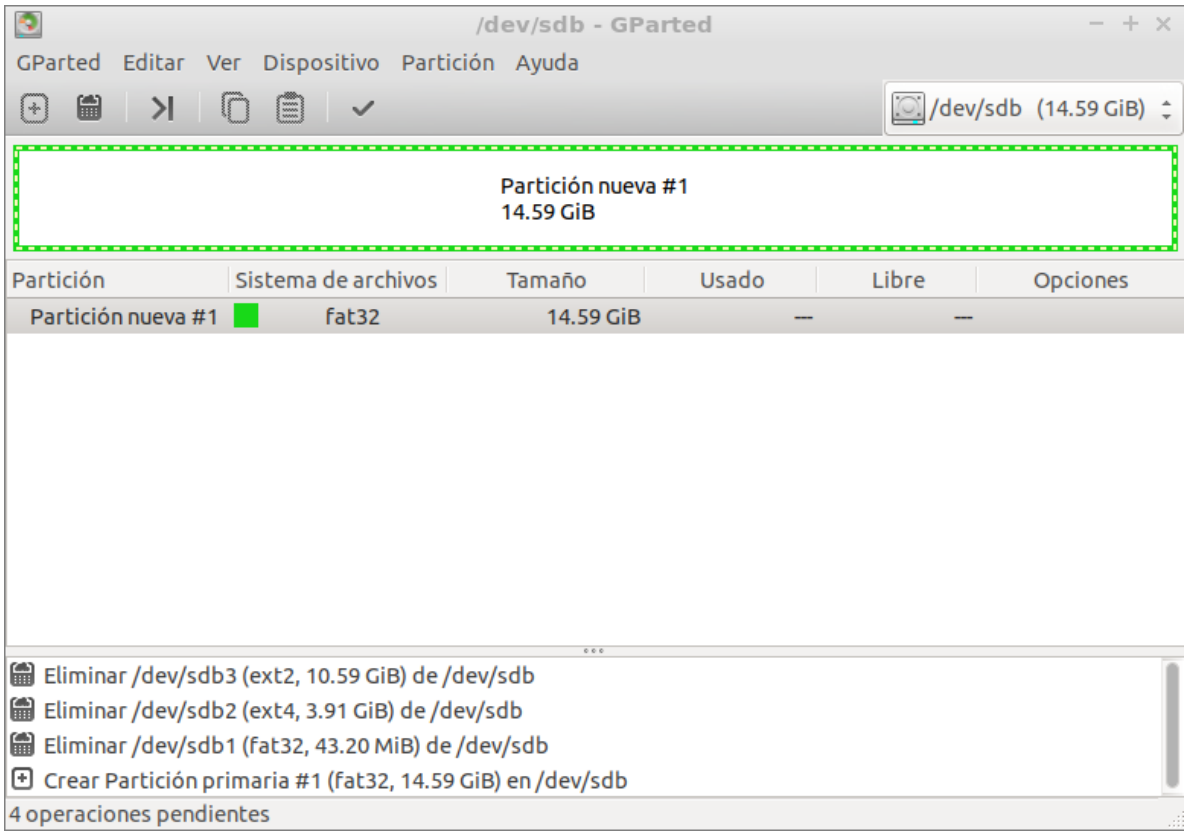
Raspbian

Formateo de SD

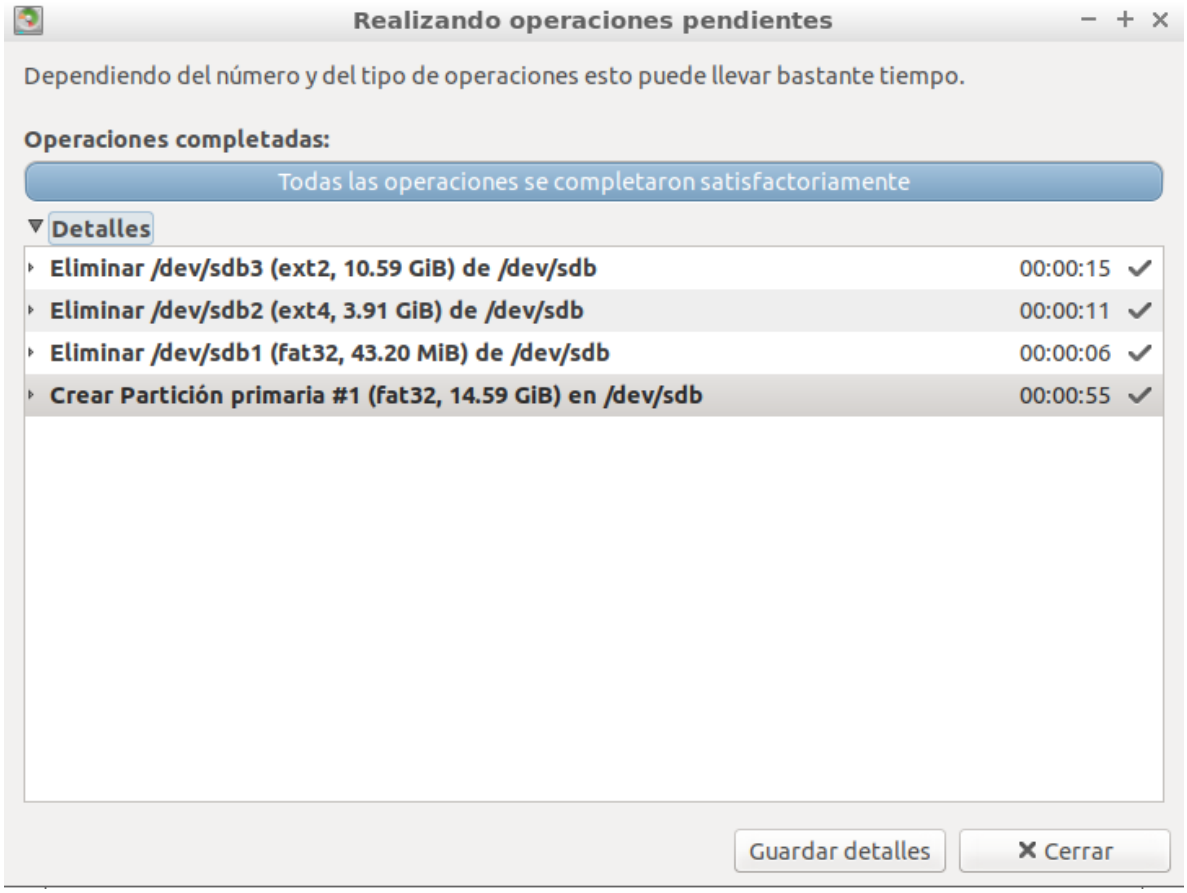
Primero hay que formatear la SD que se va a usar para instalar el sistema operativo. Para hacer esto vamos a usar Gparted, una utilidad de Linux.

Se debe primero seleccionar el dispositivo montado, y desmontar cada partición que hay en el. Luego borrar esas particiones y crear una nueva. Por simplicidad para un nuevo uso lo formateamos con un formato de `fat32`.

Aquí se ve como borramos las particiones:



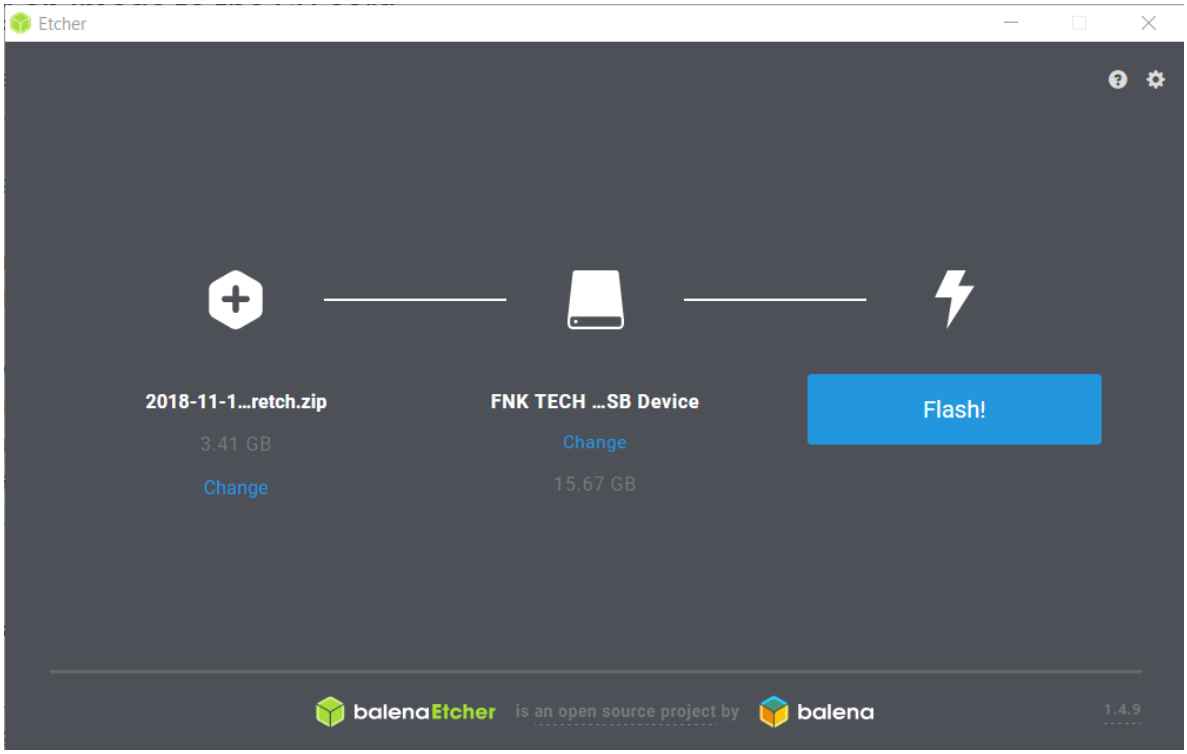
Aquí las operaciones se han completado y tenemos la SD con su tamaño total lista para usarse nuevamente.

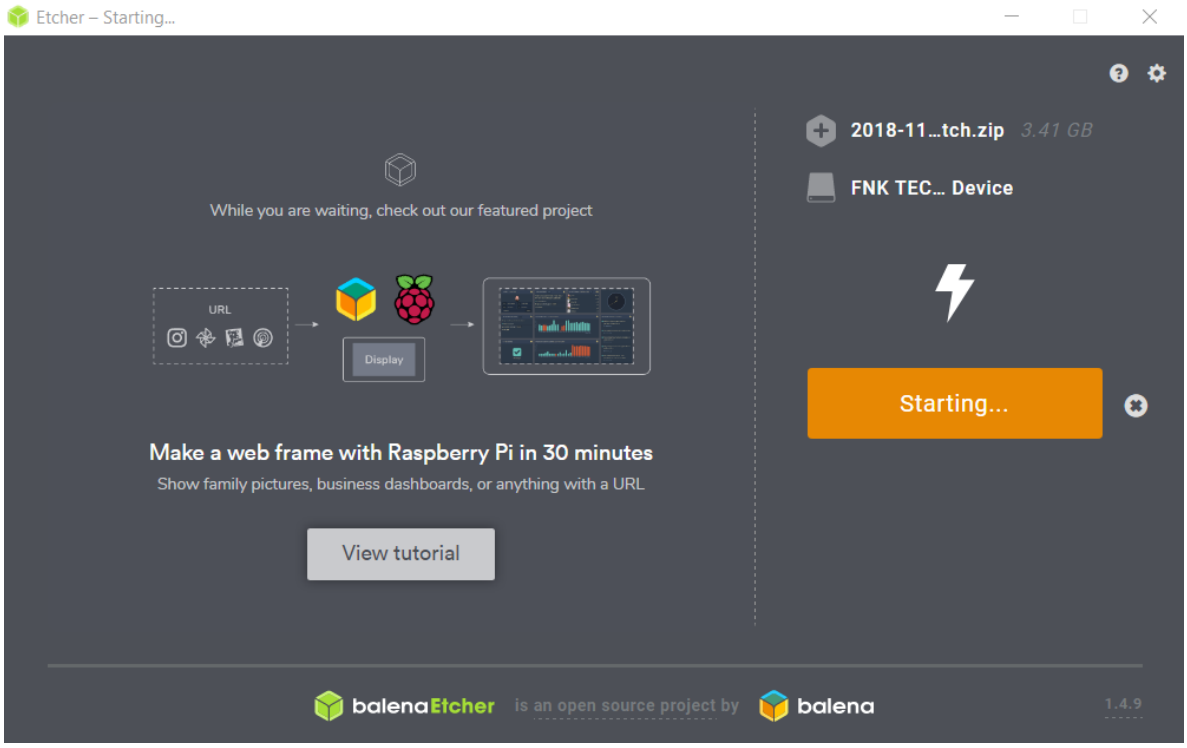


Instalación de Raspbian

Se usa la herramienta Etcher. Se puede descargar desde https://www.balena.io/etcher/

Luego se selecciona el archivo comprimido que contiene al sistema operativo, el dispositivo flash donde se va a cargar y se inicia.

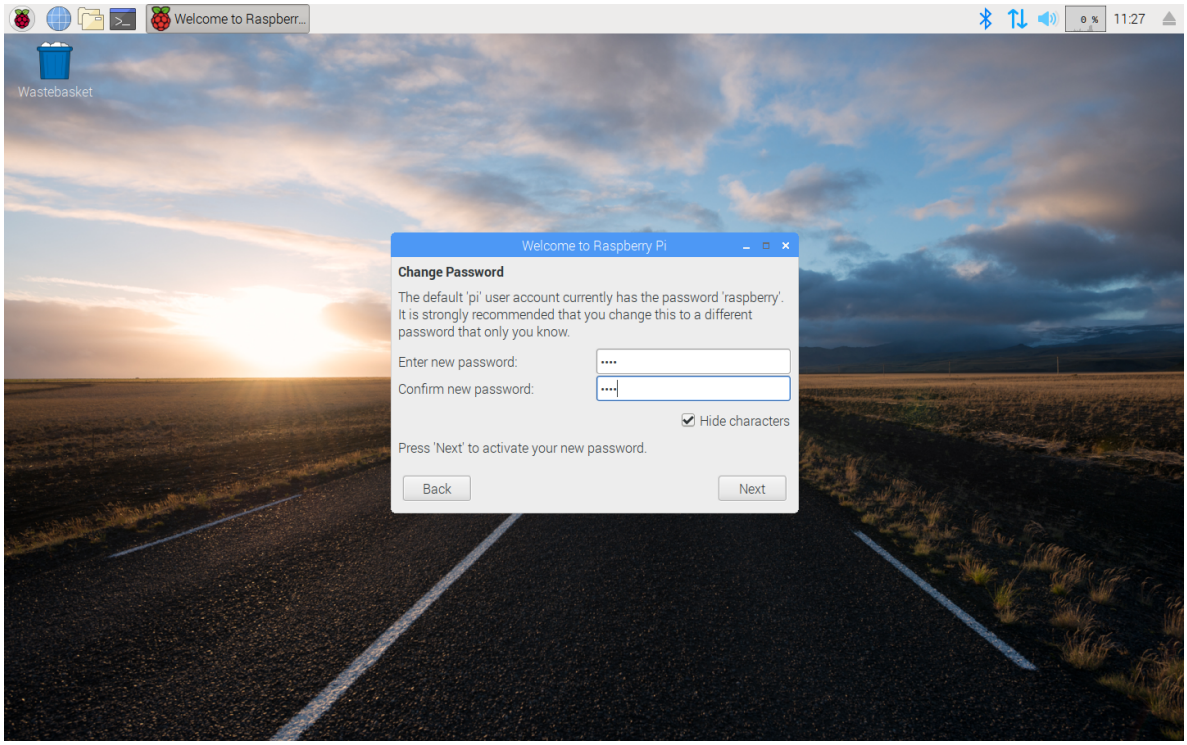




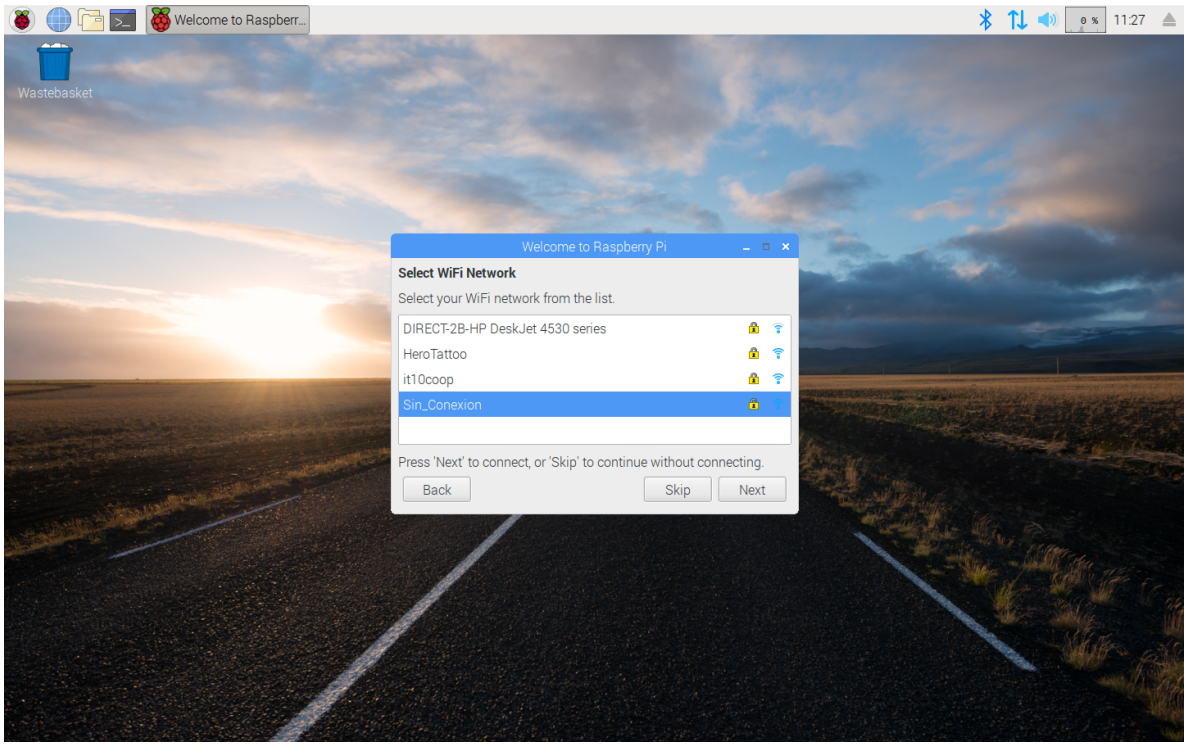
Una vez terminado el proceso ya se puede insertar la tarjeta micro SD en la Raspberry Pi 3B+.

Configuración inicial Raspbian

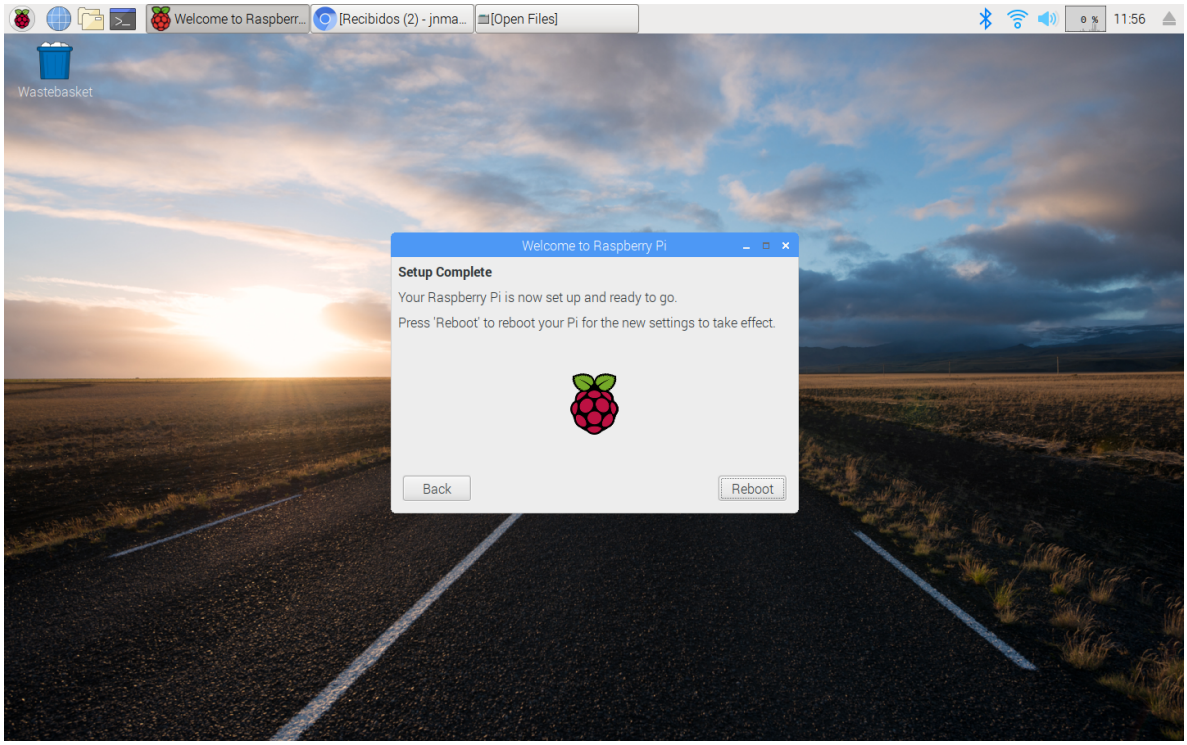
Se crea una nueva password que es `it10` para el usuario `pi`.



Luego se debe seleccionar una red WiFi, para poder realizar actualizaciones.



Y por ultimo, se reinicia para aplicar los cambios.



Habilitar SSH en Raspbian

Por motivos de seguridad, las nuevas imágenes de Raspbian vienen con el server ssh deshabilitado. Esto es porque el usuario y el password por defecto son de dominio público y una RPi (Raspberry Pi) accesible por ssh podría ser fácilmente atacada por gente indeseable si su contraseña no ha sido cambiada. Evidentemente la RPi en un entorno seguro no correría en principio riesgos, pero si la exponemos directamente a Internet (abriendo el puerto 22) o la conectamos en un lugar público, los riesgos son más que evidentes.

En primer lugar se recomienda cambiar la contraseña con este comando:

`passwd`

En nuestro caso la cambiamos inicialmente al momento de instalar Raspbian. Existen varios métodos para activar el server ssh. Teniendo acceso local a la RPi, lo más fácil es utilizar el comando `sudo raspi-config` -> `5 Interfacing Options` -> P2 SSH -> `Sí.`

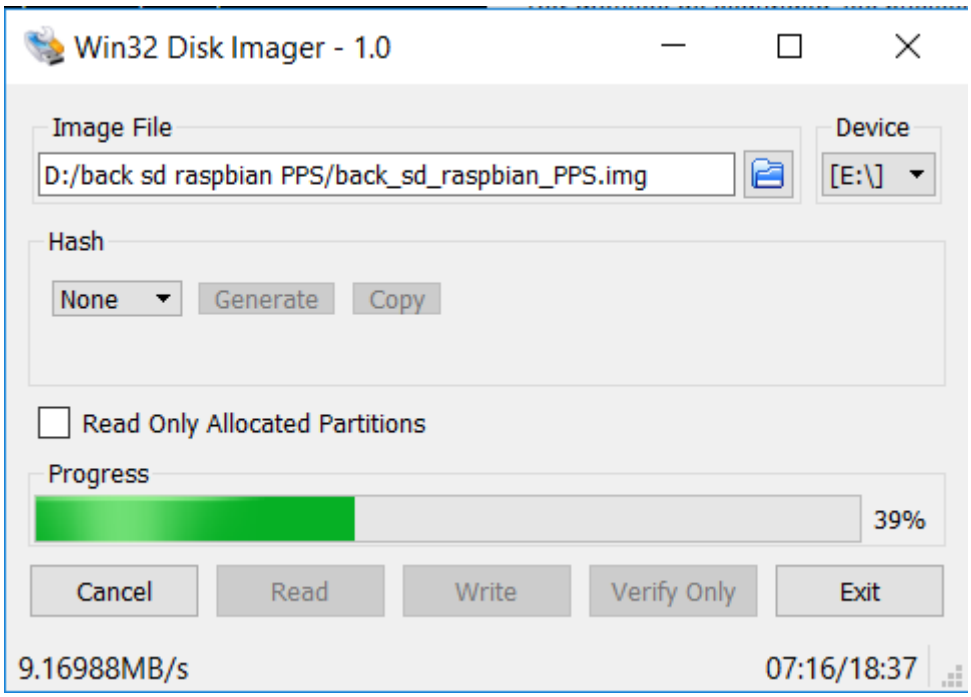
Tras el reinicio de la RPi, se puede comprobar que el servicio está activo con este comando:

`service ssh status`

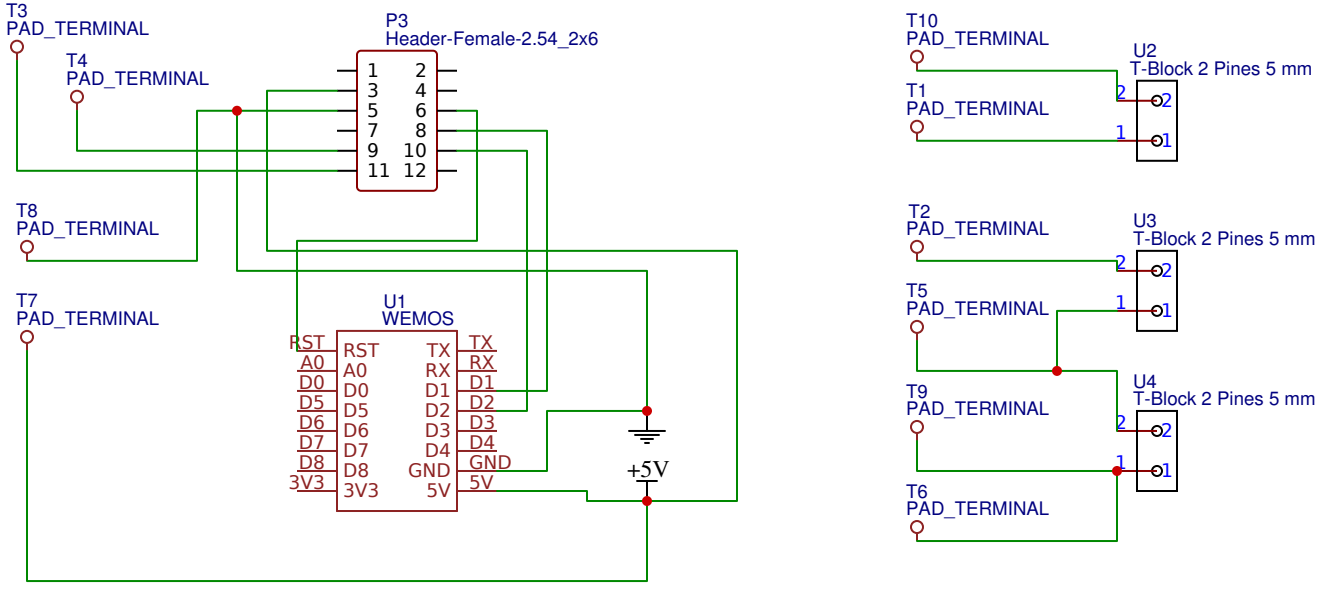
Crear copia de seguridad de sd con Raspbian

Usando Windows, se debe instalar el software **Win32 Disk Imager**, que se puede descargar desde https://sourceforge.net/projects/win32diskimager/.

Una vez instalado el programa, se conecta la unidad sd, se indica la que corresponde para la que se hara la copia de seguridad. En este caso es la unidad **E:** . Luego se da una ubicacion y nombre de la copia de seguridad, con extension **.img**. Por ultimo se da click en el boton **Read**.



Para restaurar con la copia hecha previamente se conecta la unidad y con el mismo software, indicando la unidad y la ubicación de la copia, se presiona sobre **Write**.



Esquemático Modulo IoT - Edukit10

REV: 1.0

Company: IT10

Sheet: 1/1

Date: 2019-03-13

Drawn By: Joaquín Manchado

Anexo VII

InfluxDB

Se instala con el comando: `sudo apt-get install influxdb`. No es necesario configurar nada previamente, ya que las nuevas versiones están listas para entrar como administrador.

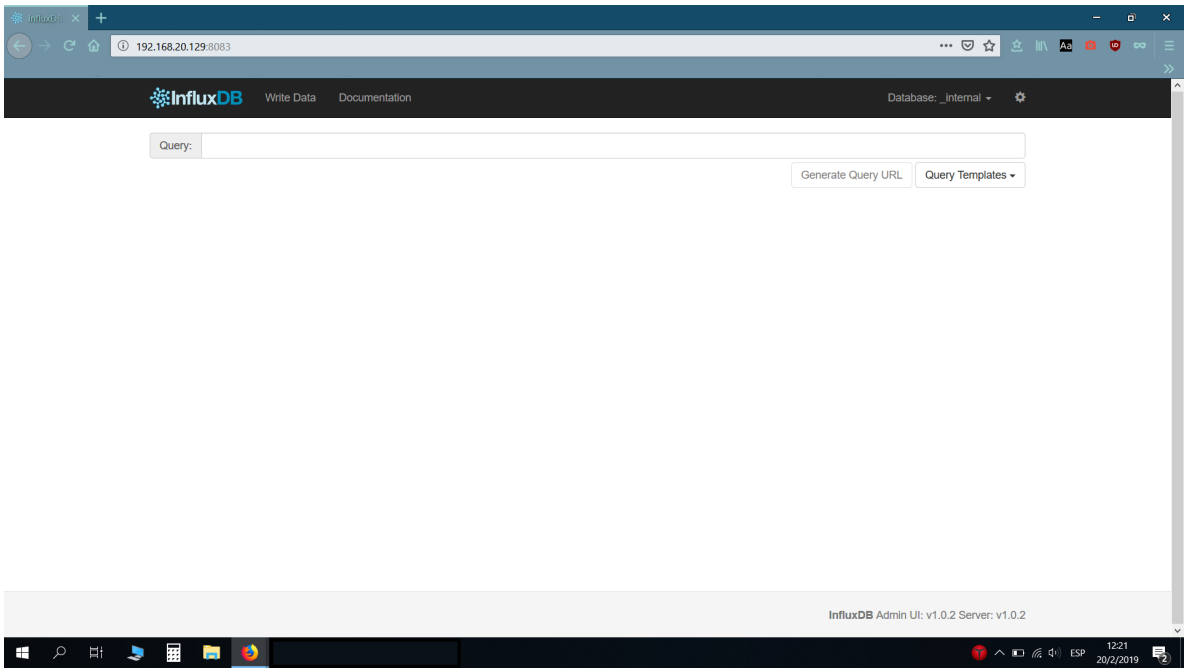
Para poder acceder, utilizar y modificar la base de datos se requiere un cliente. Para esto lo instalamos con `sudo apt-get install influxdb-client`.

Para iniciar la base de datos no es necesario hacerlo a través del servicio, es decir `sudo service influxdb start`. Solo hace falta iniciarlo así:

```
1 sudo influxd run --config /etc/influxdb/influxdb.conf &
```

Lo corremos en segundo plano para poder utilizar el cliente en la misma terminal, sobre todo si pretendemos usar una sola consola con ssh.

También podríamos hacer las consultas o lo que sea necesario a través de un cliente web



Configuraciones previas

Para acceder al cliente por el terminal, solo ingresamos `influx` en la misma. Puede que sea necesario aclarar en algún caso al host al que se conecta, esto se hace: `influx -host IP`. Esa IP es la local, o en todo caso la del equipo que maneje la base de datos.

Primero debemos crear un usuario como el de Raspbian para la base de datos:

```
1 CREATE USER "pi" WITH PASSWORD 'it10' WITH ALL PRIVILEGES
```

Anexo VIII

Grafana

Instalación

Para la Raspberry Pi 3 B+ se puede instalar por terminal. Primero se descarga:

```
wget https://dl.grafana.com/oss/release/grafana_5.4.3_armhf.deb
sudo dpkg -i grafana_5.4.3_armhf.deb
```

Luego se indica los siguiente para instalar Grafana:

```
    ### NOT starting on installation, please execute the following statements to configure grafana to start automatically using systemd
 sudo /bin/systemctl daemon-reload
 sudo /bin/systemctl enable grafana-server
    ### You can start grafana-server by executing
 sudo /bin/systemctl start grafana-server
```

Por ultimo habilitamos el servicio y activamos el demonio para que inicie la herramienta.

```
    sudo /bin/systemctl daemon-reload
    sudo /bin/systemctl enable grafana-server
    sudo /bin/systemctl restart grafana-server
    sudo service grafana-server start
    sudo systemctl enable grafana-server.service
    sudo /lib/systemd/systemd-sysv-install enable grafana-server
```

Ahora sabiendo la dirección IP del equipo, ingresando en un navegador web con `http://<IP>:3000/` se puede acceder a Grafana y desarrollar un dashboard.

Anexo IX

Node-RED sobre Raspbian

Instalación de Node-RED

Se usa el comando siguiente en la terminal:

```
    bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
```

En instalaciones mínimas de Debian puede que haga falta correr `sudo apt-get install build-essential` antes del comando anterior. No es necesario en nuestro caso porque ya estaba instalado.

Descripción del script

Los nodos se pueden administrar ahora a través del palette manager incorporado. Sin embargo, la instalación predeterminada de Pi precarga algunos nodos globalmente, por lo que no se pueden administrar y actualizar fácilmente. La intención del script es para:

1. Actualizar un usuario existente a LTS 8.x o 10.x Node.js y al último Node-RED.
2. Migrar los nodos instalados globalmente existentes al espacio de los usuarios `~ / .node-red` para que puedan administrarse a través del palette manager.
3. Opcionalmente (re) instalar los nodos adicionales que están preinstalados en una imagen completa de Raspbian Pi
4. Nota: NO actualiza ningún usuario instalado nodos existentes. Esto debe hacerse manualmente por el usuario

Nota: NO actualiza ningún usuario instalado nodos existentes. Esto debe hacerse manualmente por el usuario

Aunque dirigido al usuario de Pi, el script también se ejecutará en cualquier sistema operativo basado en Debian, como Ubuntu, y por lo tanto se puede usar en otras plataformas de hardware, aunque no se ha probado ampliamente.

El comando anterior ejecuta muchos comandos como `sudo` y elimina los directorios existentes de Node.js y Node-RED.

El script realizará los siguientes pasos:

1. Pregunta si desea (re) instalar los nodos Pi adicionales
2. Guardar una lista de los nodos Node-RED instalados globalmente que se encuentran en `/ usr / lib / node_modules`
3. apt-get remove nodered
4. Elimina todos los archivos binarios del nodo rojo de `/ usr / bin y / usr / local / bin`
5. Elimina cualquier módulo de Node-RED de `/ usr / lib / node_modules` y `/ usr / local / lib / node_modules`
6. Detecta si Node.js se instaló desde el paquete Node.js o Debian
7. Si no es v8 o más nuevo, elimínarlo según corresponda e instalar el último v8 o v10 LTS (que no usa apt).
8. Limpia la caché npm y la caché .node-gyp para eliminar cualquier versión anterior del código
9. Instala la última versión de Node-RED.
10. Reinstalar bajo la cuenta de usuario cualquier nodo que se haya instalado previamente globalmente
11. Reinstalar los nodos Pi adicionales si es necesario

12. Reconstruir todos los nodos: para volver a compilar los binarios para que coincidan con la última versión de Node.js
13. Agregue los comandos node-red-start, node-red-stop y node-red-log a / usr / bin
14. Añadir menú de acceso directo y el icono
15. Agregar script systemd y establecer usuario
16. Si en un Pi agrega una temperatura de CPU -> ejemplo de IoT

Corriendo Node-RED

Para iniciar Node-RED, hay dos formas:

- En el escritorio, seleecionar Menu -> Programming -> Node-RED.
- O por medio de la terminal, `node-red-start`

Nota: cerrar la ventana (o `ctrl + c`) no frena la ejecucion de Node-RED (seguirá corriendo en segundo plano)

Para parar Node-RED, ejecutar el comando `node-red-stop`.

Para ver el log, ejecutar el comando `node-red-log`.

Inicio automático en el booteo

Usar comando:

```
sudo systemctl enable nodered.service
```

Y de la misma manera, hacer `sudo systemctl disable nodered.service` para deshabilitar el inico automatico de Node-RED en el booteo.

Instalación de nuevos nodos

Instalaremos nuevos nodos para trabajar, estos son:

- Dashboard
  - `npm install node-red-dashboard` Explicar un poco
- Mqtt-Broker
  - `npm install node-red-contrib-mqtt-broker` Explicar un poco
- InfluxDB
  - `npm install node-red-contrib-influxdb` Explicar un poco
- Algun otro...

Hay dos formas de poder instalarlos, que describiremos a continuación.

- **Usando el editor:**

Desde la version 0.15 se pueden instalar nodos directamente usando el editor. para esto se debe ir a **Manage Palette** desde el **menu**, luego seleccionar la pestaña **install**. Desde aqui se pueden buscar nuevos nodos, instalarlos, actualizarlos, habilitar o deshabilitar nodos existentes.

- Usando nodos con paquetes npm

Se puede instalarde manera local en el directorio del usuario(por defecto, $HOME/.node-red):

```
    cd $HOME/.node-red
    npm install <npm-package-name>
```

Después de esto se debe hacer **stop** y **restart** sobre Node-RED para cargar los nuevos nodos.

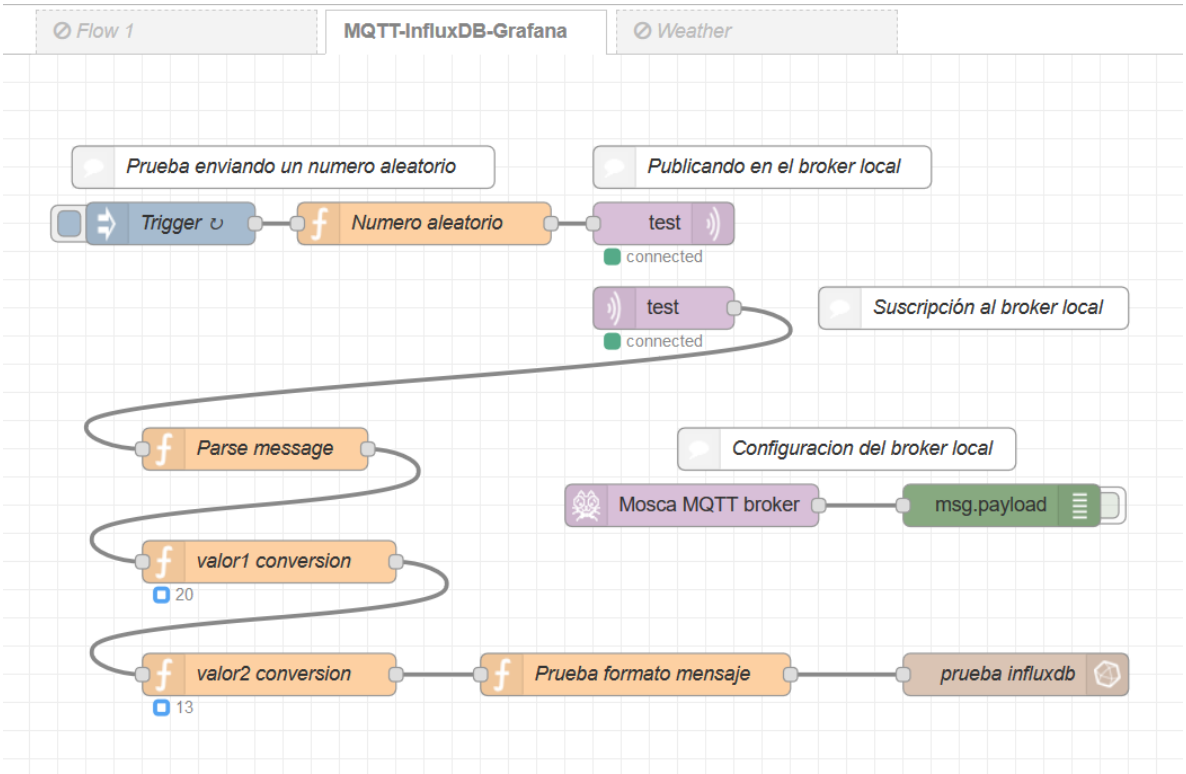
Anexo X

Integración Node-RED/ InfluxDB/ Grafana

Se procede a hacer una prueba integrando las tres tecnologías para luego poder construir un escenario real. Por el momento los datos que se trabajaran son aleatorios para tener una carga simulada al guardar y mostrar a través de gráficos.

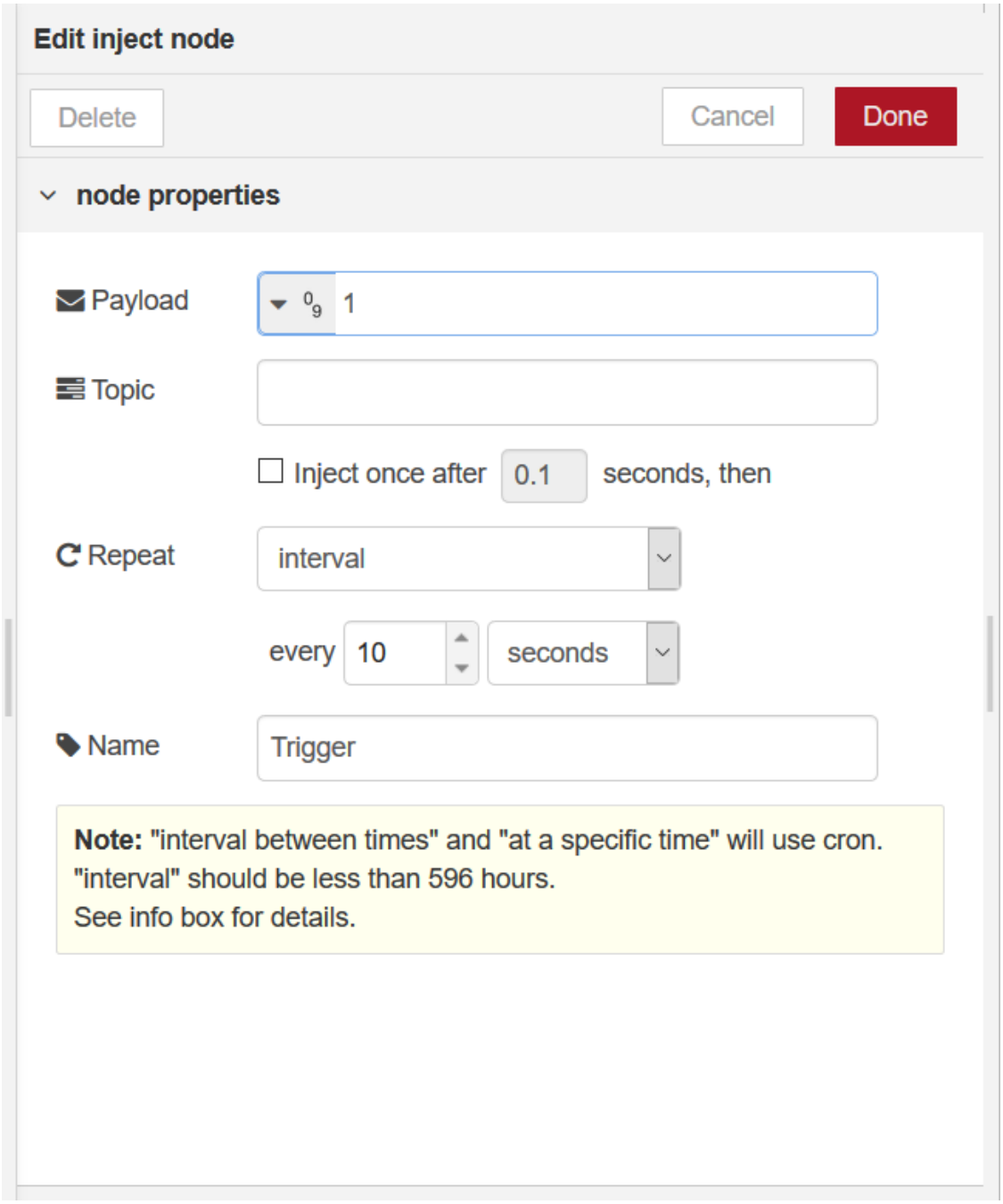
Fase 1: Node-RED

El flujo completo es el siguiente:



Describiremos nodo por nodo:

- **Trigger:** Envía un "1" cada cierto período de tiempo, que es configurable (en este caso cada 10 segundos), para que dispare la acción del bloque siguiente (**Numero aleatorio**).

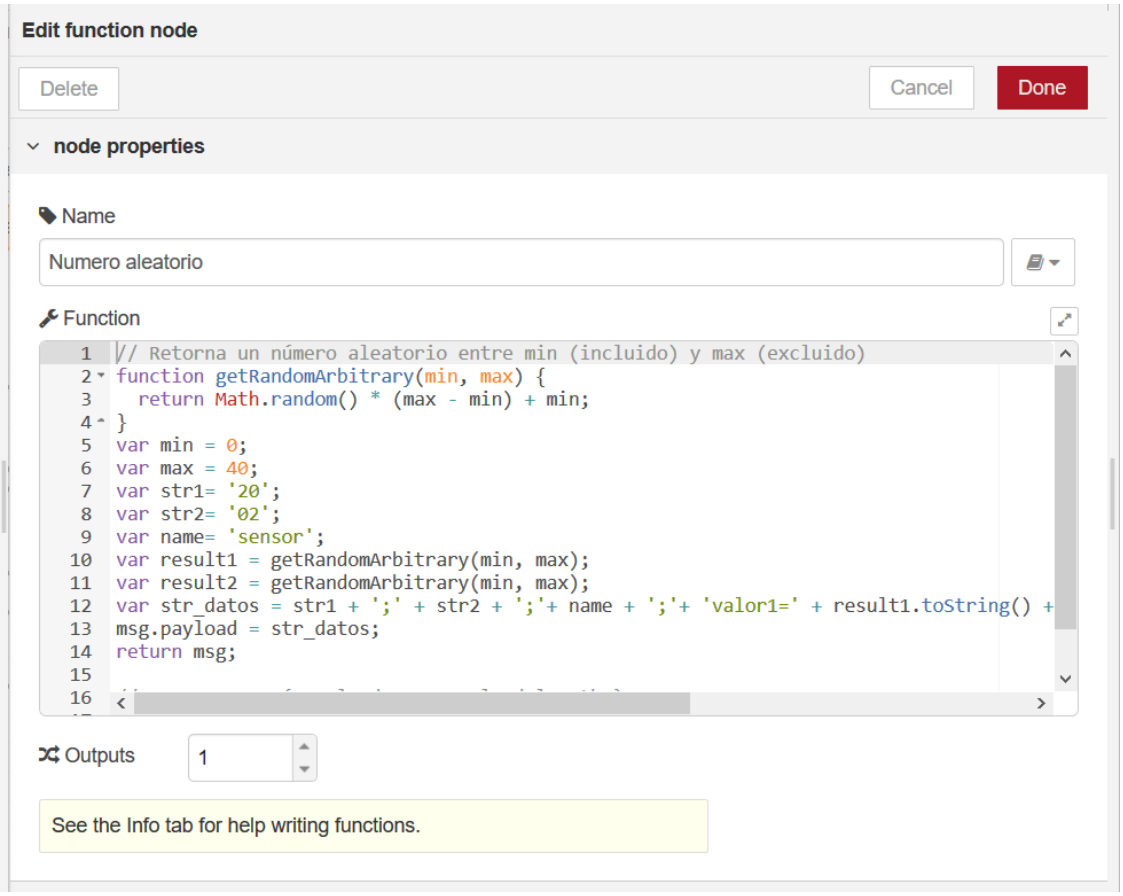


Podría enviarse cualquier otro dato, no es de importancia, ya que cuando la función recibe una entrada, se ejecuta independientemente de lo que tenga la entrada, o si se usara como argumento propio de la función.

- **Numero aleatorio:** En este bloque se genera el flujo de datos que luego se parsea para colocarlo como corresponde en la base de datos y posteriormente en Grafana. El código es el siguiente:

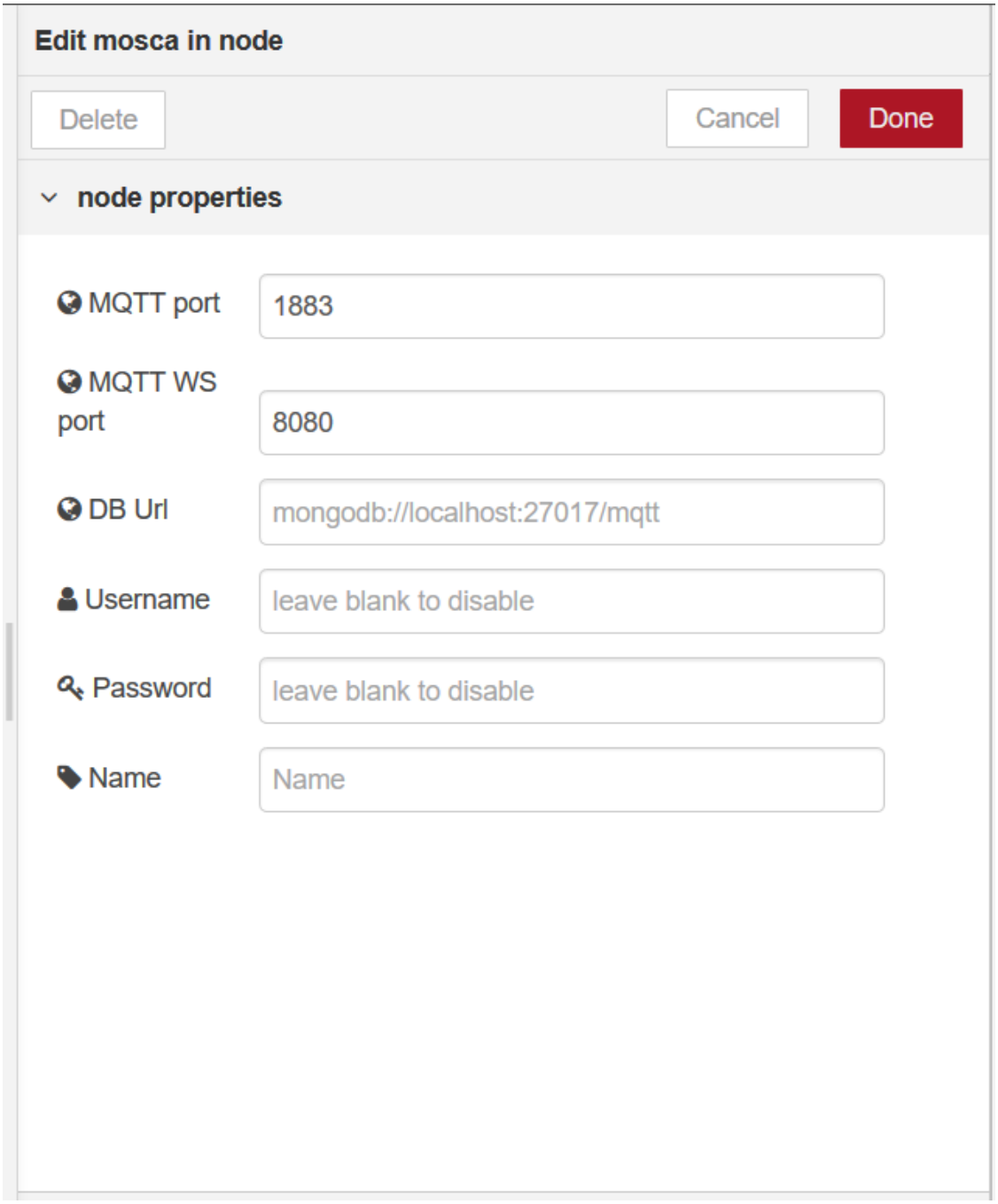
```
//############################ FUNCION
########################################
// Retorna un número aleatorio entre min (incluido) y max (excluido)
function getRandomArbitrary(min, max) {
  return Math.random() * (max - min) + min;
}
//#########################################################################
#######
var min = 0;
var max = 40;
var str1= '20';
var str2= '02';
```

```
var name= 'sensor';
var result1 = getRandomArbitrary(min, max);
var result2 = getRandomArbitrary(min, max);
var str_datos = str1 + ';' + str2 + ';'+ name + ';'+ 'valor1=' +
result1.toString() + ';'+ 'valor2=' +result2.toString()+ ';';
msg.payload = str_datos;
return msg;
```

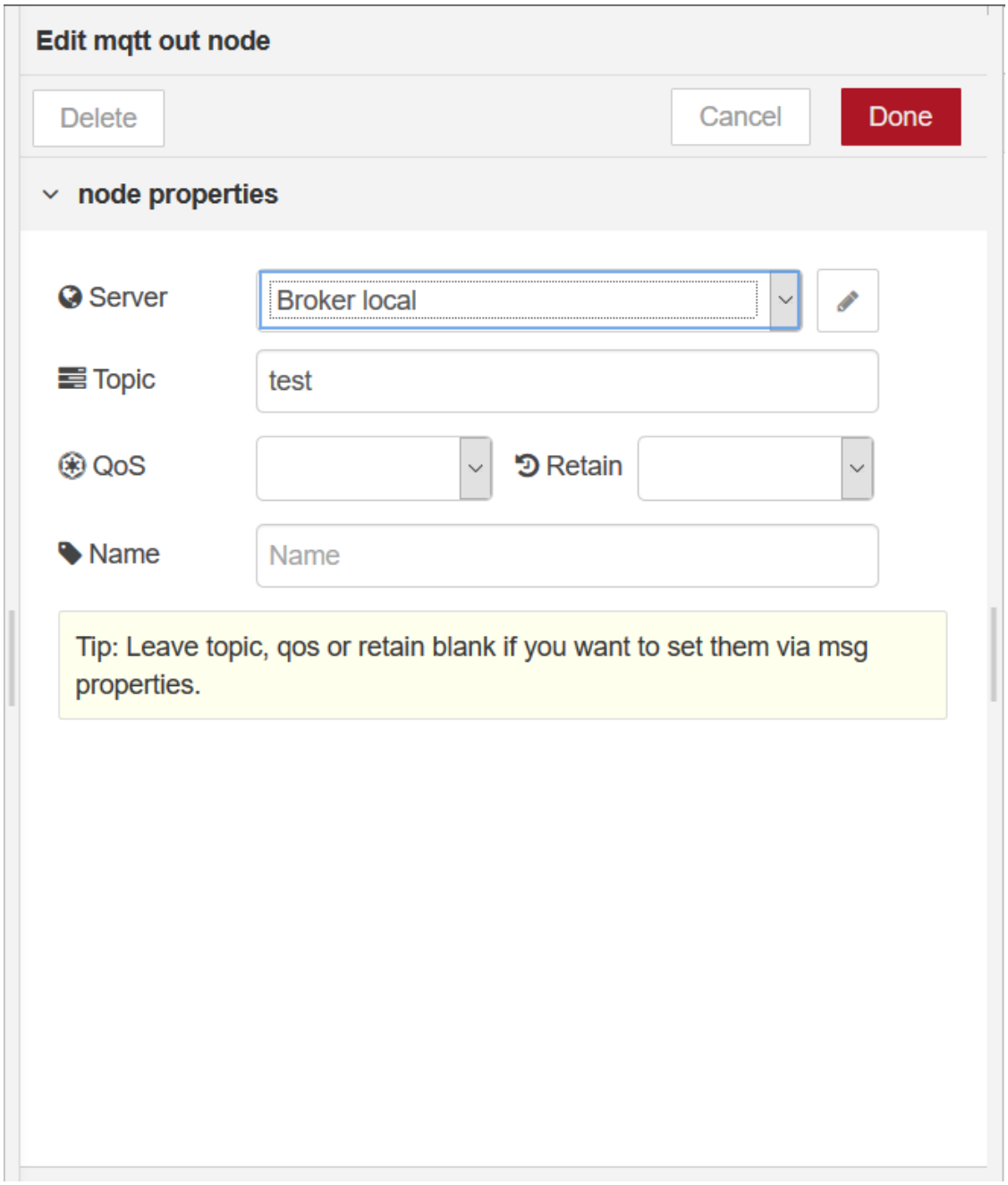


Se crea una función para que genere números aleatorios entre limites inferiores y superiores. Luego se llama a esa función poniéndole como limites las variables `min` y `max`, respectivamente. Luego se crean variables adicionales para agregar mas carga al mensaje que se enviará al broker. Todas las variables y resultados se concatenan en forma de string en `str_datos`, separados cada uno por el caracter `';'` (Esto posteriormente servirá para poder parsear el mensaje). Por ultimo se pone la variable string como carga de `msg` y se devuelve el mismo como salida.

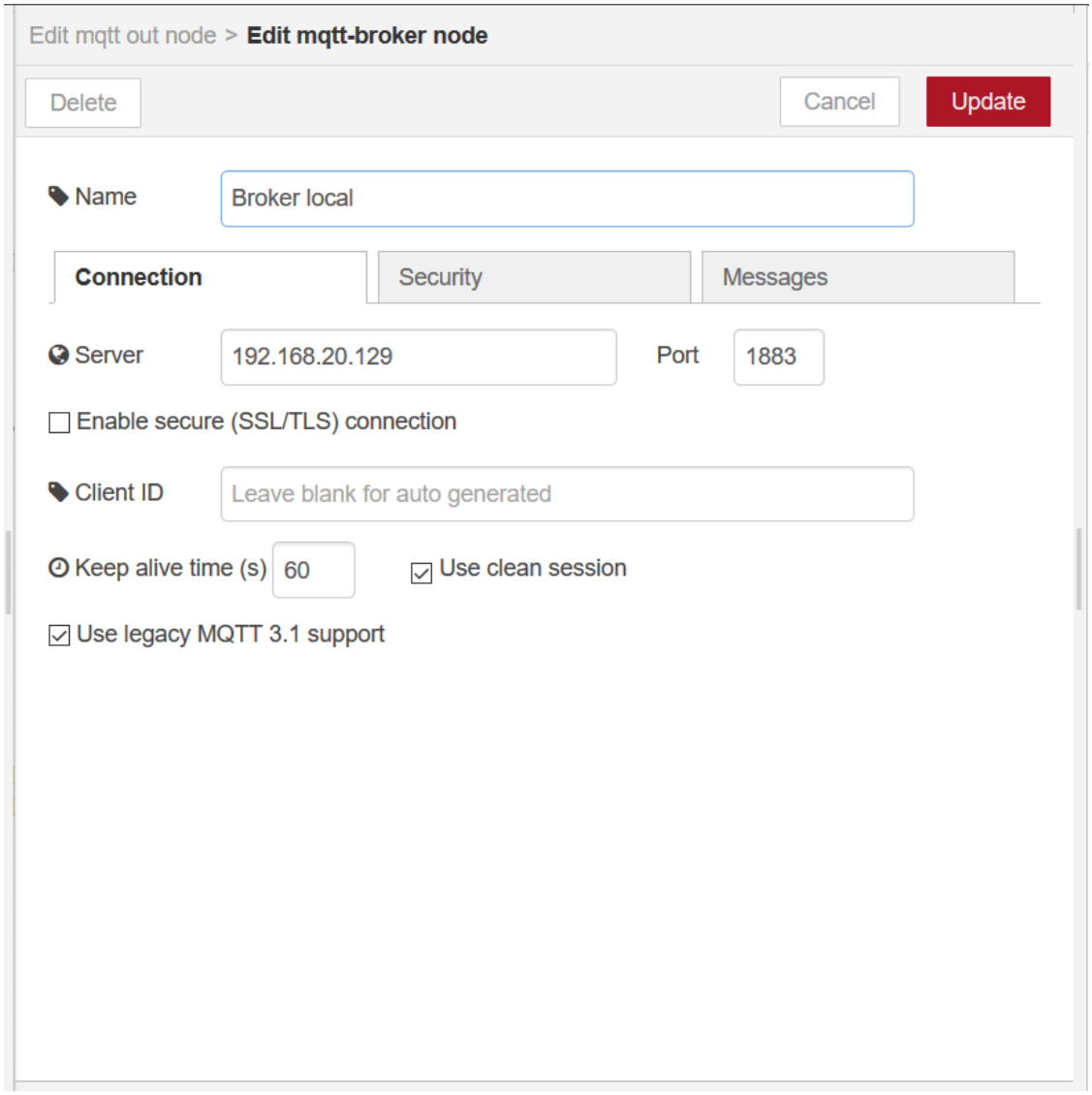
- **Mosca MQTT Broker:** Es un broker MQTT que se crea localmente. Muy útil para automatización de hogares o flujo de datos pequeño. Por defecto el broker escucha en el puerto 1883 y el puerto 8080 para el Websocket. Por el momento no le asignamos ningun usuario y contraseña ya que es una prueba.



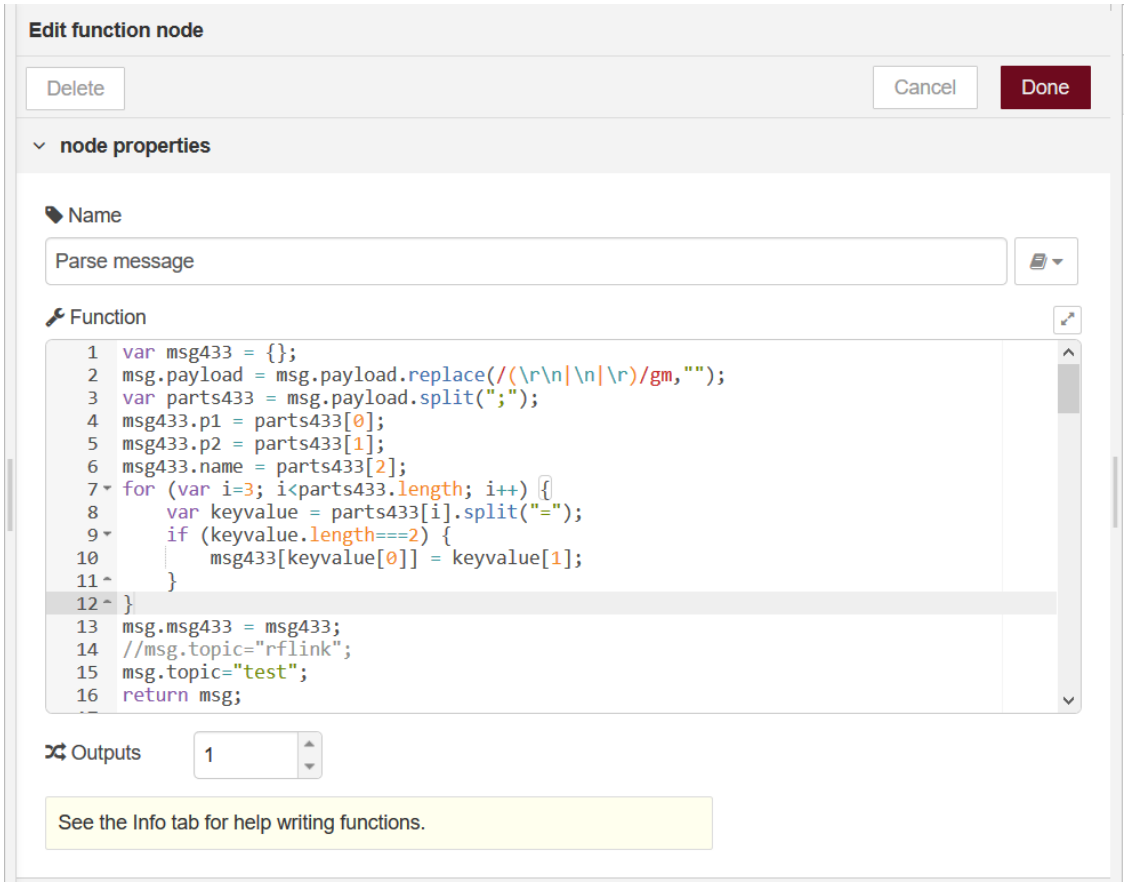
- **test:** Son dos bloques similares que tienen que ver con la publicación/suscripción respecto del broker local. Los datos aleatorios generados se publican al topic `test`.



Luego el bloque suscriptor toma los datos que se enviaron para procesarlos.



- **Parse message:** Es un bloque tipo funcion en donde se parsea el mensaje tomado del broker en forma de suscriptor al topic `test`.



El código es el siguiente:

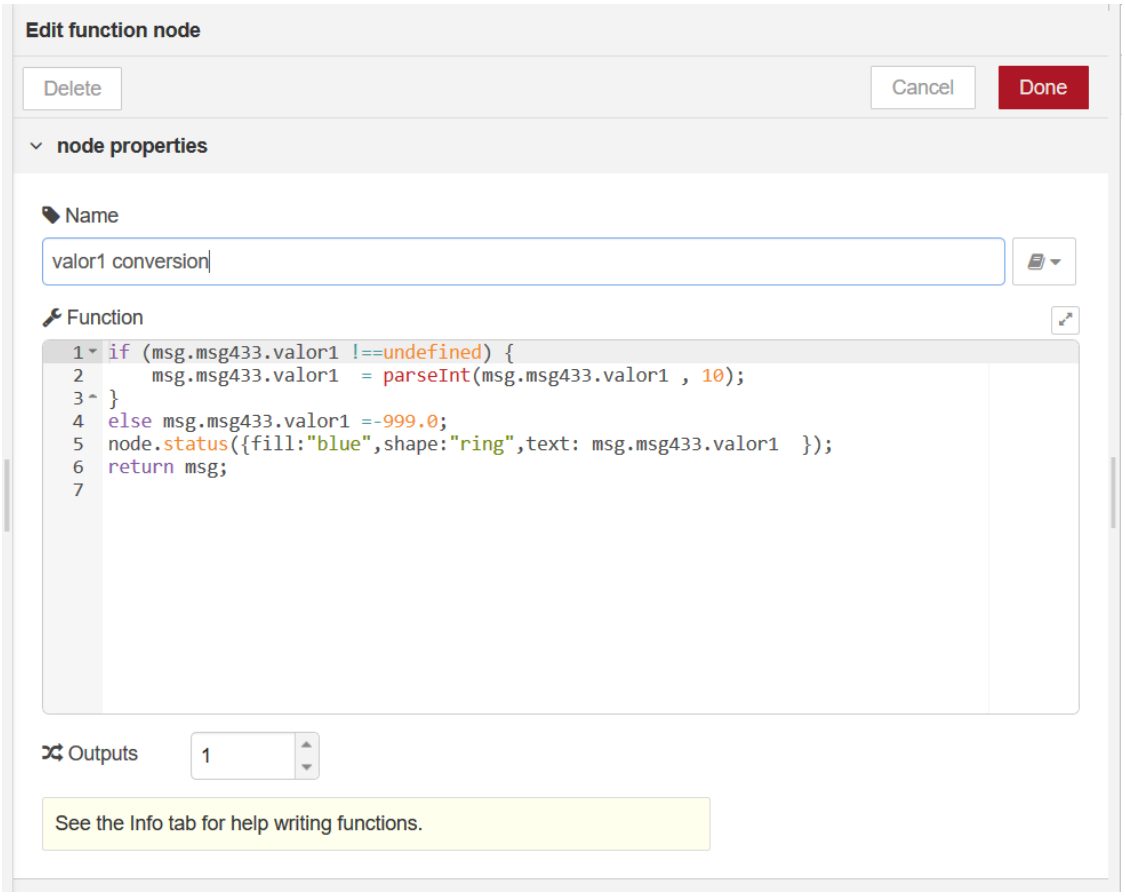
```
var msg433 = {};
msg.payload = msg.payload.replace(/(\r\n|\n|\r)/gm,"");

var parts433 = msg.payload.split(";");

msg433.p1 = parts433[0];
msg433.p2 = parts433[1];
msg433.name = parts433[2];
for (var i=3; i < parts433.length; i++) {
    var keyvalue = parts433[i].split("=");
    if (keyvalue.length===2) {
        msg433[keyvalue[0]] = keyvalue[1];
    }
}

msg.msg433 = msg433;

msg.topic="test";
return msg;
```

Primero se crea la variable `msg433` donde se ira parseando el mensaje que se recibe. Luego eliminamos todos los saltos de línea posibles con la instrucción `msg.payload.replace(/(\r\n|\n|\r)/gm,"")`. Entendiendo esto:

- **\n**: Nueva línea
- **\r**: Retorno de carro
- **gm**: Flags de patrones globales.

- **g**: Modificador global. Todos los matcheos (No regresa después del primer encuentro)
- **m**: Multilínea. Hace que ^ y $ coincidan con el inicio / final de cada línea (no solo el inicio / final de la cadena) Después se separan los datos teniendo en cuenta el caracter `';'`. De las primeras tres partes, la tercera es el nombre del sensor, por una cuestión de identificar el flujo de datos. A partir de ahí, con un bucle se recorre todas las partes separadas y se queda con el valor que fue enviado, propiamente dicho. Se agrega una propiedad nueva al mensaje que sigue en flujo, que es el topic. Servirá para setearlo mas adelante. Finalmente se devuelve el mensaje formateado y parseado.

- **valor1 conversion:** Bloque de función.

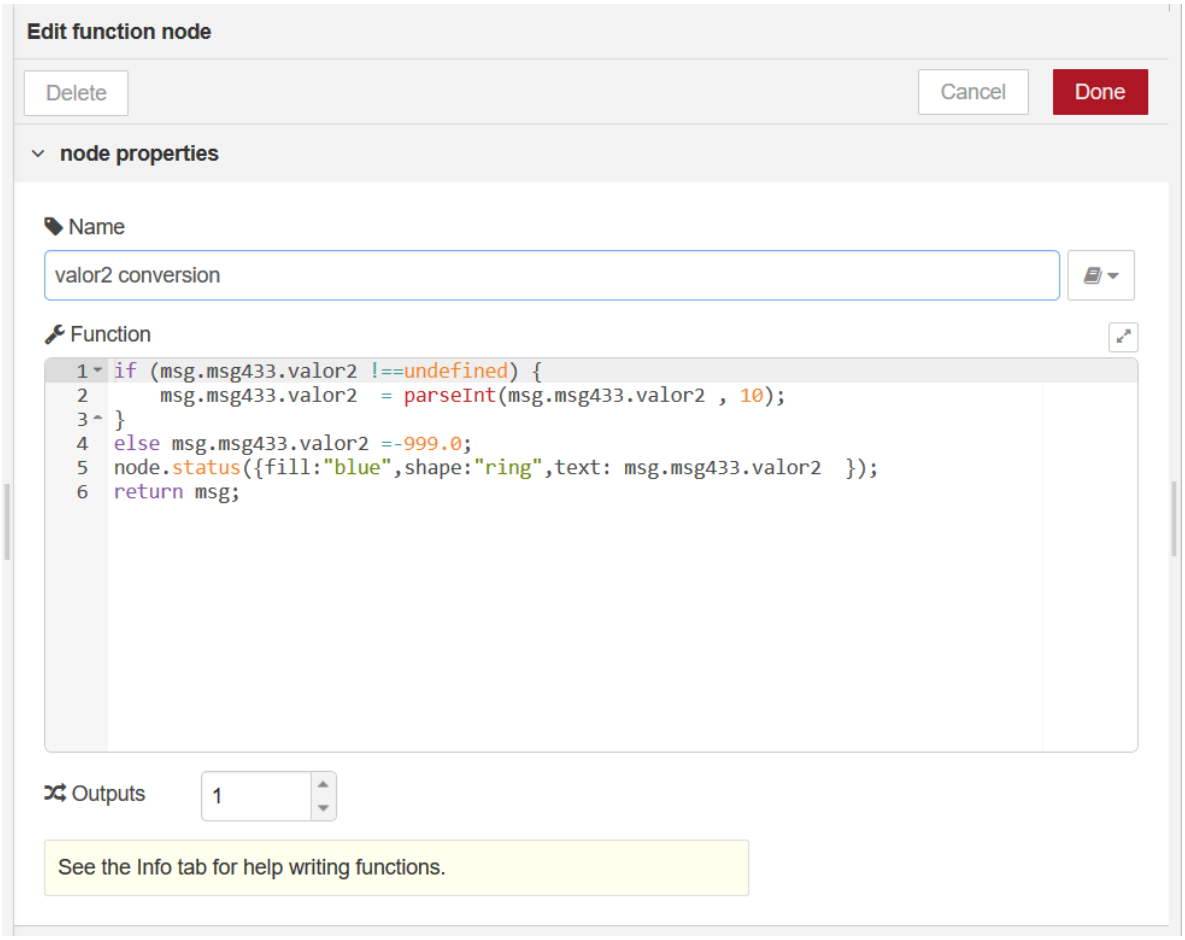


El código es el siguiente:

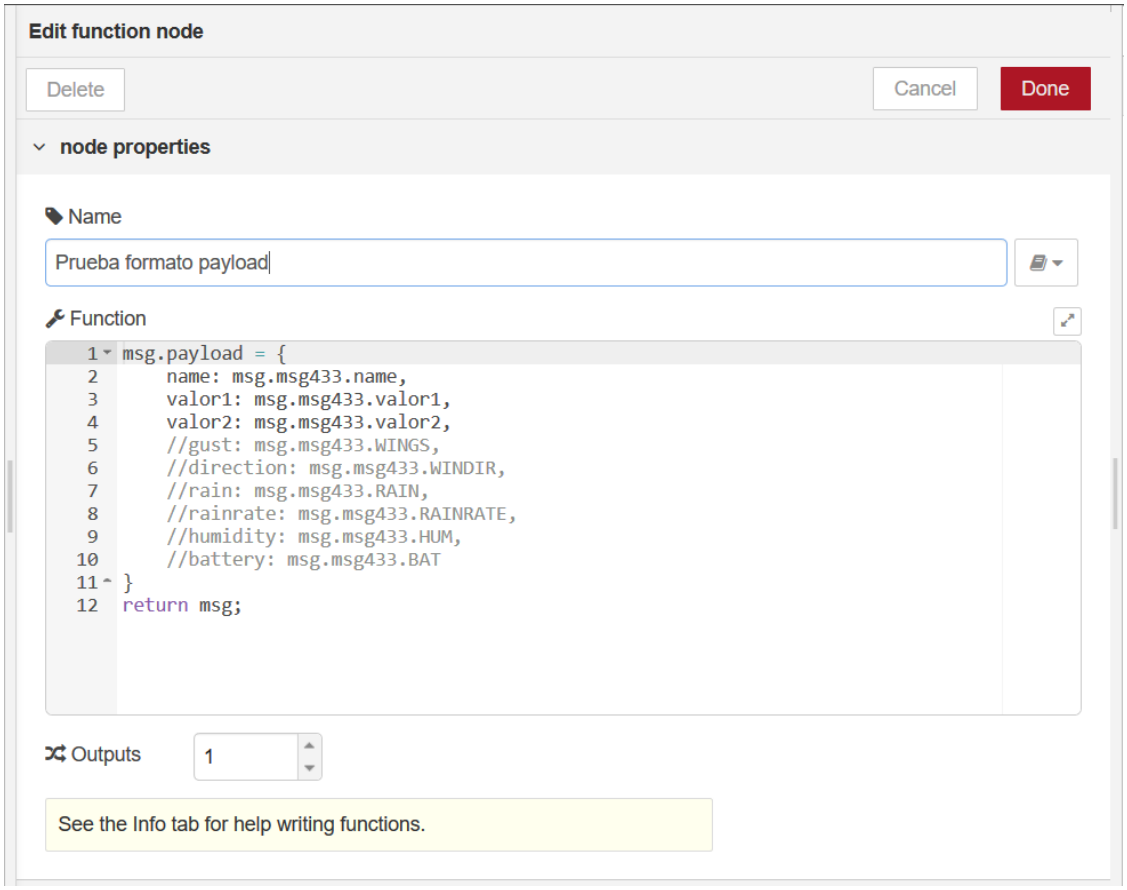
```
if (msg.msg433.valor1 !== undefined) {
    msg.msg433.valor1  = parseInt(msg.msg433.valor1 , 10);
}
else msg.msg433.valor1 =-999.0;
node.status({fill:"blue",shape:"ring",text: msg.msg433.valor1  });
return msg;
```

Se indica que si `valor1` no esta definido como parte del objeto `msg`, lo toma y lo incluye en el mismo, como un tipo de dato entero, y decimal. Esto viene de la instrucción `msg.msg433.valor1 = parseInt(msg.msg433.valor1 , 10)`. El "10" indica base decimal. Si estuviera definido, se setea con un **-999.0**. Además para poder observar como esta funcionando el nodo y ver que valor esta pasando por ahí se incluye la línea `node.status({fill:"blue",shape:"ring",text: msg.msg433.valor1 })`. Sigue el mensaje con las respectivas partes siguientes.

- **valor2 conversion:** Bloque de función. Funciona exactamente igual que el anterior, nada más que trabaja con la variable `valor2`.



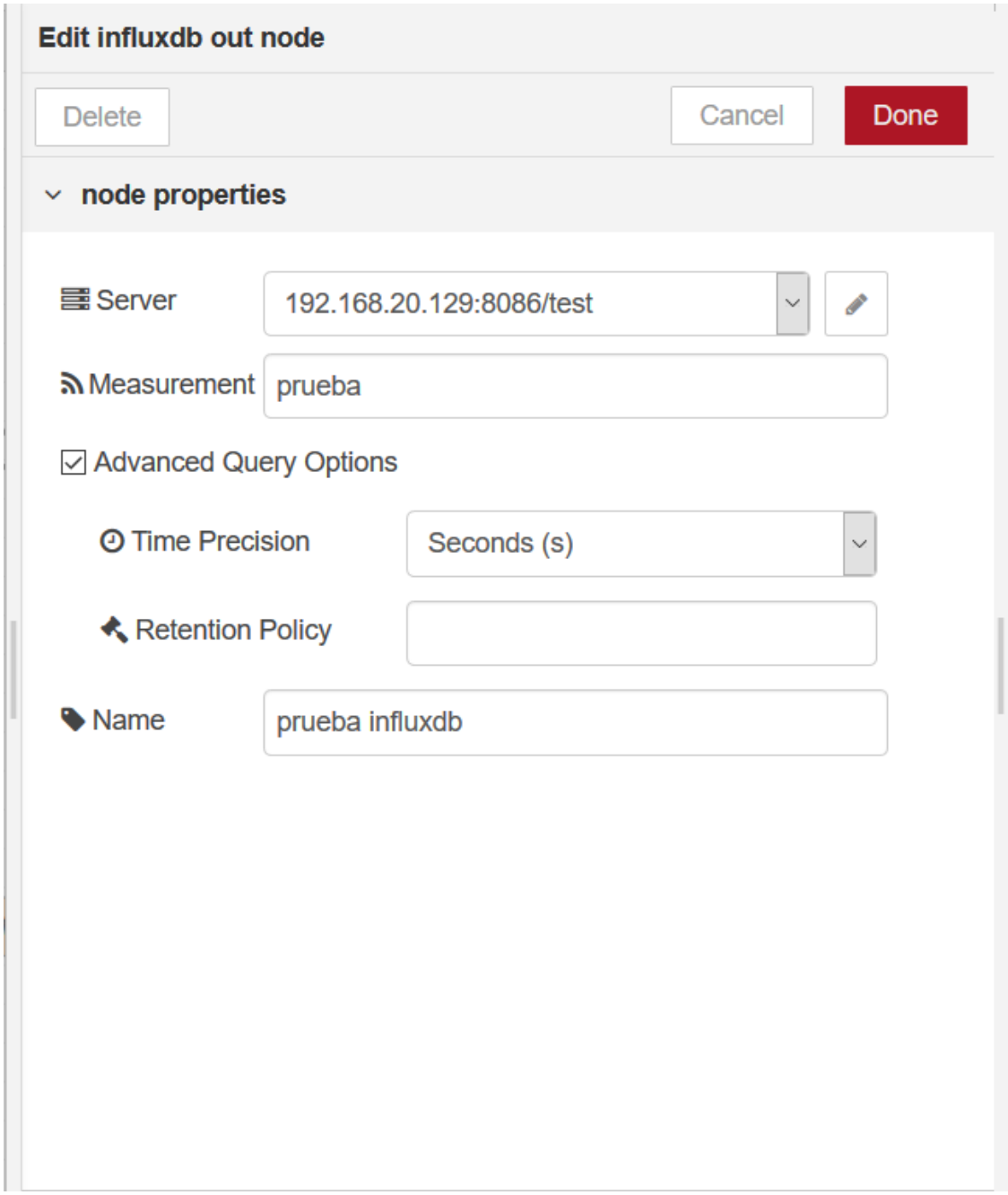
- **Prueba formato mensaje:** Nodo de función. Asigna como propiedad los valores previos del mensaje para formatearlos de la manera que lo necesita la base de datos InfluxDB.



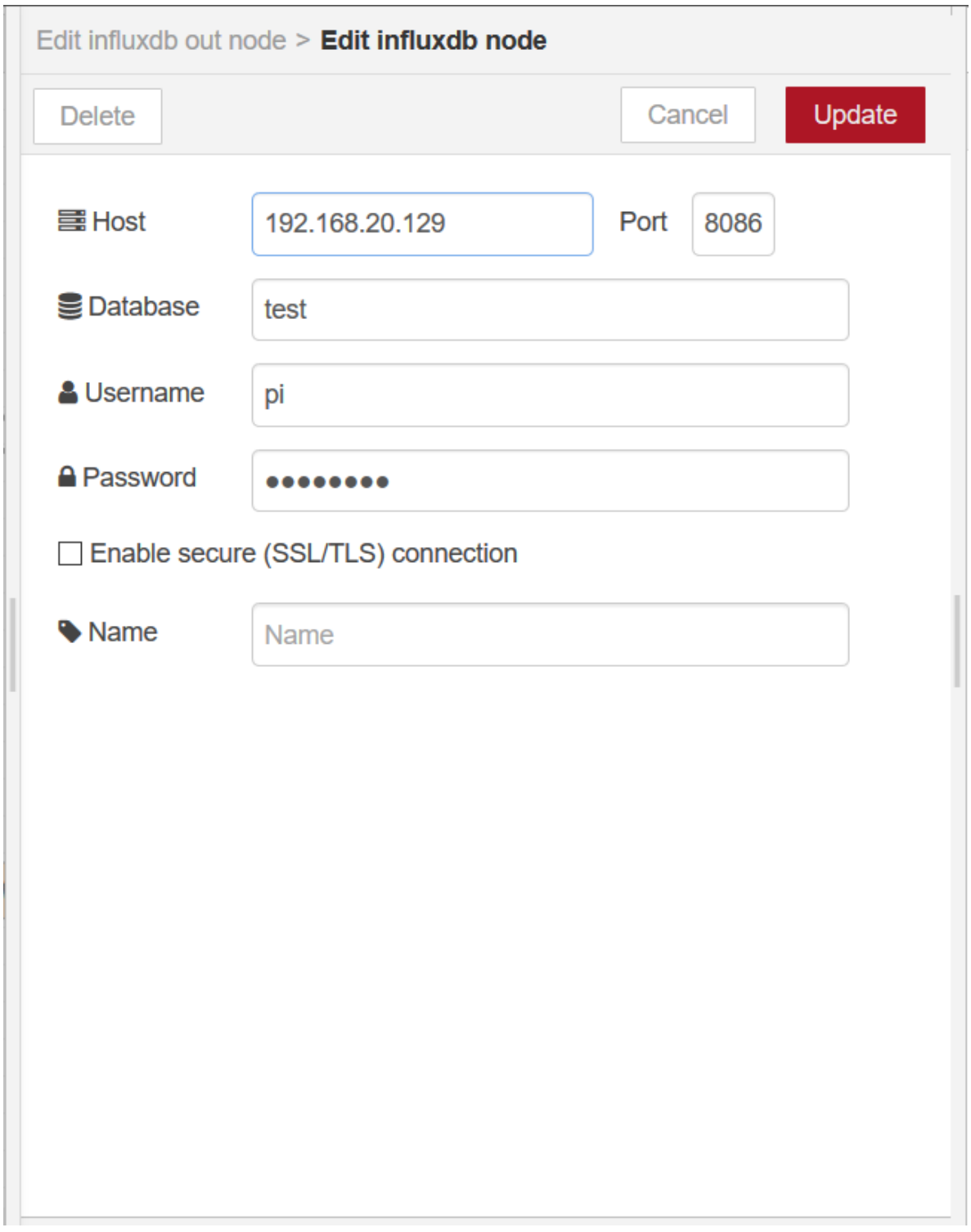
El código es el siguiente:

```
msg.payload = {
  name: msg.msg433.name,
  valor1: msg.msg433.valor1,
  valor2: msg.msg433.valor2,
}
return msg;
```

- **prueba influxdb:** Nodo de influxDB.



Se debe indicar la dirección del servidor (Dirección IP local) y el nombre de la base de datos (*test*). Además el *Measurement* donde se insertarán los datos. En nuestro caso se llama **prueba**. Además se le puede indicar la precisión del tiempo y alguna política de retención. No será necesario modificar nada en ese sentido para este ejemplo. En cuanto a la configuración del nodo, se indica el nombre de la base de datos, el puerto por el que esta escuchando, usuario y contraseña con la que debe acceder e insertar los datos. Para el caso se usa `pi:it10`

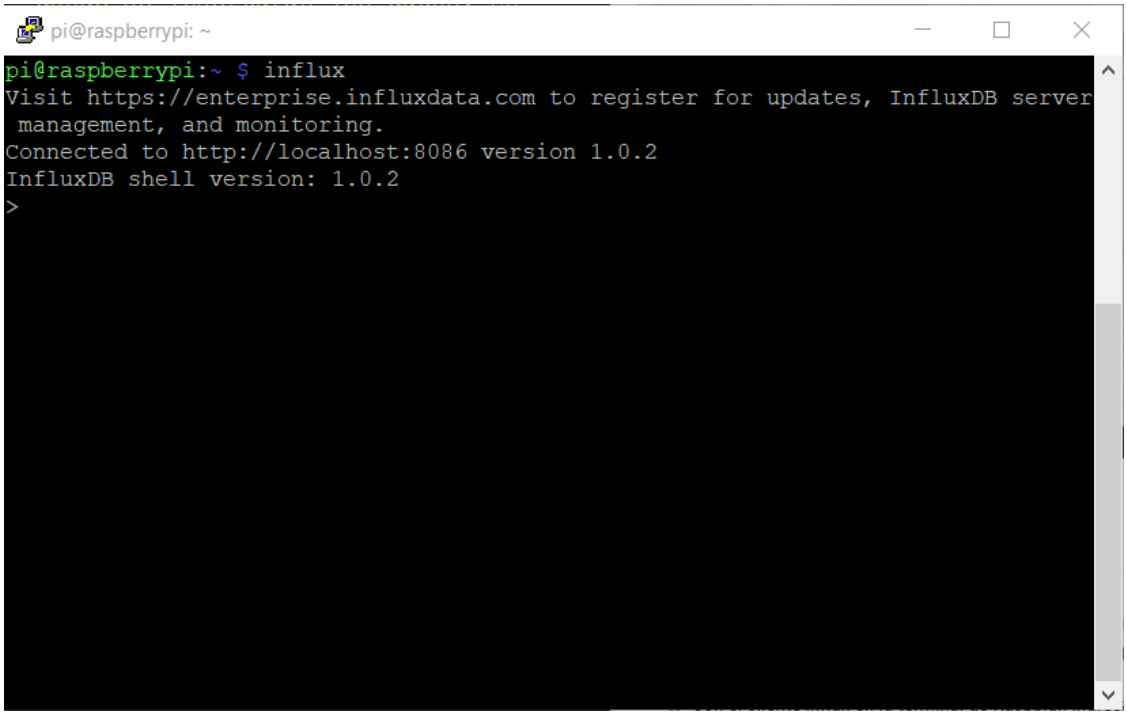


Fase 2: InfluxDB

Con InfluxDB ya instalado solo hace falta crear la base de datos de prueba, ya que Node-RED envía los datos, pero si no hay una base de datos donde insertarse, estos nunca se envían. Se procede de la siguiente manera:

1. Abrir el cliente de influxDB en la terminal de Raspbian:

```
influx
```

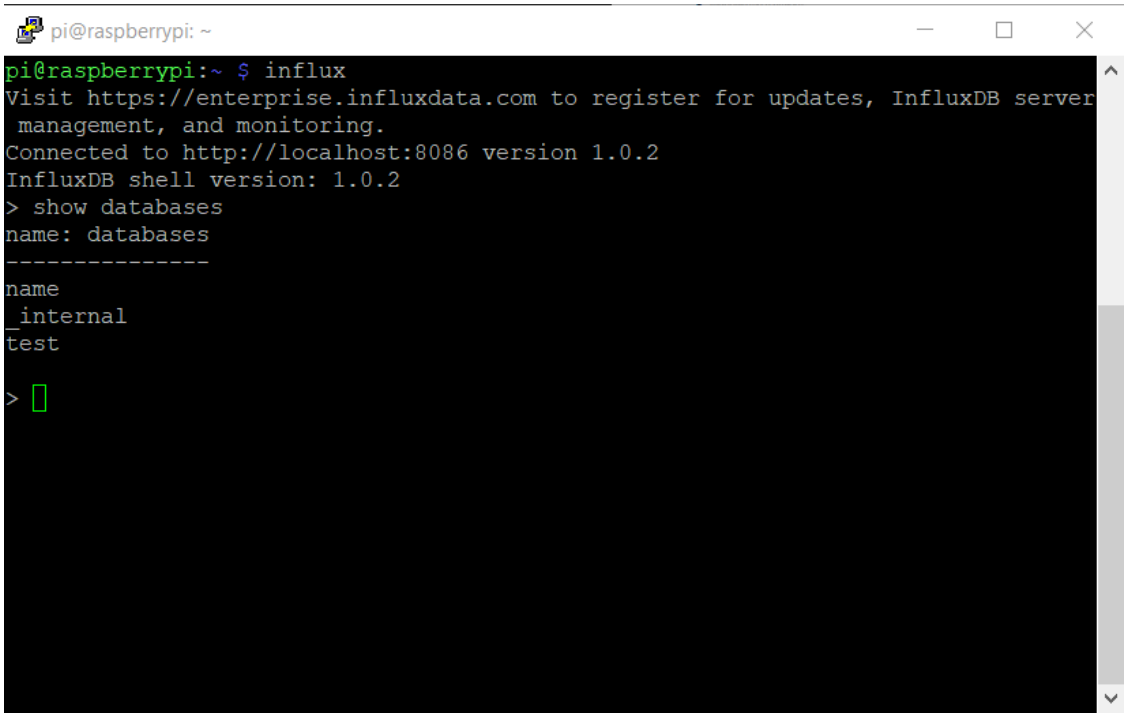


2. Creamos la base de datos `test`:

```
CREATE DATABASE test
```

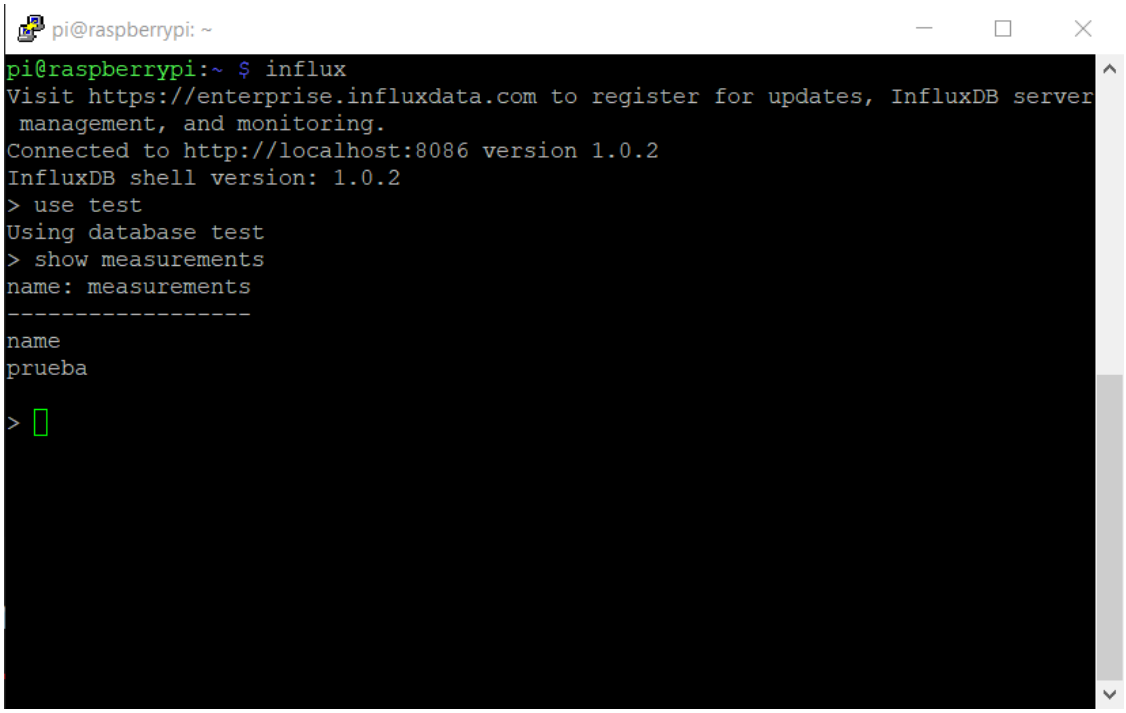
3. Mostramos para ver que se creo correctamente:

```
show databases
```

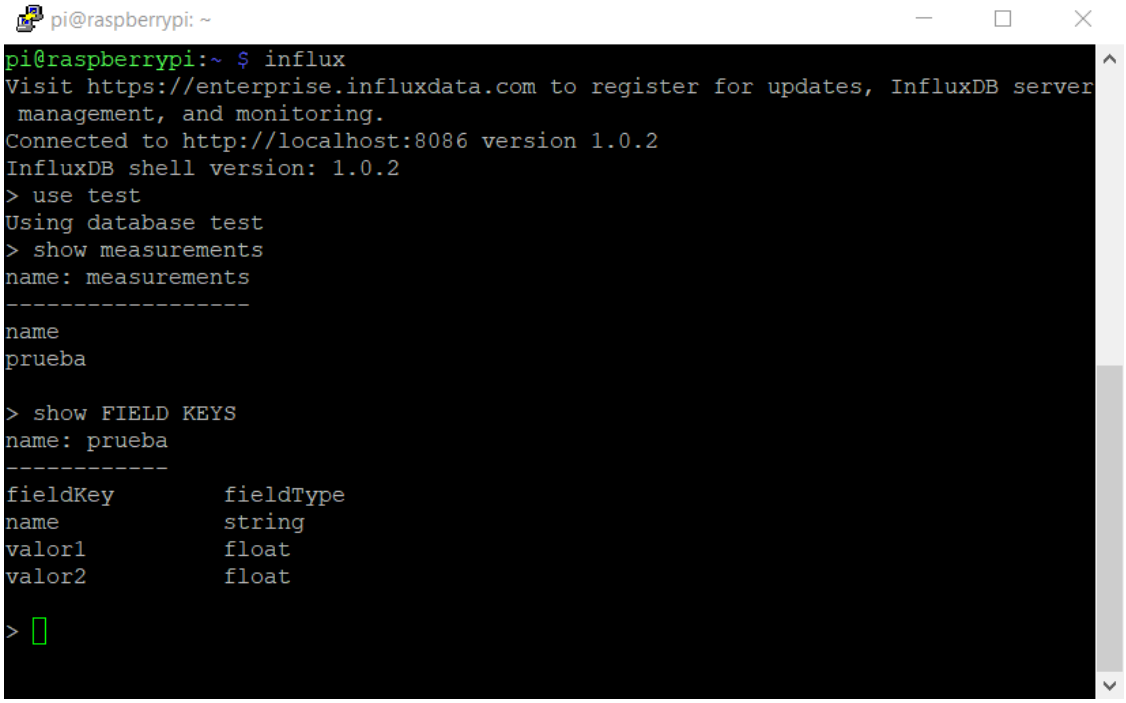


4. Una vez hecho lo anterior, Hacemos **Deploy** nuevamente en Node-RED, para que cree el *Measurement* **prueba**, con los respectivos *Field Keys* comenzar a recibir los primeros valores. Luego lo observarnos con el cliente influx.

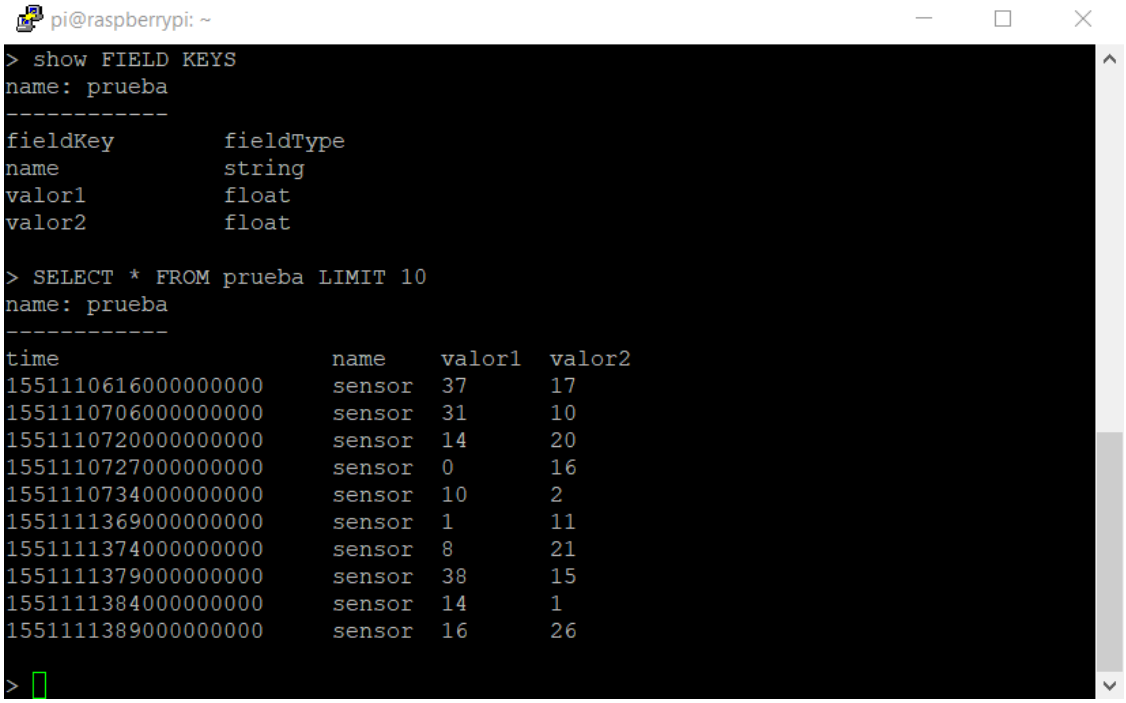
```
use test
show measurements
```




```
show FIELD KEYS
```



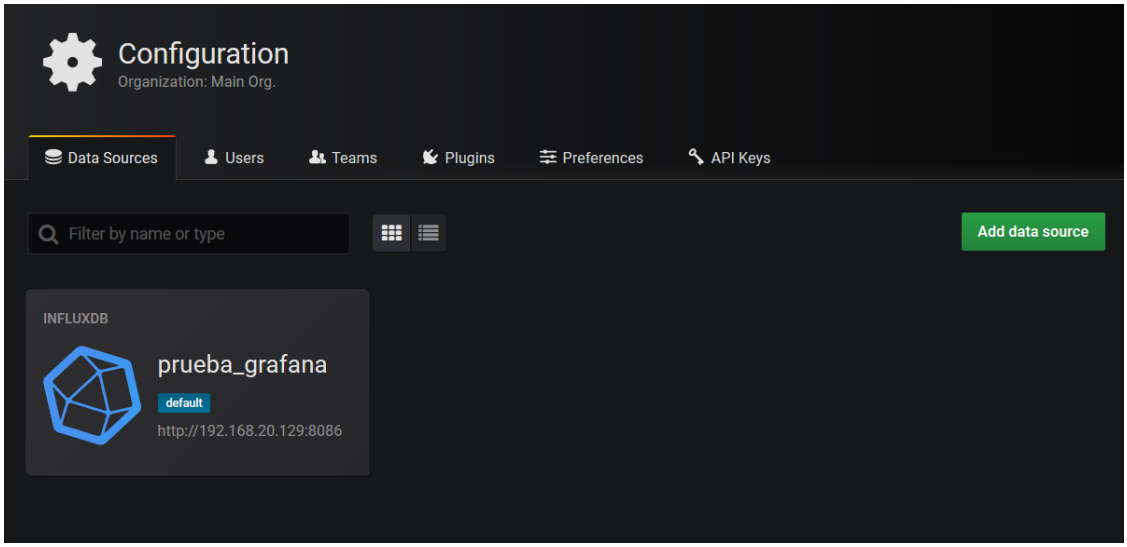

```
SELECT * FROM prueba LIMIT 10
```



Fase 3 - Grafana

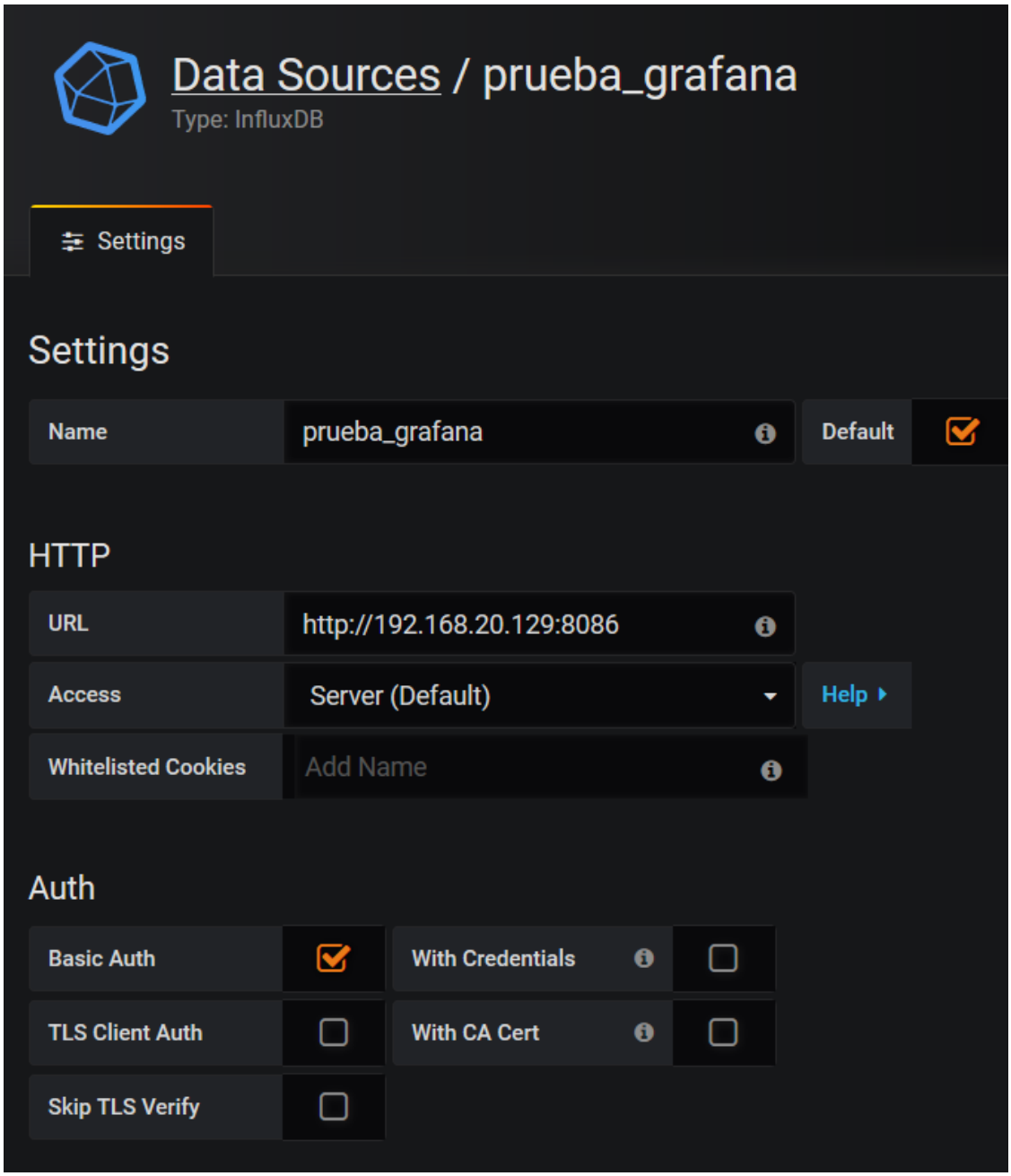
Grafana toma los datos insertados en la base de datos temporal y los pone a disposición en una grafica de nuestro gusto.

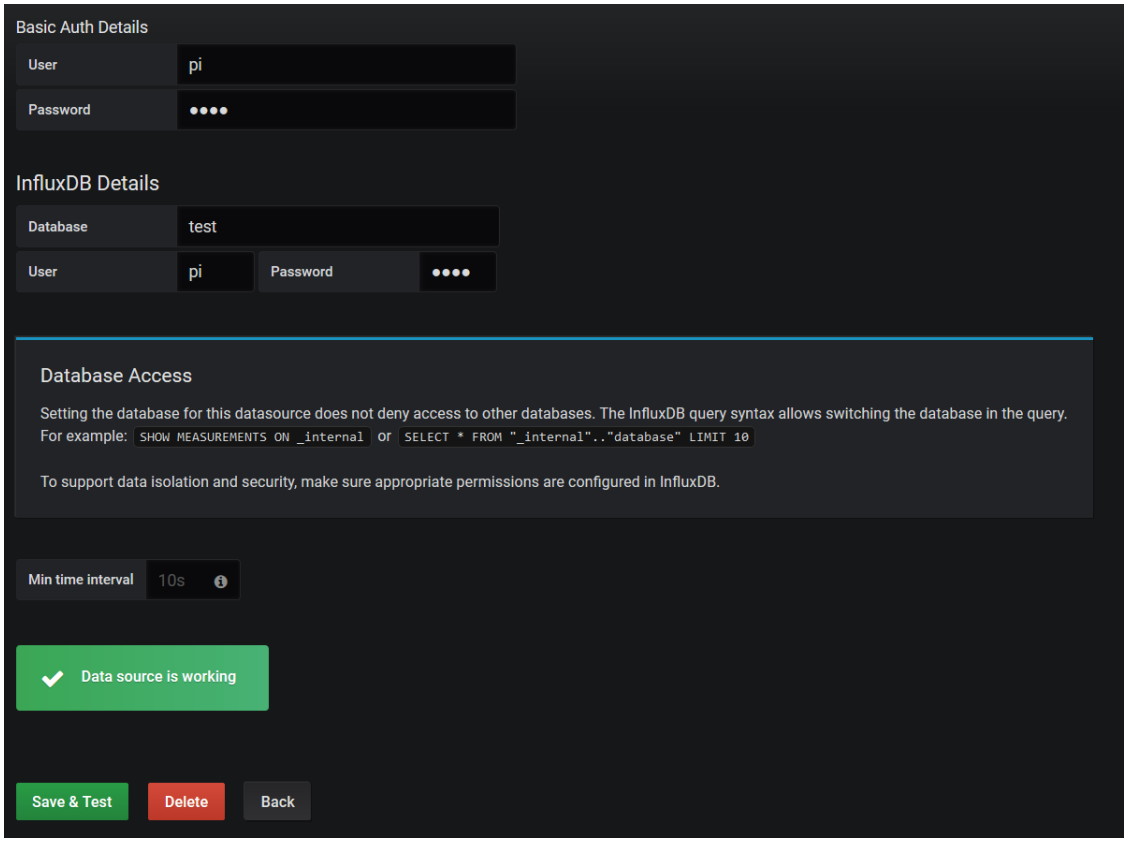
1. Ingresar a la aplicación web con nuestra IP local o a la que estamos accediendo en la misma red, en este caso `http://192.168.20.129:3000` e ingresar con el *usuario:contraseña* correspondiente.
2. Asignar una fuente de datos. Ir a configuración, sobre el panel izquierdo y seleccionar `Data sources`. Luego presionar sobre `Add data source`.



En la imagen aparece ya agregada, pero el procedimiento es el mismo para agregar cualquier *data source*.

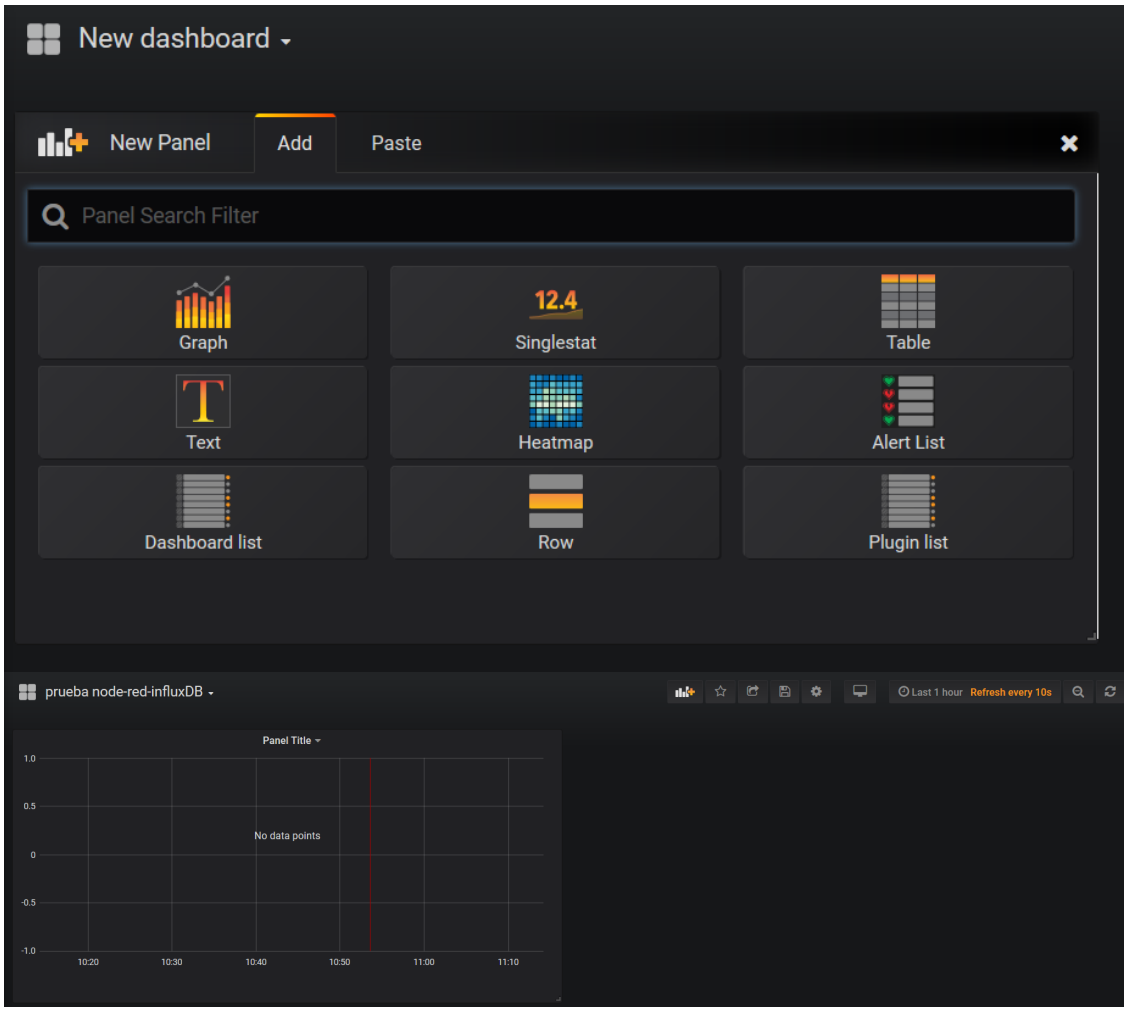
3. Configurar la fuente de datos. Aquí se pone la información necesaria para acceder a la base de datos Influx.



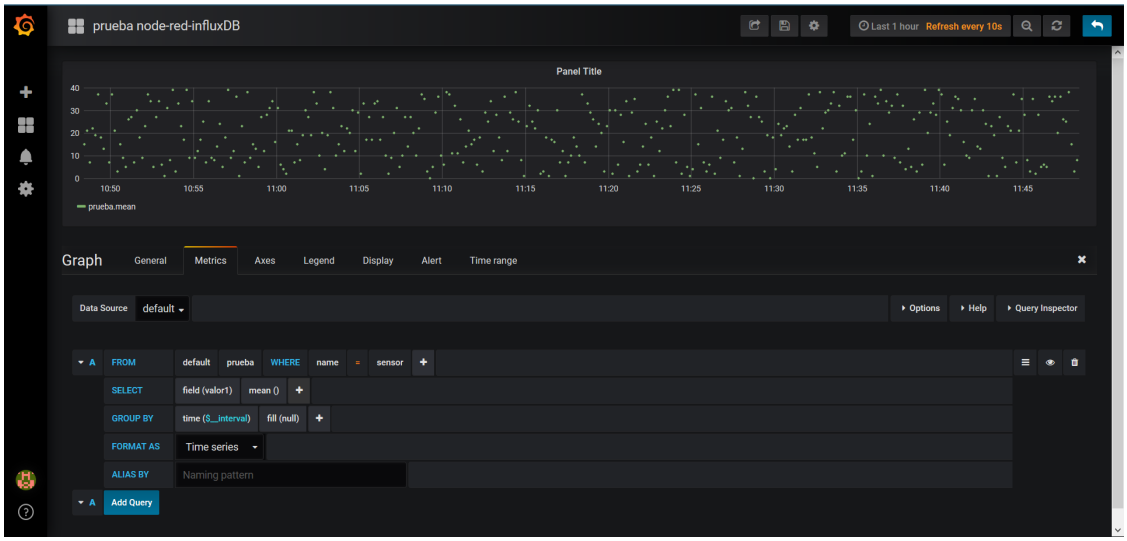


Se le pone un nombre (en este caso *prueba_grafana*). En las opciones HTTP se pone la dirección IP local o de la maquina a la que se accede en la red, con el puerto 8086, que es el puerto por defecto en el que la base de datos escucha las peticiones HTTP. Dejamos la opción `Access: Server(Default)`. En cuanto a la autenticación, solo ponemos la básica, que son nuestro *usuario:contraseña* de Raspbian (**pi:it10**). Por ultimo en los detalles de InfluxDB, se indica la base de datos a la que se quiere acceder. En este caso a `test`, con el *usuario:contraseña* **pi:it10**. Esta ultima parte de autenticación no es necesaria ya que se accede de manera local con el mismo usuario pi. Se guarda y se procede a crear un dashboard.

4. Crear un dashboard. Aquí se decide que tipo de panel se va a utilizar. Por el momento solo usaremos gráficos.

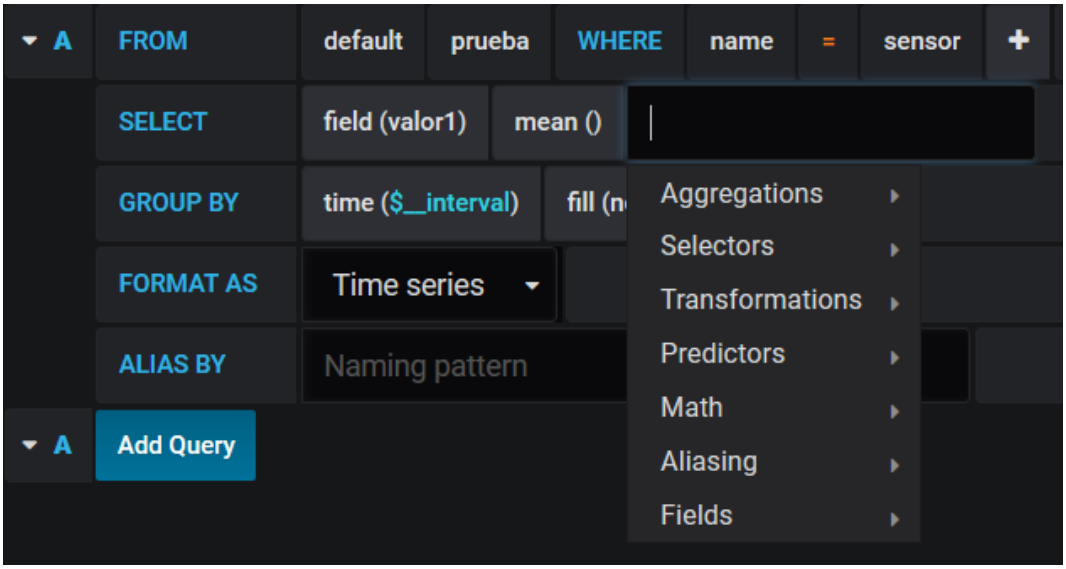


Sobre el `Panel Title` se hace click, luego `Edit`.



En la pestaña metrics esta la parte esencial para mostrar los datos. Es donde se encuentra la peticion a la base de datos. Por el momento solo realizaremos una sola para este panel.

- En la línea `FROM` : *Data source Measurement* `WHERE` *Field* `=` *valor*. Esta línea debemos indicar la fuente como **default** o **prueba_grafana**, **prueba** como *Measurement*. `WHERE` nos sirve para identificar datos particulares dentro de la misma base de datos, como por ejemplo todo lo referido a un sensor, entonces se indica **name** como *Field* y **sensor** valor de ese field. En el caso que tuviéramos mas sensores de esta forma se puede discriminar uno de otro.
- En la línea `SELECT` : *field() mean()*. En field, indicamos que para **valor1** realizaremos la grafica con sus datos, y realizando la media de los datos aportados en el intervalo de tiempo que se ha consultado (había quedado por defecto 10 segundos).

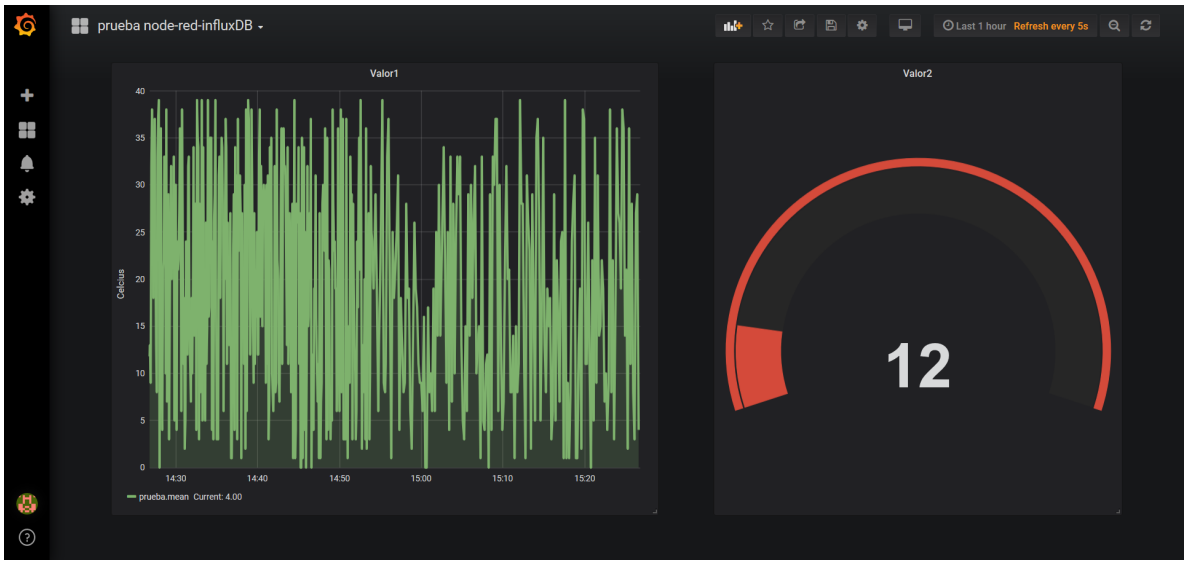


Se puede agregar o modificar la caracteristica de como se procesan los datos, no es solamente **mean()**, Para el caso es el mas conveniente para visualizar.

- En la línea `GROUP BY`: `time($_interval)` `fill(null)`. Aquí se indica el tiempo de acceso a los datos y como mostrarlos. Dejamos por defecto `time` y en fill podemos dejarlo como **null**, que se visualiza en forma de puntos, o con **none**, en forma de linea continua.
- En la línea `FORMAT AS`: `Time Series` lo dejamos por defecto.

5. Para ver mas cuestiones títulos, leyendas, formas de grafico, etc, están las demás pestañas. No lo analizaremos por el momento.

El resultado de toda esta configuración es la siguiente:



ANEXO XI

Proyecto "Invernadero Inteligente IoT" usando WeMos D1 Mini, Node-RED, InfluxDB y Grafana

Introducción

Gracias al módulo IoT de la Edukit10 se puede montar un invernadero inteligente. Además, usando otras herramientas y plataformas open-source se puede potenciar. La idea general es tomar datos del ambiente a través de sensores con la Edukit, vía WiFi con una placa WeMos D1 mini se envían los datos usando protocolo MQTT a un broker instalado en una Raspberry Pi, que luego serán leídos y guardados en una base de datos temporal, InfluxDB. Por último, se busca obtener graficas en tiempo real, con Grafana, de lo que esta sucediendo en nuestro invernadero.

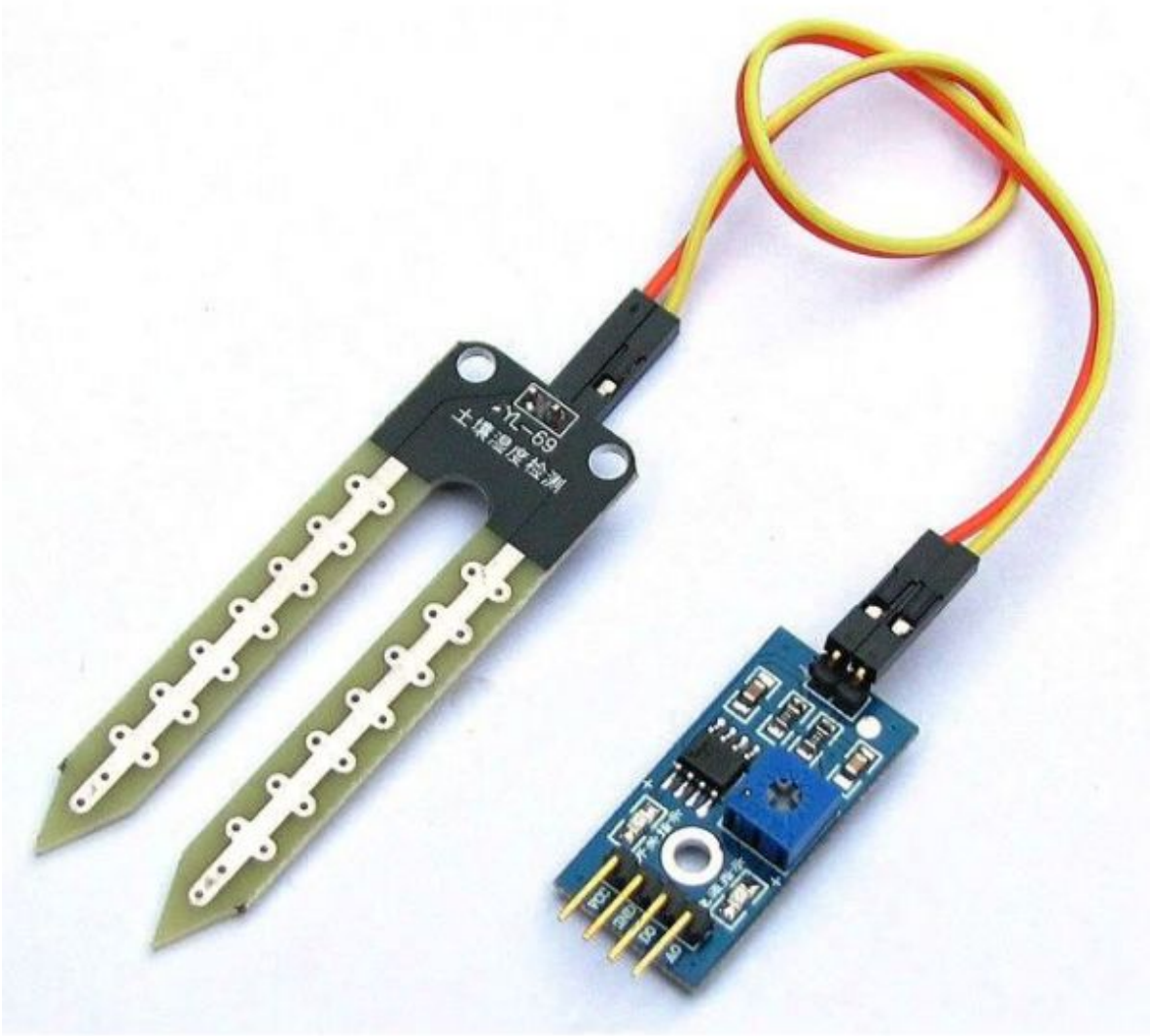
¿Que necesitamos?

- Edukit10 y su respectivo cable USB
- WeMos D1 mini
- Raspberry Pi 3+
- Cable USB-microUSB
- Sensor de monóxido de carbono MQ-9
- Sensor de humedad para tierra YL-69
- Protoboard
- Cables macho-macho y macho-hembra

Conexión de elementos

Por una cuestión de orden, se creo una placa PCB, pero con una protoboard se puede lograr los mismos resultados.

Sensor de humedad



Este sensor puede medir la cantidad de humedad presente en el suelo que lo rodea empleando dos electrodos que pasan corriente a través del suelo, y lee la resistencia. Mayor presencia de agua hace que la tierra conduzca electricidad más fácil (Menor resistencia), mientras que un suelo seco es un conductor pobre de la electricidad (Mayor resistencia).

Para aplicación como sensor de humedad en suelo por largos periodos de tiempo, se recomienda alimentar el módulo electrónico automáticamente para encenderlo únicamente al momento de tomar las mediciones, apagándolo inmediatamente al terminar y así minimizar la corrosión electrolítica. También se puede intercambiar las conexiones del elemento sensor periódicamente para que los dos electrodos roten de polaridad.

Características:

- Medida análoga de la humedad con salida de variación de voltaje (AO)
- Señal digital de superación de umbral con salida para el usuario (DO) y LED indicador. La sensibilidad de disparo se puede ajustar mediante trimmer. Esta función es provista por un comparador con LM393
- Pines de conexión de la tarjeta: VCC: alimentación, GND: Tierra, DO: Salida digital indicadora de superación de umbral, AO: Salida análoga de la medición de humedad
- LED indicador de encendido
- Voltaje de alimentación: 2 V a 6 V
- Dos agujeros de sujeción en el sensor de diámetro 3 mm aprox. y un agujero de sujeción en el módulo electrónico de 2 mm aprox.
- Dimensiones aprox: Sensor 6 cm x 2 cm. Módulo electrónico 4 cm x 1.5 cm

Esquema de montaje

Las conexiones de este sensor son bastante simples, la sonda no posee polaridad y por lo tanto es igual como la conectemos al modulo de medición, el resto de conexiones son de alimentación y la salida del modulo que será análoga o digital dependiendo de cual queramos usar (en este caso usaremos la análoga).

Las conexiones que usaremos para el prototipo son:

Modulo medición	**Edukit**
VCC	5V
GND	GND
A0	A4

Sensor de gases MQ-9

Este sensor de gas es sensible al monóxido de carbono (CO), pero también es sensible a gases inflamables (Gas Natural y Butano). Tiene alta sensibilidad (ajustable mediante potenciómetro) y un tiempo de respuesta rápido. Dispone de una capa sensible de Dióxido de Estaño (SnO2). El monóxido de carbono se detecta a temperatura de calentamiento baja (1.4V) mediante ciclos de temperatura elevada y baja. La conductividad eléctrica incrementa con la concentración de monóxido de carbono en el aire. El sensor tiene 6 pines, 4 para la medición de la señal y 2 para el calentador, sin embargo el módulo dispone directamente de un pin analógico, otro digital, además de alimentación y masa.

Características:

- Tensión de alimentación: 5V
- Concentración: 20-2000ppm CO, 500ppm-10000ppm CH4, 500ppm-10000ppm LPG
- Temperatura de calentamiento: Alta (5V), Baja (1.4V).
- Sensibilidad ajustable con potenciómetro.
- Resistencia de calentamiento: 31Ω±3Ω(Temperatura ambiente)

- Potencia de calentamiento: ≤350mW
- Temperatura y Humedad: -20°C 50°C
- RL: 1K

Propiedades eléctricas:

- Voltaje de entrada: DC5V consumo de energía (actual): 150mA
- Salida do: TTL digitales 0 y 1 (0.1 y 5 v)
- Sa: cerca de 0.1-0.3 V (relativamente limpio), la mayor concentración de tensión 4 V
- Nota especial: después de que el sensor se activa, necesita calentarse alrededor de 20 segundos, los datos medidos se estabilizan.

Esquema de montaje

El esquema eléctrico es sencillo. Alimentamos el módulo conectando GND y 5V a los pines correspondientes de la Edukit.

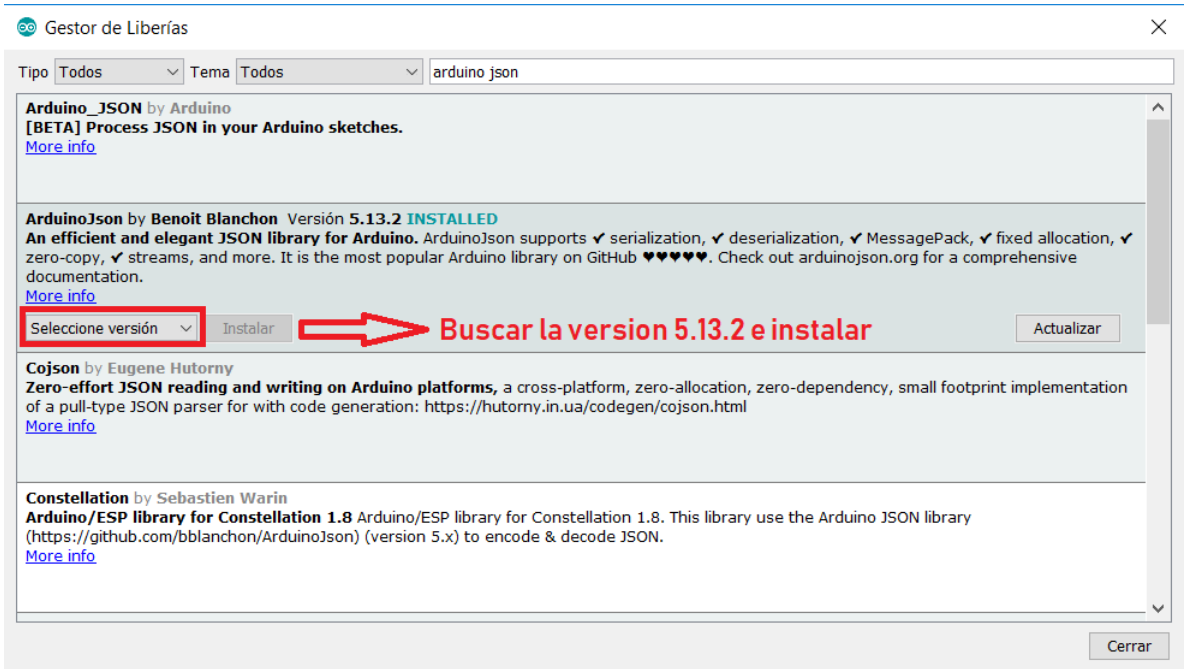
En este caso la el pin digital no se conecta a nada, solo el analógico, que es por donde se leerán los datos obtenidos por el sensor.

Las conexiones que usaremos para el prototipo son:

Modulo medición	Edukit
VCC	5V
GND	GND
A0	A5

Edukit10

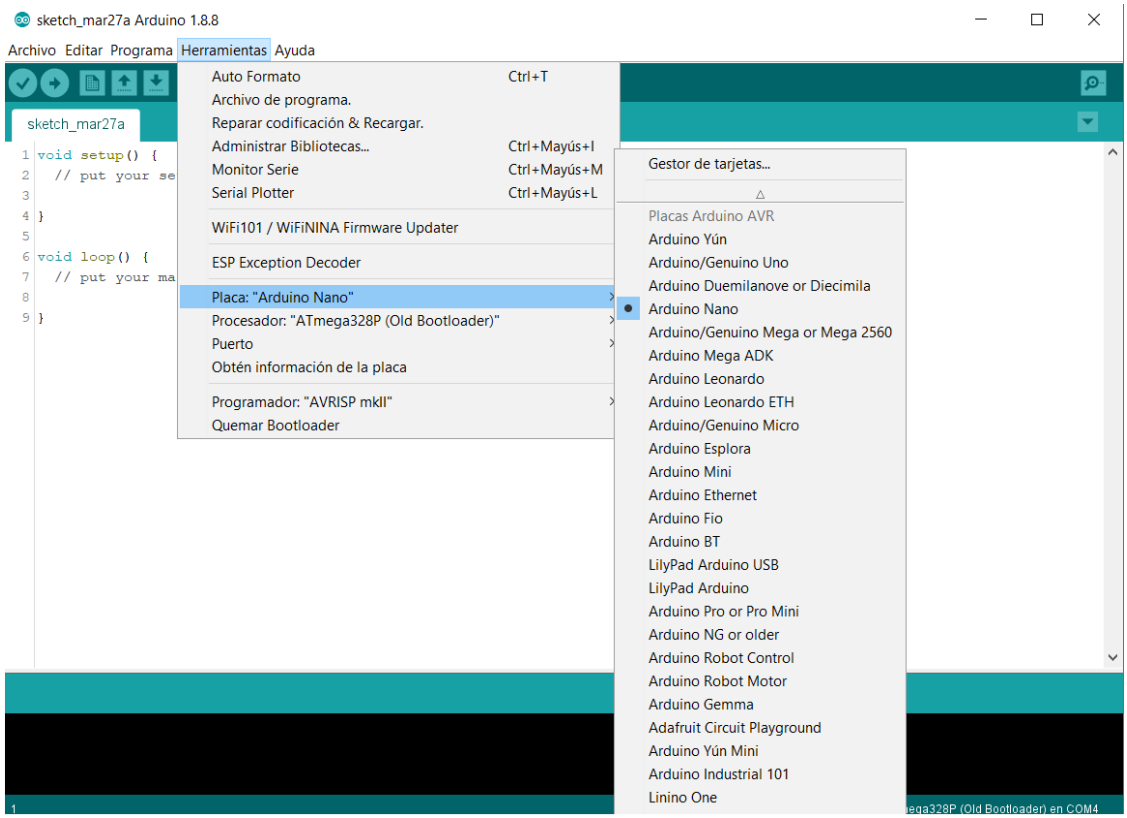
Con la placa Edukit conectada a la PC, ingresamos al Arduino IDE. Luego deberemos descargar la librería Arduino Json(https://arduinojson.org/). Para esto, vamos a `Programa` -> `Inlcuir Librería` -> `Administrar Bibliotecas`. Una vez allí, deberemos usar el buscador.



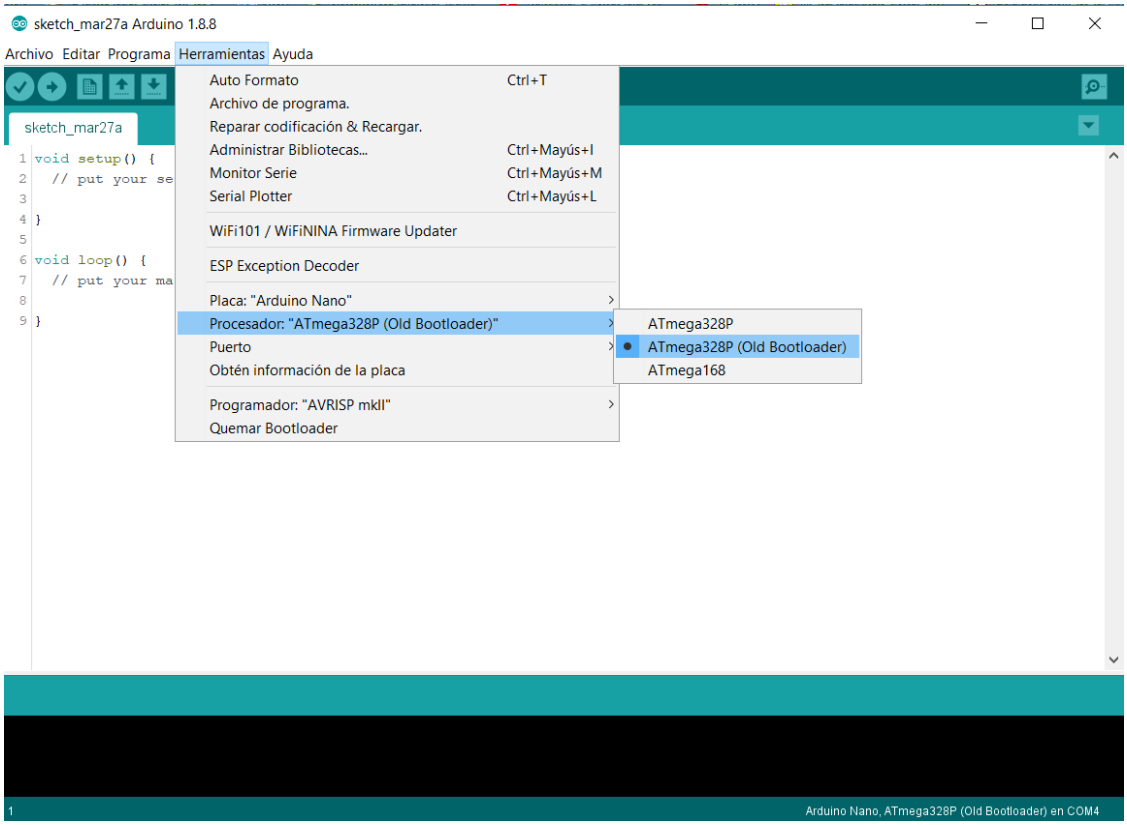
Seleccionamos la librería desarrollada por Benoit Blanchon. Tener en cuenta que se esta utilizando la version 5.13.2

Luego procedemos a configurar Arduino IDE para cargar nuestro codigo. En el menu `Herramientas` seleccionamos lo siguiente:

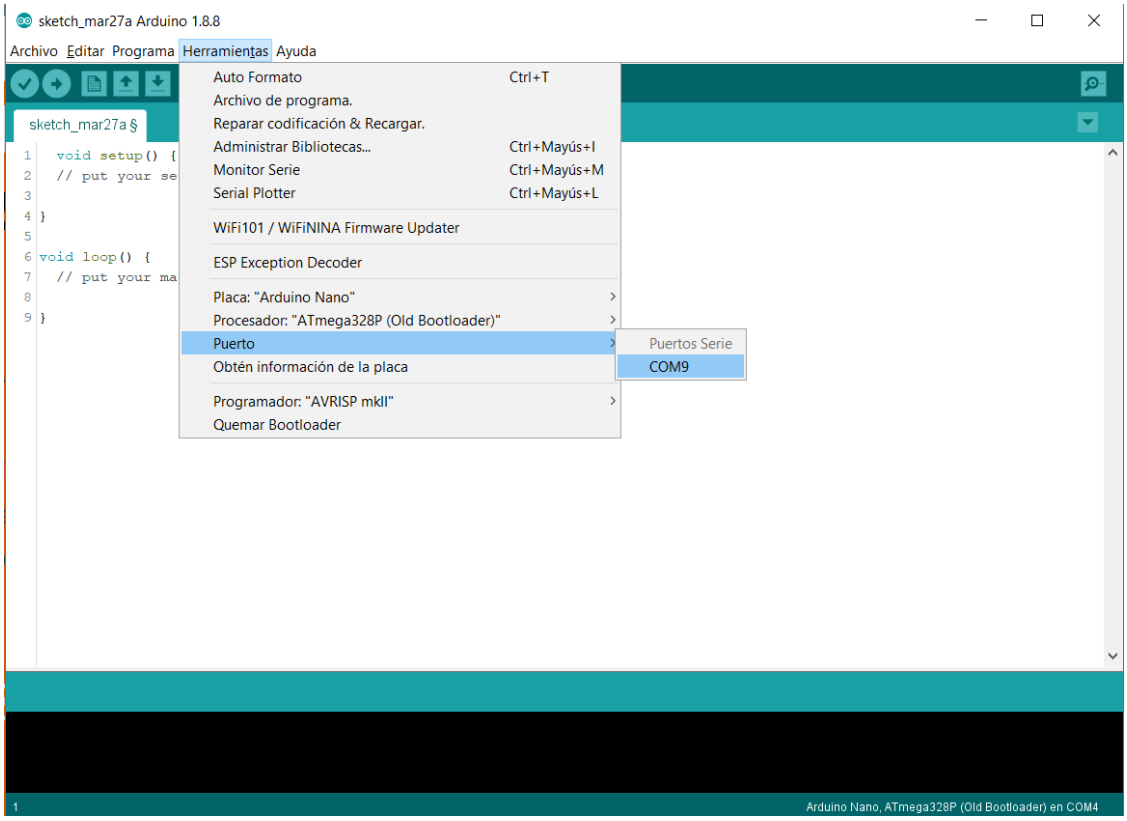
- Placa: "Arduino Nano"



- Procesador: "ATmega328p (Old Bootloader)



- Puerto: El que sea necesario. El sistema operativo debería detectarlo cuando se conecta al equipo. (En este caso COM9, puede ser otro)



Una vez configurado como se cargará el programa a la Edukit, se ingresa el siguiente código:

```
#include <SoftwareSerial.h>
#include <ArduinoJson.h>

SoftwareSerial mySerial(10, 9); // RX, TX

//################################ Sensor CO - MQ-9 ########################################
const int MQ9_PIN = A5; //PIN ANALOGICO DE MQ-9
//#########################################################################################

//################################ Humedad - YL69 ##########################################
const int HUM_PIN = A4; //PIN ANALOGICO DE YL69
int Humedad=0;//Variable utilizada para calcular la humedad
int Lectura_Analogica=0;//variable para leer el valor del pin analogico
//#########################################################################################

//################################ Sensor de Luz - LDR #####################################
const int LDR_PIN = A7;
int val_LDR = 0;
//#########################################################################################

//################################ Temperatura - LM35 ######################################
const int TEMP_PIN = A6;
int lecturaADC = 0;
double voltajeLM35 = 0.0;
double TemperaturaLM35 = 0.0;
```

```
//#########################################################################
###############

//############################## Variables Generales
###################################
unsigned long tiempo1 = 0;
unsigned long tiempo2 = 0;

unsigned long tiempo_pub1;
unsigned long tiempo_pub2;
int size_char = 12;
const int DELAY = 2000;
const int seg = 1000;

String topic = "testpps/";
//Esto es lo que quiero mandar para interactuar con InfluxDB:
//String mensaje = 'nodo' + ',' + MQ9_valor + ',' + hum + ',' + ldr + ','
+ temp;
String topic_redundante = "testpps/control/";
//#########################################################################
###############

void setup()
{
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  Serial.println("Inicio!");
  mySerial.begin(9600);

  tiempo1=millis();
}

void loop()
{
  tiempo2=millis();
  if(tiempo2 > (tiempo1+(30*seg))){
    tiempo1=millis();

//############################## Sensor CO - MQ-9
########################################
    int raw_adc_MQ9 = analogRead(MQ9_PIN);
    float value_adc_MQ9 = raw_adc_MQ9 * (5.0 / 1023.0);
    char MQ9_valor[size_char];
    dtostrf(value_adc_MQ9, 5, 2, MQ9_valor);

//############################## Humedad - YL69
########################################
    Lectura_Analogica = analogRead(HUM_PIN);//Leer el valor del
potenciometro
    Humedad = map(Lectura_Analogica, 0, 1023, 100, 0);//Escala para
utilizarlo con el servo
    char hum[size_char];
    dtostrf(Humedad, 5, 2, hum);
    //(variable float, ancho, cantidad de decimales, variable char)

```

```
75  //############################ Temperatura - LM35
    ######################################
76      lecturaADC = analogRead(TEMP_PIN);
77      voltajeLM35 = ((double)lecturaADC/1023)*5;
78      TemperaturaLM35 = voltajeLM35/0.01;
79      //char temp[5];
80      char temp[size_char];
81      dtostrf(TemperaturaLM35, 5, 2, temp);
82
83  //############################## Sensor de Luz - LDR
    ####################################
84      val_LDR = analogRead(LDR_PIN);
85      char ldr[size_char];
86      dtostrf(val_LDR, 5, 2, ldr);
87
88  //############################## Publicación al broker
    ####################################
89      String mensaje;
90      mensaje.concat("edukit");
91      mensaje.concat(",");
92      mensaje.concat("mq9=");
93      mensaje.concat(MQ9_valor);
94      mensaje.concat(",");
95      mensaje.concat("hum=");
96      mensaje.concat(hum);
97      mensaje.concat(",");
98      mensaje.concat("ldr=");
99      mensaje.concat(ldr);
100     mensaje.concat(",");
101     mensaje.concat("temp=");
102     mensaje.concat(temp);
103
104     pub_mqtt(topic, mensaje);
105
106 // ESPERANDO FINALIZAR TRANSMISION SERIAL
107     tiempo_pub1 = millis();
108     tiempo_pub2 = millis();
109     while (tiempo_pub2 < (5*seg)+tiempo_pub1) {
110       tiempo_pub2 = millis();
111     }
112     /*Hay que darle algo de tiempo entre publicacion y publicacion al
    broker porque arduino no procesa todo al instante. Se le da un tiempo de
    3segundos entre envio y envio.*/
113
114     StaticJsonBuffer<80> jsonBuffer;
115     char json[80];
116     JsonObject& root = jsonBuffer.createObject();
117
118     root["nodo"] = "edukit";
119     //Se formatea el char del sensor MQ-9 a string y luego a Float, porque
    asi ya viene redondeado a 2 decimales
120     String inString = "";
121     inString += MQ9_valor;
122     root["mq9"] = inString.toFloat();
123     inString = "";
124
125     root["hum"] = Humedad;
126     root["ldr"] = val_LDR;
```

```
127
128     inString += temp;
129     root["temp"] = inString.toFloat();
130
131     root.printTo(json, sizeof(json));
132     pub_mqtt(topic_redundante,json);
133   }
134 }
135 //#########################---------------FUNCIONES-------------------
    ########################
136 void pub_mqtt(String topic, String payload) //que sea recurrente
137 {
138   String a;
139   a = topic + ';' + payload;
140   Serial.println(a);
141   mySerial.println(a);
142 }
```

Como vimos antes, se necesitan las librerías `ArduinoJson` y `SoftwareSerial` (esta última ya incluida en la IDE de Arduino). Se hace comunicación serial por software con la WeMos D1 Mini a través de los pines 10 y 9, como Rx y TX, respectivamente.

La primer parte del código esta compuesta por declaración de variables y pines. Si bien se toma una decisión de conexión en cuanto al sensor MQ-9 y el YL-69, se podrían usar otros, que sean analógicos. Para más información revisar la documentación y el detalle del pinout de la placa.

La segunda parte del código, ya mas orientada a la ejecución, se encarga de la toma de datos de los sensores, conversión analógica/digital, y publicación al broker MQTT. Para todos los sensores, excepto el LDR, la conversión analógica/digital se hace teniendo en cuenta 10 bits, cuantizados en 5 volt, la tensión continua que toma cada uno de ellos. Por ejemplo:

```
1 float_value_adc_MQ = raw_adc_MQ9 * (5.0 / 1023.0);
```

Luego se convierten a una cadena de caracteres que tiene como máximo dos decimales con `dtostrf`.

En la parte de publicación al broker se van concatenando las cadenas de caracteres con el dato de cada sensor, separado en nombre y valor. Esto más tarde lo va a leer Node-RED para darle el formato necesario para guardarlo en la base de datos temporal InfluxDB. Se puede ver que hay dos publicaciones, es solo por una cuestión de redundancia de datos, en otro topic, que no estará leyendo particularmente Node-RED, y que además se escribe como JSON, para un posterior tratamiento si se considera necesario.

Una cuestión no menor es que hay dos temporizadores. Uno que se provee con la finalidad setear el tiempo entre transmisiones de datos a la base de datos, y el otro separar en tiempo las transmisiones a los dos topics MQTT. El segundo existe porque al usar un enlace serial entre la WeMos D1 mini y la Edukit tiene una transmisión bastante lenta, y termina siendo un cuello de botella para el flujo de datos. Para el volumen de sensores que se están manejando se toma como prudente un tiempo de 5 segundos entre transmisión y transmisión. Puede variar de acuerdo a la tasa de transmisión elegida para la conexión serial por software.

WeMos D1 Mini

El WeMos D1 mini ESP8266 es una tarjeta de desarrollo similar a Arduino, especialmente orientada al Internet de las cosas (IoT). Está basado en el SoC (System on Chip) ESP8266, un chip altamente integrado, diseñado para las necesidades de un mundo conectado. Integra un potente procesador con Arquitectura de 32 bits (más potente que el Arduino Due) y conectividad Wifi.

WeMos está diseñado especialmente para trabajar en protoboard o soldado sobre una placa. Posee un regulador de voltaje en placa que le permite alimentarse directamente del puerto USB. Los pines de entradas/salidas trabajan a 3.3V. El chip CH340G se encarga de la comunicación USB-Serial.

De forma muy resumida estas son algunas de las principales características:

- Velocidad: 80MHz/160MHz
- Flash: 4M bytes
- Tensión funcionamiento: 3.3V
- Entradas y salidas digitales: 11, todos (salvo el D0) con PWM, interrupciones, e I2C
- Entradas analógicas: 1 (Max. 3.2V)
- Conector Micro-USB

Una de las características que hacen especial al microcontrolador ESP8266 es el hecho de poder usar firmware alternativos para poder hacer usos del mismo sin necesidad de tener que programar. Entre esas alternativas nos inclinamos por Tasmota, firmware muy completo que permite el uso de numerosos dispositivos de una forma más genérica.

Tasmota

Firmware alternativo para dispositivos basados en ESP8266 con web, temporizadores, actualizaciones de firmware 'Over The Air' (OTA) y compatibilidad con sensores, lo que permite el control bajo Serial, HTTP, MQTT y KNX, para su uso en Smart Home Systems. Escrito para Arduino IDE y PlatformIO. Desarrollado principalmente para Sonoff, pero ahora compatible con muchos mas dispositivos, entre ellos la WeMos D1 mini.

Puede haber versiones nuevas, pero la que usaremos es la **v6.4.1**. Se puede descargar desde github, haciendo una clonación del repositorio o simplemente descargándolo. Desde aquí es posible obtener el firmware https://github.com/jnmanchado/Tasmota.

No es igual a la versión original del desarollador, ya que se hicieron algunas modificaciones al código para obtener lo que necesitamos. Queremos usar el firmware para publicar a un broker MQTT a través de la Edukit, que esta conectada a la WeMos D1 mini vía Software Serial.

Puesta a punto de WeMos D1 mini

Software necesario

1. PlatformIO (todas las bibliotecas y configuraciones necesarias están preconfiguradas en [platformio.ini])
2. Visual Studio Code

En este proyecto usaremos PlatformIO.

Otros requerimientos

- El código fuente del firmware (Sonoff - Tasmota)
- Un broker MQTT
- Un cliente MQTT para interactuar con el módulo

Herramientas para la carga del firmware

Visual Studio Code

Cómo preparar y configurar Visual Studio Code con PlatformIO para la compilación y carga de Tasmota.

Descargar e instalar Visual Studio Code

Descargar [Visual Studio Code] (VSC) desde su pagina web

Instalar la extensión PlatformIO

Instalar la extensión *PlatformIO IDE* en VSC.

Seleccionar `Ver` - `Extensiones` y escribir *PlatformIO* en el cuadro de búsqueda.

Asegurarse de seleccionar la extensión oficial de PlatformIO.org *PlatformIO IDE* y seleccionar Instalar. Aceptar para instalar dependencias.

Copiar archivos

Copiar todos los archivos del código fuente de la versión Tasmota en la carpeta de trabajo de VSC.

Compilar Tasmota

Iniciar VSC y seleccionar `File` - `Open folder...` para apuntar a la carpeta de trabajo.

Nota: Presionar `Ctrl + Shift + P` y escribir **PlatformIO** para ver todas las opciones.

Seleccionar el firmware deseado editando el archivo `platformio.ini` según sea necesario.

Cargar Tasmota

Habilitar las opciones deseadas en `platformio.ini` para la carga serial como:

```
; *** Subir el método de restablecimiento de serie para Wemos y NodeMCU
upload_port = COM5
; upload_speed = 512000
upload_speed = 115200
; upload_resetmethod = nodemcu
```

Para el caso de la WeMos D1 mini y el S.O. Lubuntu 18.04 usado, el `upload_port` en vez de ser **COM5** es **/dev/ttyUSB0**. Además hay que dar permisos para que platformIO pueda acceder al dispositivo.

Luego **Build**(si es necesario) y **Upload**. Consejo:

```
En caso de que VSC muestre una gran cantidad de errores usando `PlatformIO - Intellisense`, una posible "solución" es cambiar el cpp Intelli Sense a "TAG PARSER"

Esta configuración se puede cambiar en la configuración del área de trabajo mediante: Use `Ctrl` + `Shift` + `P` y escribir `Preferences: Open Workspace Settings` y escribir `intelli Sense` en el cuadro de búsqueda. Ahora cambiar el valor de `Intelli Sense Engine` a `Tag Parser`.
```

Configuración inicial desde `my_user_config.h`

Modificando el código antes de cargarlo

Modificando el archivo `my_user_config.h` se puede tener una configuración inicial personalizada sin necesidad de operar el dispositivo desde la WebUI. Se define el modulo para trabajar con WeMos D1 Mini con:

```
    #define MODULE          GENERIC
```

En cuanto a la información de red:

```
    #define WIFI_IP_ADDRESS        "0.0.0.0"         // [IpAddress1] Set to 0.0.0.0 for using DHCP or enter a static IP address
    #define WIFI_GATEWAY           "192.168.1.1"     // [IpAddress2] If not using DHCP set Gateway IP address
    #define WIFI_SUBNETMASK        "255.255.255.0"   // [IpAddress3] If not using DHCP set Network mask
    #define WIFI_DNS               "192.168.1.1"     // [IpAddress4] If not using DHCP set DNS IP address (might be equal to WIFI_GATEWAY)

    #define STA_SSID1              "[Ssid1]"               // [Ssid1] Wifi SSID
    #define STA_PASS1              "[Password1]"               // [Password1] Wifi password
    #define STA_SSID2              "[Ssid2]"               // [Ssid2] Optional alternate AP Wifi SSID
    #define STA_PASS2              "[Password2]"               // [Password2] Optional alternate AP Wifi password
    #define WIFI_CONFIG_TOOL       WIFI_RETRY        // [WifiConfig] Default tool if wifi fails to connect

```

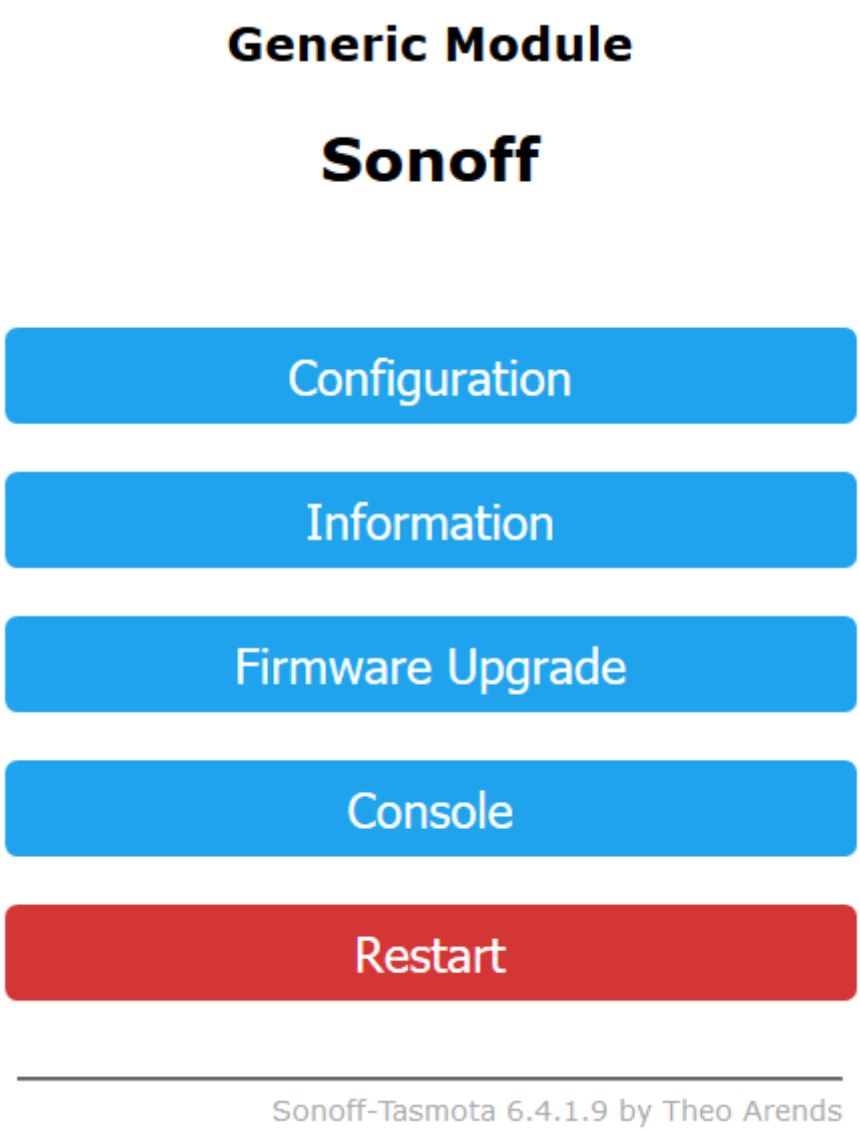
Para configurar el broker MQTT:

```
    #define MQTT_HOST              "direccion-MqttHost"               // [MqttHost]
    [...]
    #define MQTT_PORT              1883              // [MqttPort] MQTT port (10123 on CloudMQTT)
    #define MQTT_USER              "[MqttUser]"       // [MqttUser] MQTT user
    #define MQTT_PASS              "[MqttPassword]"       // [MqttPassword] MQTT password
    [...]
    #define MQTT_TOPIC             "Nombre-topic"           // [Topic] (unique) MQTT device topic, set to 'PROJECT "_%06X"' for unique topic including device MAC address
```

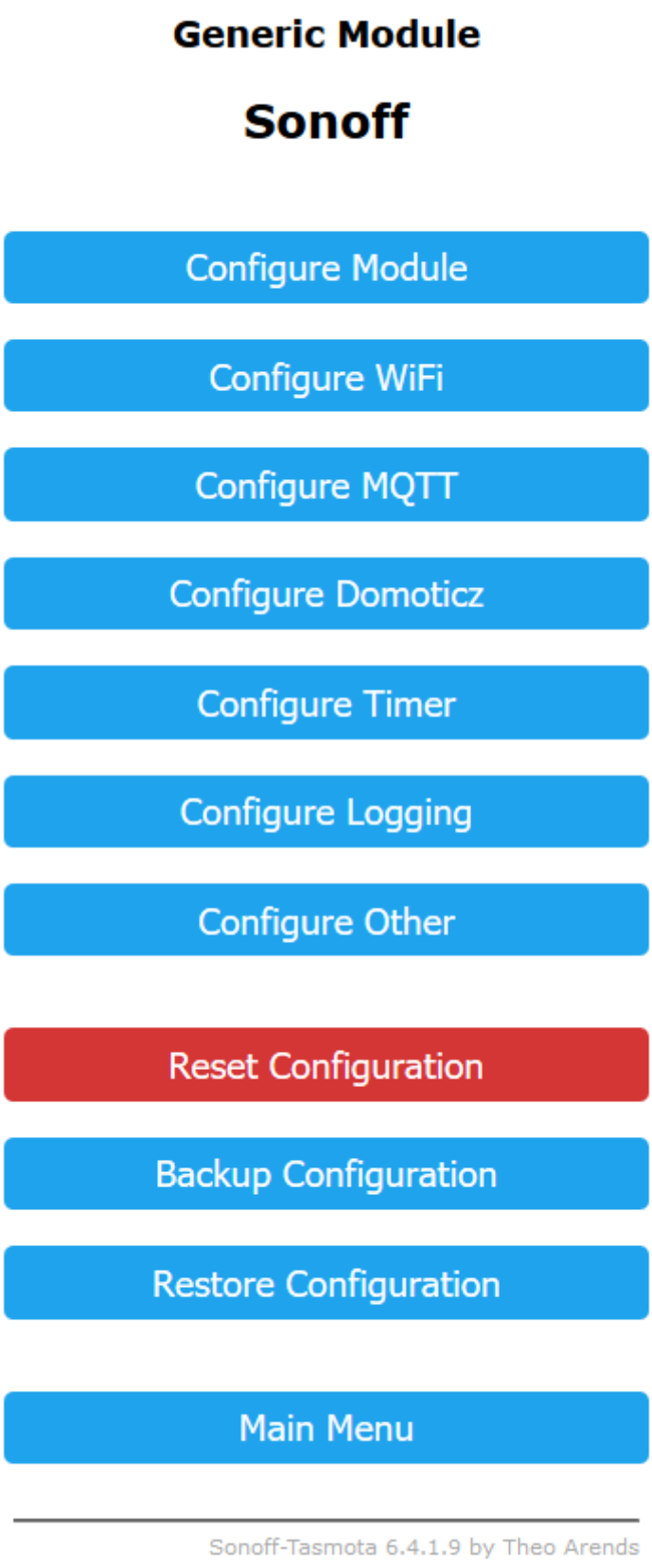
Luego puede haber mas o menos parámetros configurables, pero que es posible hacerlo desde un primer momento antes de compilar y flashear la placa, y de esta forma evitar usar la WebUI.

Configuración del dispositivo desde la WebUI

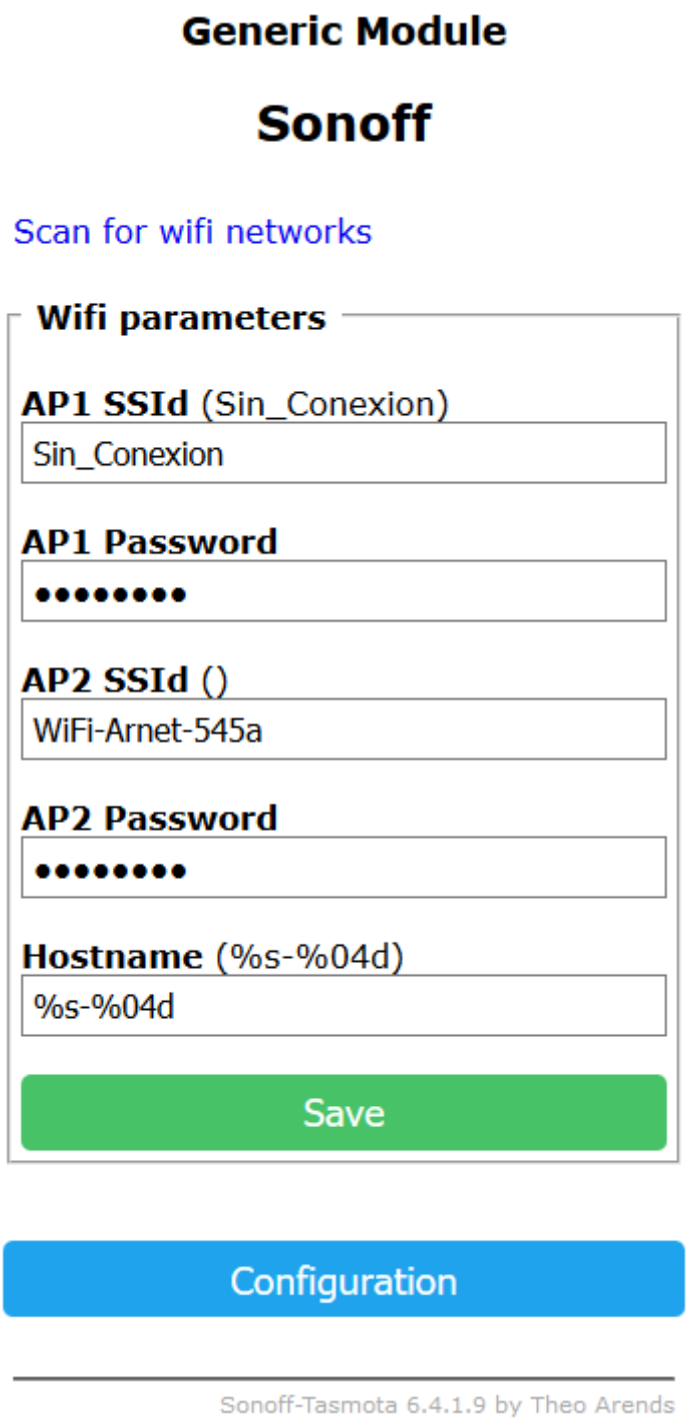
Si no se le da información de red para conectarse a su red WiFi, puede cargar el código y unirse a la red propia de Tasmota, y luego modificar los parámetros. Una vez conectado a la misma red, se accede al dispositivo entrando con un navegador web a http:\[ip-WeMos] (reemplazar por la IP correspondiente). Obtendremos algo como lo siguiente:



Ingresando a `Configuration` se pueden modificar todos los parámetros necesarios para poder trabajar. En nuestro caso necesitamos configurar el tipo de modulo, información de red y el broker MQTT.

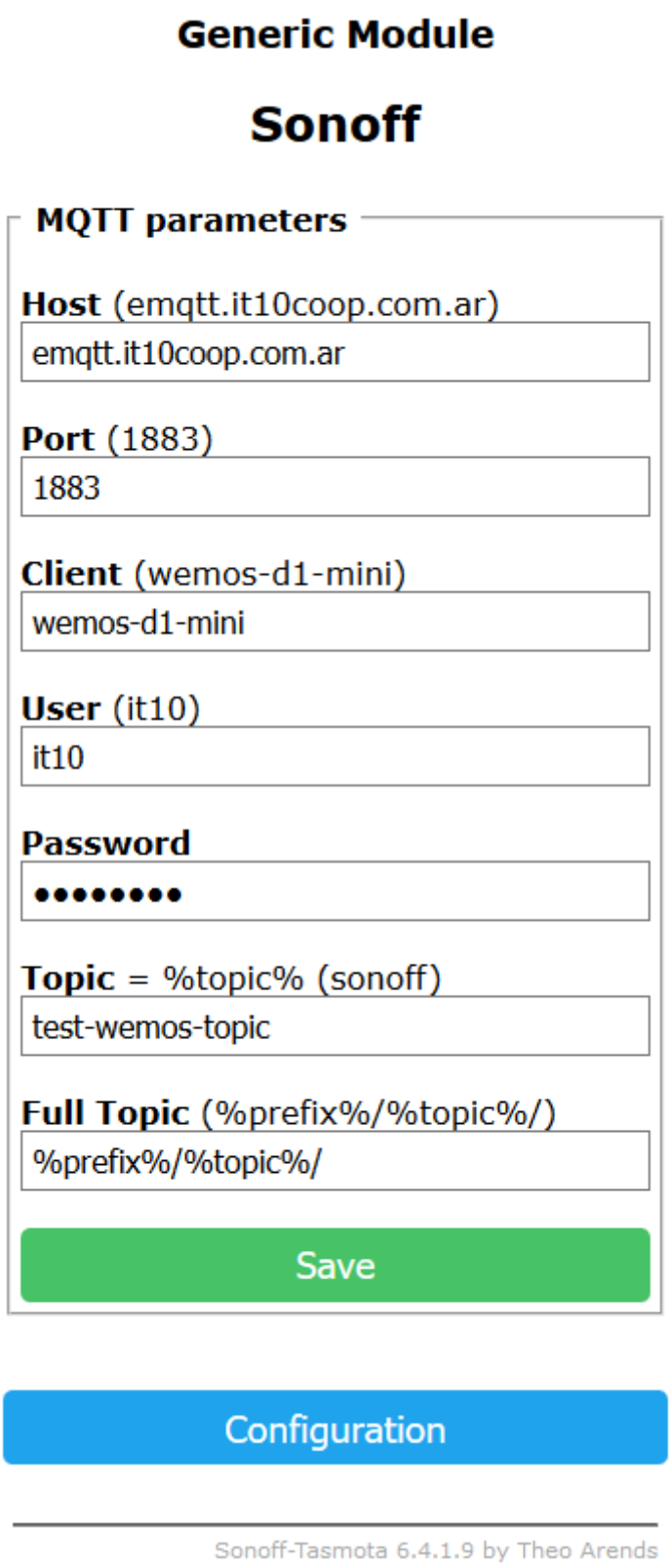


- Conexión a una red Ingresando a `Configure Wifi` se pueden configurar dos redes con sus SSID y contraseñas respectivas. Esto es porque por defecto Tasmota intenta conectarse a la primer red, si falla, prueba con la segunda, para garantizar la conexión Wireless en todo momento.



Presionar **Save** para guardar configuración.

- Broker MQTT Se procede a configurar los parametros para poder interactuar con el dispositivo a traves del protocolo de mensajería MQTT.



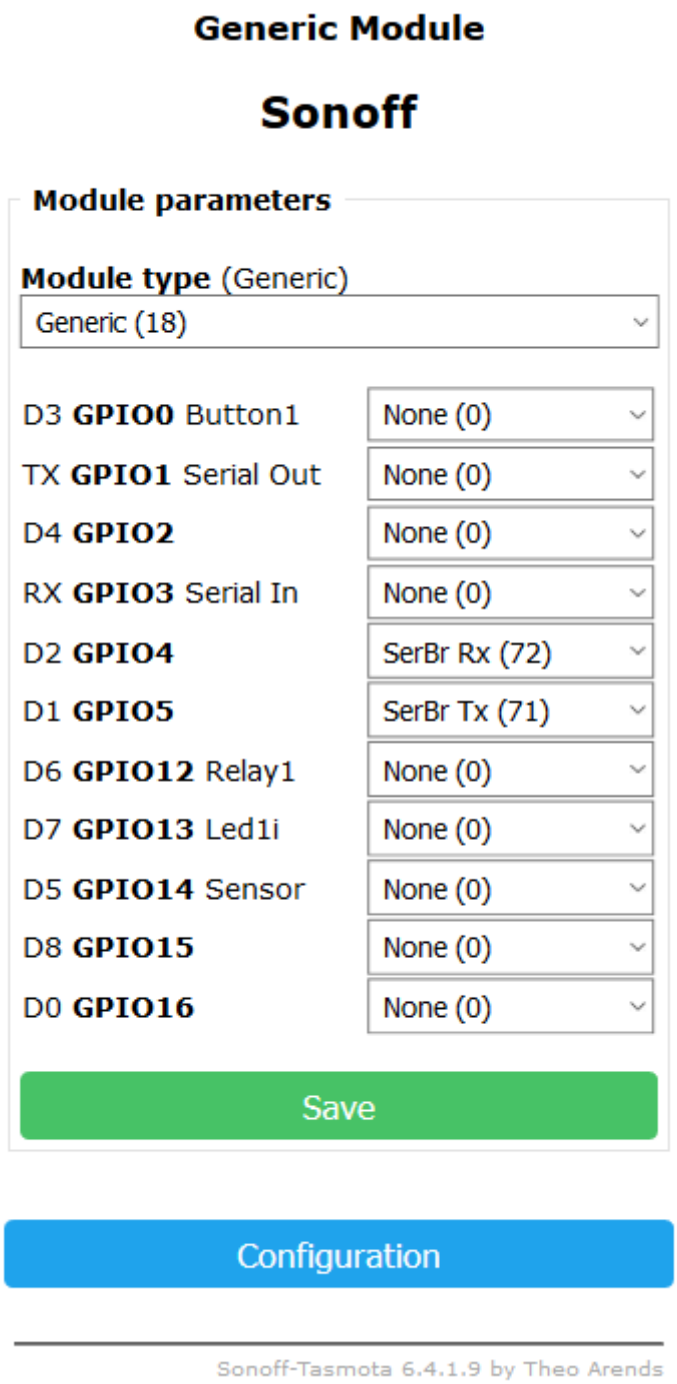
Presionar **Save** para guardar configuración.

Pines para comunicación serial por software de WeMos D1 mini

El ESP8266 tiene dos UARTS (puertos serie): UART0 en los pines 1 y 3 (TX0 y RX0 respectivamente), Y UART1 en los pines 2 y 8 (TX1 y RX1 respectivamente). Sin embargo, GPIO8 se utiliza para conectar el chip flash . Esto significa que UART1 solo puede transmitir datos.

UART0 también tiene control de flujo de hardware en los pines 15 y 13 (RTS0 y CTS0 respectivamente). Estos dos pines también pueden usarse como pines alternativos TX0 y RX0.

Se realiza una conexión serial por software, y para esto se utilizan pines digitales. En este caso GPIO5(D1) y GPIO4(D2) (SerBr Tx y SerBr Rx respectivamente).

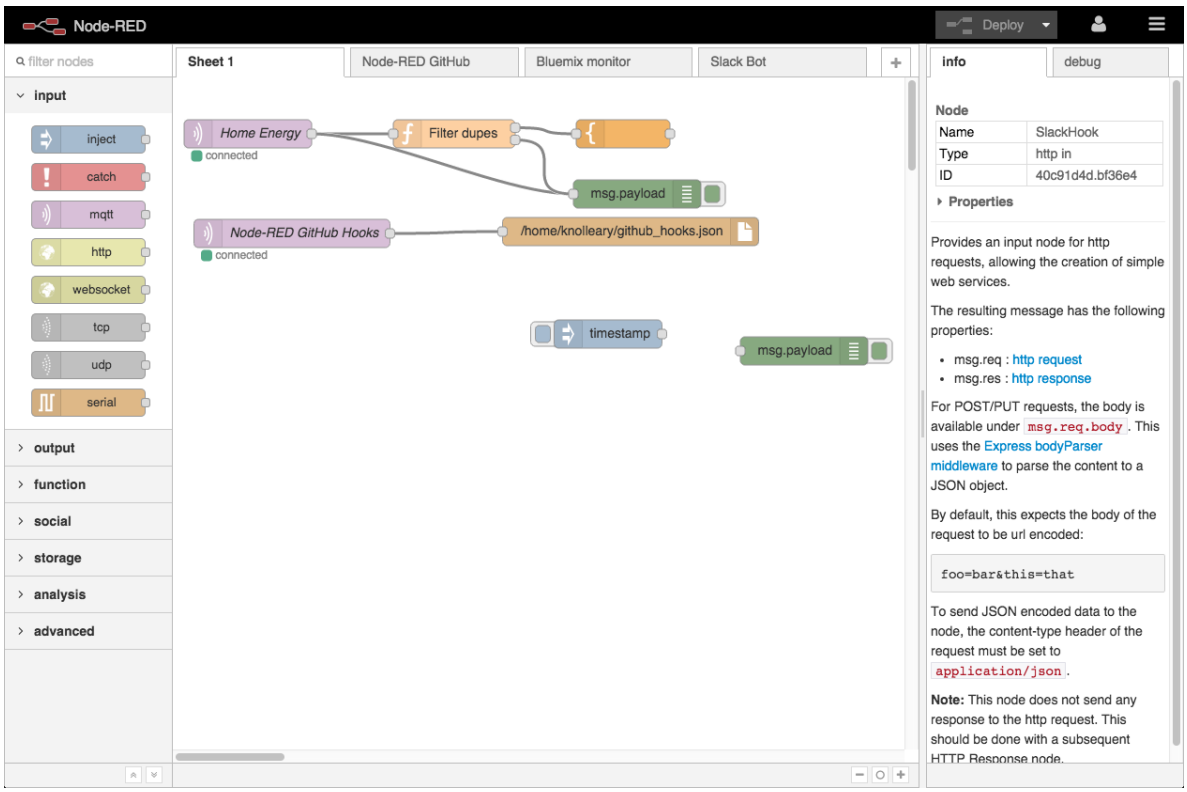


Node-RED

Node-RED (https://nodered.org/) es un motor de flujos con enfoque IoT, que permite definir gráficamente flujos de servicios, a través de protocolos estándares como REST, MQTT, Websocket, AMQP… además de ofrecer integración con apis de terceros, tales como Twitter, Facebook, Yahoo!…

Se trata de una herramienta visual muy ligera, programada en NodeJS y que puede ejecutarse desde en dispositivos tan limitados como una Raspberry, hasta en plataformas complejas como IBM Bluemix, Azure IoT o Sofia2 Platform.

El editor de flujos de Node-RED consiste en una sencilla interfaz en HTML, accesible desde cualquier navegador, en la que arrastrando y conectando nodos entre sí, es posible definir un flujo que ofrezca un servicio.



Instalación de Node-RED

Usando un equipo que corra como sistema operativo con kernel Linux, en una terminal ingresamos el comando:

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
```

En instalaciones mínimas de Debian puede que haga falta correr `sudo apt-get install build-essential` antes del comando anterior.

Corriendo Node-RED

Para iniciar Node-RED, hay dos formas:

- En el escritorio, seleccionar `Menu -> Programming -> Node-RED`
- O por medio de la terminal, `node-red-start`

Nota: cerrar la ventana (o `ctrl + c`) no frena la ejecución de Node-RED (seguirá corriendo en segundo plano)

Para parar Node-RED, ejecutar el comando `node-red-stop`.

Para ver el log, ejecutar el comando `node-red-log`.

Inicio automático en el booteo

Usar comando:

```
sudo systemctl enable nodered.service
```

Y de la misma manera, hacer `sudo systemctl disable nodered.service` para deshabilitar el inicio automático de Node-RED en el booteo.

Instalación de nuevos nodos

Instalaremos nuevos nodos para trabajar, estos son:

- Dashboard
  - `npm install node-red-dashboard` Nos permitirá realizar web dinámicas para mostrar nuestros resultados.
- Mqtt-Broker
  - `npm install node-red-contrib-mqtt-broker` Crea un broker local, que usaremos para vincular la Edukit con la base de datos InfluxDB
- InfluxDB
  - `npm install node-red-contrib-influxdb` Nodos para insertar y realizar consultas sobre una base de datos influxDB.

Hay dos formas de poder instalarlos, que describiremos a continuación.

- **Usando el editor:**

Desde la versión 0.15 se pueden instalar nodos directamente usando el editor. para esto se debe ir a **Manage Palette** desde el **menu**, luego seleccionar la pestaña **install**. Desde aquí se pueden buscar nuevos nodos, instalarlos, actualizarlos, habilitar o deshabilitar nodos existentes.

- Usando nodos con paquetes npm

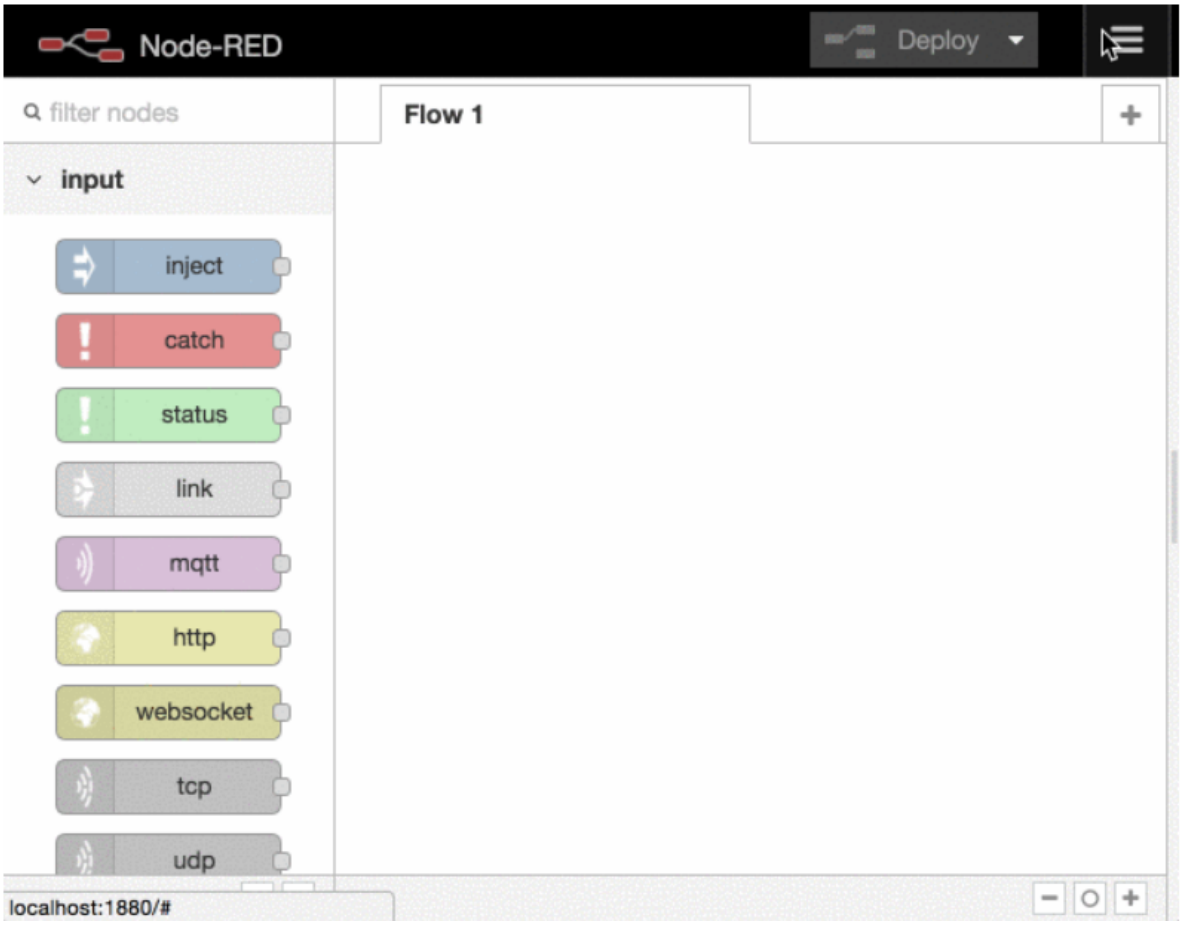
Se puede instalar de manera local en el directorio del usuario(por defecto, $HOME/.node-red):

```
1    cd $HOME/.node-red
2    npm install <npm-package-name>
```

Después de esto se debe hacer stop y restart sobre Node-RED para cargar los nuevos nodos.

Creando flujo

Para empezar a trabajar con Node-RED se debe iniciar el servicio, como se explico antes, y luego acceder por el navegador a http://localhost:1880, o bien la dirección IP del equipo remoto que tiene instalada la aplicación.



Si nos dirigimos al sector superior derecho de la web, presionamos ahí, vamos **Import**, **Clipboard** y luego copiamos y pegamos este código:

```
[
    {
        "id": "230f45d8.6f7a52",
        "type": "tab",
        "label": "Definitivo",
        "disabled": false,
        "info": ""
    },
    {
        "id": "c4539125.7c9968",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "Formato mensaje",
        "func": "msg.payload = {\n    name: msg.msg433.name,\n    mq9: msg.msg433.mq9,\n    hum: msg.msg433.hum,\n    ldr: msg.msg433.ldr,\n    temp: msg.msg433.temp,\n}\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 590,
        "y": 440,
        "wires": [
            [
                "94c0dd83.2c07d8"
            ]
        ]
    },
    {
        "id": "94c0dd83.2c07d8",
```

```
        "type": "influxdb out",
        "z": "230f45d8.6f7a52",
        "influxdb": "be79df3d.f330d",
        "name": "Influxdb",
        "measurement": "invernadero",
        "precision": "s",
        "retentionPolicy": "",
        "x": 800,
        "y": 440,
        "wires": []
    },
    {
        "id": "5de4b796.d1efa8",
        "type": "mqtt in",
        "z": "230f45d8.6f7a52",
        "name": "",
        "topic": "testpps/",
        "qos": "0",
        "broker": "16e1aa9a.dc41f5",
        "x": 130,
        "y": 100,
        "wires": [
            [
                "c6c846c1.05ed5"
            ]
        ]
    },
    {
        "id": "c6c846c1.05ed5",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "Parseo de mensaje MQTT",
```

```
59        "func": "var msg433 = {};\nmsg.payload =
msg.payload.replace(/(\\r\\n|\\n|\\r)/gm,\"\");\n\nvar parts433 =
msg.payload.split(\",\");\n\nmsg433.name = parts433[0];\nfor (var i=1;
i<parts433.length; i++) {\n    var keyvalue = parts433[i].split(\"=\");\n
   if (keyvalue.length===2) {\n        msg433[keyvalue[0]] =
keyvalue[1];\n    }\n}\n\nmsg.msg433 =
msg433;\n//msg.topic=\"rflink\";\nmsg.topic=\"testpps/\";\nreturn
msg;\n\n/*\n\n// So firstly a generic means of getting incoming items
into an object\n\nvar the433 = {};\nmsg.payload =
msg.payload.replace(/(\\r\\n|\\n|\\r)/gm,\"\");\nnode.warn(msg.payload);\
nvar parts433 = msg.payload.split(\";\");\n\nthe433.p1 =
parts433[0];\nthe433.p2 = parts433[1];\nthe433.name = parts433[2];\n\nvar
a = 3;\nwhile (a < parts433.length) {\n    var bits433 =
parts433[a].split(\"=\");\n    switch (bits433[0]) {\n        case
\"ID\": the433.id = bits433[1]; break;\n        case \"SWITCH\":
the433.switch = bits433[1]; break;\n        case \"CMD\": the433.cmd =
bits433[1]; break;\n        case \"SET_LEVEL\": the433.set_level =
parseInt(bits433[1], 10); break;\n        case \"TEMP\": the433.temp =
parseInt(bits433[1], 16) / 10; break;\n        case \"HUM\": the433.hum =
parseInt(bits433[1], 10); break;\n        case \"BARO\": the433.baro =
parseInt(bits433[1], 16); break;\n        case \"HSTATUS\":
the433.hstatus = parseInt(bits433[1], 10); break;\n        case
\"BFORECAST\": the433.bforecast = parseInt(bits433[1], 10); break;\n
   case \"UV\": the433.uv = parseInt(bits433[1], 16); break;\n
case \"LUX\": the433.lux = parseInt(bits433[1], 16); break;\n        case
\"BAT\": the433.bat = bits433[1]; break;\n        case \"RAIN\":
the433.rain = parseInt(bits433[1], 16) / 10; break;\n        case
\"RAIN\": the433.rainrate = parseInt(bits433[1], 16) / 10; break;\n
case \"WINSP\": the433.winsp = parseInt(bits433[1], 16) / 10; break;\n
     case \"AWINSP\": the433.awinsp = parseInt(bits433[1], 16) / 10;
break;\n        case \"WINGS\": the433.wings = parseInt(bits433[1], 16);
break;\n        case \"WINDIR\": the433.windir = parseInt(bits433[1],
10); break;\n        case \"WINCHL\": the433.winchl =
parseInt(bits433[1], 16); break;\n        case \"WINTMP\": the433.wintmp
= parseInt(bits433[1], 16); break;\n        case \"CHIME\": the433.chime
= parseInt(bits433[1], 10); break;\n        case \"SMOKEALERT\":
the433.smokealert = bits433[1]; break;\n        case \"PIR\": the433.pir
= bits433[1]; break;\n        case \"CO2\": the433.co2 =
parseInt(bits433[1], 10); break;\n        case \"SOUND\": the433.sound =
parseInt(bits433[1], 10); break;\n        case \"KWATT\": the433.kwatt =
parseInt(bits433[1], 16); break;\n        case \"WATT\": the433.watt =
parseInt(bits433[1], 16); break;\n        case \"CURRENT\":
the433.current = parseInt(bits433[1], 10); break;\n        case
\"CURRENT2\": the433.current2 = parseInt(bits433[1], 10); break;\n
case \"CURRENT3\": the433.current3 = parseInt(bits433[1], 10); break;\n
     case \"DIST\": the433.dist = parseInt(bits433[1], 10); break;\n
   case \"METER\": the433.meter = parseInt(bits433[1], 10); break;\n
   case \"VOLT\": the433.volt = parseInt(bits433[1], 10); break;\n
case \"RGBW\": the433.rgbc = parseInt(bits433[1].substring(0, 2), 16);\n
           the433.rgbw = parseInt(bits433[1].substring(2, 4), 16);
break;\n    }\n    a++;\n}\n\n// SO - the above is general... here is my
specific setup for temporarily displaying\n// the Acurite info\nif
((the433.p1 == \"20\") && (the433.name == \"Acurite\") && (the433.id ==
\"c826\")) {\n    if (typeof the433.temp !== 'undefined') temp =
the433.temp;\n    if (typeof the433.hum !== 'undefined') hum =
the433.hum;\n    if (typeof the433.bat !== 'undefined') bat =
the433.bat;\n    if (typeof the433.rain !== 'undefined') rain =
the433.rain;\n    if (typeof the433.winsp !== 'undefined') winsp =
```

```
the433.winsp;\n    if (typeof the433.windir !== 'undefined') windir =
the433.windir;\n\n    node.warn(\"Temperature: \" + temp + \"c\");\n
node.warn(\"Humidity: \" + hum + \"%\");\n    node.warn(\"Battery: \" +
bat);\n    node.warn(\"Rain: \" + rain + \"mm\");\n    node.warn(\"Wind
Speed: \" + winsp + \"km/h\");\n    node.warn(\"Wind Dir: \" + (windir *
22.5) + \" degrees\");\n}\n\n*/",
        "outputs": 1,
        "noerr": 0,
        "x": 340,
        "y": 180,
        "wires": [
            [
                "2b375d5f.5623ca"
            ]
        ]
    },
    {
        "id": "2b375d5f.5623ca",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "MQ-9 conversion",
        "func": "if (msg.msg433.mq9 !== undefined) {\n    msg.msg433.mq9
= parseFloat(msg.msg433.mq9 , 10);\n}\nelse msg.msg433.mq9
=-999.0;\nnode.status({fill:\"blue\",shape:\"ring\",text: msg.msg433.mq9
});\nreturn msg;\n",
        "outputs": 1,
        "noerr": 0,
        "x": 310,
        "y": 260,
        "wires": [
            [
                "8f034c1a.1b7428"
            ]
        ]
    },
    {
        "id": "8f034c1a.1b7428",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "Humedad conversion",
        "func": "if (msg.msg433.hum !==undefined) {\n    msg.msg433.hum =
parseFloat(msg.msg433.hum , 10);\n}\nelse msg.msg433.hum
=-999.0;\nnode.status({fill:\"blue\",shape:\"ring\",text: msg.msg433.hum
});\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 320,
        "y": 320,
        "wires": [
            [
                "7bb503f4.b5778c"
            ]
        ]
    },
    {
        "id": "cf6930a7.38dd9",
        "type": "comment",
        "z": "230f45d8.6f7a52",
```

```
        "name": "Suscripción al broker local",
        "info": "",
        "x": 190,
        "y": 40,
        "wires": []
    },
    {
        "id": "bc978e8d.5cd0b8",
        "type": "comment",
        "z": "230f45d8.6f7a52",
        "name": "Configuracion del broker local",
        "info": "",
        "x": 680,
        "y": 260,
        "wires": []
    },
    {
        "id": "cc0daab2.cf179",
        "type": "ui_template",
        "z": "230f45d8.6f7a52",
        "group": "124a1fa3.a5e4c8",
        "name": "frame mq9",
        "order": 2,
        "width": "0",
        "height": "0",
        "format": "<!--<iframe src=\"http://192.168.20.129:3000/d-
solo/TQrnackRk/prueba-valor1?
orgId=1&from=1551883611046&to=$msg.payload&refresh=10s&panelId=2\"
width=\"450\" height=\"200\" frameborder=\"0\"></iframe>-->\n<!--<iframe
src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-
edukit?orgId=1&panelId=18&from=1552863382441&to=1552949782445\"
width=\"543\" height=\"330\" frameborder=\"0\"></iframe> --->\n<iframe
src=\"http://192.168.20.131:3000/d-solo/CelyNFmgz/edicion_escenario1-
edukit_2?orgId=1&refresh=10s&from=1553010978259&to=
($msg.payload)&panelId=18\" width=\"543\" height=\"330\"
frameborder=\"0\"></iframe>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "templateScope": "local",
        "x": 1270,
        "y": 260,
        "wires": [
            []
        ]
    },
    {
        "id": "7bb503f4.b5778c",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "LDR conversion",
        "func": "if (msg.msg433.ldr !==undefined) {\n    msg.msg433.ldr =
parseInt(msg.msg433.ldr , 10);\n}\nelse msg.msg433.ldr
=-999.0;\nnode.status({fill:\"blue\",shape:\"ring\",text: msg.msg433.ldr
});\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 300,
        "y": 380,
```

```
        "wires": [
            [
                "55bdfe1c.5ced5"
            ]
        ]
    },
    {
        "id": "55bdfe1c.5ced5",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "Temperatura conversion",
        "func": "if (msg.msg433.temp !==undefined) {\n    msg.msg433.temp = parseFloat(msg.msg433.temp , 10);\n}\nelse msg.msg433.temp =-999.0;\nnode.status({fill:\"blue\",shape:\"ring\",text: msg.msg433.temp });\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 330,
        "y": 440,
        "wires": [
            [
                "c4539125.7c9968"
            ]
        ]
    },
    {
        "id": "8cf93b28.f10ff8",
        "type": "ui_template",
        "z": "230f45d8.6f7a52",
        "group": "b8c3e6e8.f28eb",
        "name": "Frame Hora",
        "order": 5,
        "width": 0,
        "height": 0,
        "format": "<iframe src=\"http://192.168.20.131:3000/d-solo/CelyNFmgz/escenario1-edukit?orgId=1&from=1552863382441&to=1552949782445&panelId=4\" width=\"1100\" height=\"150\" frameborder=\"0\"></iframe>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "templateScope": "local",
        "x": 1270,
        "y": 140,
        "wires": [
            []
        ]
    },
    {
        "id": "eabf72f.462a41",
        "type": "ui_template",
        "z": "230f45d8.6f7a52",
        "group": "d579fb4d.09324",
        "name": "Frame Temperatura",
        "order": 1,
        "width": "0",
        "height": "0",
```

```
        "format": "<!--<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit?orgId=1&panelId=14&from=1552863382441&to=$msg.payload\" width=\"543\" height=\"325\" frameborder=\"0\"></iframe>-->\n<!--<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&refresh=10s&from=1553010690649&to=$msg.payload&panelId=14\" width=\"543\" height=\"325\" frameborder=\"0\"></iframe>-->\n<iframe src=\"http://192.168.20.131:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&refresh=10s&from=1553089902485&to=$msg.payload&panelId=14\" width=\"543\" height=\"325\" frameborder=\"0\"></iframe>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "templateScope": "local",
        "x": 1290,
        "y": 180,
        "wires": [
            []
        ]
    },
    {
        "id": "b2f0f93.9458d88",
        "type": "ui_template",
        "z": "230f45d8.6f7a52",
        "group": "d579fb4d.09324",
        "name": "Frame Humedad",
        "order": 4,
        "width": 0,
        "height": 0,
        "format": "<!---<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit?orgId=1&refresh=10s&panelId=10&from=1552863382441&to=$msg.payload\" width=\"543\" height=\"325\" frameborder=\"0\"></iframe>-->\n<!--<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&refresh=10s&from=1553010852123&to=$msg.payload&panelId=10\" width=\"543\" height=\"325\" frameborder=\"0\"></iframe>-->\n<iframe src=\"http://192.168.20.131:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&refresh=10s&from=1553089966488&to=$msg.payload&panelId=10\" width=\"543\" height=\"325\" frameborder=\"0\"></iframe>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "templateScope": "local",
        "x": 1290,
        "y": 220,
        "wires": [
            []
        ]
    },
    {
        "id": "8f78ecbe.d9ef1",
        "type": "ui_template",
        "z": "230f45d8.6f7a52",
        "group": "124a1fa3.a5e4c8",
        "name": "Frame LDR",
        "order": 1,
```

```
        "width": "0",
        "height": "0",
        "format": "<!--<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit?orgId=1&refresh=10s&panelId=8&from=1552863382441&to=1552949782445\" width=\"543\" height=\"325\" frameborder=\"0\"></iframe> ---->\n<iframe src=\"http://192.168.20.131:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&refresh=10s&from=1553011900229&to=($msg.payload)&panelId=8\" width=\"543\" height=\"325\" frameborder=\"0\"></iframe>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "templateScope": "local",
        "x": 1270,
        "y": 300,
        "wires": [
            []
        ]
    },
    {
        "id": "ac1e53eb.bd3398",
        "type": "comment",
        "z": "230f45d8.6f7a52",
        "name": "Vista rapida",
        "info": "",
        "x": 1270,
        "y": 100,
        "wires": []
    },
    {
        "id": "728df83a.c5c39",
        "type": "comment",
        "z": "230f45d8.6f7a52",
        "name": "Exportar datos historicos",
        "info": "",
        "x": 2430,
        "y": 100,
        "wires": []
    },
    {
        "id": "c1909fc1.25fe1",
        "type": "ui_template",
        "z": "230f45d8.6f7a52",
        "group": "5d22a336.67eb5c",
        "name": "Historico mq9",
        "order": 9,
        "width": "20",
        "height": "6",
```

```
        "format": "<!--<iframe src=\"http://192.168.20.129:3000/d-solo/TQrnackRk/prueba-valor1?orgId=1&from=1551883611046&to=$msg.payload&refresh=10s&panelId=2\" width=\"450\" height=\"200\" frameborder=\"0\"></iframe>-->\n<!--<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit?orgId=1&panelId=15&from=1552863382441&to=1552949782445\" width=\"1100\" height=\"300\" frameborder=\"0\"></iframe>-->\n<!--<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&refresh=10s&from=1553012112762&to=($msg.payload)&panelId=15\" width=\"1100\" height=\"300\" frameborder=\"0\"></iframe>-->\n<div ng-bind-html=\"msg.payload\"></div>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "templateScope": "local",
        "x": 2600,
        "y": 1020,
        "wires": [
            []
        ]
    },
    {
        "id": "c6890547.33cfe",
        "type": "ui_template",
        "z": "230f45d8.6f7a52",
        "group": "a7f71eb3.656578",
        "name": "Historico Temperatura",
        "order": 3,
        "width": "20",
        "height": "6",
        "format": "<!--<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/escenario1-edukit?orgId=1&from=1552863382441&to=1552949782445&panelId=2\" width=\"1100\" height=\"300\" frameborder=\"0\"></iframe>-->\n<!--<iframe src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&refresh=10s&from=1553012005696&to=($msg.payload)&panelId=2\" width=\"1100\" height=\"300\" frameborder=\"0\"></iframe>-->\n<div ng-bind-html=\"msg.payload\"></div>",
        "storeOutMessages": true,
        "fwdInMessages": true,
        "templateScope": "local",
        "x": 2620,
        "y": 780,
        "wires": [
            []
        ]
    },
    {
        "id": "e8985261.d749e",
        "type": "ui_template",
        "z": "230f45d8.6f7a52",
        "group": "8f212678.706158",
        "name": "Historico Humedad",
        "order": 5,
        "width": "20",
        "height": "6",
```

```
316         "format": "<!--<iframe src=\"http://192.168.20.129:3000/d-
solo/CelyNFmgz/edicion_escenario1-edukit?
orgId=1&panelId=16&from=1552863382441&to=1552949782445\" width=\"1100\"
height=\"300\" frameborder=\"0\"></iframe>-->\n<!--<iframe
src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-
edukit_2?orgId=1&refresh=10s&from=1553012070609&to=
($msg.payload)&panelId=16\" width=\"1100\" height=\"300\"
frameborder=\"0\"></iframe>-->\n<div ng-bind-html=\"msg.payload\">
</div>",
317         "storeOutMessages": true,
318         "fwdInMessages": true,
319         "templateScope": "local",
320         "x": 2610,
321         "y": 860,
322         "wires": [
323             []
324         ]
325     },
326     {
327         "id": "4fb3d53c.4351b4",
328         "type": "ui_template",
329         "z": "230f45d8.6f7a52",
330         "group": "afecb5f7.f3fbc8",
331         "name": "Historico LDR",
332         "order": 7,
333         "width": "20",
334         "height": "6",
335         "format": "<!--<iframe src=\"http://192.168.20.129:3000/d-
solo/CelyNFmgz/edicion_escenario1-edukit?
orgId=1&panelId=17&from=1552863382441&to=1552949782445\" width=\"1100\"
height=\"300\" frameborder=\"0\"></iframe>-->\n<!--<iframe
src=\"http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-
edukit_2?orgId=1&refresh=10s&from=1553012156317&to=
($msg.payload)&panelId=17\" width=\"1100\" height=\"300\"
frameborder=\"0\"></iframe>-->\n<div ng-bind-html=\"msg.payload\">
</div>",
336         "storeOutMessages": true,
337         "fwdInMessages": true,
338         "templateScope": "local",
339         "x": 2600,
340         "y": 940,
341         "wires": [
342             []
343         ]
344     },
345     {
346         "id": "684d701c.b69238",
347         "type": "mosca in",
348         "z": "230f45d8.6f7a52",
349         "mqtt_port": 1883,
350         "mqtt_ws_port": 8080,
351         "name": "",
352         "username": "",
353         "password": "",
354         "dburl": "",
355         "x": 650,
356         "y": 300,
357         "wires": [
```

```
            []
        ]
    },
    {
        "id": "e92020cd.8880a8",
        "type": "file",
        "z": "230f45d8.6f7a52",
        "name": "",
        "filename": "/home/pi/temp.csv",
        "appendNewline": true,
        "createDir": false,
        "overwriteFile": "true",
        "x": 2730,
        "y": 180,
        "wires": [
            [
                "1042c388.0e4e5c"
            ]
        ]
    },
    {
        "id": "46f9e84.7996b18",
        "type": "csv",
        "z": "230f45d8.6f7a52",
        "name": "",
        "sep": ",",
        "hdrin": false,
        "hdrout": true,
        "multi": "one",
        "ret": "\\r\\n",
        "temp": "time,name,temp",
        "skip": "0",
        "x": 2550,
        "y": 180,
        "wires": [
            [
                "e92020cd.8880a8"
            ]
        ]
    },
    {
        "id": "77ff6db4.33b624",
        "type": "ui_ui_control",
        "z": "230f45d8.6f7a52",
        "name": "Show/Hide",
        "x": 1750,
        "y": 180,
        "wires": [
            []
        ]
    },
    {
        "id": "9a0d3aea.19a5c",
        "type": "ui_date_picker",
        "z": "230f45d8.6f7a52",
        "name": "Fecha_inicial_temp",
        "label": "Fecha inicial",
        "group": "8ed24b80.43df2",
```

```
        "order": 1,
        "width": "9",
        "height": "1",
        "passthru": false,
        "topic": "Fecha_inicial_temp",
        "x": 1950,
        "y": 160,
        "wires": [
            [
                "d294b3a4.3c9a8"
            ]
        ]
    },
    {
        "id": "8c416833.629d78",
        "type": "ui_button",
        "z": "230f45d8.6f7a52",
        "name": "Exportar temp",
        "group": "a7f71eb3.656578",
        "order": 0,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Exportar",
        "tooltip": "Exportar datos historicos de temperatura en formato csv",
        "color": "",
        "bgcolor": "",
        "icon": "fa-file-o ",
        "payload": "{\"group\":{\"show\":[\"Registros_export_temp\",\"Registros_Fecha_2_temp\",\"Registros_boton_cancel_export\"]}}",
        "payloadType": "json",
        "topic": "",
        "x": 1560,
        "y": 160,
        "wires": [
            [
                "77ff6db4.33b624"
            ]
        ]
    },
    {
        "id": "da33d4e0.efa66",
        "type": "ui_date_picker",
        "z": "230f45d8.6f7a52",
        "name": "Fecha_final_temp",
        "label": "Fecha final",
        "group": "8ed24b80.43df2",
        "order": 2,
        "width": "9",
        "height": "1",
        "passthru": false,
        "topic": "Fecha_final_temp",
        "x": 1950,
        "y": 200,
        "wires": [
            [
```

```
                "d294b3a4.3c9a8"
            ]
        ]
    },
    {
        "id": "cc5101d9.917468",
        "type": "ui_button",
        "z": "2305f45d8.6f7a52",
        "name": "Cancel temp",
        "group": "8ed24b80.43df2",
        "order": 7,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Cancelar",
        "tooltip": "",
        "color": "",
        "bgcolor": "",
        "icon": "fa-times ",
        "payload": "{\"group\":{\"hide\":
[\"Registros_export_temp\",\"Registros_Fecha_2_temp\",\"Registros_boton_cancel_export\"]}}",
        "payloadType": "json",
        "topic": "",
        "x": 1570,
        "y": 200,
        "wires": [
            [
                "77ff6db4.33b624"
            ]
        ]
    },
    {
        "id": "6a10d4c3.802384",
        "type": "comment",
        "z": "2305f45d8.6f7a52",
        "name": "Seccion Web",
        "info": "",
        "x": 2070,
        "y": 20,
        "wires": []
    },
    {
        "id": "38869e96.9a44da",
        "type": "ui_ui_control",
        "z": "2305f45d8.6f7a52",
        "name": "Show/Hide",
        "x": 1750,
        "y": 300,
        "wires": [
            []
        ]
    },
    {
        "id": "2da9ffd4.15dff",
        "type": "ui_date_picker",
        "z": "2305f45d8.6f7a52",
        "name": "Fecha_inicial_hum",
```

```
        "label": "Fecha inicial",
        "group": "852840d5.64a668",
        "order": 1,
        "width": "9",
        "height": "1",
        "passthru": false,
        "topic": "Fecha_inicial_hum",
        "x": 1950,
        "y": 280,
        "wires": [
            [
                "2bcf0ab.41ddd76"
            ]
        ]
    },
    {
        "id": "1b1a4e8e.f3e4f9",
        "type": "ui_date_picker",
        "z": "230f45d8.6f7a52",
        "name": "Fecha_final_hum",
        "label": "Fecha final",
        "group": "852840d5.64a668",
        "order": 2,
        "width": "9",
        "height": "1",
        "passthru": false,
        "topic": "Fecha_final_hum",
        "x": 1950,
        "y": 320,
        "wires": [
            [
                "2bcf0ab.41ddd76"
            ]
        ]
    },
    {
        "id": "402d943d.28bb84",
        "type": "ui_ui_control",
        "z": "230f45d8.6f7a52",
        "name": "Show/Hide",
        "x": 1750,
        "y": 440,
        "wires": [
            []
        ]
    },
    {
        "id": "a870d237.03fa78",
        "type": "ui_date_picker",
        "z": "230f45d8.6f7a52",
        "name": "Fecha_inicial_ldr",
        "label": "Fecha inicial",
        "group": "224c07c1.06a2d",
        "order": 3,
        "width": "9",
        "height": "1",
        "passthru": false,
        "topic": "Fecha_inicial_ldr",
```

```
        "x": 1950,
        "y": 420,
        "wires": [
            [
                "f29a3e6c.d4a7d8"
            ]
        ]
    },
    {
        "id": "c3948d16.f046b8",
        "type": "ui_date_picker",
        "z": "230f45d8.6f7a52",
        "name": "Fecha_final_ldr",
        "label": "Fecha final",
        "group": "224c07c1.06a2d",
        "order": 4,
        "width": "9",
        "height": "1",
        "passthru": false,
        "topic": "Fecha_final_ldr",
        "x": 1940,
        "y": 460,
        "wires": [
            [
                "f29a3e6c.d4a7d8"
            ]
        ]
    },
    {
        "id": "b20893e3.5eb888",
        "type": "ui_ui_control",
        "z": "230f45d8.6f7a52",
        "name": "Show/Hide",
        "x": 1750,
        "y": 580,
        "wires": [
            []
        ]
    },
    {
        "id": "6481979c.4413d",
        "type": "ui_date_picker",
        "z": "230f45d8.6f7a52",
        "name": "Fecha_inicial_mq9",
        "label": "Fecha inicial",
        "group": "29206391.d60f3c",
        "order": 5,
        "width": "9",
        "height": "1",
        "passthru": false,
        "topic": "Fecha_inicial_mq9",
        "x": 1950,
        "y": 560,
        "wires": [
            [
                "d70a417d.669b08"
            ]
        ]
```

```
    },
    {
        "id": "c8d347d8.2fc6e",
        "type": "ui_date_picker",
        "z": "230f45d8.6f7a52",
        "name": "Fecha_final_mq9",
        "label": "Fecha final",
        "group": "29206391.d60f3c",
        "order": 6,
        "width": "9",
        "height": "1",
        "passthru": false,
        "topic": "Fecha_final_mq9",
        "x": 1950,
        "y": 600,
        "wires": [
            [
                "d70a417d.669b08"
            ]
        ]
    },
    {
        "id": "c151b034.ba1148",
        "type": "ui_button",
        "z": "230f45d8.6f7a52",
        "name": "Exportar hum",
        "group": "8f212678.706158",
        "order": 0,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Exportar",
        "tooltip": "Exportar datos historicos de humedad en formato csv",
        "color": "",
        "bgcolor": "",
        "icon": "fa-file-o ",
        "payload": "{\"group\":{\"show\":[\"Registros_export_hum\"]}}",
        "payloadType": "json",
        "topic": "",
        "x": 1560,
        "y": 280,
        "wires": [
            [
                "38869e96.9a44da"
            ]
        ]
    },
    {
        "id": "28d742da.b1139e",
        "type": "ui_button",
        "z": "230f45d8.6f7a52",
        "name": "Cancel hum",
        "group": "852840d5.64a668",
        "order": 8,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Cancelar",
```

```
        "tooltip": "",
        "color": "",
        "bgcolor": "",
        "icon": "fa-times ",
        "payload": "{\"group\":{\"hide\":[\"Registros_export_hum\"]}}",
        "payloadType": "json",
        "topic": "",
        "x": 1570,
        "y": 320,
        "wires": [
            [
                "38869e96.9a44da"
            ]
        ]
    },
    {
        "id": "834b3480.145228",
        "type": "ui_button",
        "z": "230f45d8.6f7a52",
        "name": "Exportar ldr",
        "group": "afecb5f7.f3fbc8",
        "order": 0,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Exportar",
        "tooltip": "Exportar datos historicos de niveles de luz en
formato csv",
        "color": "",
        "bgcolor": "",
        "icon": "fa-file-o ",
        "payload": "{\"group\":{\"show\":[\"Registros_export_ldr\"]}}",
        "payloadType": "json",
        "topic": "",
        "x": 1570,
        "y": 420,
        "wires": [
            [
                "402d943d.28bb84"
            ]
        ]
    },
    {
        "id": "e981f4c9.eac4e8",
        "type": "ui_button",
        "z": "230f45d8.6f7a52",
        "name": "Cancel ldr",
        "group": "224c07c1.06a2d",
        "order": 9,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Cancelar",
        "tooltip": "",
        "color": "",
        "bgcolor": "",
        "icon": "fa-times ",
        "payload": "{\"group\":{\"hide\":[\"Registros_export_ldr\"]}}",
```

```
        "payloadType": "json",
        "topic": "",
        "x": 1580,
        "y": 460,
        "wires": [
            [
                "402d943d.28bb84"
            ]
        ]
    },
    {
        "id": "23e2742c.6a351c",
        "type": "ui_button",
        "z": "230f45d8.6f7a52",
        "name": "Exportar mq9",
        "group": "5d22a336.67eb5c",
        "order": 0,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Exportar",
        "tooltip": "Exportar datos historicos de concentración de CO en formato csv",
        "color": "",
        "bgcolor": "",
        "icon": "fa-file-o ",
        "payload": "{\"group\":{\"show\":[\"Registros_export_mq9\"]}}",
        "payloadType": "json",
        "topic": "",
        "x": 1560,
        "y": 560,
        "wires": [
            [
                "b20893e3.5eb888"
            ]
        ]
    },
    {
        "id": "dfd6e162.99691",
        "type": "ui_button",
        "z": "230f45d8.6f7a52",
        "name": "Cancel mq9",
        "group": "29206391.d60f3c",
        "order": 10,
        "width": "2",
        "height": "1",
        "passthru": false,
        "label": "Cancelar",
        "tooltip": "",
        "color": "",
        "bgcolor": "",
        "icon": "fa-times ",
        "payload": "{\"group\":{\"hide\":[\"Registros_export_mq9\"]}}",
        "payloadType": "json",
        "topic": "",
        "x": 1570,
        "y": 600,
        "wires": [
```

```
            [
                "b20893e3.5eb888"
            ]
        ]
    },
    {
        "id": "9c1128fe.cd83a",
        "type": "influxdb in",
        "z": "230f45d8.6f7a52",
        "influxdb": "be79df3d.f330d",
        "name": "",
        "query": "",
        "rawOutput": false,
        "precision": "s",
        "retentionPolicy": "",
        "x": 2430,
        "y": 220,
        "wires": [
            [
                "46f9e84.7996b18"
            ]
        ]
    },
    {
        "id": "d294b3a4.3c9a8",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "Envio de consulta con fechas",
        "func": "context.Fecha_inicial_temp = context.Fecha_inicial_temp || 0;\ncontext.Fecha_final_temp = context.Fecha_final_temp || 0;\n\nif (msg.topic === 'Fecha_inicial_temp') {\n  context.Fecha_inicial_temp = msg.payload;\n} else if (msg.topic === 'Fecha_final_temp') {\n context.Fecha_final_temp = msg.payload;\n}\n\nif (context.Fecha_inicial_temp !== 0 && context.Fecha_final_temp !== 0){\n //msg.query='SELECT * FROM \"invernadero\" WHERE \"time\" > ' + context.Fecha_inicial_temp + '000000000' + ' AND (time < ' + context.Fecha_final_temp + '000000000' + ')'\n    msg.query='SELECT \"name\",\"temp\" FROM invernadero WHERE time > ' + context.Fecha_inicial_temp + '000000' + ' AND (time < ' + context.Fecha_final_temp + '000000' + ')'\n    context.Fecha_inicial_temp = 0;\n    context.Fecha_final_temp = 0;\n    return msg;\n}\nelse {\n return null\n}\n\n\n\n//var msg = {};\n//var a = context.get(fecha1);\n//context.set(a)=fecha1;\n//var b = context.get(fecha2);\n//msg.fecha1=\"1553100903000000000\";\n//msg.fecha2=\"1553101023000000000\";\n//msg.fecha1=a;\n//msg.fecha2=b;\n//.payload = msg;\n//return msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 2220,
        "y": 180,
        "wires": [
            [
                "9c1128fe.cd83a"
            ]
        ]
    },
    {
        "id": "2bcf0ab.41ddd76",
```

```
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "Envio de consulta con fechas",
        "func": "context.Fecha_inicial_hum = context.Fecha_inicial_hum || 0;\ncontext.Fecha_final_hum = context.Fecha_final_hum || 0;\n\nif (msg.topic === 'Fecha_inicial_hum') {\n  context.Fecha_inicial_hum = msg.payload;\n} else if (msg.topic === 'Fecha_final_hum') {\n context.Fecha_final_hum = msg.payload;\n}\n\nif (context.Fecha_inicial_hum !== 0 && context.Fecha_final_hum !== 0){\n msg.query='SELECT \"name\",\"hum\" FROM invernadero WHERE time > ' + context.Fecha_inicial_hum + '000000' + ' AND (time < ' + context.Fecha_final_hum + '000000' + ')'\n    context.Fecha_inicial_hum = 0;\n    context.Fecha_final_hum = 0;\n    return msg;\n}\nelse {\n return null\n}",
        "outputs": 1,
        "noerr": 0,
        "x": 2220,
        "y": 300,
        "wires": [
            [
                "e5364de.5baa13"
            ]
        ]
    },
    {
        "id": "f29a3e6c.d4a7d8",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "Envio de consulta con fechas",
        "func": "context.Fecha_inicial_ldr = context.Fecha_inicial_ldr || 0;\ncontext.Fecha_final_ldr = context.Fecha_final_ldr || 0;\n\nif (msg.topic === 'Fecha_inicial_ldr') {\n  context.Fecha_inicial_ldr = msg.payload;\n} else if (msg.topic === 'Fecha_final_ldr') {\n context.Fecha_final_ldr = msg.payload;\n}\n\nif (context.Fecha_inicial_ldr !== 0 && context.Fecha_final_ldr !== 0){\n msg.query='SELECT \"name\",\"ldr\" FROM invernadero WHERE time > ' + context.Fecha_inicial_ldr + '000000' + ' AND (time < ' + context.Fecha_final_ldr + '000000' + ')'\n    context.Fecha_inicial_ldr = 0;\n    context.Fecha_final_ldr = 0;\n    return msg;\n}\nelse {\n return null\n}\n\n\n\n//var msg = {};\n//var a = context.get(fecha1);\n//context.set(a)=fecha1;\n//var b = context.get(fecha2);\n//msg.fecha1=\"1553100903000000000\";\n//msg.fecha2=\"1553101023000000000\";\n//msg.fecha1=a;\n//msg.fecha2=b;\n//.payload = msg;\n//return msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 2200,
        "y": 440,
        "wires": [
            [
                "a22e3988.585bf"
            ]
        ]
    },
    {
        "id": "d70a417d.669b08",
        "type": "function",
        "z": "230f45d8.6f7a52",
```

```
        "name": "Envio de consulta con fechas",
        "func": "context.Fecha_inicial_mq9 = context.Fecha_inicial_mq9 || 0;\ncontext.Fecha_final_mq9 = context.Fecha_final_mq9 || 0;\n\nif (msg.topic === 'Fecha_inicial_mq9') {\n  context.Fecha_inicial_mq9 = msg.payload;\n} else if (msg.topic === 'Fecha_final_mq9') {\n context.Fecha_final_mq9 = msg.payload;\n}\n\nif (context.Fecha_inicial_mq9 !== 0 && context.Fecha_final_mq9 !== 0){\n msg.query='SELECT \"name\",\"mq9\" FROM invernadero WHERE time > ' + context.Fecha_inicial_mq9 + '000000' + ' AND (time < ' + context.Fecha_final_mq9 + '000000' + ')'\n    context.Fecha_inicial_mq9 = 0;\n    context.Fecha_final_mq9 = 0;\n    return msg;\n}\nelse {\n return null\n}\n\n\n\n//var msg = {};\n//var a = context.get(fecha1);\n//context.set(a)=fecha1;\n//var b = context.get(fecha2);\n//msg.fecha1=\"1553100903000000000\";\n//msg.fecha2=\"1553101023000000000\";\n//msg.fecha1=a;\n//msg.fecha2=b;\n//.payload = msg;\n//return msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 2200,
        "y": 580,
        "wires": [
            [
                "e249a631.816a4"
            ]
        ]
    },
    {
        "id": "e249a631.816a4",
        "type": "influxdb in",
        "z": "230f45d8.6f7a52",
        "influxdb": "be79df3d.f330d",
        "name": "",
        "query": "",
        "rawOutput": false,
        "precision": "s",
        "retentionPolicy": "",
        "x": 2430,
        "y": 620,
        "wires": [
            [
                "749caeef.5c8ac"
            ]
        ]
    },
    {
        "id": "a22e3988.585bf",
        "type": "influxdb in",
        "z": "230f45d8.6f7a52",
        "influxdb": "be79df3d.f330d",
        "name": "",
        "query": "",
        "rawOutput": false,
        "precision": "s",
        "retentionPolicy": "",
        "x": 2430,
        "y": 480,
        "wires": [
            [
```

```
                "fe3c9d68.93ec28"
            ]
        ]
    },
    {
        "id": "e5364de.5baa13",
        "type": "influxdb in",
        "z": "230f45d8.6f7a52",
        "influxdb": "be79df3d.f330d",
        "name": "",
        "query": "",
        "rawOutput": false,
        "precision": "s",
        "retentionPolicy": "",
        "x": 2430,
        "y": 340,
        "wires": [
            [
                "76067011.d0d0b8"
            ]
        ]
    },
    {
        "id": "76067011.d0d0b8",
        "type": "csv",
        "z": "230f45d8.6f7a52",
        "name": "",
        "sep": ",",
        "hdrin": false,
        "hdrout": true,
        "multi": "one",
        "ret": "\\n",
        "temp": "time,name,hum",
        "skip": "0",
        "x": 2550,
        "y": 300,
        "wires": [
            [
                "fb694316.9ace48"
            ]
        ]
    },
    {
        "id": "fe3c9d68.93ec28",
        "type": "csv",
        "z": "230f45d8.6f7a52",
        "name": "",
        "sep": ",",
        "hdrin": false,
        "hdrout": true,
        "multi": "one",
        "ret": "\\n",
        "temp": "time,name,ldr",
        "skip": "0",
        "x": 2550,
        "y": 440,
        "wires": [
            [
```

```
                "2cbb7a16.a15426"
            ]
        ]
    },
    {
        "id": "749caeef.5c8ac",
        "type": "csv",
        "z": "230f45d8.6f7a52",
        "name": "",
        "sep": ",",
        "hdrin": false,
        "hdrout": true,
        "multi": "one",
        "ret": "\\n",
        "temp": "time,name,mq9",
        "skip": "0",
        "x": 2550,
        "y": 580,
        "wires": [
            [
                "bf55b6bb.77b918"
            ]
        ]
    },
    {
        "id": "fb694316.9ace48",
        "type": "file",
        "z": "230f45d8.6f7a52",
        "name": "",
        "filename": "/home/pi/hum.csv",
        "appendNewline": true,
        "createDir": false,
        "overwriteFile": "true",
        "x": 2730,
        "y": 300,
        "wires": [
            [
                "75e5519e.732cd"
            ]
        ]
    },
    {
        "id": "2cbb7a16.a15426",
        "type": "file",
        "z": "230f45d8.6f7a52",
        "name": "",
        "filename": "/home/pi/ldr.csv",
        "appendNewline": true,
        "createDir": false,
        "overwriteFile": "true",
        "x": 2720,
        "y": 440,
        "wires": [
            [
                "d37e950a.c11f3"
            ]
        ]
    },
```

```
    {
        "id": "bf55b6bb.77b918",
        "type": "file",
        "z": "230f45d8.6f7a52",
        "name": "",
        "filename": "/home/pi/mq9.csv",
        "appendNewline": true,
        "createDir": false,
        "overwriteFile": "true",
        "x": 2730,
        "y": 580,
        "wires": [
            [
                "33659eb8.4a5502"
            ]
        ]
    },
    {
        "id": "f5698941.9b4fa8",
        "type": "ui_toast",
        "z": "230f45d8.6f7a52",
        "position": "top right",
        "displayTime": "3",
        "highlight": "",
        "outputs": 0,
        "ok": "OK",
        "cancel": "",
        "topic": "Datos exportados con exito",
        "name": "Tarea completada",
        "x": 2930,
        "y": 220,
        "wires": []
    },
    {
        "id": "1042c388.0e4e5c",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "",
        "func": "msg.payload=\"/home/pi/temp.csv\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 2910,
        "y": 180,
        "wires": [
            [
                "f5698941.9b4fa8"
            ]
        ]
    },
    {
        "id": "75e5519e.732cd",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "",
        "func": "msg.payload=\"/home/pi/hum.csv\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 2910,
```

```
        "y": 300,
        "wires": [
            [
                "b6aabdb6.42b2c"
            ]
        ]
    },
    {
        "id": "b6aabdb6.42b2c",
        "type": "ui_toast",
        "z": "230f45d8.6f7a52",
        "position": "top right",
        "displayTime": "3",
        "highlight": "",
        "outputs": 0,
        "ok": "OK",
        "cancel": "",
        "topic": "Datos exportados con exito",
        "name": "Tarea completada",
        "x": 2930,
        "y": 340,
        "wires": []
    },
    {
        "id": "d37e950a.c11f3",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "",
        "func": "msg.payload=\"/home/pi/ldr.csv\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 2910,
        "y": 440,
        "wires": [
            [
                "845ef2b8.6d6da8"
            ]
        ]
    },
    {
        "id": "845ef2b8.6d6da8",
        "type": "ui_toast",
        "z": "230f45d8.6f7a52",
        "position": "top right",
        "displayTime": "3",
        "highlight": "",
        "outputs": 0,
        "ok": "OK",
        "cancel": "",
        "topic": "Datos exportados con exito",
        "name": "Tarea completada",
        "x": 2930,
        "y": 480,
        "wires": []
    },
    {
        "id": "33659eb8.4a5502",
        "type": "function",
```

```
        "z": "230f45d8.6f7a52",
        "name": "",
        "func": "msg.payload=\"/home/pi/mq9.csv\";\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 2910,
        "y": 580,
        "wires": [
            [
                "f9a08d12.bf3048"
            ]
        ]
    },
    {
        "id": "f9a08d12.bf3048",
        "type": "ui_toast",
        "z": "230f45d8.6f7a52",
        "position": "top right",
        "displayTime": "3",
        "highlight": "",
        "outputs": 0,
        "ok": "OK",
        "cancel": "",
        "topic": "Datos exportados con exito",
        "name": "Tarea completada",
        "x": 2930,
        "y": 620,
        "wires": []
    },
    {
        "id": "e2df35d7.5139a8",
        "type": "template",
        "z": "230f45d8.6f7a52",
        "name": "Historico Temperatura",
        "field": "template",
        "fieldType": "msg",
        "format": "handlebars",
        "syntax": "mustache",
        "template": "<iframe src=\"http://192.168.20.131:3000/d-
solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&
{{payload}}&panelId=2\" width=\"1100\" height=\"300\" frameborder=\"0\">
</iframe>",
        "output": "str",
        "x": 2400,
        "y": 780,
        "wires": [
            [
                "c6890547.33cfe"
            ]
        ]
    },
    {
        "id": "bb344085.ea39d",
        "type": "ui_button",
        "z": "230f45d8.6f7a52",
        "name": "confirmar",
        "group": "d04e1b2.f8adae8",
        "order": 4,
```

```
        "width": "3",
        "height": "1",
        "passthru": false,
        "label": "Confirmar",
        "tooltip": "Confirma los intervalos definidos. Presionar una vez que este seguro de los datos ingresados. ",
        "color": "",
        "bgcolor": "",
        "icon": "fa-check-circle ",
        "payload": "confirmar",
        "payloadType": "str",
        "topic": "confirmar",
        "x": 1560,
        "y": 980,
        "wires": [
            [
                "322bac1e.1b0f44",
                "d454004f.29974",
                "ffd57fe9.3f8d8"
            ]
        ]
    },
    {
        "id": "7f90fbfd.c86404",
        "type": "ui_form",
        "z": "230f45d8.6f7a52",
        "name": "f1",
        "label": "Momento inicial",
        "group": "dc137fb4.d8e79",
        "order": 1,
        "width": "6",
        "height": "1",
        "options": [
            {
                "label": "",
                "value": "f1",
                "type": "date",
                "required": true
            },
            {
                "label": "Hora del dia (entre 0 y 24))",
                "value": "h1",
                "type": "number",
                "required": true
            }
        ],
        "formValue": {
            "f1": "",
            "h1": ""
        },
        "payload": "",
        "submit": "Cargar",
        "cancel": "Cancelar",
        "topic": "f1",
        "x": 1570,
        "y": 800,
        "wires": [
            [
```

```
                "322bac1e.1b0f44",
                "d454004f.29974"
            ]
        ]
    },
    {
        "id": "8619e095.f071f",
        "type": "ui_form",
        "z": "230f45d8.6f7a52",
        "name": "f2",
        "label": "Momento final",
        "group": "41fbf2c8.27dd2c",
        "order": 1,
        "width": "6",
        "height": "1",
        "options": [
            {
                "label": "",
                "value": "f2",
                "type": "date",
                "required": true
            },
            {
                "label": "Hora del dia (entre 0 y 24))",
                "value": "h2",
                "type": "number",
                "required": true
            }
        ],
        "formValue": {
            "f2": "",
            "h2": ""
        },
        "payload": "",
        "submit": "Cargar",
        "cancel": "Cancelar",
        "topic": "f2",
        "x": 1570,
        "y": 860,
        "wires": [
            [
                "322bac1e.1b0f44",
                "d454004f.29974"
            ]
        ]
    },
    {
        "id": "ffd57fe9.3f8d8",
        "type": "ui_dropdown",
        "z": "230f45d8.6f7a52",
        "name": "",
        "label": "Refrescar cada ",
        "tooltip": "Seleccionar un intervalo de tiempo para refrescar los graficos y obtener información en tiempo real",
        "place": "Seleccione intervalo",
        "group": "d04e1b2.f8adae8",
        "order": 1,
        "width": "6",
```

```
        "height": "1",
        "passthru": false,
        "options": [
            {
                "label": "Off",
                "value": "Off",
                "type": "str"
            },
            {
                "label": "5s",
                "value": "5s",
                "type": "str"
            },
            {
                "label": "10s",
                "value": "10s",
                "type": "str"
            },
            {
                "label": "30s",
                "value": "30s",
                "type": "str"
            },
            {
                "label": "1m",
                "value": "1m",
                "type": "str"
            },
            {
                "label": "5m",
                "value": "5m",
                "type": "str"
            },
            {
                "label": "15m",
                "value": "15m",
                "type": "str"
            },
            {
                "label": "30m",
                "value": "30m",
                "type": "str"
            },
            {
                "label": "1h",
                "value": "1h",
                "type": "str"
            },
            {
                "label": "2h",
                "value": "2h",
                "type": "str"
            },
            {
                "label": "1d",
                "value": "1d",
                "type": "str"
            }
```

```
        ],
        "payload": "",
        "topic": "refresh",
        "x": 1540,
        "y": 920,
        "wires": [
            [
                "322bac1e.1b0f44",
                "d454004f.29974"
            ]
        ]
    },
    {
        "id": "322bac1e.1b0f44",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "refresh, from y to - Datos Historicos",
        "func": "context.f1 = context.f1 || 0;\ncontext.f2 = context.f2 || 0;\ncontext.h1 = context.h1 || -1;\ncontext.h2 = context.h2 || -1;\ncontext.ref = context.ref || 'Off';\ncontext.con = context.con || 'Nada';\n\nif (msg.topic === 'f1') {\n  context.f1 = msg.payload.f1;\n context.h1 = msg.payload.h1;\n} else if (msg.topic === 'f2') {\n context.f2 = msg.payload.f2;\n  context.h2 = msg.payload.h2;\n} else if (msg.topic === 'refresh') {\n  context.ref = msg.payload;\n} else if (msg.topic === 'confirmar') {\n  context.con = msg.payload;\n} else if (msg.topic === 'reset') {\n  context.reset = msg.payload;\n}\n\nif (context.f1 !== 0 && context.f2 !== 0 && context.ref !== 'Off' && context.con == 'confirmar'){\n    var fecha1 = new Date(context.f1); // Your timezone!\n    var epoch1 = fecha1.getTime();\n    \n    var fecha2 = new Date(context.f2); // Your timezone!\n    var epoch2 = fecha2.getTime();\n    \n    var hora1 = context.h1;\n    var hora2 = context.h2;\n    epoch1 = epoch1 + hora1*3600*1000; //3600 segundos en una hora, 1000 miliseg por cada seg\n    epoch2 = epoch2 + hora2*3600*1000;\n    \n    msg.payload='refresh=' + context.ref + '&from='+ epoch1.toString() + '&to='+ epoch2.toString();\n\n context.f1 = 0;\n    context.f2 = 0;\n    context.h1 = -1;\n context.h2 = -1;\n    context.ref = 'Off';\n    context.con = 'NADA';\n \n    return msg;\n}\nelse {\n    return null\n}",
        "outputs": 1,
        "noerr": 0,
        "x": 1980,
        "y": 840,
        "wires": [
            [
                "e2df35d7.5139a8",
                "f498edee.ab468",
                "39dea26a.fb1Od6",
                "4f824516.3c5ddc"
            ]
        ]
    },
    {
        "id": "f498edee.ab468",
        "type": "template",
        "z": "230f45d8.6f7a52",
        "name": "Historico Humedad",
        "field": "template",
        "fieldType": "msg",
```

```
        "format": "handlebars",
        "syntax": "mustache",
        "template": "<iframe src=\"http://192.168.20.131:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&{{payload}}&panelId=16\" width=\"1100\" height=\"300\" frameborder=\"0\"></iframe>",
        "output": "str",
        "x": 2390,
        "y": 860,
        "wires": [
            [
                "e8985261.d749e"
            ]
        ]
    },
    {
        "id": "39dea26a.fb10d6",
        "type": "template",
        "z": "230f45d8.6f7a52",
        "name": "Historico LDR",
        "field": "template",
        "fieldType": "msg",
        "format": "handlebars",
        "syntax": "mustache",
        "template": "<iframe src=\"http://192.168.20.131:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&{{payload}}&panelId=17\" width=\"1100\" height=\"300\" frameborder=\"0\"></iframe>",
        "output": "str",
        "x": 2380,
        "y": 940,
        "wires": [
            [
                "4fb3d53c.4351b4"
            ]
        ]
    },
    {
        "id": "4f824516.3c5ddc",
        "type": "template",
        "z": "230f45d8.6f7a52",
        "name": "Historico mq9",
        "field": "template",
        "fieldType": "msg",
        "format": "handlebars",
        "syntax": "mustache",
        "template": "<iframe src=\"http://192.168.20.131:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?orgId=1&{{payload}}&panelId=15\" width=\"1100\" height=\"300\" frameborder=\"0\"></iframe>",
        "output": "str",
        "x": 2380,
        "y": 1020,
        "wires": [
            [
                "c1909fc1.25fe1"
            ]
        ]
```

```
    },
    {
        "id": "b44ffe1f.41f14",
        "type": "comment",
        "z": "230f45d8.6f7a52",
        "name": "Datos historicos -  Graficos e intervalo de muestra",
        "info": "",
        "x": 2080,
        "y": 720,
        "wires": []
    },
    {
        "id": "4d4268a3.1ad878",
        "type": "ui_text",
        "z": "230f45d8.6f7a52",
        "group": "d04e1b2.f8adae8",
        "order": 3,
        "width": "6",
        "height": "2",
        "name": "",
        "label": "Tiempo seteado: ",
        "format": "{{msg.payload}}",
        "layout": "row-left",
        "x": 2130,
        "y": 1020,
        "wires": []
    },
    {
        "id": "d454004f.29974",
        "type": "function",
        "z": "230f45d8.6f7a52",
        "name": "Mostrar actual",
```

```
        "func": "context.f1 = context.f1 || 0;\ncontext.f2 = context.f2 || 0;\ncontext.h1 = context.h1 || -1;\ncontext.h2 = context.h2 || -1;\ncontext.ref = context.ref || 'Off';\ncontext.con = context.con || 'Nada';\n\nif (msg.topic === 'f1') {\n  context.f1 = msg.payload.f1;\n context.h1 = msg.payload.h1;\n} else if (msg.topic === 'f2') {\n context.f2 = msg.payload.f2;\n  context.h2 = msg.payload.h2;\n} else if (msg.topic === 'refresh') {\n  context.ref = msg.payload;\n} else if (msg.topic === 'confirmar') {\n  context.con = msg.payload;\n}\n\nif (context.f1 !== 0 && context.f2 !== 0 && context.ref !== 'Off' && context.con == 'confirmar'){\n    var fecha1 = new Date(context.f1); // Your timezone!\n    var epoch1 = fecha1.getTime();\n    \n    var fecha2 = new Date(context.f2); // Your timezone!\n    var epoch2 = fecha2.getTime();\n    \n    var hora1 = context.h1;\n    var hora2 = context.h2;\n    epoch1 = epoch1 + hora1*3600*1000; //3600 segundos en una hora, 1000 miliseg por cada seg\n    epoch2 = epoch2 + hora2*3600*1000;\n    \n    epoch1_show = new Date(epoch1).toLocaleString();\n    epoch2_show = new Date(epoch2).toLocaleString();\n    \n    epoch1_show = epoch1_show.replace(/-/g,\":\");\n    epoch1_show = epoch1_show.replace(\" \",\":\");\n    var partes1 = epoch1_show.split(\":\");\n    \n    epoch2_show = epoch2_show.replace(/-/g,\":\");\n    epoch2_show = epoch2_show.replace(\" \",\":\");\n    var partes2 = epoch2_show.split(\":\");\n\n    var dia1 = partes1[2];\n    var mes1 = partes1[1];\n    var anio1 = partes1[0];\n    var h1 = partes1[3];\n var m1 = partes1[4];\n    var s1 = partes1[5];\n    \n    var dia2 = partes2[2];\n    var mes2 = partes2[1];\n    var anio2 = partes2[0];\n var h2 = partes2[3];\n    var m2 = partes2[4];\n    var s2 = partes2[5];\n    \n    msg.payload = 'Inicio:' + dia1 + '/' + mes1 + '/' + anio1 + '-' + h1 + 'hs' + '<br>Final: ' + \" \" + dia2 + '/' + mes2 + '/' + anio2 + '-' + h2 + 'hs';\n\n    context.f1 = 0;\n    context.f2 = 0;\n    context.h1 = -1;\n    context.h2 = -1;\n    context.ref = 'Off';\n    context.con = 'NADA';\n    \n    return msg;\n}\nelse {\n return null\n}",
        "outputs": 1,
        "noerr": 0,
        "x": 1920,
        "y": 920,
        "wires": [
            [
                "4d4268a3.1ad878"
            ]
        ]
    },
    {
        "id": "be79df3d.f330d",
        "type": "influxdb",
        "z": "",
        "hostname": "localhost",
        "port": "8086",
        "protocol": "http",
        "database": "invernadero-IT10",
        "name": "",
        "usetls": false,
        "tls": ""
    },
    {
        "id": "16e1aa9a.dc41f5",
```

```
        "type": "mqtt-broker",
        "z": "",
        "name": "Broker local",
        "broker": "localhost",
        "port": "1883",
        "clientid": "",
        "usetls": false,
        "compatmode": true,
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "closeTopic": "",
        "closeQos": "0",
        "closePayload": "",
        "willTopic": "",
        "willQos": "0",
        "willPayload": ""
    },
    {
        "id": "124a1fa3.a5e4c8",
        "type": "ui_group",
        "z": "",
        "name": "Datos Instantaneos - mq9/LDR",
        "tab": "efb1f443.02175",
        "order": 3,
        "disp": false,
        "width": "10",
        "collapse": false
    },
    {
        "id": "b8c3e6e8.f28eb",
        "type": "ui_group",
        "z": "",
        "name": "Hora",
        "tab": "efb1f443.02175",
        "order": 1,
        "disp": false,
        "width": "20",
        "collapse": false
    },
    {
        "id": "d579fb4d.09324",
        "type": "ui_group",
        "z": "",
        "name": "Datos Instantaneos - Temp/humedad",
        "tab": "efb1f443.02175",
        "order": 2,
        "disp": false,
        "width": "10",
        "collapse": false
    },
    {
        "id": "5d22a336.67eb5c",
        "type": "ui_group",
        "z": "",
        "name": "Datos historicos de concentración de CO",
```

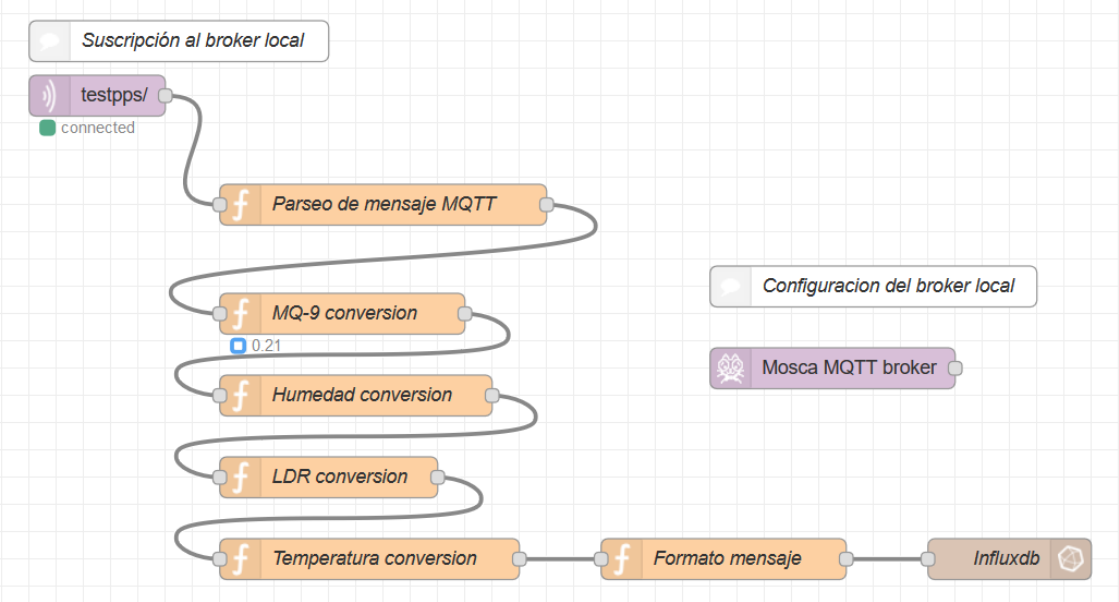
```
        "tab": "c06a1a1e.69ede",
        "order": 10,
        "disp": true,
        "width": "20",
        "collapse": true
    },
    {
        "id": "a7f71eb3.656578",
        "type": "ui_group",
        "z": "",
        "name": "Datos historicos de temperatura ambiente",
        "tab": "c06a1a1e.69ede",
        "order": 4,
        "disp": true,
        "width": "20",
        "collapse": true
    },
    {
        "id": "8f212678.706158",
        "type": "ui_group",
        "z": "",
        "name": "Datos historicos de humedad",
        "tab": "c06a1a1e.69ede",
        "order": 6,
        "disp": true,
        "width": "20",
        "collapse": true
    },
    {
        "id": "afecb5f7.f3fbc8",
        "type": "ui_group",
        "z": "",
        "name": "Datos historicos de niveles de luz",
        "tab": "c06a1a1e.69ede",
        "order": 8,
        "disp": true,
        "width": "20",
        "collapse": true
    },
    {
        "id": "8ed24b80.43df2",
        "type": "ui_group",
        "z": "",
        "name": "export temp",
        "tab": "c06a1a1e.69ede",
        "order": 5,
        "disp": false,
        "width": "20",
        "collapse": false
    },
    {
        "id": "852840d5.64a668",
        "type": "ui_group",
        "z": "",
        "name": "export hum",
        "tab": "c06a1a1e.69ede",
        "order": 7,
        "disp": false,
```

```
        "width": "20",
        "collapse": false
    },
    {
        "id": "224c07c1.06a2d",
        "type": "ui_group",
        "z": "",
        "name": "export ldr",
        "tab": "c06a1a1e.69ede",
        "order": 9,
        "disp": false,
        "width": "20",
        "collapse": false
    },
    {
        "id": "29206391.d60f3c",
        "type": "ui_group",
        "z": "",
        "name": "export mq9",
        "tab": "c06a1a1e.69ede",
        "order": 11,
        "disp": false,
        "width": "20",
        "collapse": false
    },
    {
        "id": "d04e1b2.f8adae8",
        "type": "ui_group",
        "z": "",
        "name": "Tiempo de refresco de gráficos",
        "tab": "c06a1a1e.69ede",
        "order": 3,
        "disp": false,
        "width": "6",
        "collapse": false
    },
    {
        "id": "dc137fb4.d8e79",
        "type": "ui_group",
        "z": "",
        "name": "Mom inicial",
        "tab": "c06a1a1e.69ede",
        "order": 1,
        "disp": false,
        "width": "7",
        "collapse": false
    },
    {
        "id": "41fbf2c8.27dd2c",
        "type": "ui_group",
        "z": "",
        "name": "Mom final",
        "tab": "c06a1a1e.69ede",
        "order": 2,
        "disp": false,
        "width": "7",
        "collapse": false
    },
```

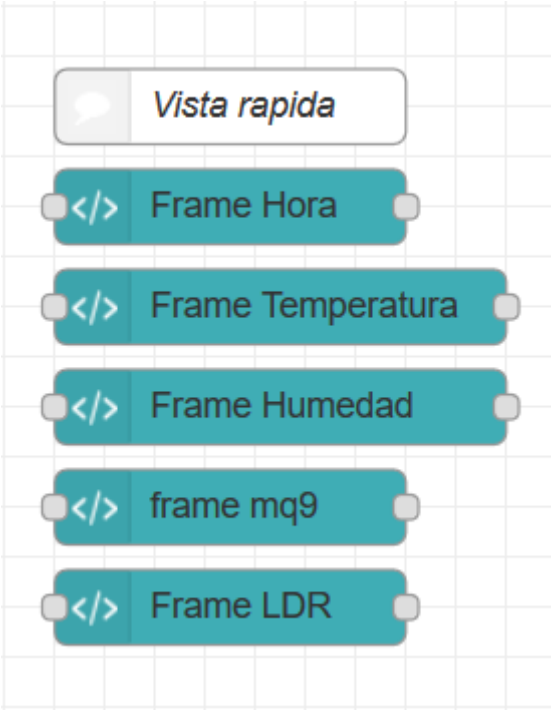
```
    {
        "id": "efb1f443.02175",
        "type": "ui_tab",
        "z": "",
        "name": "Invernadero",
        "icon": "fa-leaf",
        "order": 1,
        "disabled": false,
        "hidden": false
    },
    {
        "id": "c06a1a1e.69ede",
        "type": "ui_tab",
        "z": "",
        "name": "Registros",
        "icon": "fa-area-chart",
        "order": 2,
        "disabled": false,
        "hidden": false
    }
]
```


Es la información de todos los nodos creados con la configuración necesaria para nuestros propósitos. Si seccionamos un poco todo los nodos:

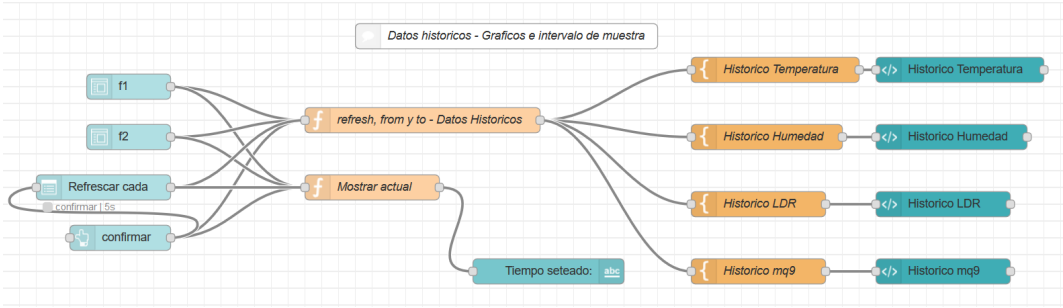
- **Nodos principales:** Son los que se encargan de tomar los datos que son publicados al broker MQTT, que también es creado aquí, parsear la información, y darle un formato correcto para insertarse en la base de datos.



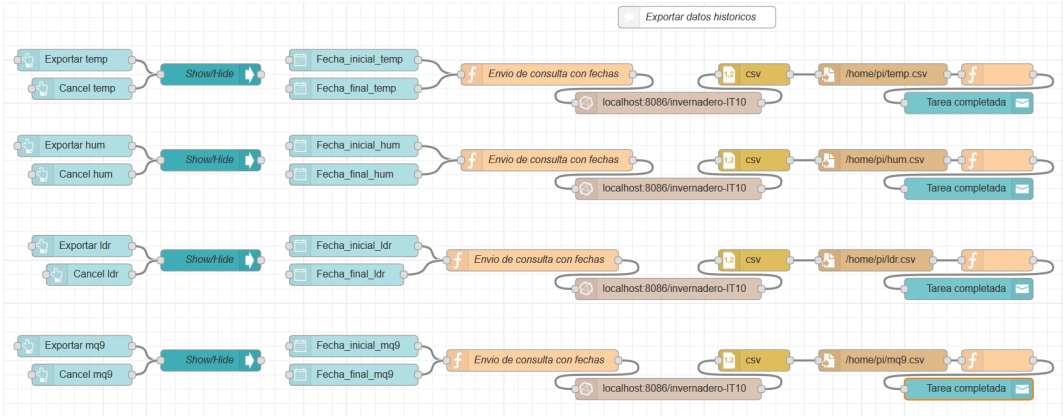
- **Nodos para visualización web:** Gracias a los nodos **Dashboard** se puede crear una web rápidamente, que usaremos para visualizar en tiempo real lo que esta pasando en nuestro invernadero. Podemos verlo en 3 partes:
    - **Nodos con datos instantáneos:** Nos muestra lo que sucede actualmente en el invernadero, con el último dato enviado de temperatura, humedad, luz y concentración de CO.



- **Nodos de datos historicos:** Nos permite visualizar datos tomados hisoticamente por los sensores, seteando el intervalo de tiempo que se quiere ver.



- **Nodos para exportar datos historicos:** Los utilizamos para poder extraer los datos en un archivo csv sobre cada sensor en particular.



Es necesario modificar la dirección IP en todos los nodos que hagan referencia a gráficos. Al momento de realizar el proyecto cuenta con la dirección IP `192.168.20.131`, que es donde esta alojado Node-RED, Grafana, InfluxDB y el broker local. Se aconseja cambiar a la dirección que corresponda, sin modificar los puertos, para poder ejecutarlo correctamente. Para hacer esto hay que ingresar a cada nodo y verificar si hay direcciones IP que sirvan para configuración o ejecución del flujo de Node-RED. Están asociados a las aplicaciones antes mencionadas. También así si se utilizan usuarios y contraseñas para securizar los procesos.

Una vez realizado lo dicho anteriormente, se puede ir al botón **Deploy**, sobre la esquina superior derecha. Se puede observar que no se están enviando datos a la base de datos porque la misma no esta creada todavía, y es el paso siguiente.

InfluxDB

InfluxDB, desarrollada por influxdata, es una base de datos basada en series de tiempo (*time-series database*). Maneja de forma eficiente estas series de datos, con miles de inserciones por segundo, y de los que necesitamos hacer cálculos, búsquedas, y agregar información, como medias, máximos, etc, y todo ello en tiempo real. Dentro de esta categoría, InfluxDB es una base de datos que supera a los esquemas SQL y NoSQL. Tiene una versión comunitaria de código abierto.

Se instala con el comando: `sudo apt-get install influxdb`. Las nuevas versiones están listas para entrar como administrador.

Para poder acceder, utilizar y modificar la base de datos se requiere un cliente. Para esto lo instalamos con `sudo apt-get install influxdb-client`.

Para iniciar la base de datos no es necesario hacerlo a través del servicio, es decir `sudo service influxdb start`. Solo hace falta iniciarlo así:

```
sudo influxd run --config /etc/influxdb/influxdb.conf &
```

Lo corremos en segundo plano para poder utilizar el cliente en la misma terminal, sobre todo si pretendemos usar una sola consola con ssh.

También podríamos hacer las consultas o lo que sea necesario a través de un cliente web.

Configuraciones previas

Para acceder al cliente por el terminal, solo ingresamos `influx` en la misma. Puede que sea necesario aclarar en algún caso al host al que se conecta, esto se hace: `influx -host IP`. Esa IP es la local, o en todo caso la del equipo que maneje la base de datos.

Primero debemos crear un usuario como el de Raspbian (ya que se desarrollo todo sobre una Raspberry Pi) para la base de datos:

```
CREATE USER "pi" WITH PASSWORD 'it10' WITH ALL PRIVILEGES
```

Si se usa otra plataforma o una PC, cambiar **pi** e **it10** por su respectivo usuario y contraseña.

Crear Base de datos

Con InfluxDB ya instalado solo hace falta crear la base de datos, ya que Node-RED envía los datos, pero si no hay una base de datos donde insertarse, estos nunca se envían y dará un error en la interfaz de Node-RED. Se procede de la siguiente manera:

1. Abrir el cliente de influxDB en la terminal de Raspbian:

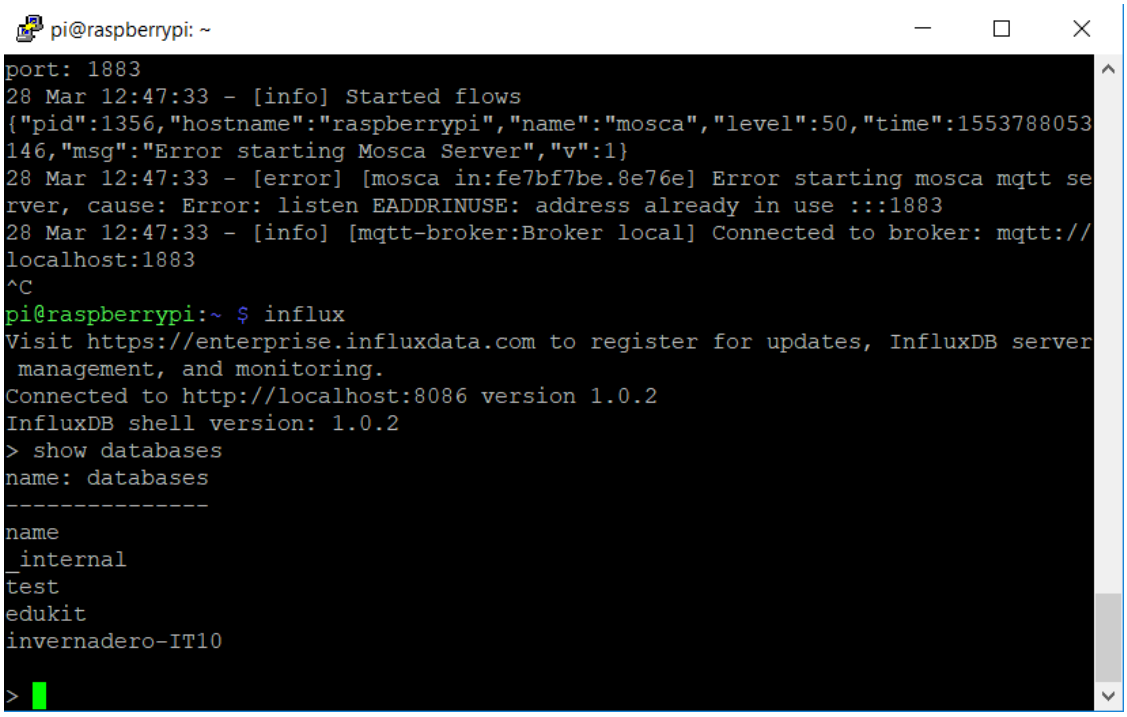
```
influx
```

2. Creamos la base de datos `invernadero-IT10`:

```
CREATE DATABASE invernadero-IT10
```

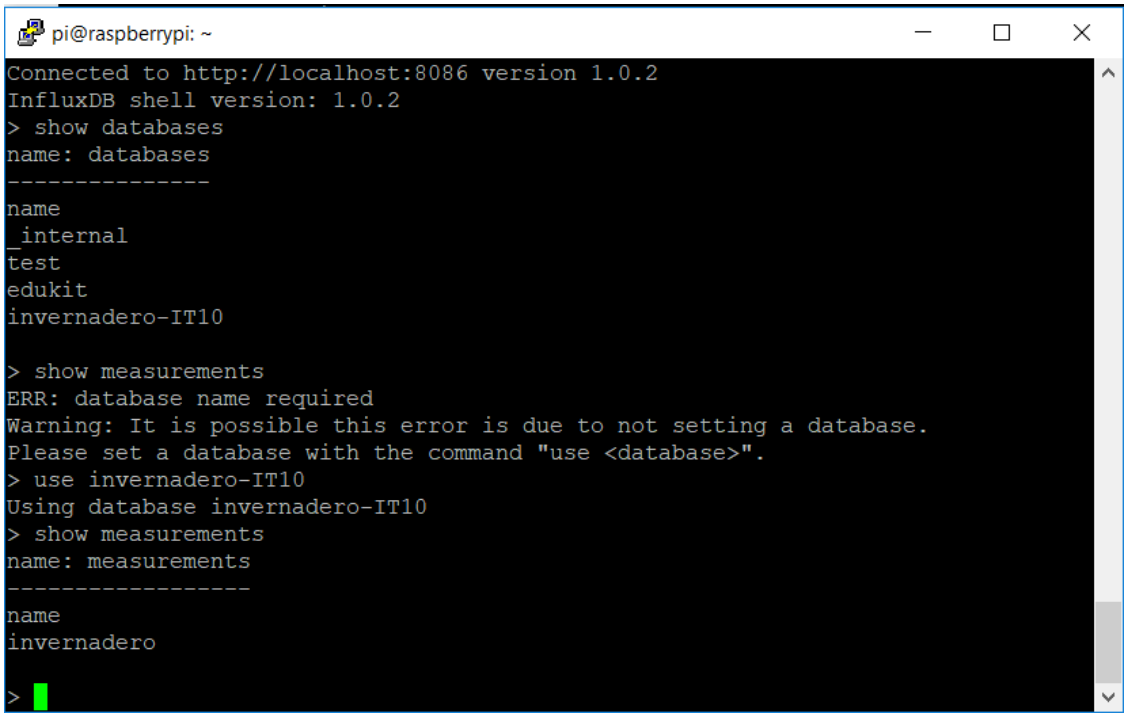
3. Mostramos para ver que se creo correctamente:

```
show databases
```

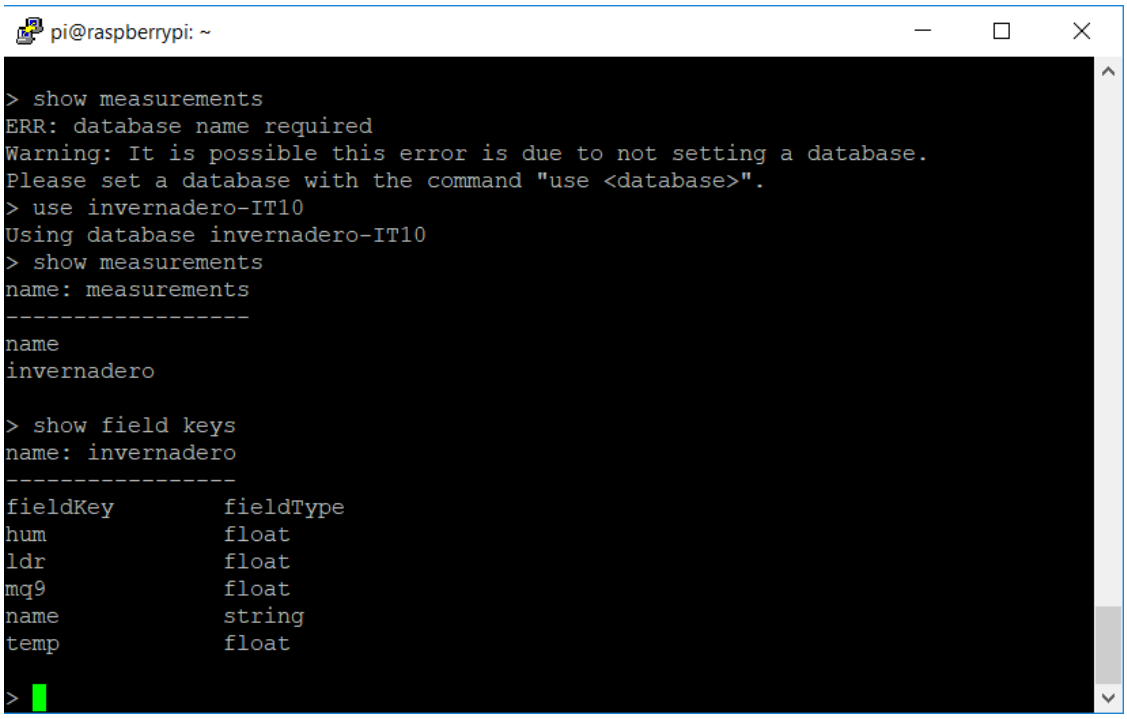


4. Una vez hecho lo anterior, Hacemos **Deploy** nuevamente en Node-RED, para que cree el *Measurement* **invernadero**, con los respectivos *Field Keys* comenzar a recibir los primeros valores. Luego lo observaremos con el cliente influx.

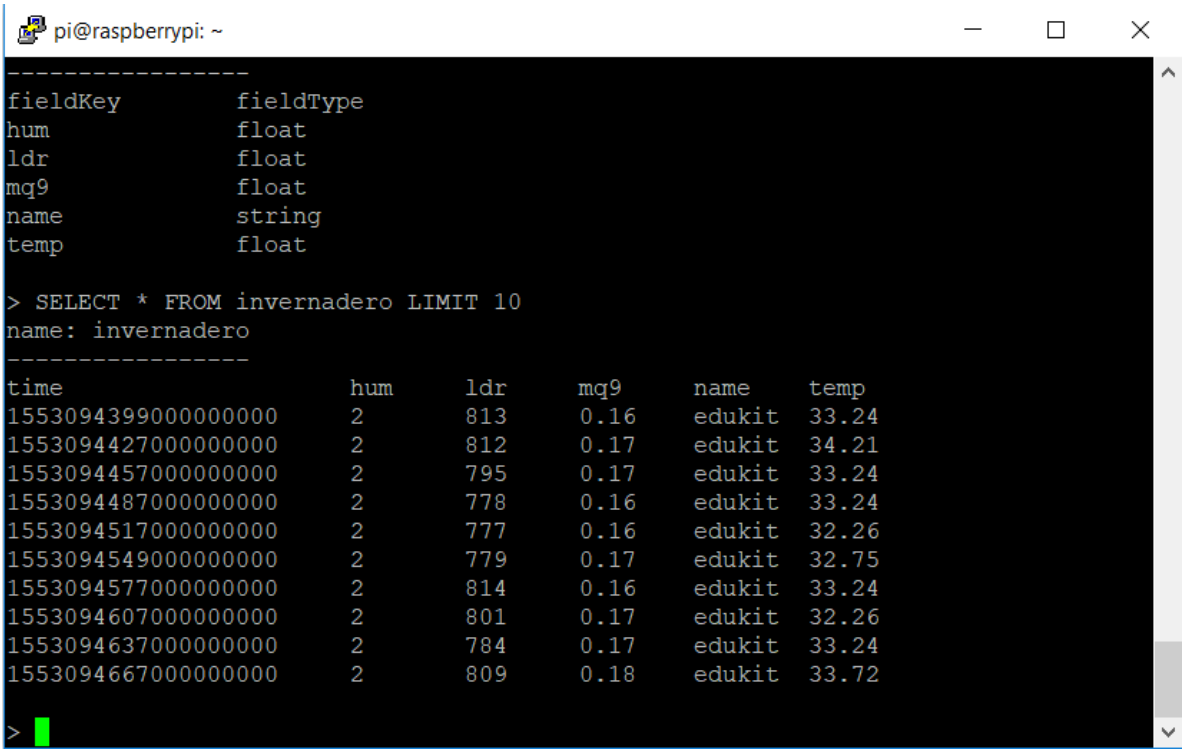
```
use invernadero-IT10
show measurements
```




```
show FIELD KEYS
```




```
SELECT * FROM invernadero LIMIT 10
```



Grafana

Grafana es una herramienta de código abierto para el análisis y visualización de métricas. Está escrita en Lenguaje Go (creado por Google) y Node.js LTS. Además cuenta con una fuerte Interfaz de Programación de Aplicaciones (API); es una aplicación que ha venido escalando posiciones, con una comunidad entusiasta de más de 600 colaboradores bien integrados. Su código fuente está publicado en GitHub.

A partir de una serie de datos recolectados se puede obtener un panorama gráfico presentado en una forma elegante y amigable. Grafana permite consultar, visualizar, alertar y comprender métricas que pueden ser recolectadas y/o procesadas por aplicaciones de terceros. Puede recopilar de forma nativa datos de *Cloudwatch, Graphite, Elasticsearch, OpenTSDB, Prometheus, Hosted Metrics e InfluxDB*.

Entre sus características posee:

- 37 complementos de fuentes de datos.
- 28 complementos para el panel.
- 15 complementos de aplicaciones.
- Más de 600 paneles de control creados para aplicaciones populares.

Instalación

Para la Raspberry Pi 3 B+ se puede instalar por terminal. Primero se descarga:

```
wget https://dl.grafana.com/oss/release/grafana_5.4.3_armhf.deb
sudo dpkg -i grafana_5.4.3_armhf.deb
```

Y cuando termina la descarga se descomprime e instala:

```
sudo dpkg -i grafana_5.4.3_armhf.deb
```

Luego se indica los siguiente para instalar Grafana:

```
    ### NOT starting on installation, please execute the following statements to configure grafana to start automatically using systemd
 sudo /bin/systemctl daemon-reload
 sudo /bin/systemctl enable grafana-server
    ### You can start grafana-server by executing
 sudo /bin/systemctl start grafana-server
```

Por ultimo habilitamos el servicio y activamos el demonio para que inicie la herramienta.

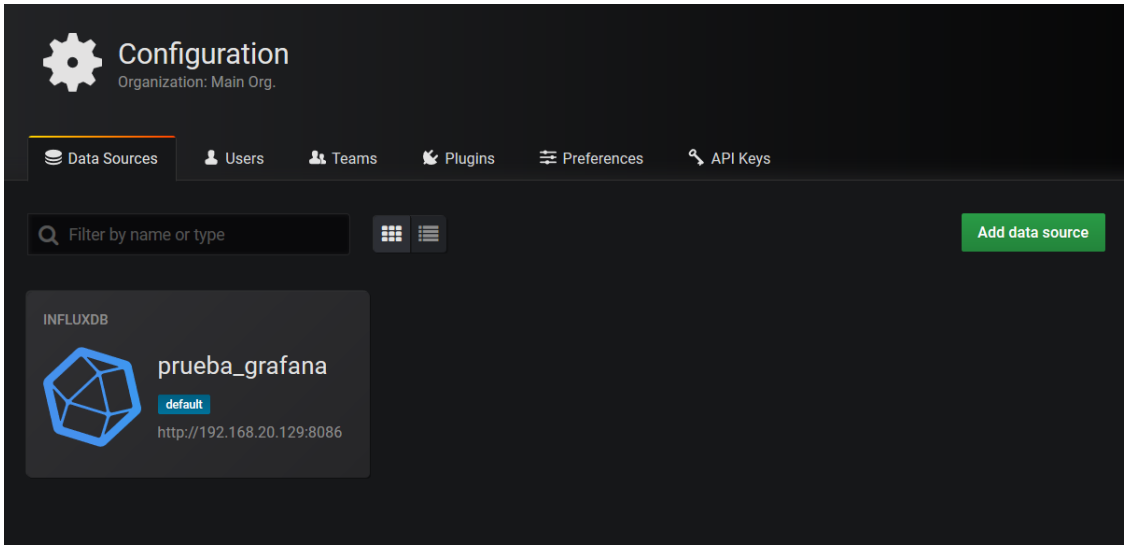
```
    sudo /bin/systemctl daemon-reload
    sudo /bin/systemctl enable grafana-server
    sudo /bin/systemctl restart grafana-server
    sudo service grafana-server start
    sudo systemctl enable grafana-server.service
    sudo /lib/systemd/systemd-sysv-install enable grafana-server
```

Ahora sabiendo la dirección IP del equipo, ingresando en un navegador web con `http://<IP>:3000/` se puede acceder a Grafana y desarrollar un dashboard.

Agregando un datasource

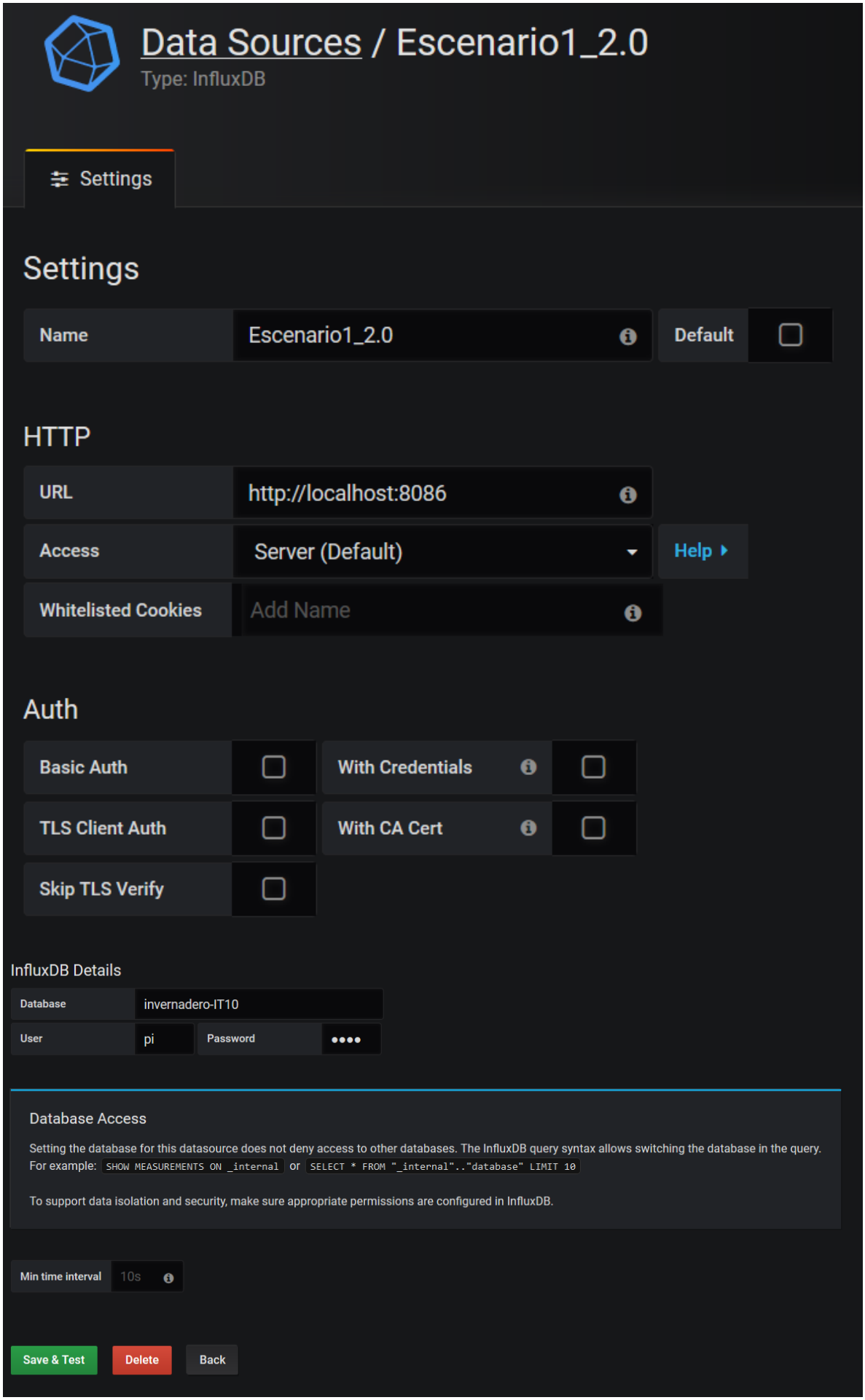
El datasource es la fuente de datos que se usara para generar las graficas. Grafana toma los datos insertados en la base de datos temporal y los pone a disposición en una grafica de nuestro gusto.

1. Ingresar a la aplicación web con nuestra IP local o a la que estamos accediendo en la misma red, en este caso `http://192.168.20.129:3000` e ingresar con el usuario:contraseña correspondiente.
2. Asignar una fuente de datos. Ir a configuración, sobre el panel izquierdo y seleccionar `Data sources`. Luego presionar sobre `Add data source`.



En la imagen aparece ya agregada, pero el procedimiento es el mismo para agregar cualquier *data source*.

3. Configurar la fuente de datos. Aquí se pone la información necesaria para acceder a la base de datos InfluxDB.

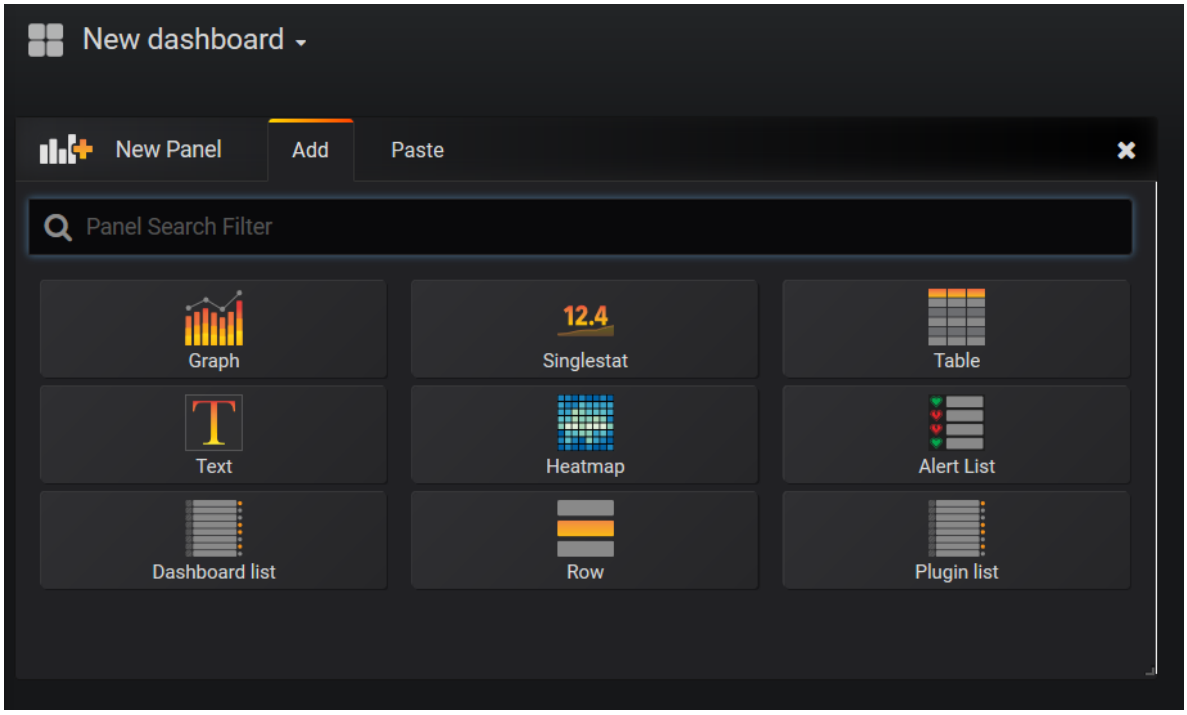


Se le pone un nombre (en este caso *Escenario_2.0*). En las opciones HTTP se pone la dirección IP local o de la maquina a la que se accede en la red, con el puerto 8086, que es el puerto por defecto en el que la base de datos escucha las peticiones HTTP. Dejamos la opción `Access: Server(Default)`. En cuanto a la autenticación, solo ponemos la básica, que son nuestro *usuario:contraseña* de Raspbian (`pi:it10`). Por ultimo en los detalles

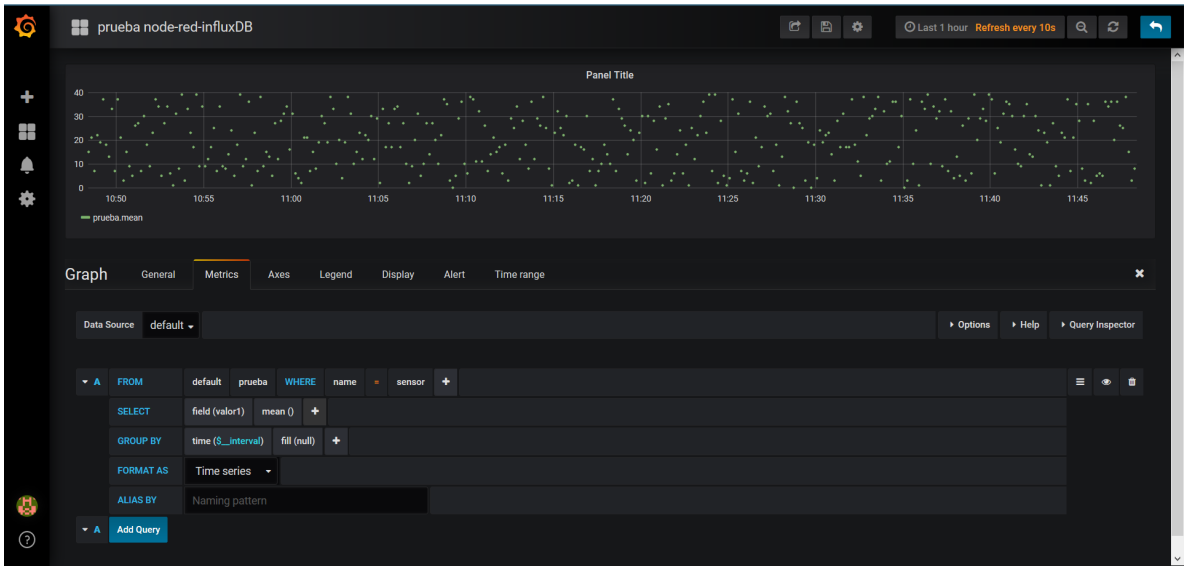
de InfluxDB, se indica la base de datos a la que se quiere acceder. En este caso a `invernadero-IT10`, con el *usuario:contraseña* `pi:it10`. Esta ultima parte de autenticación no es necesaria ya que se accede de manera local con el mismo usuario pi. Se guarda y se procede a crear un dashboard.

Creando el dashboard

Aquí se decide que tipo de panel se va a utilizar.

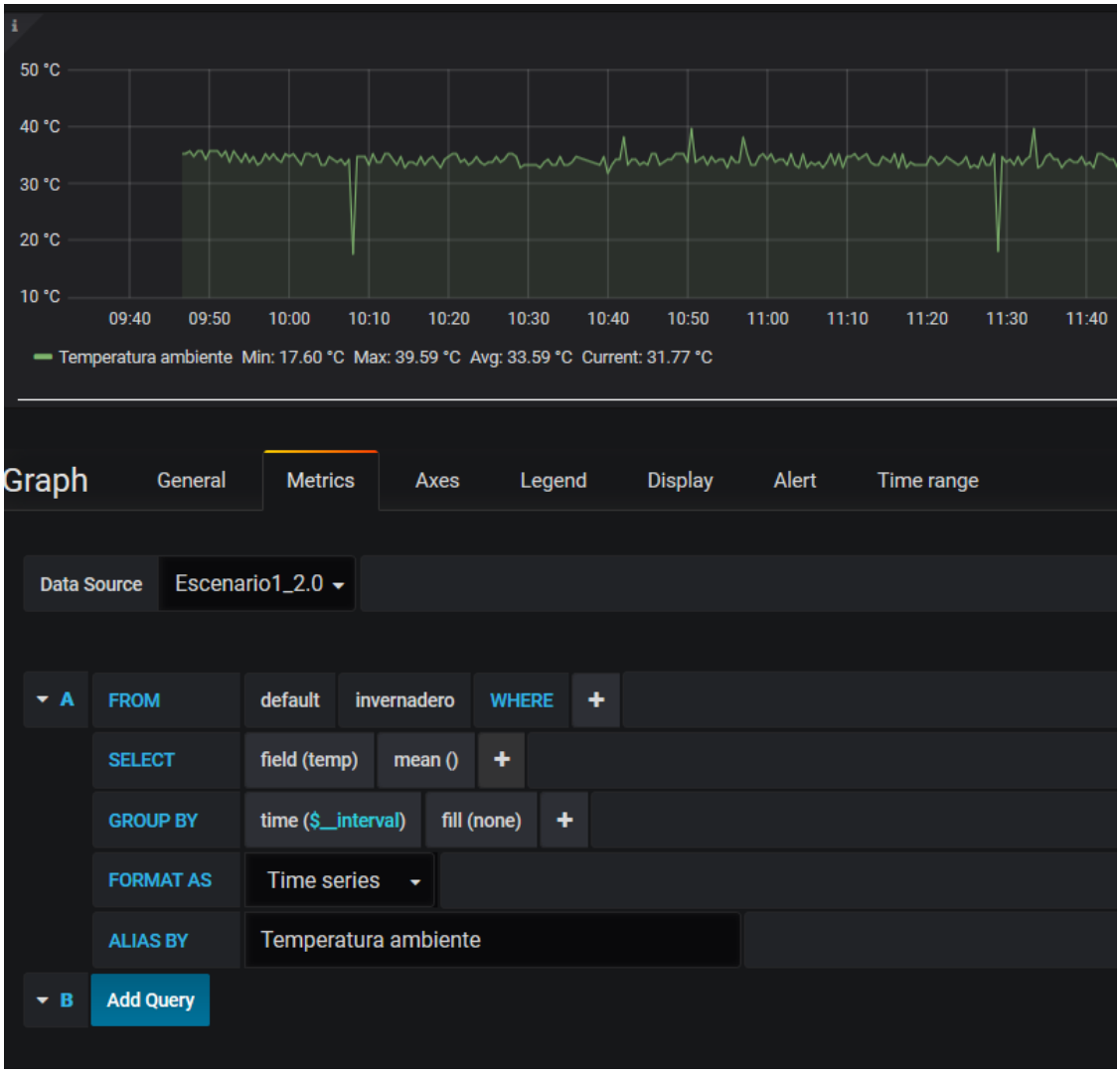


Sobre el `Panel Title` se hace click, luego `Edit`.



En la pestaña metrics esta la parte esencial para mostrar los datos. Es donde se encuentra la petición a la base de datos. Por el momento solo realizaremos una sola para este panel.

- En la línea `FROM`: *Data source Measurement* `WHERE` *Field* `=` *valor*. Esta línea debemos indicar la fuente como **default** o **invernadero-IT10**, **invernadero** como *Measurement*. `WHERE` nos sirve para identificar datos particulares dentro de la misma base de datos, como por ejemplo todo lo referido a un sensor, entonces se indica **name** como *Field* y **edukit** valor de ese field. En el caso que tuviéramos mas sensores de esta forma se puede discriminar uno de otro.
- En la línea `SELECT`: *field() mean()*. En field, indicamos que para **temp** realizaremos la grafica con sus datos, y realizando la media de los datos aportados en el intervalo de tiempo que se ha consultado.



- Se puede agregar o modificar la característica de como se procesan los datos, no es solamente **mean()**, Para el caso es el mas conveniente para visualizar.
- En la línea `GROUP BY`: `time($_interval)` `fill(null)`. Aquí se indica el tiempo de acceso a los datos y como mostrarlos. Dejamos por defecto `time` y en **fill** podemos dejarlo como **null**, que se visualiza en forma de puntos, o con **none**, en forma de línea continua. Dejaremos **none** Es importante dejar por defecto `time($_interval)`, ya que esto nos permite cambiar el rango de fechas a visualizar de forma dinámica desde el dashboard, muy útil para nuestros propósitos.
- En la línea `FORMAT AS`: `Time Series` lo dejamos por defecto.

Para el caso de mostrar valores instantáneos la configuración es muy similar, solo cambiaríamos el campo `SELECT`: *field() mean()* por *last()*, que sería el último valor de la base de datos.

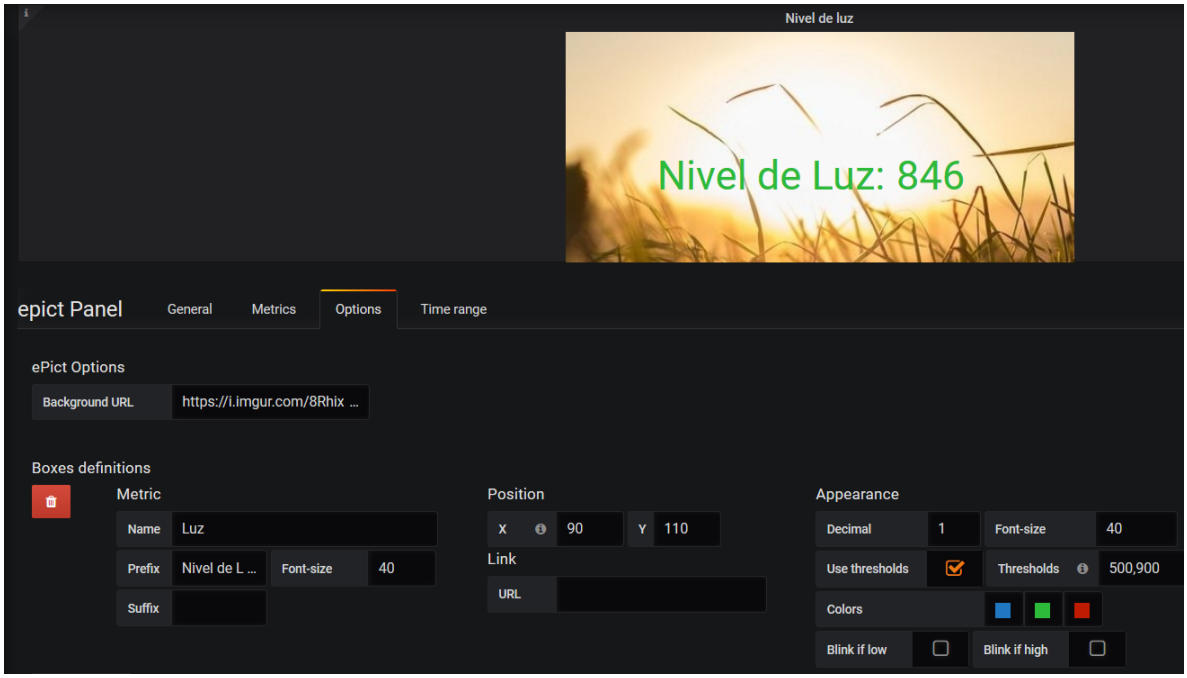
Si queremos un panel que contenga una imagen será necesario descargar uno nuevo. Los que vienen por defecto no tienen esa opción. Descargaremos el panel **epict Panel**, que tiene sus instrucciones en https://grafana.com/plugins/larona-epict-panel/installation, pero es sencillamente ingresar por la terminal:

```
grafana-cli plugins install larona-epict-panel
```

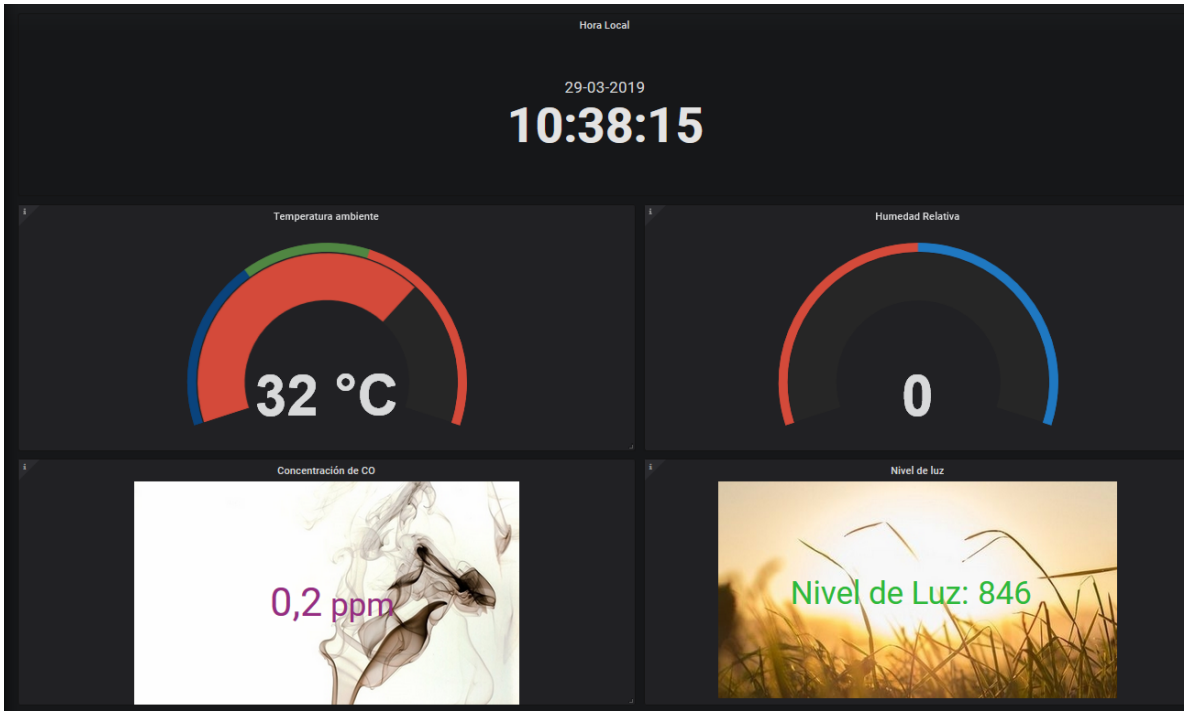
Luego solo resta reiniciar el servicio de Grafana.

```
sudo service grafana-server restart
```

La configuración es muy similar a los demás en cuanto las métricas. Para insertar una imagen, se debe ir a la pestaña **Options**, y en **Backgroud URL** copiar la URL de la imagen deseada.



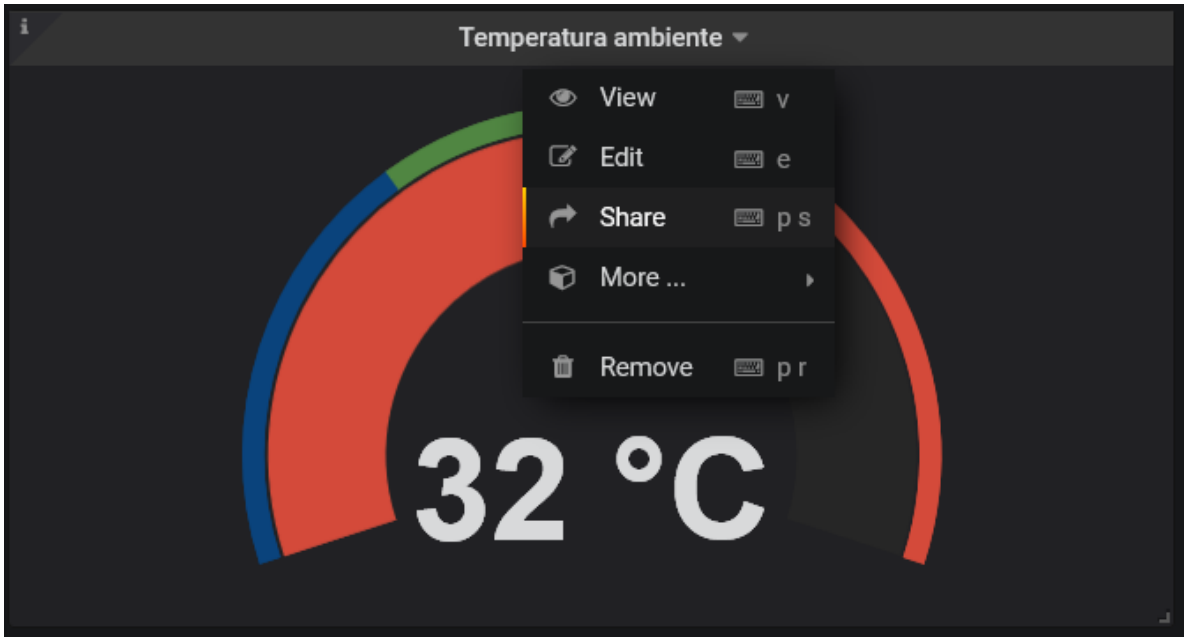
Para ver mas cuestiones títulos, leyendas, formas de grafico, etc, están las demás pestañas. El resultado de toda esta configuración es la siguiente:



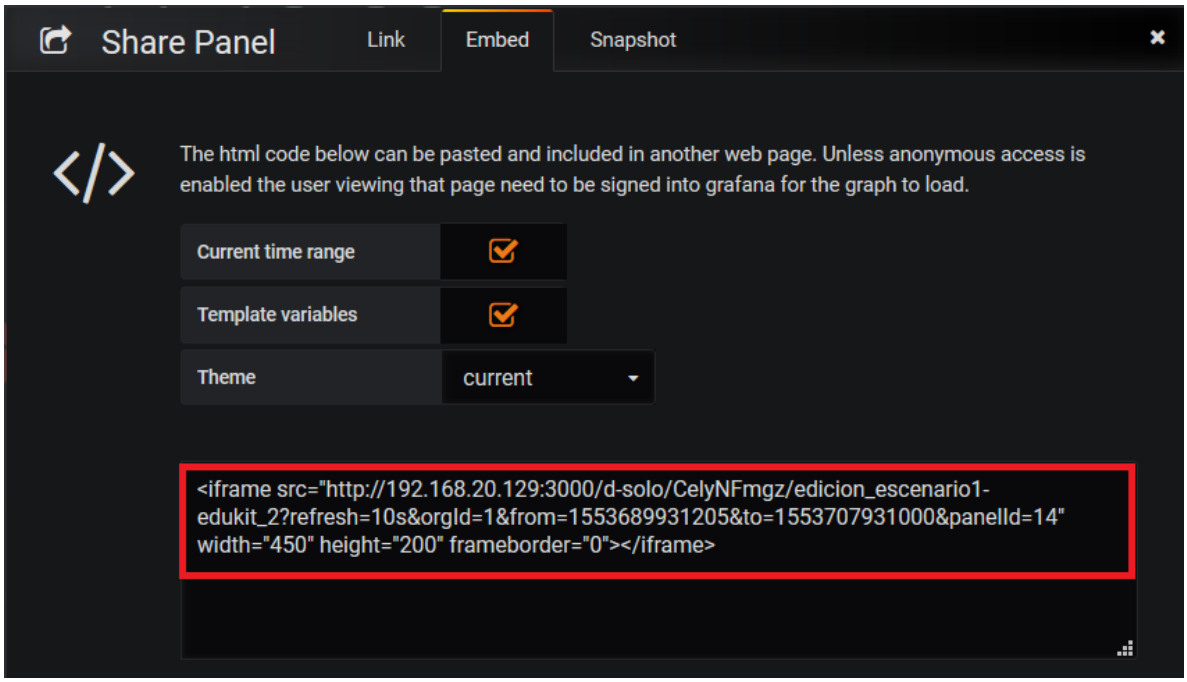
Recordar guardar el dashboard cuando se finalicen los cambios, para luego poder exportarlos.

Embebiendo gráficos

Hasta acá solo podemos ver los gráficos ingresando a Grafana, pero la idea es poder tenerlos disponibles en nuestra web que desarrollamos con Node-RED. Para eso debemos ir uno por uno de los gráficos que queremos embeber, presionar en su título y luego en **Share**.



Después ir hasta la pestaña **Embed** y copiar el código HTML. Tener estos códigos presentes, que mas adelantes los usaremos en Node-RED.



Permisos para usuarios

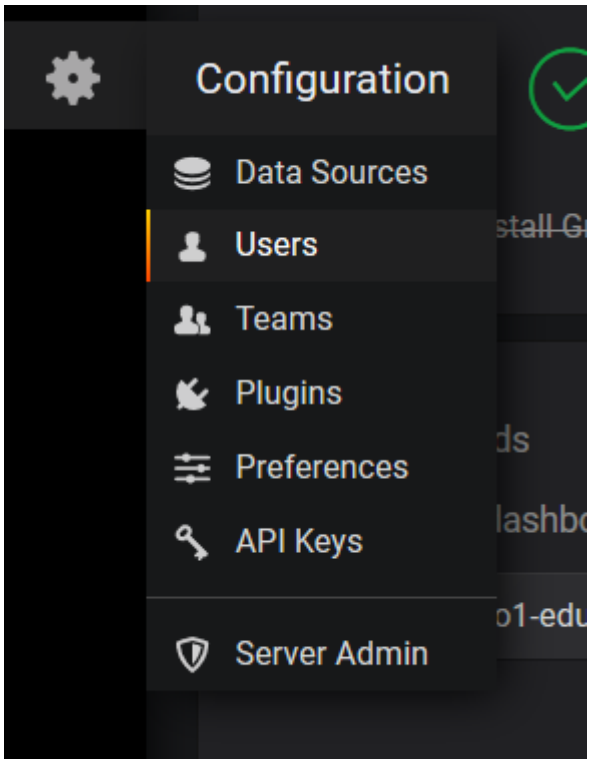
Nosotros como administradores ya tenemos todos los permisos posibles. Pero para poder embeber gráficos es necesario que ese "usuario que va a consultar el gráfico" que es Node-RED tenga al menos permisos para ver.

En el archivo de configuración de Grafana, que se ubica en `/etc/grafana/grafana.ini` nos dirigimos a la sección **Users** y **Anonymous Auth**. Hay que descomentar las siguientes líneas (borrar el punto y coma), salvo para el caso `org_name` que pondremos true, y en `org_role` que pondremos el nombre del Team que crearemos posteriormente:

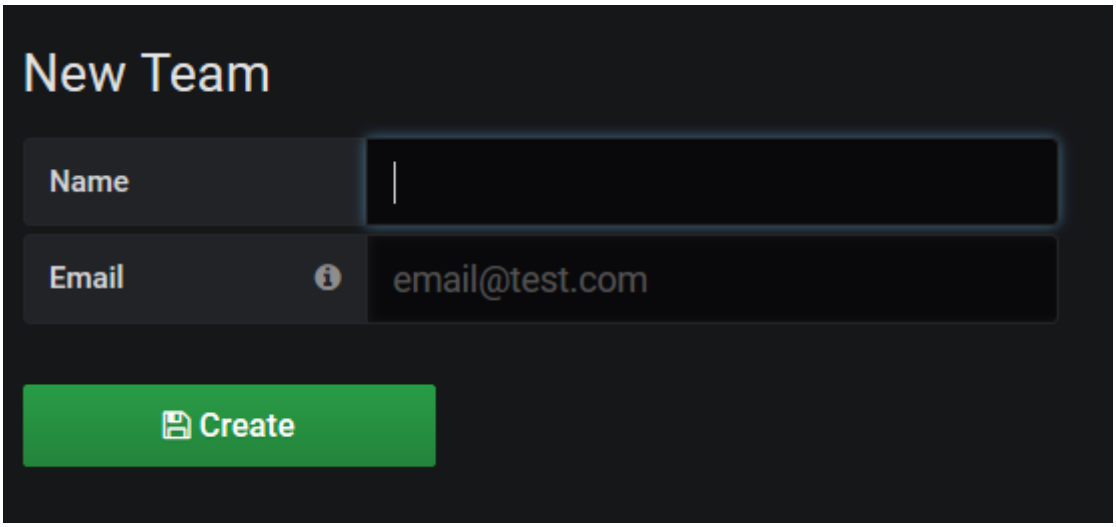
```
1    # Set to true to automatically assign new users to the default
organization (id$
2 auto_assign_org = true
3
```

```
    # Default role new users will be automatically assigned (if disabled
above is s$
auto_assign_org_role = Viewer

[auth.anonymous]
    # enable anonymous access
enabled = true

    # specify organization name that should be used for unauthenticated
users
org_name = IT10

    # specify role for unauthenticated users
org_role = Viewer

```

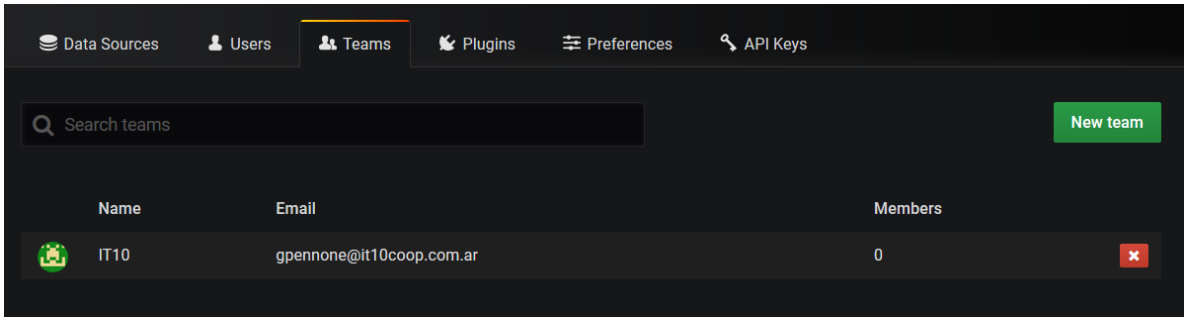
Luego, ya en la interfaz web, vamos a **Configuration**, **Users** y a la pestaña **Teams**.



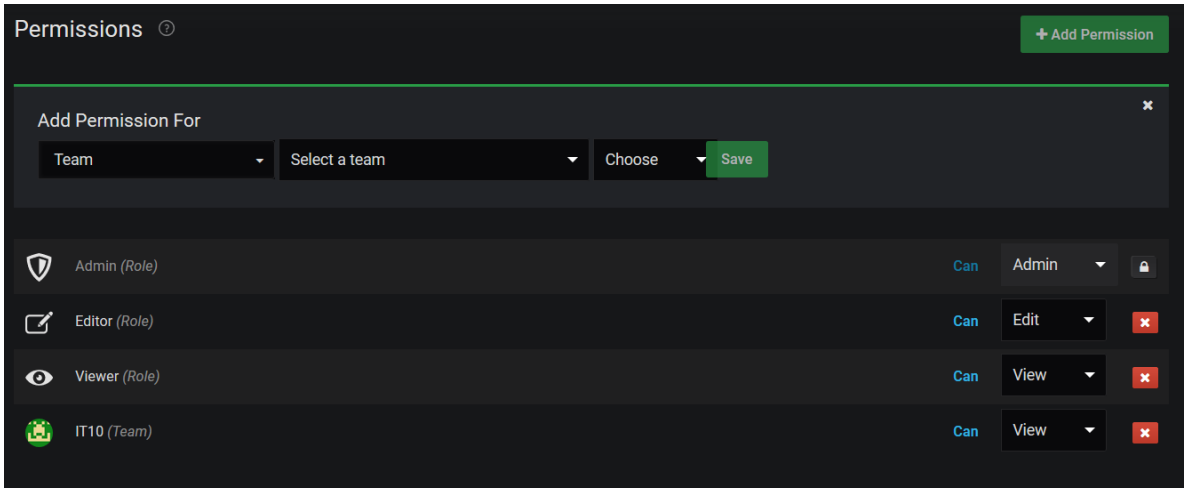
Crearemos un nuevo **Team**, que es donde por defecto entraran los nuevos usuarios que quieran ver los gráficos.



Ponemos un nombre y un mail al que hagan referencia (puede ser cualquiera). Y listo, Team creado.



Por último, debemos darle efectivamente permisos para ver a estos usuarios. Hay que dirigirse a las opciones del dashboard que creamos en un principio, a la sección **Permissions** y otorgar permisos de **Viewer**. Guardar para finalizar.



Y con esto ya Node-RED puede mostrar los gráficos que embebimos en sus nodos. Es necesario hacer un *restart* del servicio de `grafana-server` para que se carguen las modificaciones.

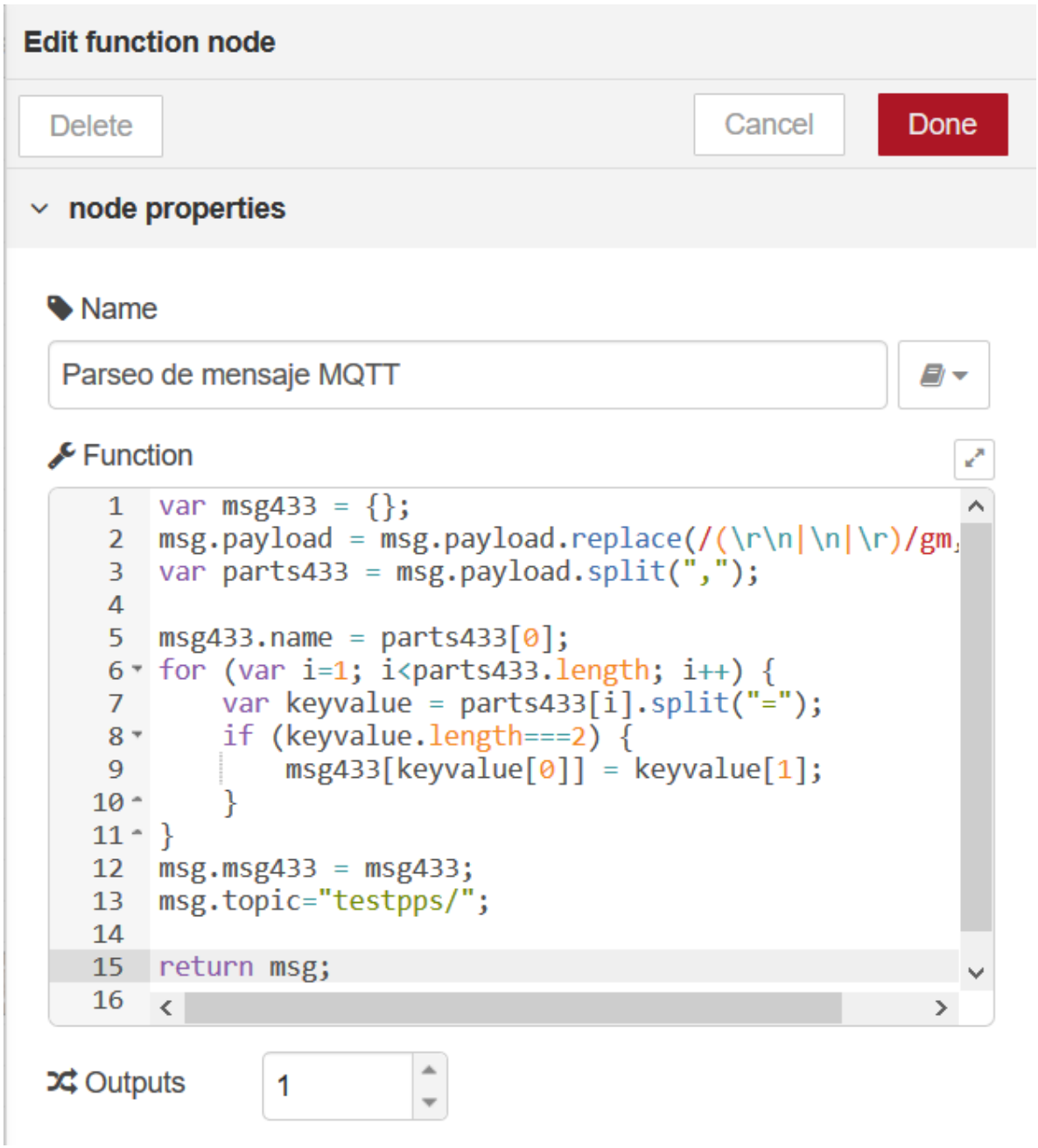
Integración de plataforma y herramientas

Ya tenemos listas todas las plataformas configuradas, y si bien Node-RED es el eslabón más importante que vincula todo, la configuración de algunos nodos son particulares de cada desarrollo. Ahora si podemos meternos dentro de los nodos para entender que sucede en cada parte. Haremos un recorrido desde que llega el dato al broker MQTT.

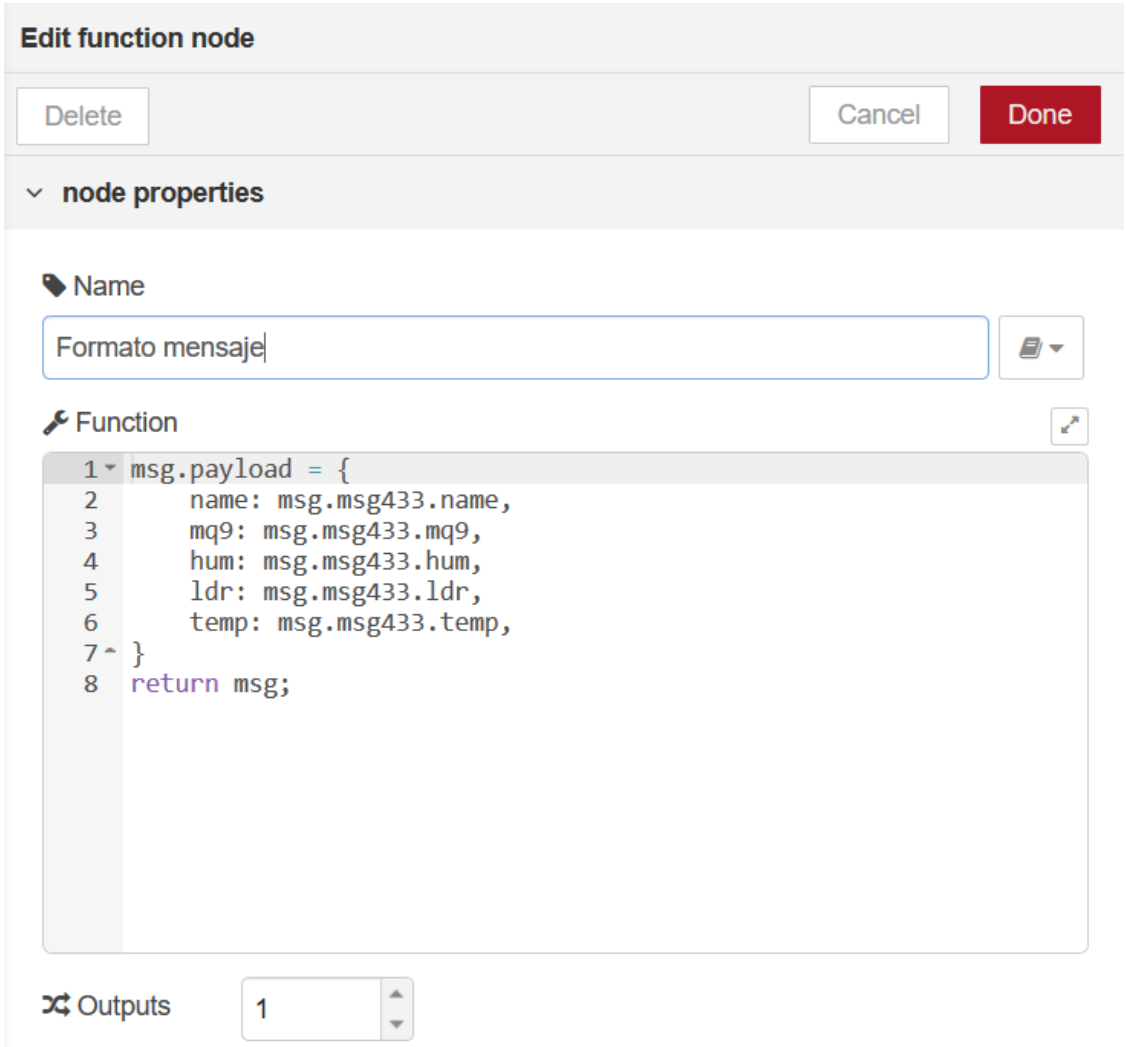
1. Nos suscribimos al broker MOSCA para escuchar los datos que ingresen al topic **testpps/**.
2. Se parsean los datos , ya que el formato es poco amigable:

```
edukit,mq9= 0.52,hum= 2.00,ldr=828.00,temp=34.21
```

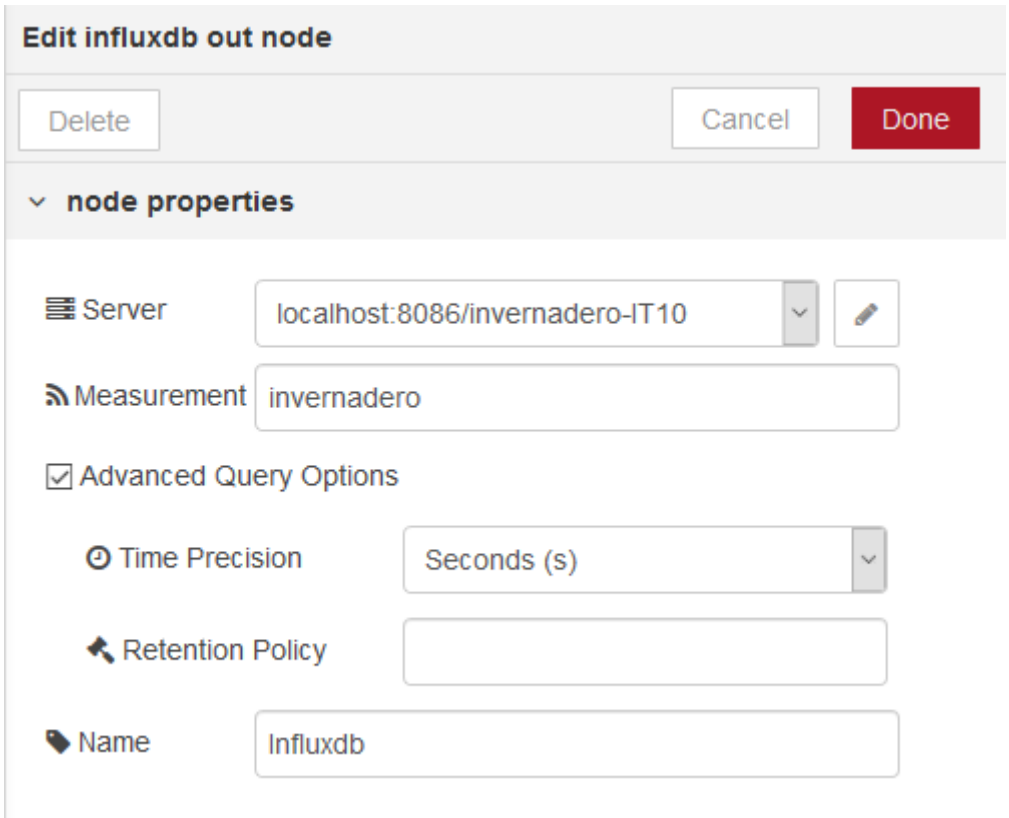
La función que parsea esto es la siguiente:



3. Posteriormente se añaden los valores como una propiedad de un objeto **msg** de Javascript y se termina de integrar en un payload que ingresará a la base de datos InfluxDB.

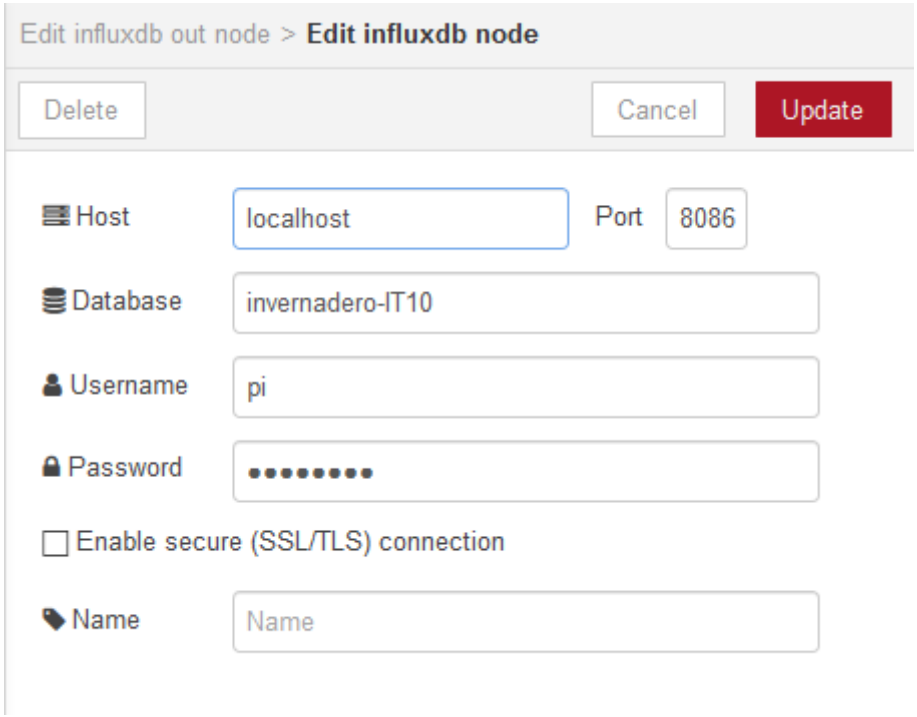


4. Se configura el nodo Influxdb para darle acceso a nuestra base de datos.

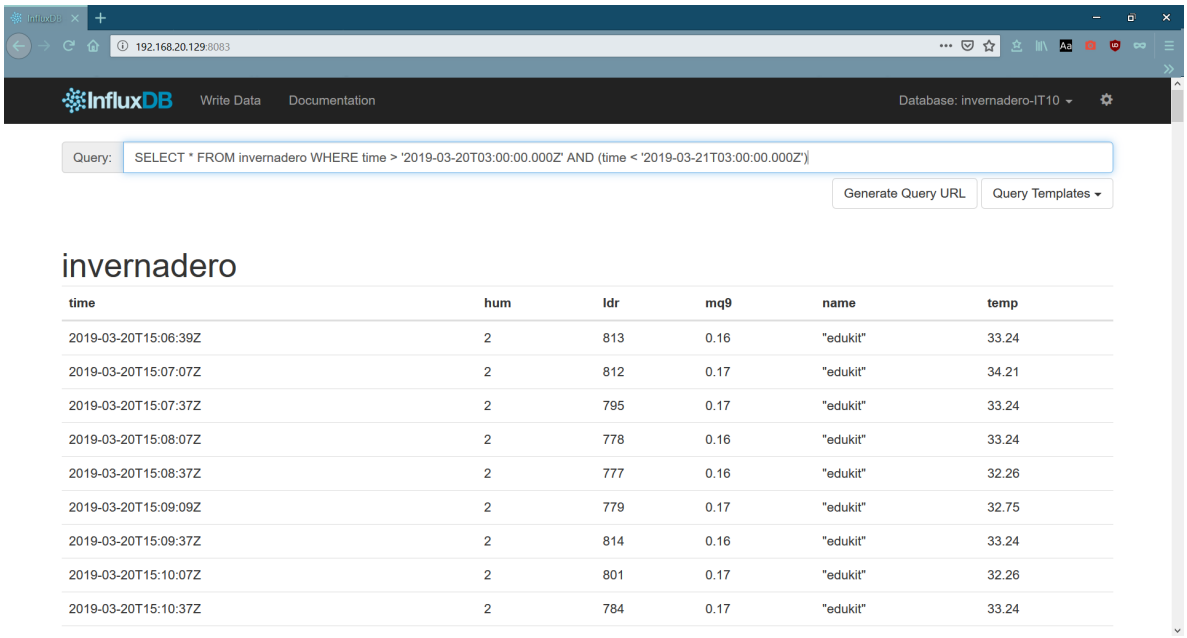


En el servidor se da la información de usuario y contraseña, además del nombre de la base de datos. Sin toda esta

información los datos no se guardarían, y grafana no podría generar gráficos.



5. Una vez desplegado el flujo de Node-RED ya se puede observar que ya hay datos en la base de datos, tanto por los gráficos, o por el cliente **influx** o también por el cliente web de InfluxDB



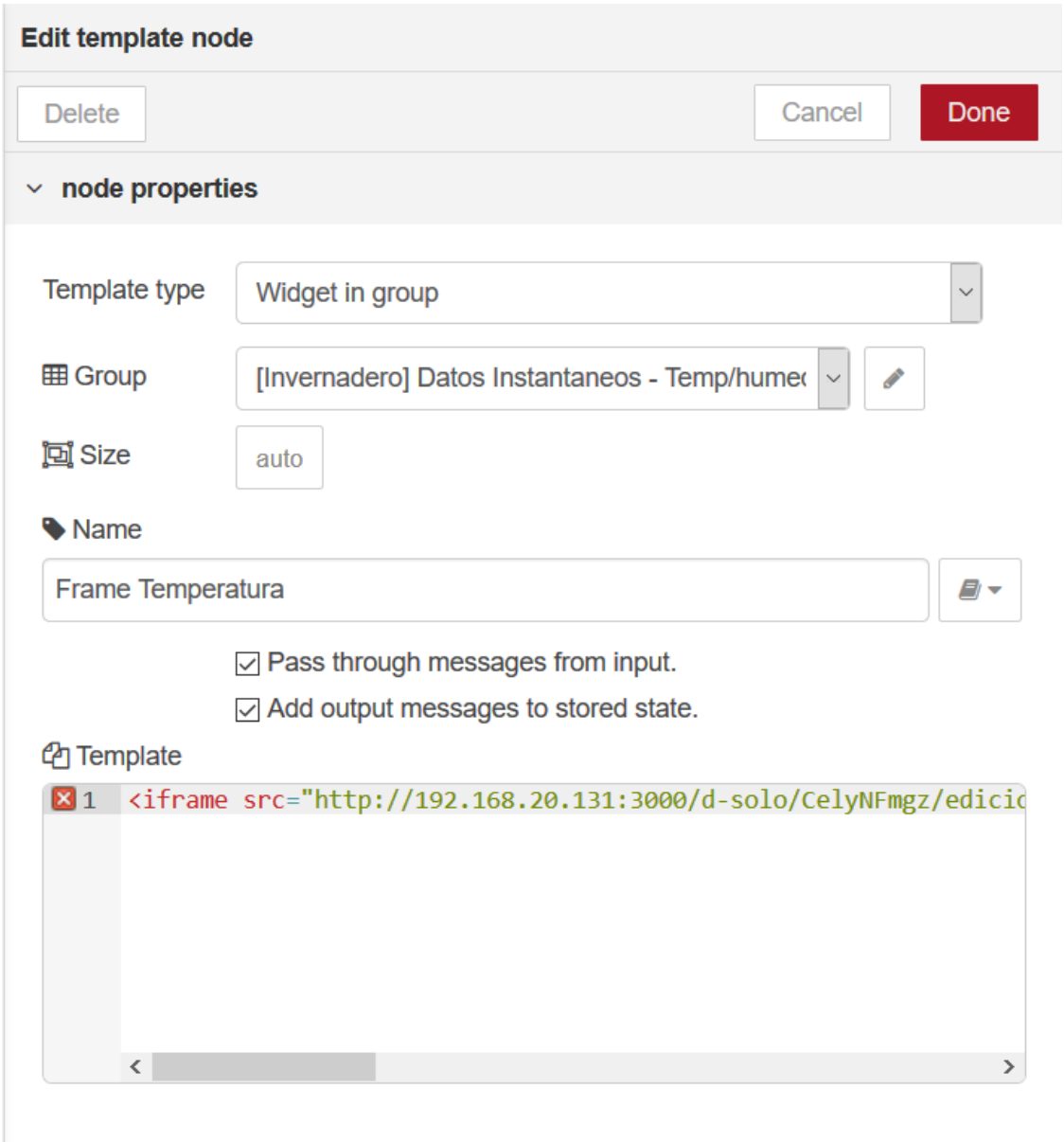
time	hum	ldr	mq9	name	temp
2019-03-20T15:06:39Z	2	813	0.16	"edukit"	33.24
2019-03-20T15:07:07Z	2	812	0.17	"edukit"	34.21
2019-03-20T15:07:37Z	2	795	0.17	"edukit"	33.24
2019-03-20T15:08:07Z	2	778	0.16	"edukit"	33.24
2019-03-20T15:08:37Z	2	777	0.16	"edukit"	32.26
2019-03-20T15:09:09Z	2	779	0.17	"edukit"	32.75
2019-03-20T15:09:37Z	2	814	0.16	"edukit"	33.24
2019-03-20T15:10:07Z	2	801	0.17	"edukit"	32.26
2019-03-20T15:10:37Z	2	784	0.17	"edukit"	33.24

Sección web

Vista rápida

1. En el nodo *Frame Hora* hay que ingresar el código HTML que sacamos de grafana del panel que muestra la hora (ya integrado al instalar la aplicación). En cuanto al campo **Group** ya fue definido en el código que brindamos del flujo de Node-RED.

2. Para los valores instantáneos de los sensores los pasos son los mismos.



Como se ve en el código HTML hace referencia a la dirección IP de Grafana. Esto es muy importante de entender, esta dirección puede variar (y de hecho lo hará) de equipo a equipo y red IP (son conocimientos que no se explicarán en el presente proyecto).

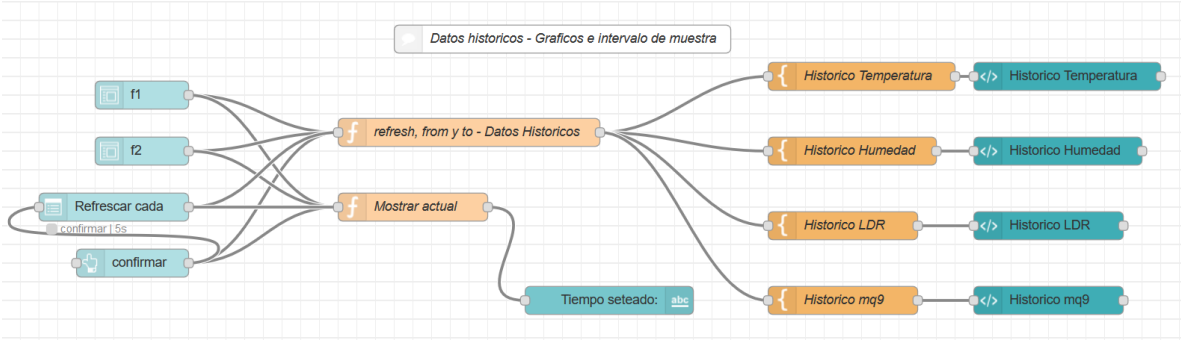
Pasamos a la segunda pestaña, de **Registros**, que podemos acceder desde

Registros históricos

Valores en función del tiempo e intervalo de muestra

Es interesante poder mostrar los gráficos de los datos históricos de los sensores en un intervalo de tiempo que consideremos de estudio. Para esto necesitamos dos formularios, que cada uno tendrá una fecha y una hora, y se corresponderán con el inicio y final de muestra. También es importante determinar el intervalo de refresco.

No profundizaremos en detalles en cuanto los formularios y botones para setear parámetros, pero si es importante entender como se actualizan las ventanas de tiempo en los gráficos.



En el nodo función **refresh, from y to - Datos históricos** se envía como parte de la salida un mensaje con la información de la ventana temporal de muestra. Para ser mas específicos, tomemos de ejemplo el grafico de temperatura ambiental. El código HTML:

```
<iframe src="http://192.168.20.129:3000/d-solo/CelyNFmgz/edicion_escenario1-edukit_2?refresh=10s&orgId=1&from=1553689931205&to=1553707931000&panelId=2" width="450" height="200" frameborder="0"></iframe>
```

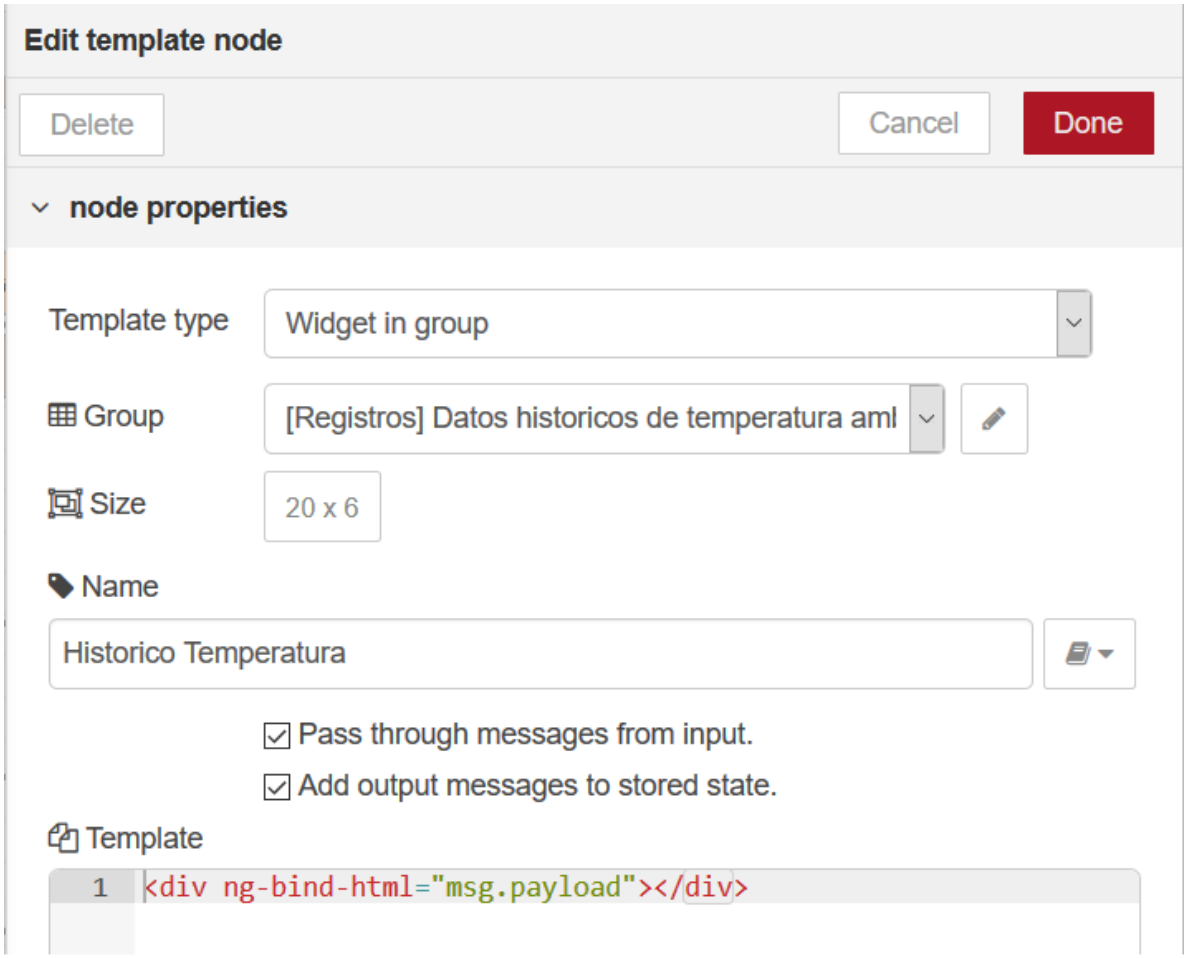
En la parte

```
refresh=10s&orgId=1&from=1553689931205&to=1553707931000
```

esta la información de interés. Los formularios y la función antes descripta modifica esto, y luego se actualiza nuestro gráfico.

El código completo se copia en el nodo **template** de color naranja:

En el último nodo **template** de color verde solo toma la carga del mensaje que sale del nodo anterior (msg.payload) y ahora si es mostrado en la web. Su configuración es la siguiente:



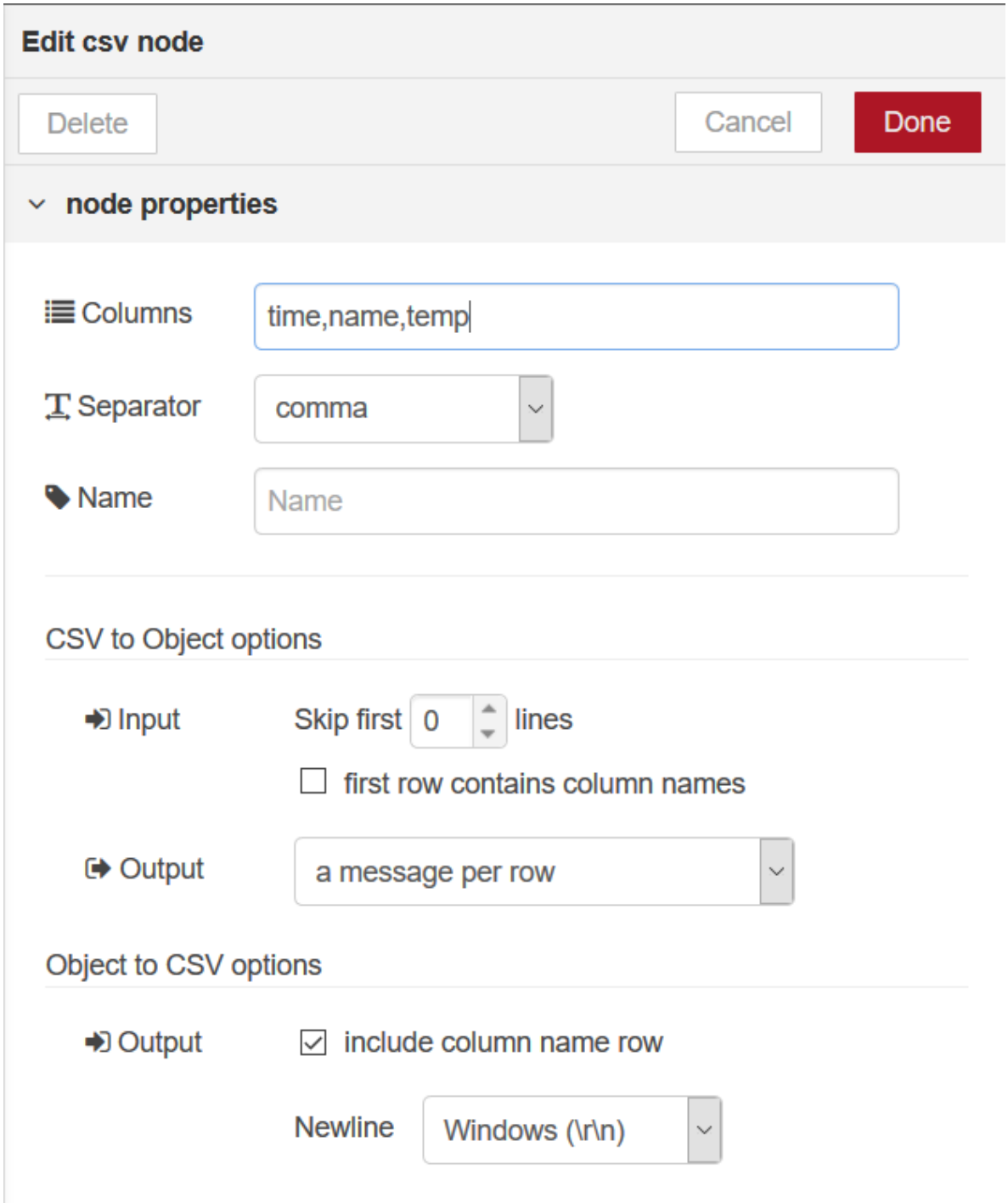
Exportando datos de los gráficos

Si quisiéramos exportar los datos que obtuvimos de los sensores para un posterior análisis se necesario hacer una consulta a la base de datos y no a Grafana.

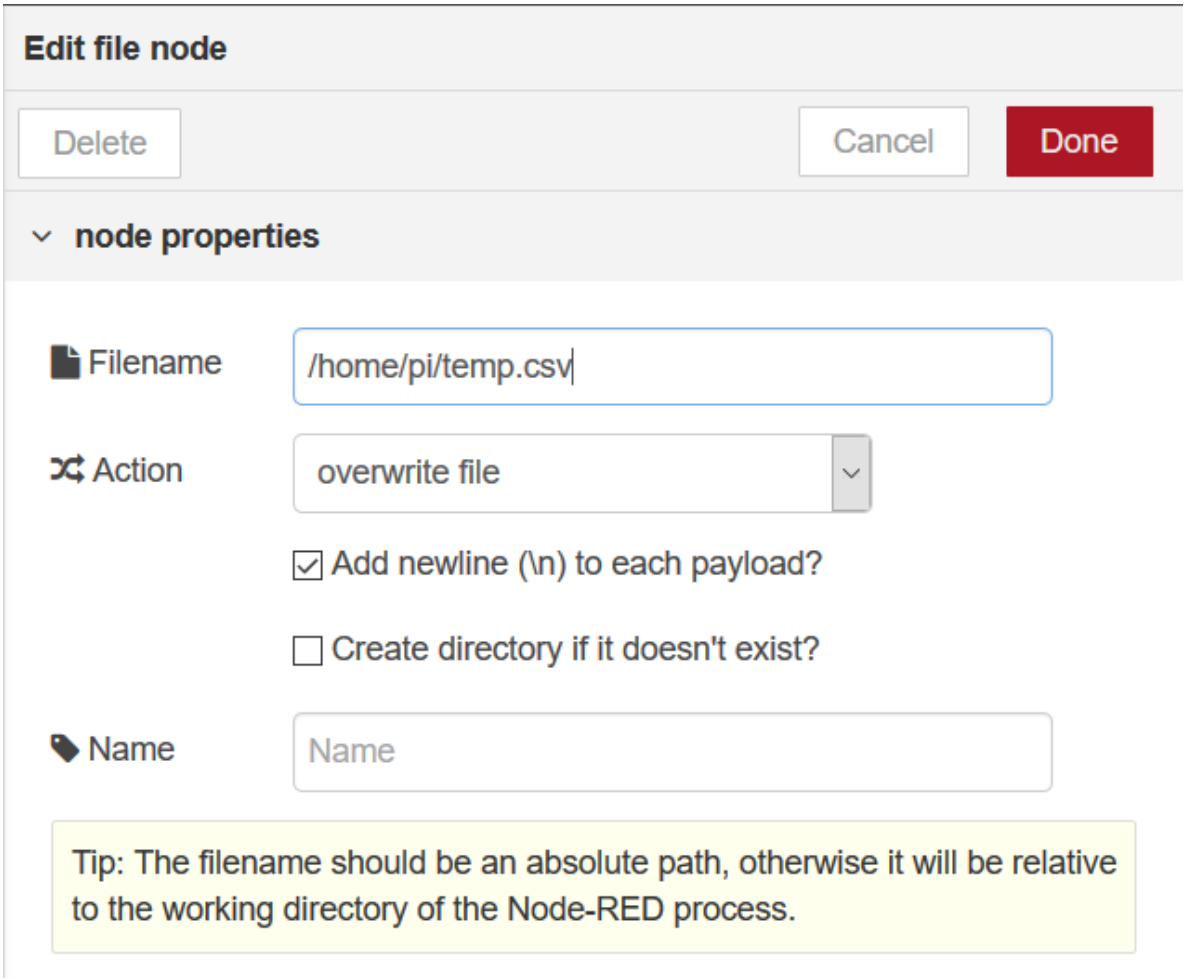
Para hacerlo con el nodo **date picker** se setea el día de inicio y de fin como ventana temporal para poder exportar los datos. Esta información no es más que estampas de tiempo, que luego en el nodo siguiente en una función se da el formato que necesita para hacer la consulta. La consulta en crudo a influxDB es:

```
SELECT "name","temp" FROM invernadero WHERE time > [estampa de tiempo de inicio] AND (time < [estampa de tiempo final])
```

La información devuelta por la base de datos se guarda en un archivo que poder defecto en el caso de la Raspberry Pi, en `/home/pi/[nombre-dato.csv]`. No es posible elegir le directorio ni el nombre, entonces si se quiere cambiar esta propiedad se debe acceder a los nodos **csv** de cada medición y en el nodo **file** también de cada medición. Nodo *csv*:


Edit csv node
Delete
Cancel
Done
node properties
Columns
time,name,temp
Separator
comma
Name
Name
CSV to Object options
Input
Skip first
0
lines
first row contains column names
Output
a message per row
Object to CSV options
Output
include column name row
Newline
Windows (\r\n)

Nodo *file*:



Un agregado interesante es que cada vez que se exporta una notificación en la web nos indica que el proceso se llevó a cabo con éxito. De lo contrario es probable que no se haya exportado la información.

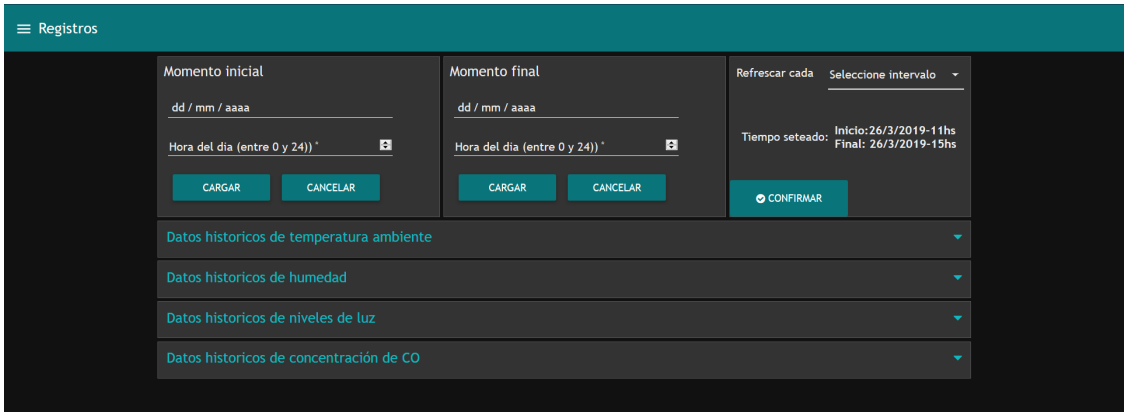
Visualización y resultados

Para acceder a la web y ver los resultados se debe ingresar con un navegador a http://localhost:1880/ui/. Cambiar `localhost` por la IP correspondiente si se quiere acceder desde un equipo remoto. La visualización de la web pestaña con datos instantáneos en un display con resolución 1920x1080 es la siguiente:

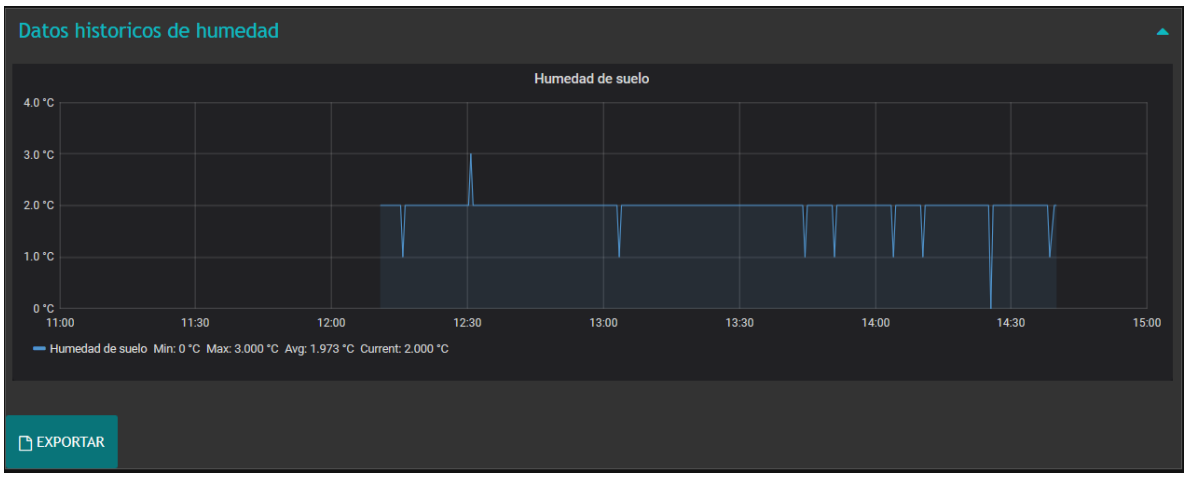
- Página principal:

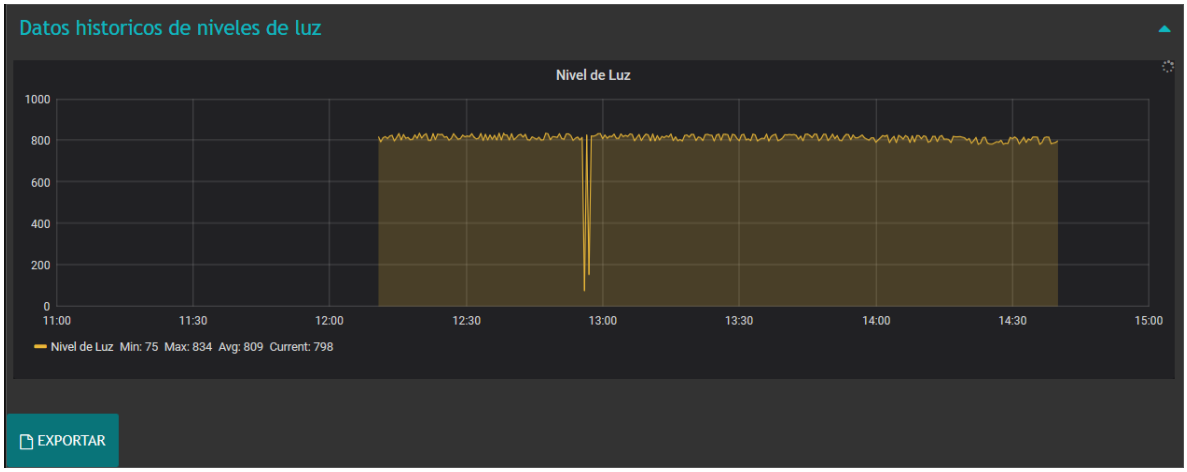
- Despliegue de pestañas:

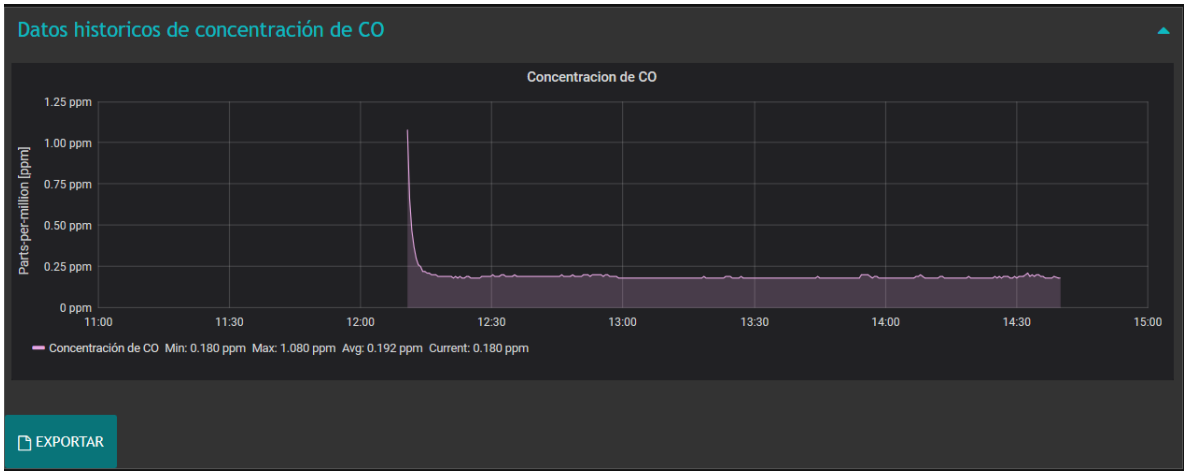
- Selección de ventana temporal para muestra de gráficos:



- Gráficos enventanado:







- Menú para exportar gráficos:

Configuraciones adicionales

Seguridad en Node-RED

Puede ser crítico que alguien acceda a nuestros nodos de Node-RED, pero se puede securizar con un usuario y contraseña, de manera que el flujo siga funcionando pero solamente le administrador pueda acceder y modificar cosas.

Para securizar Node-RED necesitamos primero descargar `node-red-admin`, que lo haremos desde el directorio donde están los archivos de Node-RED con la línea siguiente desde la terminal:

```
sudo npm install -g node-red-admin
```

Luego de la instalación debemos generar un hash para las password que nosotros deseemos. En nuestro caso usaremos `it10`.

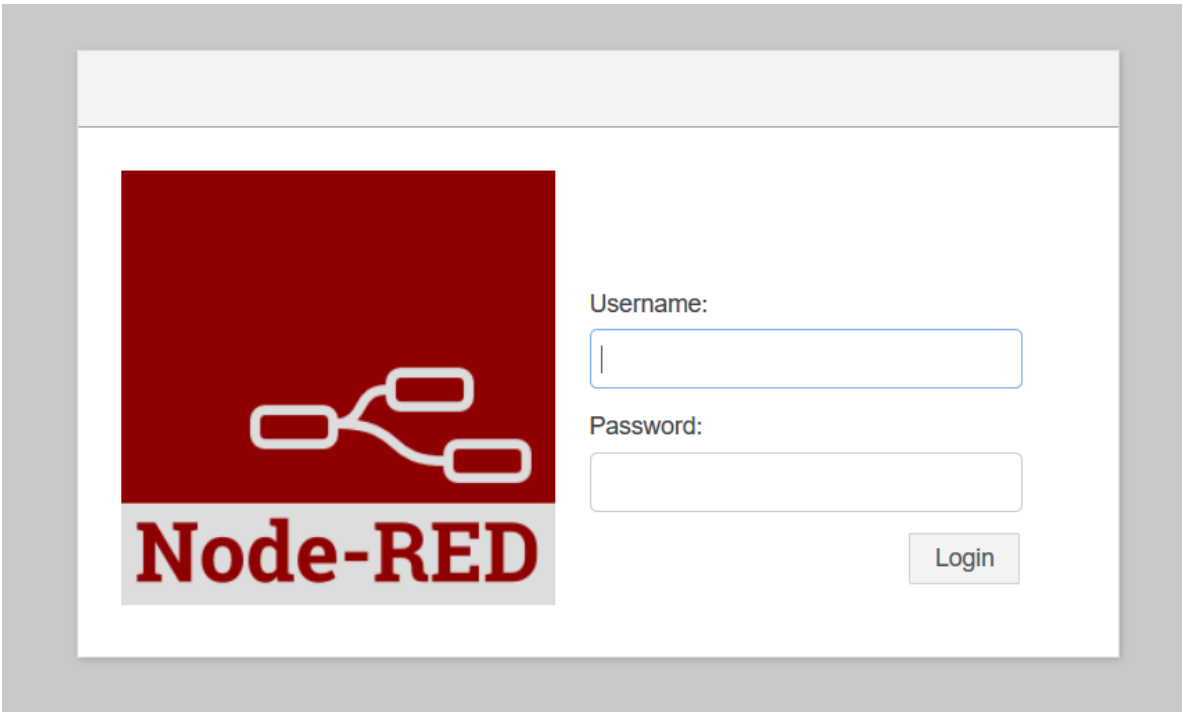
```
node-red-admin hash-pw
```

La herramienta nos pedirá la password, se ingresa, y nos devolverá el hash. Este mismo lo colocaremos en el archivo de opciones de Node-RED `settings.js`. Accediendo al archivo, se debe editar lo siguiente:

```
adminAuth: {
    type: "credentials",
    users: [{
        username: "[NOMBRE_USUARIO]",
        password: "[HASH_GENERADO]",
        permissions: "*"
    }]
}
```

El asterisco en *permissions* indica que tiene permisos de administrador. Se puede generar otro usuario con otro permiso, como de lectura (se pone *read* en ese campo).

Podemos ver que cada vez que accedemos a la web local nos pedirá identificarnos.



ESP8266EX Datasheet

Version 4.3

Espressif Systems IOT Team

http://bbs.espressif.com/

Disclaimer and Copyright Notice

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member Logo is a trademark of the WiFi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

Table of Contents

1. General Overview

1.1. Introduction

Espressif Systems' Smart Connectivity Platform (ESCP) is a set of high performance, high integration wireless SOCs, designed for space and power constrained mobile platform designers. It provides unsurpassed ability to embed WiFi capabilities within other systems, or to function as a standalone application, with the lowest cost, and minimal space requirement.

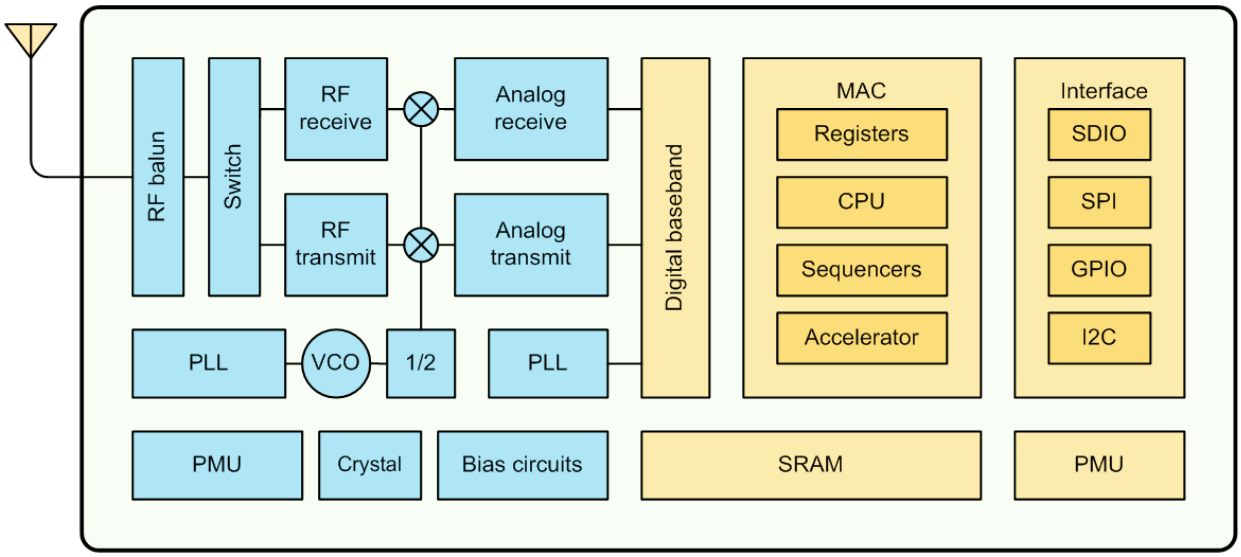


Figure 1 ESP8266EX Block Diagram

ESP8266EX offers a complete and self-contained WiFi networking solution; it can be used to host the application or to offload WiFi networking functions from another application processor.

When ESP8266EX hosts the application, it boots up directly from an external flash. In has integrated cache to improve the performance of the system in such applications.

Alternately, serving as a WiFi adapter, wireless internet access can be added to any micro controller-based design with simple connectivity (SPI/SDIO or I2C/UART interface).

ESP8266EX is among the most integrated WiFi chip in the industry; it integrates the antenna switches, RF balun, power amplifier, low noise receive amplifier, filters, power management modules, it requires minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor, with on-chip SRAM, besides the WiFi functionalities. ESP8266EX is often integrated with external sensors and other application specific devices through its GPIOs; sample codes for such applications are provided in the software development kit (SDK).

Espressif Systems' Smart Connectivity Platform (ESCP) demonstrates sophisticated system-level features include fast sleep/wake context switching for energy-efficient VoIP, adaptive radio biasing for low-power operation, advance signal processing, and spur cancellation and radio co-existence features for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

1.2. Features

- 802.11 b/g/n
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- WiFi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IR Remote Control, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4s guard interval
- Deep sleep power <10uA, Power down leakage current < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20 dBm output power in 802.11b mode
- Operating temperature range -40C ~ 125C
- FCC, CE, TELEC, WiFi Alliance, and SRRC certified

1.3. Parameters

Table 1 Parameters

Categories	Items	Values
WiFi Paramters	Certificates	FCC/CE/TELEC/SRRC
	WiFi Protocles	802.11 b/g/n
	Frequency Range	2.4G-2.5G (2400M-2483.5M)
	Tx Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
	Types of Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware Paramaters	Peripheral Bus	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/PWM
	Operating Voltage	3.0~3.6V
	Operating Current	Average value: 80mA
	Operating Temperature Range	-40°~125°
	Ambient Temperature Range	Normal temperature
	Package Size	5x5mm
	External Interface	N/A
Software Parameters	WiFi mode	station/softAP/SoftAP+station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Ssoftware Development	Supports Cloud Server Development / SDK for custom firmware development
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP

	User Configuration	AT Instruction Set, Cloud Server, Android/ iOS App

1.4. Ultra Low Power Technology

ESP8266EX has been designed for mobile, wearable electronics and Internet of Things applications with the aim of achieving the lowest power consumption with a combination of several proprietary techniques. The power saving architecture operates mainly in 3 modes: active mode, sleep mode and deep sleep mode.

By using advance power management techniques and logic to power-down functions not required and to control switching between sleep and active modes, ESP8266EX consumes about than 60uA in deep sleep mode (with RTC clock still running) and less than 1.0mA (DTIM=3) or less than 0.5mA (DTIM=10) to stay connected to the access point.

When in sleep mode, only the calibrated real-time clock and watchdog remains active. The real-time clock can be programmed to wake up the ESP8266EX at any required interval.

The ESP8266EX can be programmed to wake up when a specified condition is detected. This minimal wake-up time feature of the ESP8266EX can be utilized by mobile device SOCs, allowing them to remain in the low-power standby mode until WiFi is needed.

In order to satisfy the power demand of mobile and wearable electronics, ESP8266EX can be programmed to reduce the output power of the PA to fit various application profiles, by trading off range for power consumption.

1.5. Major Applications

Major fields of ESP8266EX applications to Internet-of-Things include:

- Home Appliances
- Home Automation
- Smart Plug and lights
- Mesh Network
- Industrial Wireless Control
- Baby Monitors
- IP Cameras
- Sensor Networks
- Wearable Electronics

- WiFi Location-aware Devices
- Security ID Tags
- WiFi Position System Beacons

2. Hardware Overview

2.1. Pin Definitions

The pin assignments for 32-pin QFN package is illustrated in Fig.2.

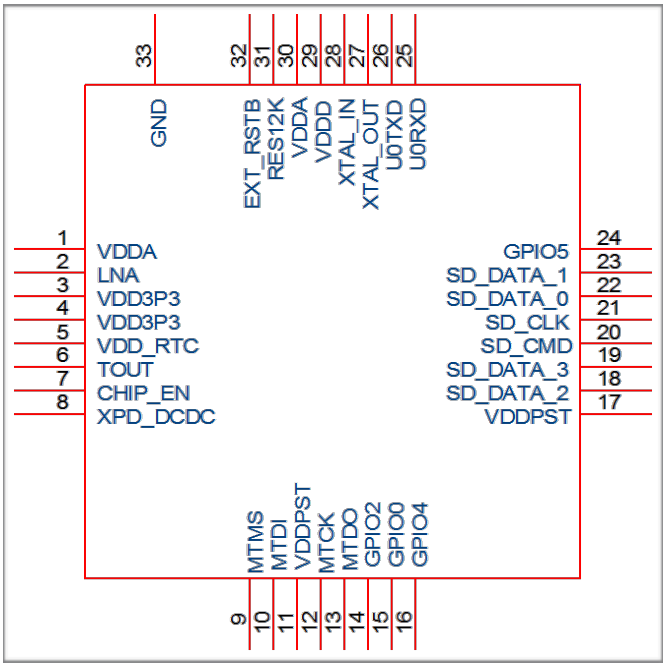


Figure 2 Pin Assignments

Table 2 below presents an overview on the general pin attributes and the functions of each pin.

Table 2 Pin Definitions

Pin	Name	Type	Function
1	VDDA	P	Analog Power 3.0 ~3.6V
2	LNA	I/O	RF Antenna Interface. Chip Output Impedance=50Ω No matching required but we recommend that the π-type matching network is retained.
3	VDD3P3	P	Amplifier Power 3.0~3.6V
4	VDD3P3	P	Amplifier Power 3.0~3.6V
5	VDD_RTC	P	NC (1.1V)

6	TOUT	I	ADC Pin (note: an internal pin of the chip) can be used to check the power voltage of VDD3P3 (Pin 3 and Pin4) or the input voltage of TOUT (Pin 6). These two functions cannot be used simultaneously.
7	CHIP_EN	I	Chip Enable. High: On, chip works properly; Low: Off, small current
8	XPD_DCDC	I/O	Deep-Sleep Wakeup； GPIO16
9	MTMS	I/O	GPIO14; HSPI_CLK
10	MTDI	I/O	GPIO12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8V~3.3V)
12	MTCK	I/O	GPIO13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART Tx during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8V~3.3V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 200Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200Ω); SPI_MSIO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART Tx during flash progamming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 3.0V~3.6V
30	VDDA	P	Analog Power 3.0V~3.6V
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: Active)

Note: GPIO2, GPIO0, MTDO can be configurable as 3-bit SDIO mode.

2.2. Electrical Characteristics

Table 3 ESP8266EX Electrical Characteristics

Parameters		Conditions	Min	Typical	Max	Unit
Storage Temperature Range			-40	Normal	125	°C
Maximum Soldering Temperature		IPC/JEDEC J-STD-020			260	°C
Working Voltage Value			3.0	3.3	3.6	V
I/O	V_{IL}/V_{IH}		-0.3/0.75V_{IO}		0.25V_{IO}/3.6	V
	V_{OL}/V_{OH}		N/0.8V_{IO}		0.1V_{IO}/N	
	I_{MAX}				12	mA
Electrostatic Discharge (HBM)		TAMB=25°C			2	KV
Electrostatic Discharge (CDM)		TAMB=25°C			0.5	KV

2.3. Power Consumption

The following current consumption is based on 3.3V supply, and 25°C ambient, using internal regulators. Measurements are done at antenna port without SAW filter. All the transmitter's measurements are based on 90% duty cycle, continuous transmit mode.

Table 4 Description on Power Consumption

Parameters	Min	Typical	Max	Unit
Tx802.11b, CCK 11Mbps, P OUT=+17dBm		170		mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm		140		mA
Tx 802.11n, MCS7, P OUT =+13dBm		120		mA
Rx 802.11b, 1024 bytes packet length , -80dBm		50		mA
Rx 802.11g, 1024 bytes packet length, -70dBm		56		mA
Rx 802.11n, 1024 bytes packet length, -65dBm		56		mA
Modem-Sleep①		15		mA
Light-Sleep②		0.9		mA
Deep-Sleep③		10		uA
Power Off		0.5		uA

①: Modem-Sleep requires the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it saves power to shut down the WiFi Modem circuit while maintaining a WiFi connection with no data transmission. E.g. in DTIM3, to maintain a sleep 300ms-wake 3ms cycle to receive AP's Beacon packages, the current is about 15mA

②: During Light-Sleep, the CPU may be suspended in applications like WiFi switch. Without data transmission, the WiFi Modem circuit can be turned off and CPU suspended to save power according to the 802.11 standard (U-APSD). E.g. in DTIM3, to maintain a sleep 300ms-wake 3ms cycle to receive AP's Beacon packages, the current is about 0.9mA.

③: Deep-Sleep does not require WiFi connection to be maintained. For application with long time lags between data transmission, e.g. a temperature sensor that checks the temperature every 100s, sleep 300s and waking up to connect to the AP (taking about 0.3~1s), the overall average current is less than 1mA.

2.4. Receiver Sensitivity

The following are measured under room temperature conditions with 3.3V and 1.1V power supplies.

Table 5 Receiver Sensitivity

Parameters	Min	Typical	Max	Unit
Input frequency	2412		2484	MHz
Input impedance		50		Ω
Input reflection			-10	dB
Output power of PA for 72.2Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
6Mbps (1/2 BPSK)		-93		dBm
54Mbps (3/4 64-QAM)		-75		dBm
HT20, MCS7 (65Mbps, 72.2Mbps)		-72		dBm
Adjacent Channel Rejection				
OFDM, 6Mbps		37		dB
OFDM, 54Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB

2.5. MCU

ESP8266EX is embedded with Tensilica L106 32-bit micro controller (MCU), which features extra low power consumption and 16-bit RSIC. The CPU clock speed is 80MHz. It can also reach a maximum value of 160MHz. Real Time Operation System (RTOS) is enabled. Currently, only 20% of MIPS has been occupied by the WiFi stack, the rest can all be used for user application programming and development. The following interfaces can be used to connect to the MCU embedded in ESP8266EX:

- Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can also be used to visit external flash;
- Data RAM interface (dBus), which can connected with memory controller;
- AHB interface, can be used to visit the register.

2.6. Memory Organization

2.6.1. Internal SRAM and ROM

ESP8266EX WiFi SoC is embedded with memory controller, including SRAM and ROM. MCU can visit the memory units through iBus, dBus, and AHB interfaces. All memory units can be visited upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK provided, SRAM space that is available to users is assigned as below:

- **RAM size < 36kB,** that is to say, when ESP8266EX is working under the station mode and is connected to the router, programmable space accessible to user in heap and data section is around 36kB.)
- There is no programmable ROM in the SoC, therefore, user program must be stored in an external SPI flash.

2.6.2. External SPI Flash

An external SPI flash is used together with ESP8266EX to store user programs. Theoretically speaking, up to 16 Mbyte memory capacity can be supported.

Suggested SPI Flash memory capacity:

- OTA is disabled: the minimum flash memory that can be supported is 512 kByte;
- OTA is enabled: the minimum flash memory that can be supported is 1 Mbyte.

Several SPI modes can be supported, including Standard SPI, Dual SPI, DIO SPI, QIO SPI, and Quad SPI.

Therefore, please choose the correct SPI mode when you are downloading into the flash, otherwise firmwares/programs that you downloaded may not work in the right way.

2.7. AHB and AHB Blocks

The AHB blocks performs the function of an arbiter, controls the AHB interfaces from the MAC, SDIO (host) and CPU. Depending on the address, the AHB data requests can go into one of the two slaves:

APB block, or

flash controller (usually for standalone applications).

Data requests to the memory controller are usually high speed requests, and requests to the APB block are usually register access.

The APB block acts as a decoder. It is meant only for access to programmable registers within ESP8266's main blocks. Depending on the address, the APB request can go to the radio, SI/SPI, SDIO (host), GPIO, UART, real-time clock (RTC), MAC or digital baseband.

3. Pins and Definitions

The chipset encapsulates variable analog and data transmission I/Os, descriptions and definitions of which are explained below in detail.

3.1. GPIO

3.1.1. General Purpose Input/Output Interface (GPIO)

There are up to 17 GPIO pins. They can be assigned to various functions by the firmware. Each GPIO can be configured with internal pull-up (except XPD_DCDC, which is configured with internal pull-down), input available for sampling by a software register, input triggering an edge or level CPU interrupt, input triggering a level wakeup interrupt, open-drain or push-pull output driver, or output source from a software register, or a sigma-delta PWM DAC.

These pins are multiplexed with other functions such as I2C, I2S, UART, PWM, IR Remote Control, etc.

Data I/O soldering pad is bidirectional and tri-state that include data input and output controlling buffer. Besides, I/O can be set as a specific state and remains like this. For example, if you intend to lower the power consumption of the chip, all data input and output enable signals can be set as remaining low power state. You can transport some specific state into the I/O. When the I/O is not powered by external circuits, the I/O will remain to be the state that it was used the last time. Some positive feedback is generated by the state-remaining function of the pins, therefore, if the external driving power must be stronger than the positive feedback. Even so, the driving power that is needed is within 5uA.

Table 6 Pin Definitions of GPIOs

Variables	Symbol	Min	Max	Unit
Input Low Voltage	V_{IL}	-0.3	$0.25 \times V_{IO}$	V
Input High Voltage	V_{IH}	$0.75 \times V_{IO}$	3.3	V
Input Leakage Current	I_{IL}		50	nA
Output Low Voltage	V_{OL}		$0.1 \times V_{IO}$	V
Output High Voltage	V_{OH}	$0.8 \times V_{IO}$		V
Input Pin Resistance Value	C_{pad}		2	pF
VDDIO	V_{IO}	1.8	3.3	V
Maximum Driving Power	I_{MAX}		12	mA
Temerpature	T_{amb}	-40	125	°C

All digital IO pins are protected from over-voltage with a snap-back circuit connected between the pad and ground. The snap back voltage is typically about 6V, and the holding voltage is 5.8V. This

provides protection from over-voltages and ESD. The output devices are also protected from reversed voltages with diodes.

3.2. Secure Digital Input/Output Interface (SDIO)

One Slave SDIO has been defined by ESP8266EX, the definitions of which are described in Table 7 below. 4bit 25MHz SDIO v1.1 and 4bit 50MHz SDIO v2.0 are supported.

Table 7 Pin Definitions of SDIOs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SDIO_CLK
SDIO_DATA0	22	IO7	SDIO_DATA0
SDIO_DATA1	23	IO8	SDIO_DATA1
SDIO_DATA_2	18	IO9	SDIO_DATA_2
SDIO_DATA_3	19	IO10	SDIO_DATA_3
SDIO_CMD	20	IO11	SDIO_CMD

3.3. Serial Peripheral Interface (SPI/HSPI)

Currently, one general Slave/Master SPI, one Slave SDID/SPI, and one general Slave/Master HSPI have been defined by ESP8266EX. Functions of all these pins can be implemented via hardware. The pin definitions are are described below:

3.3.1. General SPI (Master/Slave)

Table 8 Pin Definitions of General SPIs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SPICLK
SDIO_DATA0	22	IO7	SPIQ/MISO
SDIO_DATA1	23	IO8	SPID/MOSI
SDIO_DATA_2	18	IO9	SPIHD
SDIO_DATA_3	19	IO10	SPIWP
SDIO_CMD	20	IO11	SPICS0
U0TXD	26	IO1	SPICS1
GPIO0	15	IO0	SPICS2

3.3.2. SDIO / SPI (Slave)

Table 9 Pin Definitions of SDIO / SPI (Slave)

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SPI_SLAVE_CLK
SDIO_DATA0	22	IO7	SPI_SLAVE_MISO
SDIO_DATA1	23	IO8	SPI_SLAVE_INT
SDIO_DATA_2	18	IO9	NC
SDIO_DATA_3	19	IO10	SPI_SLAVE_CS
SDIO_CMD	20	IO11	SPI_SLAVE_MOSI

3.3.3. HSPI (Master/Slave)

Table 10 Pin Definitions of HSPI (Master/Slave)

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	HSPICLK
MTDI	10	IO12	HSPIQ/MISO
MTCK	12	IO13	HSPID/MOSI
MTDO	13	IO15	HPSICS

Note:

- SPI mode can be implemented via software programming. The clock frequency can reach up to a maximum value of 80MHz.
- Function of Slave SDIO/SPI interface can be implemented via hardware, and linked list DMA (Direct Memory Access) is supported, software overheads are smaller. However, there is no linked list DMA on general SPI and HSPI, and the software overheads are larger, therefore, the data transmitting speed will be restrained by software processing speed.

3.4. Inter-integrated Circuit Interface (I2C)

One I2C, which is mainly used to connect with micro controller and other peripheral equipment such as sensors, is defined by ESP8266EX. The present pin definition of I2C is as defined below:

Table 11 Pin Definitions of I2C

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	I2C_SCL
GPIO2	14	IO2	I2C_SDA

Both I2C-Master and I2C-Slave are supported. I2C interface functionality can be realized via software programming, the clock frequency can be up to around 100KHz at most. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

3.5. I2S

Currently one I2S data input interface and one I2S data output interface are defined. I2S interface is mainly used in applications such as data collection, processing, and transmission of audio data, as well as the input and output of serial data. For example, LED lights (WS2812 series) are supported. The pin definition of I2S is as defined below:

Table 12 Pin Definitions of I2S

I2S Data Input:			
Pin Name	**Pin Num**	**IO**	**Function Name**
MTDI	10	IO12	I2SI_DATA
MTCK	12	IO13	I2SI_BCK
MTMS	9	IO14	I2SI_WS
I2S Data Output:			
Pin Name	**Pin Num**	**IO**	**Function Name**
MTDO	13	IO15	I2SO_BCK
U0RXD	25	IO3	I2SO_DATA
GPIO2	14	IO2	I2SO_WS

I2S functionality can be realized via software programming, the GPIOs that will be used are multiplexed, and linked list DMA is supported.

3.6. Universal Asynchronous Receiver Transmitter (UART)

Two UART interfaces, UART0 and UART1, have been defined by ESP8266EX, the definitions are as below:

Table 13 Pin Definitions of UART Interfaces

Pin Type	Pin Name	Pin Num	IO	Function Name
UART0	U0RXD	25	IO3	U0RXD
	U0TXD	26	IO1	U0TXD
	MTDO	13	IO15	U0RTS
	MTCK	12	IO13	U0CTS
UART1	GPIO2	14	IO2	U1TXD
	SD_D1	23	IO8	U1RXD

Data transfers to/from UART interfaces can be implemented via hardware. The data transmission speed via UART interfaces can reach 115200*40 (4.5Mbps).

UART0 can be for communication. It supports fluid control. Since UART1 features only data transmit signal (Tx), it is usually used for printing log.

Notes: By default, UART0 will output some printed information when the device is powered on and is booting up. The baud rate of the printed information is closely related to the frequency of the external crystal oscillator. If the frequency of the crystal oscillator is 40MHz, then the baud rate for printing is 115200; if the frequency of the crystal oscillator is 26MHz, then the baud rate for printing is 74880. If the printed information exerts any influence on the functionality of your device, you'd better block the printing during the power-on period by changing (`U0TXD,U0RXD`) to (`MTDO,MTCK`).

3.7. Pulse-Width Modulation (PWM)

Four PWM output interfaces have been defined by ESP8266EX. They can be extended by users themselves. The present pin definitions of the PWM interfaces are defined as below:

Table 14 Pin Definitions of PWM Interfaces

Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	PWM0
MTDO	13	IO15	PWM1
MTMS	9	IO14	PWM2
GPIO4	16	IO4	PWM3

The functionality of PWM interfaces can be implemented via software programming. For example, in the LED smart light demo, the function of PWM is realized by interruption of the timer, the minimum resolution can reach as much as 44 ns. PWM frequency range is adjustable from 1000 us to 10000 us,

i.e., between 100Hz and 1KHz. When the PWM frequency is at 1 KHz, the duty ratio will reach 1/22727, and over 14 bit resolution will be achieved at 1KHz refresh rate.

3.8. IR Remote Control

Currently, only one Infrared remote control interface is defined, the pin definition is as below:

Table 14 Pin Definition of IR Remote Control

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO12	IR Tx
GPIO5	24	IO5	IR Rx

The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are used by this interface. The frequency of modulated carrier signal is 38KHz, while the duty ratio of the square wave is 1/3. The length of data transmission, which is around 1m, is determined by two factors: one is the maximum value of rated current, the other is internal current-limiting resistance value in the infrared receiver. The larger the resistance value, the lower the current, so is the power, and vice versa. The transmission angle is between 15° and 30°, and is mainly determined by the radiation direction of the infrared receiver.

Notes: Among the eight interfaces mentioned above, most of them can be multiplexed. Pin definitions that can be defined is not limited to the eight ones herein mentioned, customers can self customise the functions of the pins according to their specific application scenarios. Functions of these pins can be implemented via software programming and hardware.

3.9. ADC (Analog-to-digital Converter)

ESP8266EX is embedded with a 10-bit precision SARADC. Currently, TOUT (Pin6) is defined as ADC interface, the definition of which is described below:

Pin Name	Pin Num	Function Name
TOUT	6	ADC Interface

Table 16 Pin Definition of ADC

The following two applications can be implemented using ADC (Pin6). However, these two applications cannot be implemented concurrently.

- Test the power supply voltage of VDD3P3 (Pin 3 and Pin 4).

The function used to test the power supply voltage on PA_VDD pin is: `uint16 system_get_vdd33(void)`

- Test the input voltage of TOUT (Pin 6):

The function used to test the input voltage of TOUT is: `uint16 system_adc_read(void)`

`RF_init` parameter in the following passage refers to `esp_init_data_default.bin`

Application One:	Test the power supply voltage of VDD3P3 (Pin 3 and Pin 4).
Hardware Design:	TOUT must be dangled.
`RF_init` Parameter:	The 107th byte of `esp_init_data_default.bin` (0 - 127 byte), "vdd33_const", must set to be 0xFF, i.e., the value of "vdd33_const" is 255.
RF Calibration Process:	Optimize the RF circuit conditions based on the testing results of VDD3P3 (Pin 3 and Pin 4).
User Programming:	Use `system_get_vdd33` instead of `system_adc_read`.

Application Two:	Test the input voltage of TOUT (Pin 6).
Hardware Design:	The input voltage range is 0 to 1.0 V when TOUT is connected to external circuit.
`RF_init` Parameter:	The value of the 107th byte of `esp_init_data_default.bin` (0 - 127 byte), "vdd33_const", must be set to be the real power supply voltage of Pin 3 and Pin 4. The working power voltage range of ESP8266EX is between 1.8V and 3.6V, while the unit of "vdd33_const" is 0.1V, therefore, the effective value range of "vdd33_const" is 18 to 36.
RF Calibration Process:	Optimize the RF circuit conditions based on the value of "vdd33_const". The permissible error is ±0.2V.
User Programming:	Use `system_adc_read` instead of `system_get_vdd33`.

Note One:

In `RF_init` parameter `esp_init_data_default.bin` (0 - 127 byte), the 107th byte is defined as "vdd33_const". Definitions of "vdd33_const" is described below:

（1）If vdd33_const = 0xff, the power voltage of Pin 3 and Pin 4 will be tested by the internal self-calibration process of ESP8266EX chipset itself. RF circuit conditions should be optimized according to the testing results.

（2）If 18 =< vdd33_const =< 36, ESP8266EX RF Calibration and optimization process is implemented via (vdd33_const/10).

（3）If vdd33_const < 18 or 36 < vdd33_const < 255, ESP8266EX RF Calibration and optimization process is implemented via the default value 3.0V.

Note Two:

Function `system_get_vdd33` is used to test the power supply voltage of VDD3P3 (Pin 3 and Pin 4). Details on this function are described below:

（1）Pin Tout must be dangled. The 107th byte of `esp_init_data_default.bin` (0 - 127 byte), "vdd33_const", must set to be 0xFF.

（2）If the 107th byte of `esp_init_data_default.bin` (0 - 127 byte), "vdd33_const", is equal to `0xff`, the returned value of function `system_get_vdd33` will be an effective value, otherwise `0xffff` will be returned.

（3）The unit of the returned value is: 1/1024 V.

Note Three:

Function system_adc_read is defined to test the input voltage of Pin TOUT (Pin 6). Details on this function are described below:

（1）The value of the 107th byte of `esp_init_data_default.bin` (0 - 127 byte), "vdd33_const", must be set to be the real power supply voltage of Pin 3 and Pin 4.

（2）If the 107th byte of `esp_init_data_default.bin` (0 - 127 byte), "vdd33_const", is NOT equal to `0xff`, the returned value of `system_adc_read` will be an effective value of the input voltage of Pin TOUT, otherwise `0xffff` will be returned.

（3）The unit of the returned value is: 1/1024 V.

3.10. LED Light and Button

ESP8266EX features up to 17 GPIOs, all of which can be assigned to realise various functions of LED lights and buttons. Definitions of some GPIOs that are assigned with certain functions in our demo application design are shown below:

Table 17 Pin Definitions of LED and Button

Pin Name	Pin Num	IO	Function Name
MTCK	12	IO13	Button (Reset)
GPIO0	15	IO0	WiFi Light
MTDI	10	IO12	Link Light

Altogether three interfaces have been defined, one is for the button, and the other two is for LED light. Generally, **MTCK** is used to control the reset button, **GPIO0** is used as an signal to indicate the WiFi working state, **MTDI** is used as a signal light to indicate communication between the device and the server.

Note: Among the nine interfaces mentioned above, most of them can be multiplexed. Pin definitions that can be defined is not limited to the eight ones herein mentioned, customers can self customise the functions of the pins according to their specific application scenarios. Functions of these pins can be implemented via software programming and hardware.

4. Firmware & Software Development Kit

The application and firmware is executed in on-chip ROM and SRAM, which loads the instructions during wake-up, through the SDIO interface, from the external flash.

The firmware implements TCP/IP, the full 802.11 b/g/n/e/i WLAN MAC protocol and WiFi Direct specification. It supports not only basic service set (BSS) operations under the distributed control function (DCF) but also P2P group operation compliant with the latest WiFi P2P protocol. Low level protocol functions are handled automatically by ESP8266:

- RTS/CTS
- acknowledgement
- fragmentation and defragmentation
- aggregation
- frame encapsulation (802.11h/RFC 1042)
- automatic beacon monitoring / scanning, and
- P2P WiFi direct

Passive or active scanning, as well as P2P discovery procedure is performed autonomously once initiated by the appropriate command. Power management is handled with minimum host interaction to minimize active duty period.

4.1. Features

The SDK includes the following library functions:

- 802.11 b/g/n/d/e/i/k/r support;
- WiFi Direct (P2P) support:
- P2P Discovery, P2P Group Owner mode, P2P Power Management
- Infrastructure BSS Station mode / P2P mode / softAP mode support;
- Hardware accelerators for CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WAPI (SMS4), WEP (RC4), CRC;
- WPA/WPA2 PSK, and WPS driver;
- Additional 802.11i security features such as pre-authentication, and TSN;
- Open Interface for various upper layer authentication schemes over EAP such as TLS, PEAP, LEAP, SIM, AKA, or customer specific;
- 802.11n support (2.4GHz);
- Supports MIMO 1×1 and 2×1, STBC, A-MPDU and A-MSDU aggregation and 0.4μs guard interval;

- WMM power save U-APSD;
- Multiple queue management to fully utilize traffic prioritization defined by 802.11e standard;
- UMA compliant and certified;
- 802.1h/RFC1042 frame encapsulation;
- Scattered DMA for optimal CPU off load on Zero Copy data transfer operations;
- Antenna diversity and selection (software managed hardware);
- Clock/power gating combined with 802.11-compliant power management dynamically adapted to current connection condition providing minimal power consumption;
- Adaptive rate fallback algorithm sets the optimum transmission rate and Tx power based on actual SNR and packet loss information;
- Automatic retransmission and response on MAC to avoid packet discarding on slow host environment;
- Seamless roaming support;
- Configurable packet traffic arbitration (PTA) with dedicated slave processor based design provides flexible and exact timing Bluetooth co-existence support for a wide range of Bluetooth Chip vendors;
- Dual and single antenna Bluetooth co-existence support with optional simultaneous receive (WiFi/Bluetooth) capability.

5. Power Management

The chip can be put into the following states:

- **OFF**: CHIP_PD pin is low. The RTC is disabled. All registers are cleared.
- **DEEP_SLEEP**: Only RTC is powered on – the rest of the chip is powered off. Recovery memory of RTC can keep basic WiFi connecting information.
- **SLEEP**: Only the RTC is operating. The crystal oscillator is disabled. Any wakeup events (MAC, host, RTC timer, external interrupts) will put the chip into the WAKEUP state.
- **WAKEUP**: In this state, the system goes from the sleep states to the PWR state. The crystal oscillator and PLLs are enabled.
- **ON**: the high speed clock is operational and sent to each block enabled by the clock control register. Lower level clock gating is implemented at the block level, including the CPU, which can be gated off using the WAITI instruction, while the system is on.

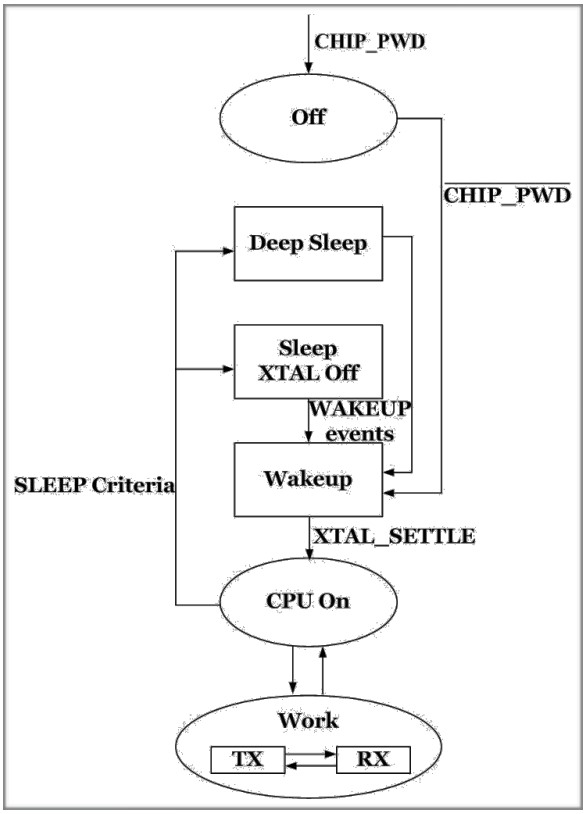


Figure 3 Illustration of Power Management

6. Clock Management

6.1. High Frequency Clock

The high frequency clock on ESP8266EX is used to drive both transmit and receive mixers. This clock is generated from the internal crystal oscillator and an external crystal. The crystal frequency can range from 26MHz to 52MHz.

While internal calibration of the crystal oscillator ensures that a wide range of crystals can be used, in general, the quality of the crystal is still a factor to consider, to have reasonable phase noise that is required for good performance. When the crystal selected is sub-optimal due to large frequency drifts or poor Q-factor, the maximum throughput and sensitivity of the WiFi system is degraded. Please refer to the application notes on how the frequency offset can be measured.

Table 18 High Frequency Clock

Parameter	Symbol	Min	Max	Unit
Frequency	FXO	26	52	MHz
Loading capacitance	CL		32	pF
Motional capacitance	CM	2	5	pF
Series resistance	RS	0	65	Ω
Frequency tolerance	ΔFXO	-15	15	ppm
Frequency vs temperature (-25°C ~ 75°C)	ΔFXO,Temp	-15	15	ppm

6.2. External Reference Requirements

For an externally generated clock, the frequency can range from 26MHz to 52MHz can be used. For good performance of the radio, the following characteristics are expected of the clock:

Table 19 External Clock Reference

Parameter	Symbol	Min	Max	Unit
Clock amplitude	VXO	0.2	1	Vpp
External clock accuracy	ΔFXO,EXT	-15	15	ppm
Phase noise @1kHz offset, 40MHz clock			-120	dBc/Hz
Phase noise @10kHz offset, 40MHz clock			-130	dBc/Hz
Phase noise @100kHz offset, 40MHz clock			-138	dBc/Hz

7. Radio

The ESP8266EX radio consists of the following main blocks:

- 2.4GHz receiver
- 2.4GHz transmitter
- High speed clock generators and crystal oscillator
- Real time clock
- Bias and regulators
- Power management

7.1. Channel Frequencies

The RF transceiver supports the following channels according to the IEEE802.11b/g/n standards.

Table 20 Frequency Channel

Channel No	Frequency (MHz)	Channel No	Frequency (MHz)
1	2412	8	2447
2	2417	9	2452
3	2422	10	2457
4	2427	11	2462
5	2432	12	2467
6	2437	13	2472
7	2442	14	2484

7.2. 2.4 GHz Receiver

The 2.4GHz receiver downconverts the RF signal to quadrature baseband signals and converts them to the digital domain with 2 high resolution high speed ADCs. To adapt to varying signal channel conditions, RF filters, automatic gain control (AGC), DC offset cancelation circuits and baseband filters are integrated within ESP8266EX.

7.3. 2.4 GHz Transmitter

The 2.4GHz transmitter up-converts the quadrature baseband signals to 2.4GHz, and drives the antenna with a high powered CMOS power amplifier. The use of digital calibration further improves the linearity of the power amplifier, enabling a state of art performance of delivering +19.5dBm average power for 802.11b transmission and +16dBm for 802.11n transmission.

Additional calibrations are integrated to cancel any imperfections of the radio, such as:

- carrier leakage,
- I/Q phase matching, and
- baseband nonlinearities

This reduces the amount of time required and test equipment required for production testing.

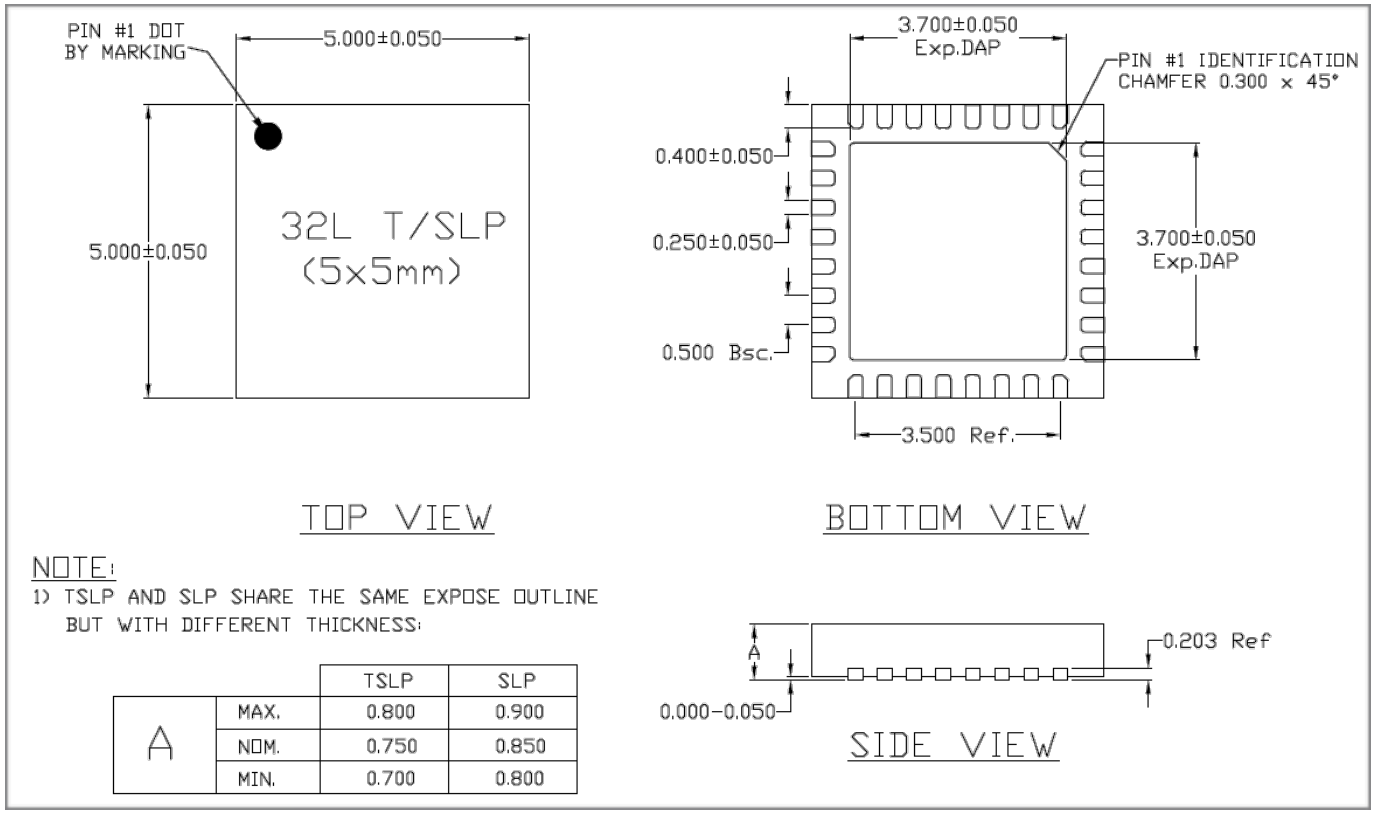
7.4. Clock Generator

The clock generator generates quadrature 2.4 GHz clock signals for the receiver and transmitter. All components of the clock generator are integrated on-chip, including:

- inductor,
- varactor, and
- loop filter

The clock generator has built-in calibration and self test circuits. Quadrature clock phases and phase noise are optimized on-chip with patented calibration algorithms to ensure the best receiver and transmitter performance.

8. Appendix: QFN32 Package Size



		TSLP	SLP
A	MAX.	0.800	0.900
	NOM.	0.750	0.850
	MIN.	0.700	0.800

SUNROM TECHNOLOGIES
Your Source for Embedded Systems

Email: info@sunrom.com or sunrom@gmail.com

Visit us at http://www.sunrom.com

Document: Datasheet Date: 28-Jul-08 Model #: 3190 Product's Page: www.sunrom.com/p-510.html

Light Dependent Resistor - LDR

Two cadmium sulphide(cds) photoconductive cells with spectral responses similar to that of the human eye. The cell resistance falls with increasing light intensity. Applications include smoke detection, automatic lighting control, batch counting and burglar alarm systems.

Applications

Photoconductive cells are used in many different types of circuits and applications.

Analog Applications

- Camera Exposure Control
- Auto Slide Focus - dual cell
- Photocopy Machines - density of toner
- Colorimetric Test Equipment
- Densitometer
- Electronic Scales - dual cell
- Automatic Gain Control – modulated light source
- Automated Rear View Mirror

Digital Applications

- Automatic Headlight Dimmer
- Night Light Control
- Oil Burner Flame Out
- Street Light Control
- Absence / Presence (beam breaker)
- Position Sensor

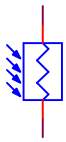
Electrical Characteristics

Parameter	Conditions	Min	Typ	Max	Unit
Cell resistance	1000 LUX 10 LUX	- -	400 9	- -	Ohm K Ohm
Dark Resistance	-	-	1	-	M Ohm
Dark Capacitance	-	-	3.5	-	pF
Rise Time	1000 LUX 10 LUX	- -	2.8 18	- -	ms ms
Fall Time	1000 LUX 10 LUX	- -	48 120	- -	ms ms
Voltage AC/DC Peak		-	-	320	V max
Current		-	-	75	mA max
Power Dissipation				100	mW max
Operating Temperature		-60	-	+75	Deg. C

Guide to source illuminations

Light source Illumination	LUX
Moonlight	0.1
60W Bulb at 1m	50
1W MES Bulb at 0.1m	100
Fluorescent Lighting	500
Bright Sunlight	30,000

FIGURE 1 CIRCUIT SYMBOL

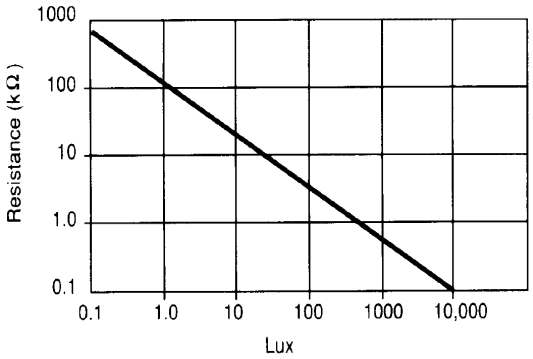


R1
LDR SUNROM #3190

Sensitivity

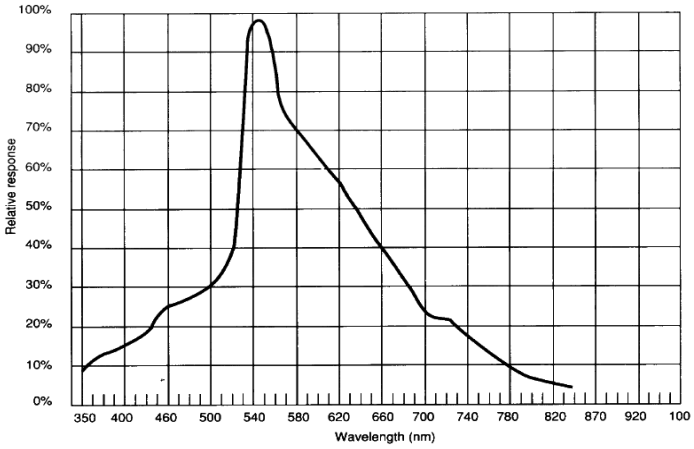
The sensitivity of a photodetector is the relationship between the light falling on the device and the resulting output signal. In the case of a photocell, one is dealing with the relationship between the incident light and the corresponding resistance of the cell.

FIGURE 2 RESISTANCE AS FUNCTION OF ILLUMINATION



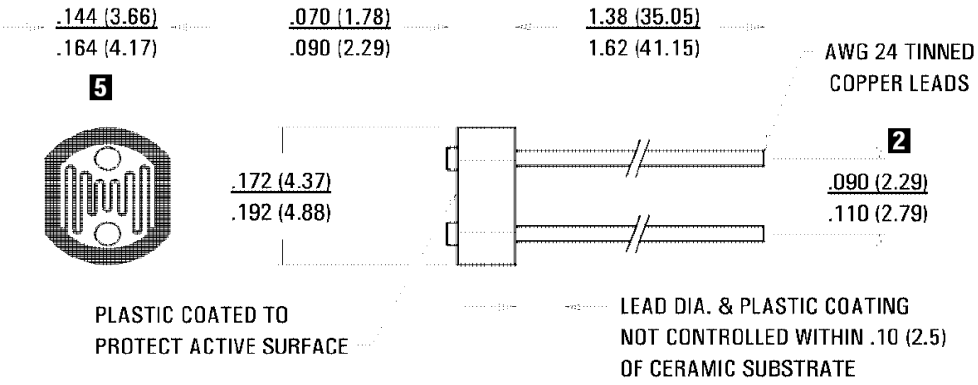
Spectral Response

Figure 3 Spectral response



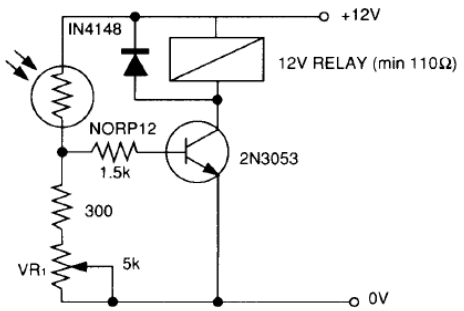
Like the human eye, the relative sensitivity of a photoconductive cell is dependent on the wavelength (color) of the incident light. Each photoconductor material type has its own unique spectral response curve or plot of the relative response of the photocell versus wavelength of light.

Dimensions



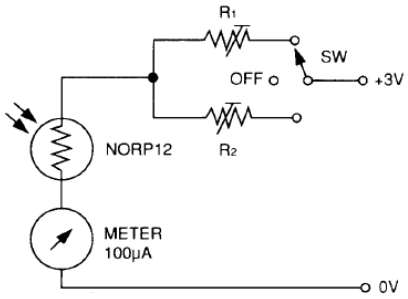
Typical Application Circuits

Figure 6 Sensitive light operated relay



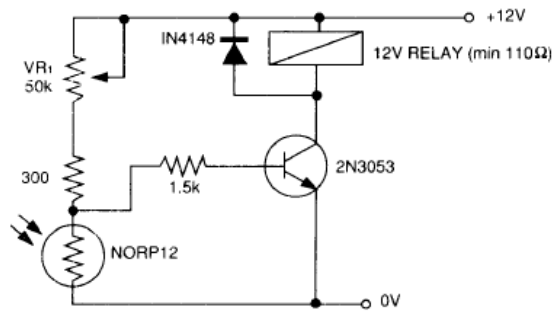
Relay energised when light level increases above the level set by VR_1

Figure 9 Logarithmic law photographic light meter



Typical value $R^1 = 100k\Omega$
$R^2 = 200k\Omega$ preset to give two overlapping ranges.
(Calibration should be made against an accurate meter.)

Figure 7 Light interruption detector



As Figure 6 relay energised when light level drops below the level set by VR_1

Figure 8 Automatic light circuit

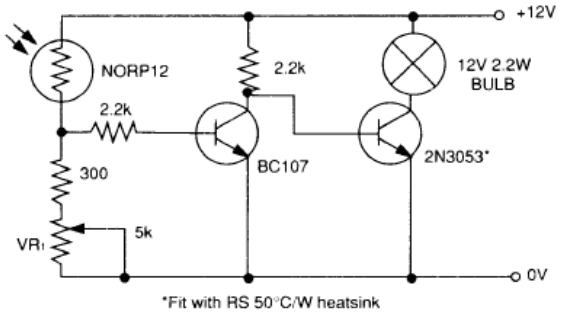
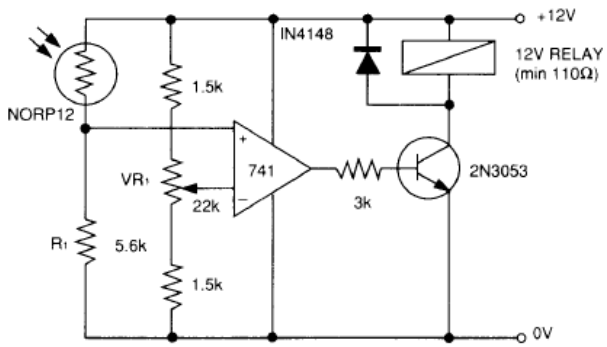


Figure 10 Extremely sensitive light operated relay



(Relay energised when light exceeds preset level.) Incorporates a balancing bridge and op-amp. R_1 and NORP12 may be interchanged for the reverse function.

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

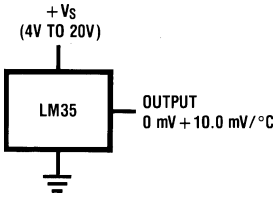
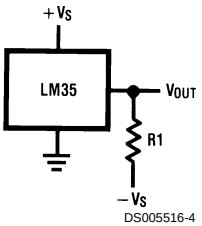


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)

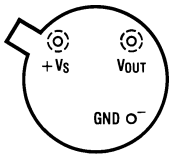


Choose $R_1 = -V_S/50\ \mu A$
V_{OUT} = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

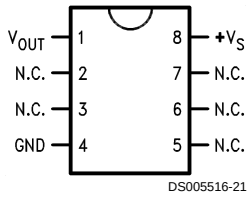
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

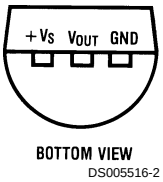
SO-8
Small Outline Molded Package



N.C. = No Connection

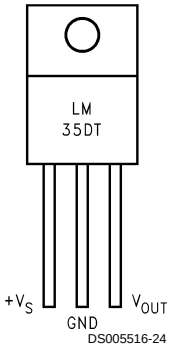
Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package



Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).

Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60˚C to +180˚C
TO-92 Package,	−60˚C to +150˚C
SO-8 Package,	−65˚C to +150˚C
TO-220 Package,	−65˚C to +150˚C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300˚C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260˚C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215˚C
Infrared (15 seconds)	220˚C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55˚C to +150˚C
LM35C, LM35CA	−40˚C to +110˚C
LM35D	0˚C to +100˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy	T_A=+25˚C	±0.2	±0.5		±0.2	±0.5		˚C
(Note 7)	T_A=−10˚C	±0.3			±0.3		±1.0	˚C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		˚C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	˚C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	**±0.18**		**±0.35**	**±0.15**		**±0.3**	˚C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.9, +10.1**		**+10.0**		**+9.9, +10.1**	mV/˚C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±3.0**	**±0.5**		**±3.0**	mV/mA
Line Regulation (Note 3)	T_A=+25˚C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	**±0.02**		**±0.1**	**±0.02**		**±0.1**	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25˚C	56	67		56	67		µA
	V_S=+5V	**105**		**131**	**91**		**114**	µA
	V_S=+30V, +25˚C	56.2	68		56.2	68		µA
	V_S=+30V	**105.5**		**133**	**91.5**		**116**	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25˚C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	**0.5**		**2.0**	**0.5**		**2.0**	µA
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.5**	**+0.39**		**+0.5**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy,	T_A=+25˚C	±0.4	±1.0		±0.4	±1.0		˚C
LM35, LM35C	T_A=−10˚C	±0.5			±0.5		±1.5	˚C
(Note 7)	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	˚C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	˚C
Accuracy, LM35D	T_A=+25˚C				±0.6	±1.5		˚C
(Note 7)	T_A=T_{MAX}				±0.9		±2.0	˚C
	T_A=T_{MIN}				±0.9		±2.0	˚C
Nonlinearity	T_{MIN}≤T_A≤T_{MAX}	**±0.3**		**±0.5**	**±0.2**		**±0.5**	˚C
(Note 8)								
Sensor Gain	T_{MIN}≤T_A≤T_{MAX}	**+10.0**	**+9.8, +10.2**		**+10.0**		**+9.8, +10.2**	mV/˚C
(Average Slope)								
Load Regulation	T_A=+25˚C	±0.4	±2.0		±0.4	±2.0		mV/mA
(Note 3) 0≤I_L≤1 mA	T_{MIN}≤T_A≤T_{MAX}	**±0.5**		**±5.0**	**±0.5**		**±5.0**	mV/mA
Line Regulation	T_A=+25˚C	±0.01	±0.1		±0.01	±0.1		mV/V
(Note 3)	4V≤V_S≤30V	**±0.02**		**±0.2**	**±0.02**		**±0.2**	mV/V
Quiescent Current	V_S=+5V, +25˚C	56	80		56	80		µA
(Note 9)	V_S=+5V	**105**		**158**	**91**		**138**	µA
	V_S=+30V, +25˚C	56.2	82		56.2	82		µA
	V_S=+30V	**105.5**		**161**	**91.5**		**141**	µA
Change of	4V≤V_S≤30V, +25˚C	0.2	2.0		0.2	2.0		µA
Quiescent Current	4V≤V_S≤30V	**0.5**		**3.0**	**0.5**		**3.0**	µA
(Note 3)								
Temperature Coefficient of Quiescent Current		**+0.39**		**+0.7**	**+0.39**		**+0.7**	µA/˚C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	˚C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			˚C

Note 1: Unless otherwise noted, these specifications apply: −55˚C≤T_J≤+150˚C for the LM35 and LM35A; −40˚C≤T_J≤+110˚C for the LM35C and LM35CA; and 0˚C≤T_J≤+100˚C for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2˚C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400˚C/W, junction to ambient, and 24˚C/W junction to case. Thermal resistance of the TO-92 package is 180˚C/W junction to ambient. Thermal resistance of the small outline molded package is 220˚C/W junction to ambient. Thermal resistance of the TO-220 package is 90˚C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in **boldface** apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/˚C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in ˚C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

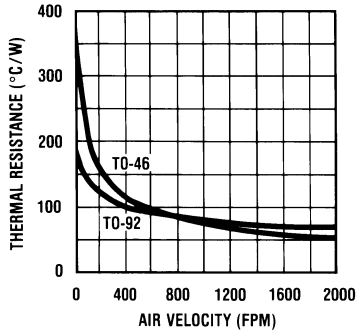
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

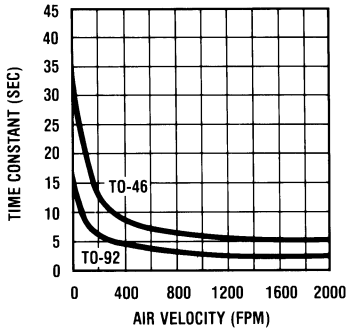
Typical Performance Characteristics

Thermal Resistance Junction to Air



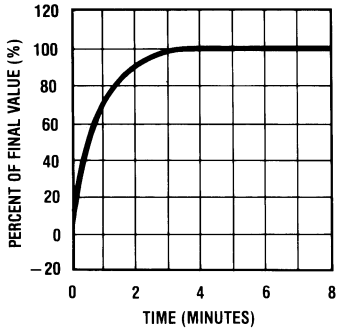
DS005516-25

Thermal Time Constant



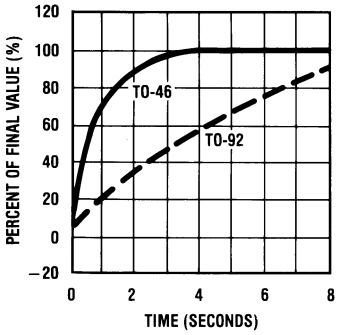
DS005516-26

Thermal Response in Still Air



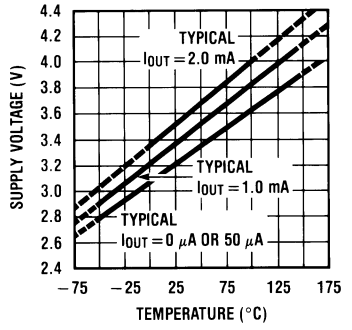
DS005516-27

Thermal Response in Stirred Oil Bath



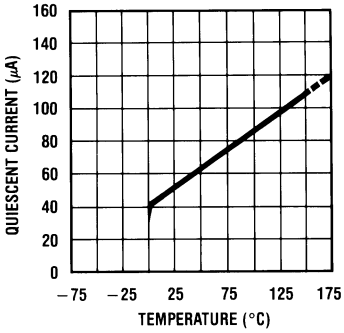
DS005516-28

Minimum Supply Voltage vs. Temperature



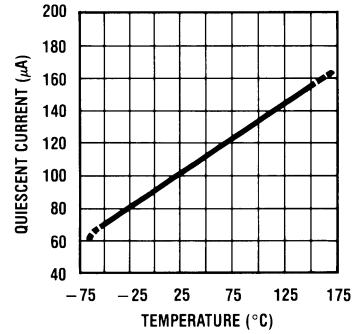
DS005516-29

Quiescent Current vs. Temperature (In Circuit of *Figure 1*.)



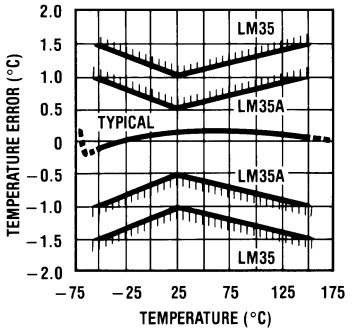
DS005516-30

Quiescent Current vs. Temperature (In Circuit of *Figure 2*.)



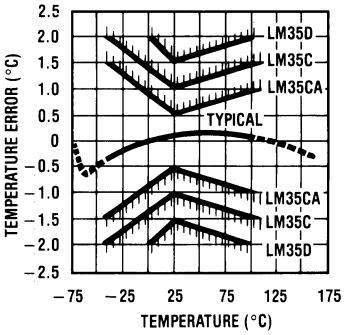
DS005516-31

Accuracy vs. Temperature (Guaranteed)



DS005516-32

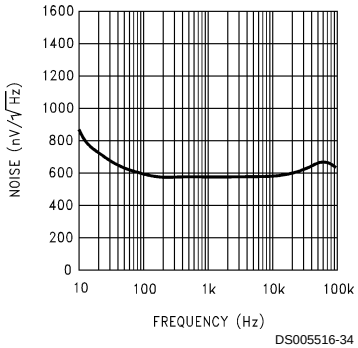
Accuracy vs. Temperature (Guaranteed)



DS005516-33

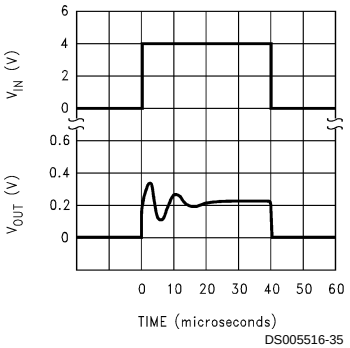
Typical Performance Characteristics (Continued)

Noise Voltage



DS005516-34

Start-Up Response



DS005516-35

Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01˚C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is expecially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400˚C/W	100˚C/W	180˚C/W	140˚C/W	220˚C/W	110˚C/W	90˚C/W
Moving air	100˚C/W	40˚C/W	90˚C/W	70˚C/W	105˚C/W	90˚C/W	26˚C/W
Still oil	100˚C/W	40˚C/W	90˚C/W	70˚C/W			
Stirred oil	50˚C/W	30˚C/W	45˚C/W	40˚C/W			
(Clamped to metal, Infinite heat sink)	(24˚C/W)				(55˚C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

Typical Applications

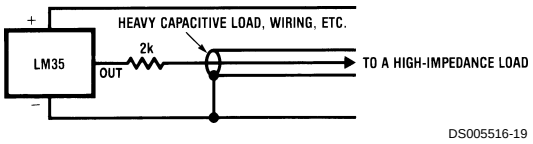


FIGURE 3. LM35 with Decoupling from Capacitive Load

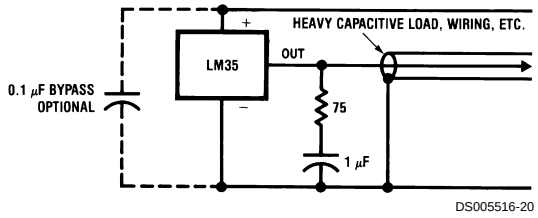


FIGURE 4. LM35 with R-C Damper

CAPACITIVE LOADS

Like most micropower circuits, the LM35 has a limited ability to drive heavy capacitive loads. The LM35 by itself is able to drive 50 pf without special precautions. If heavier loads are anticipated, it is easy to isolate or decouple the load with a resistor; see *Figure 3*. Or you can improve the tolerance of capacitance with a series R-C damper from output to ground; see *Figure 4*.

When the LM35 is applied with a 200Ω load resistor as shown in *Figure 5*, *Figure 6* or *Figure 8* it is relatively immune to wiring capacitance because the capacitance forms a bypass from ground to input, not on the output. However, as with any linear circuit connected to wires in a hostile environment, its performance can be affected adversely by intense electromagnetic sources such as relays, radio transmitters, motors with arcing brushes, SCR transients, etc, as its wiring can act as a receiving antenna and its internal junctions can act as rectifiers. For best results in such cases, a bypass capacitor from V_{IN} to ground and a series R-C damper such as 75Ω in series with 0.2 or 1 μF from output to ground are often useful. These are shown in *Figure 13*, *Figure 14*, and *Figure 16*.

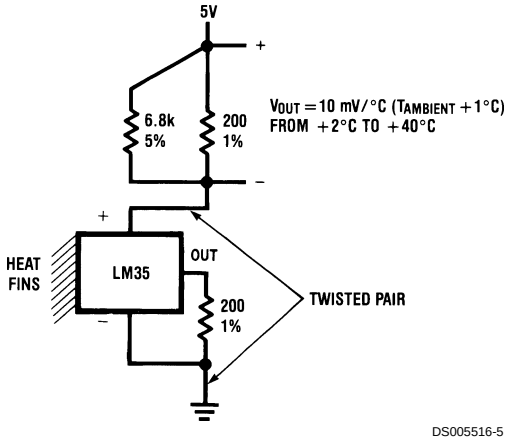


FIGURE 5. Two-Wire Remote Temperature Sensor (Grounded Sensor)

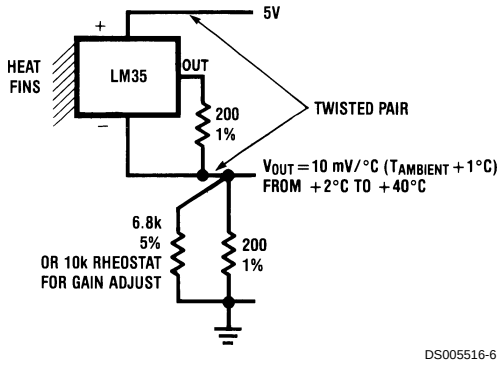


FIGURE 6. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

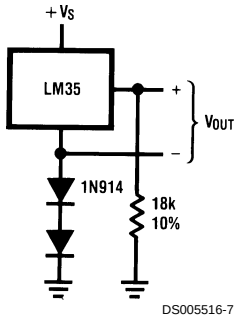


FIGURE 7. Temperature Sensor, Single Supply, −55˚ to +150˚C

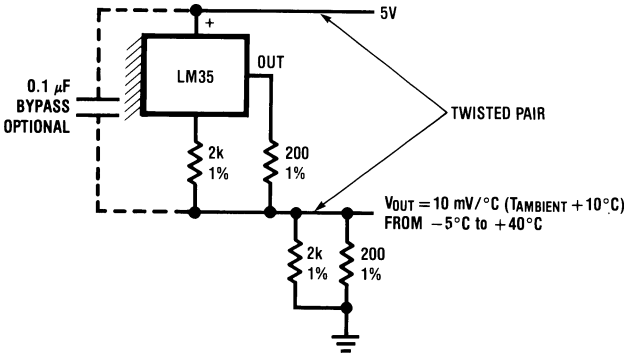


FIGURE 8. Two-Wire Remote Temperature Sensor (Output Referred to Ground)

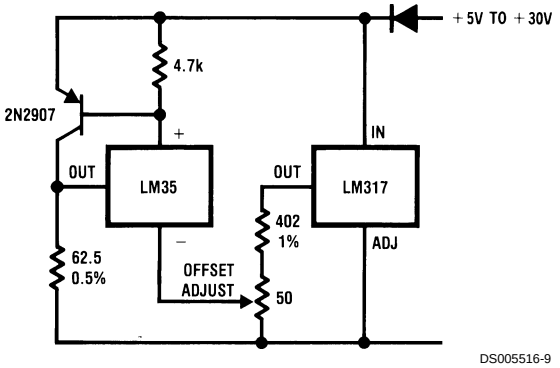


FIGURE 9. 4-To-20 mA Current Source (0˚C to +100˚C)

Typical Applications (Continued)

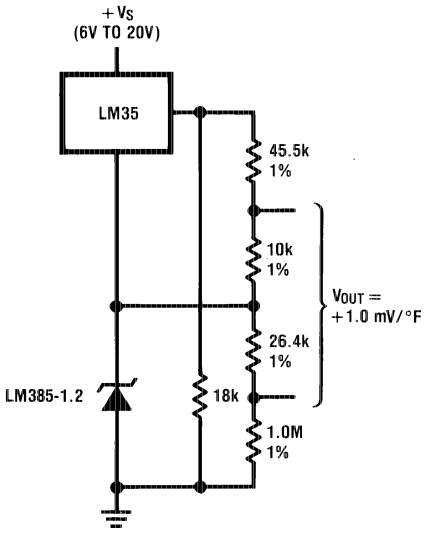


FIGURE 10. Fahrenheit Thermometer

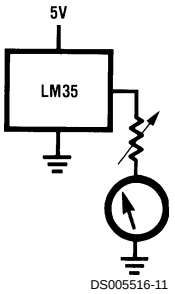


FIGURE 11. Centigrade Thermometer (Analog Meter)

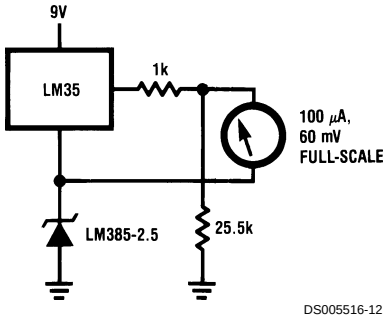


FIGURE 12. Fahrenheit ThermometerExpanded Scale Thermometer
(50˚ to 80˚ Fahrenheit, for Example Shown)

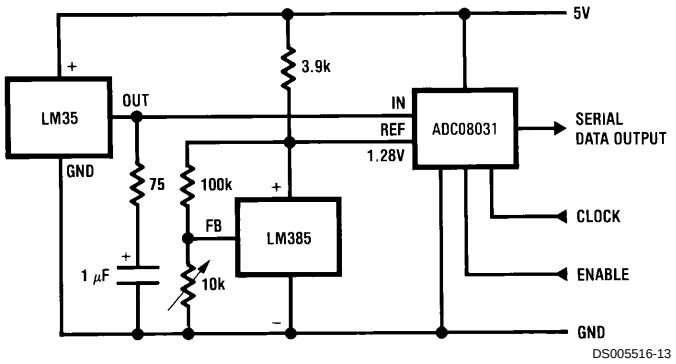


FIGURE 13. Temperature To Digital Converter (Serial Output) (+128˚C Full Scale)

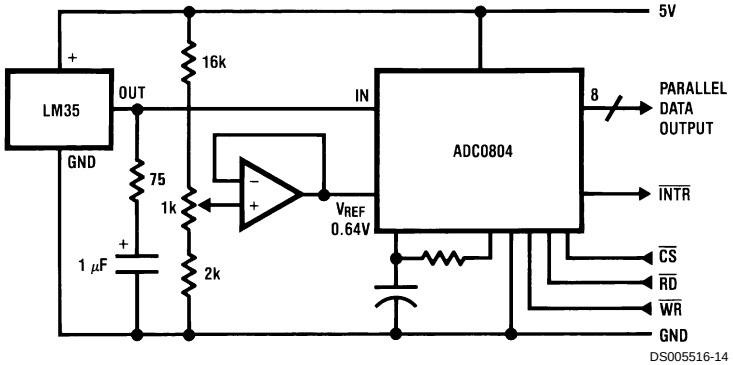
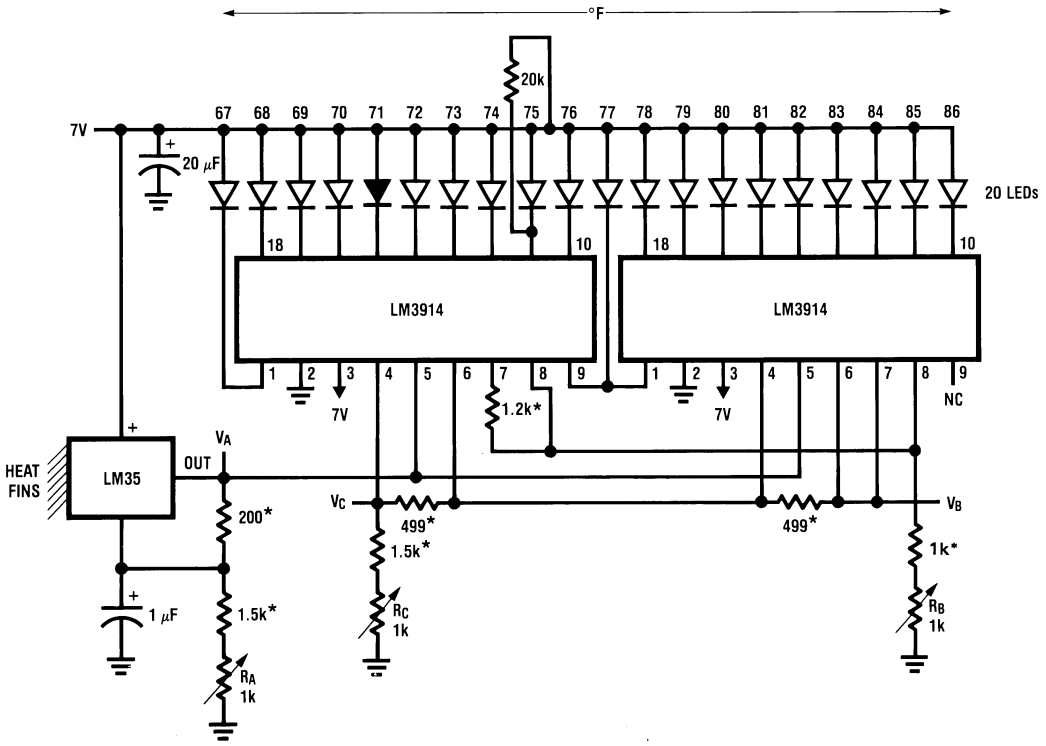


FIGURE 14. Temperature To Digital Converter (Parallel TRI-STATE™ Outputs for Standard Data Bus to µP Interface) (128˚C Full Scale)

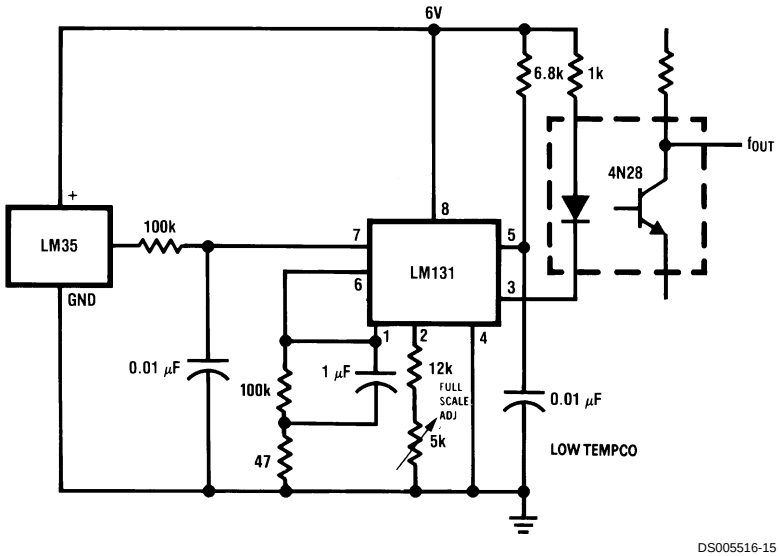
Typical Applications (Continued)



DS005516-16

*=1% or 2% film resistor
Trim R_B for V_B=3.075V
Trim R_C for V_C=1.955V
Trim R_A for V_A=0.075V + 100mV/°C x $T_{ambient}$
Example, V_A=2.275V at 22°C

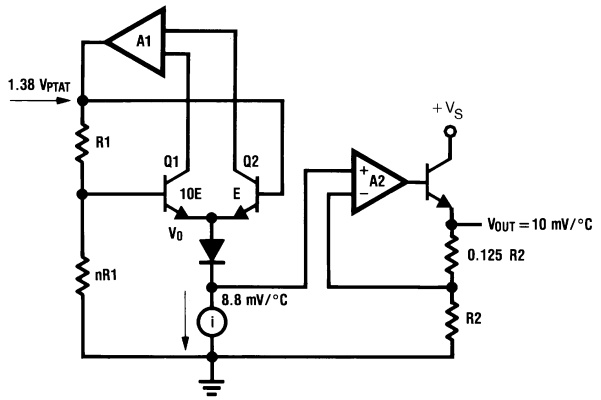
FIGURE 15. Bar-Graph Temperature Display (Dot Mode)



DS005516-15

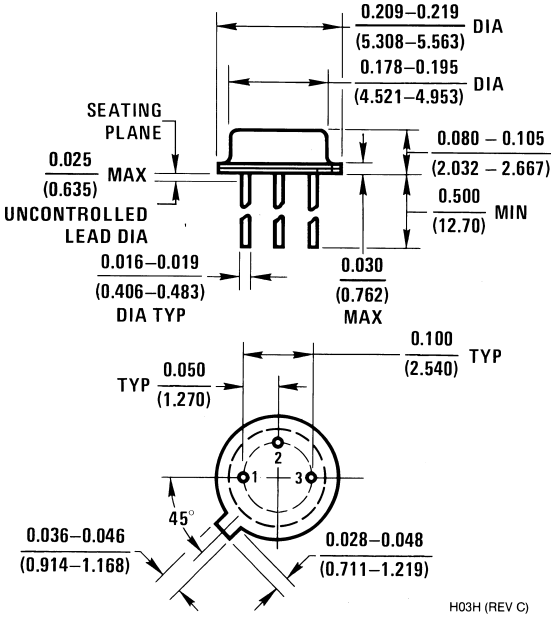
FIGURE 16. LM35 With Voltage-To-Frequency Converter And Isolated Output (2°C to +150°C; 20 Hz to 1500 Hz)

Block Diagram

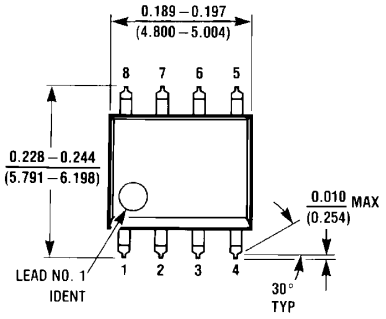


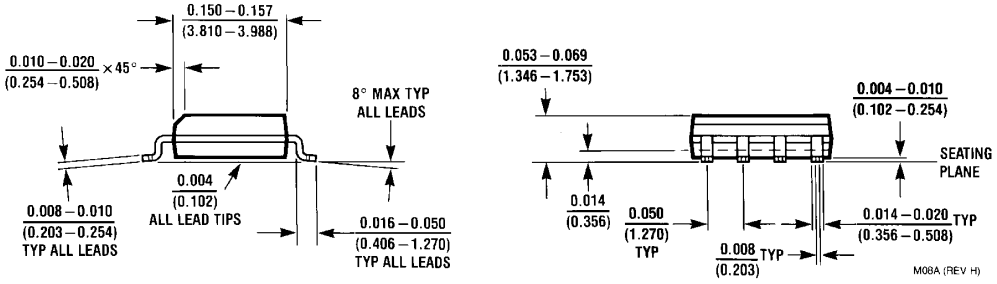
DS005516-23

Physical Dimensions inches (millimeters) unless otherwise noted



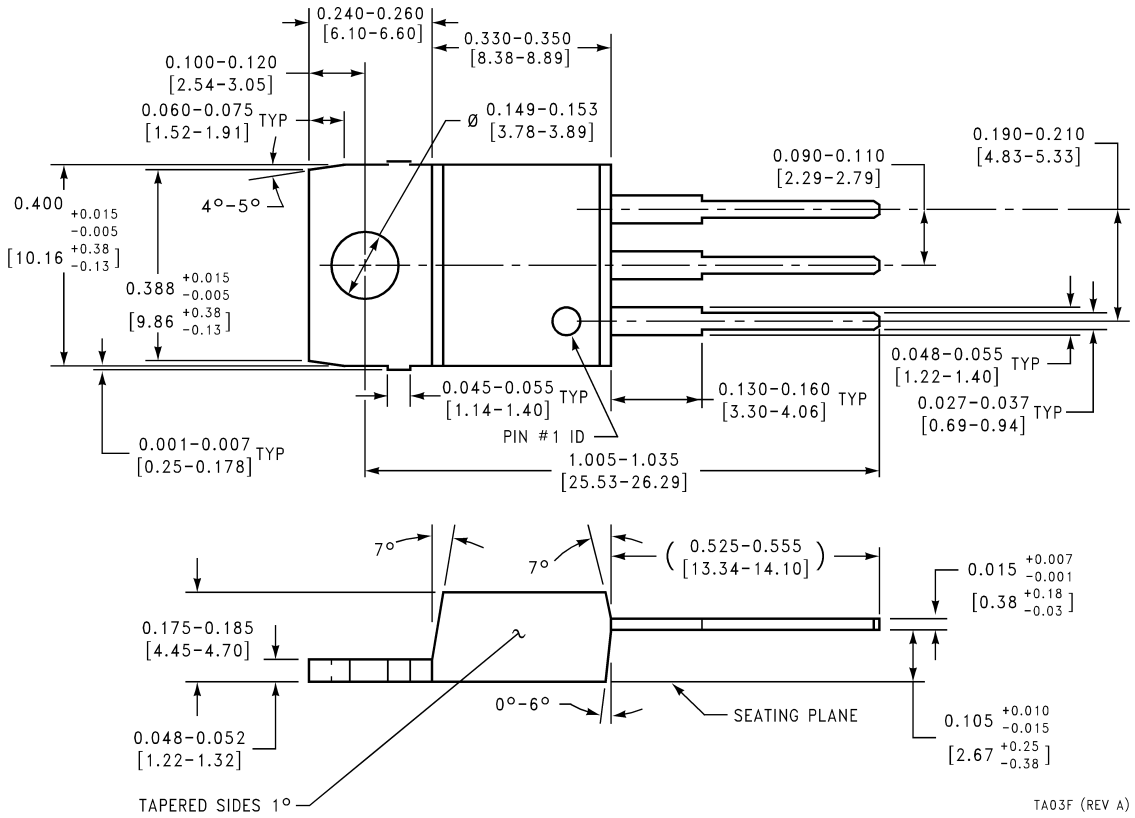
TO-46 Metal Can Package (H)
Order Number LM35H, LM35AH, LM35CH, LM35CAH, or LM35DH
NS Package Number H03H





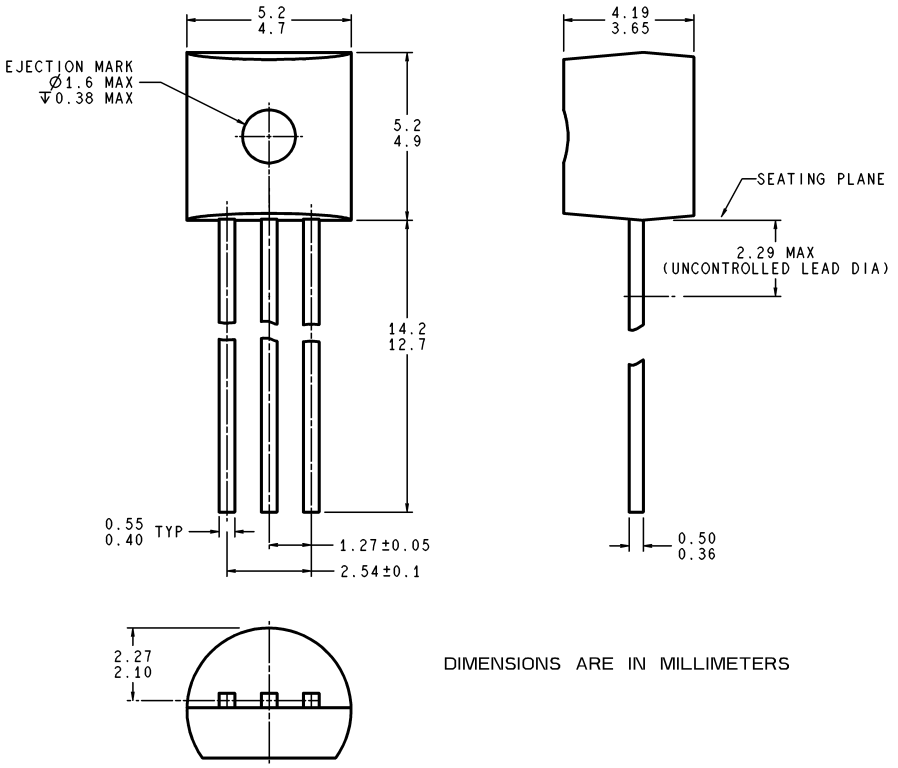
SO-8 Molded Small Outline Package (M)
Order Number LM35DM
NS Package Number M08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



Power Package TO-220 (T)
Order Number LM35DT
NS Package Number TA03F

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



TO-92 Plastic Package (Z)
Order Number LM35CZ, LM35CAZ or LM35DZ
NS Package Number Z03A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

MQ-9 Semiconductor Sensor for CO/Combustible Gas

Sensitive material of MQ-9 gas sensor is SnO_2, which with lower conductivity in clean air. It make detection by method of cycle high and low temperature, and detect CO when low temperature (heated by 1.5V). The sensor's conductivity is more higher along with the gas concentration rising. When high temperature (heated by 5.0V), it detects Methane, Propane etc combustible gas and cleans the other gases adsorbed under low temperature. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-9 gas sensor has high sensitity to Carbon Monoxide, Methane and LPG. The sensor could be used to detect different gases contains CO and combustible gases, it is with low cost and suitable for different application.

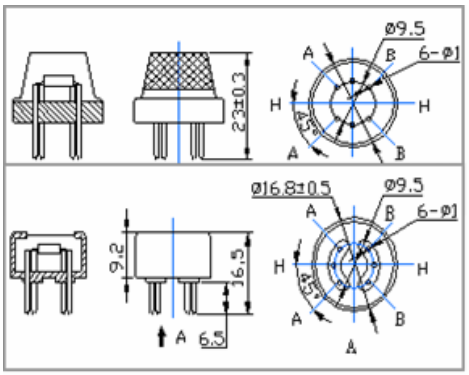
Character

* Good sensitivity to CO/Combustible gas
* High sensitivity to Methane, Propane and CO
* Long life and low cost
* Simple drive circuit

Application

* Domestic gas leakage detector
* Industrial gas detector
* Portable gas detector

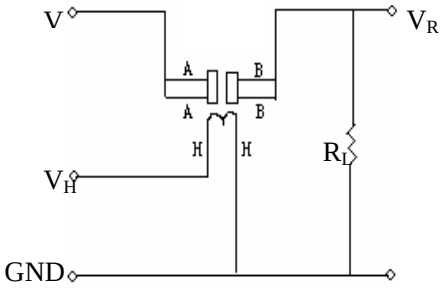
Configuration



Technical Data

Model No.			MQ-9
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite
Detection Gas			CO and combustible gas
Concentration			10-1000ppm CO 100-10000ppm combustible gas
Circuit	Loop Voltage	V_C	≤10V DC
	Heater Voltage	V_H	5.0V±0.2V ACorDC（High） 1.5V±0.1V ACorDC（Low）
	Heater Time	T_L	60±1S（High）90±1S（Low）
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤350mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 100ppm CO)
	Sensitivity	S	Rs(in air)/Rs(100ppm CO)≥5
	Slope	α	≤0.6(R_{300ppm}/R_{100ppm} CO)
Condition	Tem. Humidity		20℃±2℃; 65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V; V_H（High）: 5.0V±0.1V; V_H（Low）: 1.5V±0.1V
	Preheat time		Over 48 hours

Basic test loop



The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage（VH） and test voltage（VC）. VH used to supply certified working temperature to the sensor, while VC used to detect voltage (VRL) on load resistance（RL）whom is in series with sensor. The sensor has light polarity, Vc need DC power. VC and VH could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable RL value is needed:

Power of Sensitivity body(Ps): $Ps=Vc^2\times Rs/(Rs+RL)^2$

Tel: 86-371-67169070/80 Fax: 86-371-67169090 Email: sales@hwsensor.com

Resistance of sensor(Rs): Rs=(Vc/VRL-1)×RL

Sensitivity Characteristics

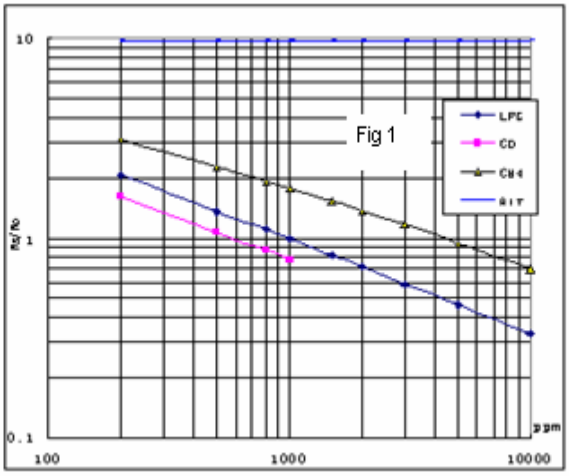


Fig.1 shows the typical sensitivity characteristics of the MQ-9, ordinate means resistance ratio of the sensor (Rs/Ro), abscissa is concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 1000ppm LPG. All test are under standard test conditions.

Influence of Temperature/Humidity

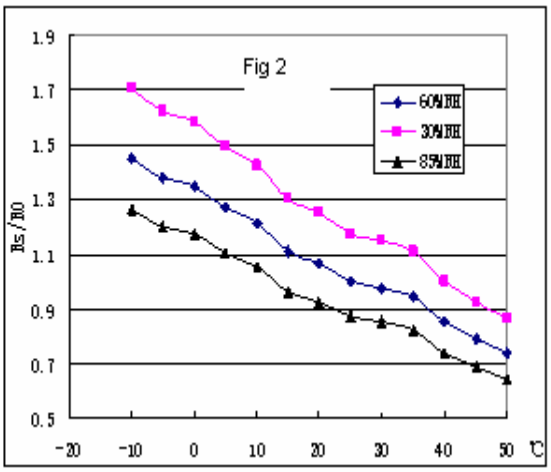
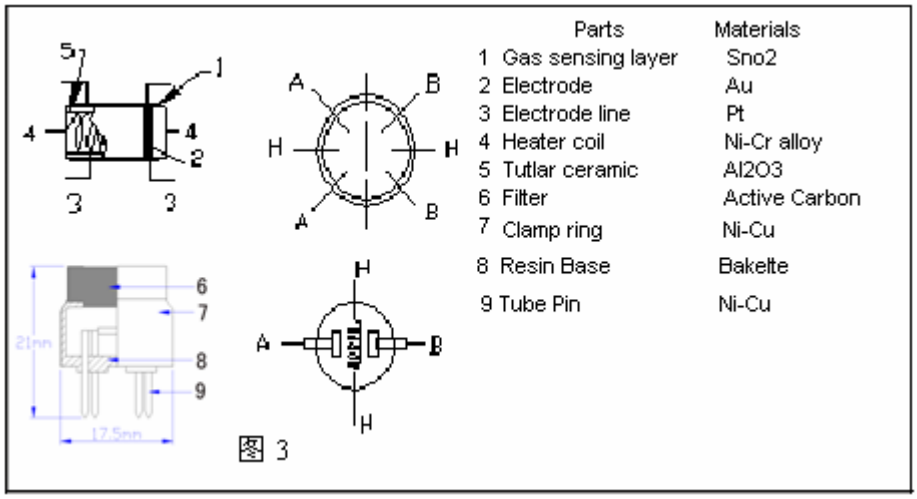


Fig.2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor (Rs/Ro), Rs means resistance of sensor in 1000ppm Propane under different tem. and humidity. Ro means resistance of the sensor in environment of 1000ppm Propane, 20℃/65%RH

Structure and configuration



Structure and configuration of MQ-9 gas sensor is shown as Fig. 3, sensor composed by micro AL2O3 ceramic tube, Tin Dioxide (SnO2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-7 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.

Notification

1 Following conditions must be prohibited

1.1 Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixature, silicon latex, putty or plastic contain silicon environment

1.2 High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H_2Sz, SO_X, Cl_2, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

1.3 Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

1.4 Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

1.5 Freezing

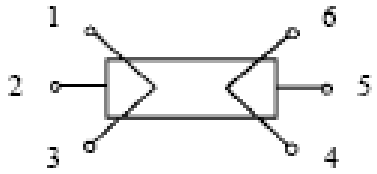
Do avoid icing on sensor'surface, otherwise sensor would lose sensitivity.

1.6 Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors' sensitivity characteristic changed badly.

1.7 Voltage on wrong pins

For 6 pins sensor, if apply voltage on 1、3 pins or 4、6 pins, it will make lead broken, and without signal when apply on 2、4 pins



2 Following conditions must be avoided

2.1 Water Condensation

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor' sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, if will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it's stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stbility before using.

2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

2.5 Vibration

Continual vibration will result in sensors down-lead response then repture. In transportation or assembling line, pneumatic screwdriver/ultrasonic welding machine can lead this vibration.

2.6 Concussion

If sensors meet strong concussion, it may lead its lead wire disconnected.

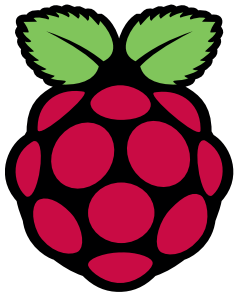
2.7 Usage

For sensor, handmade welding is optimal way. If use wave crest welding should meet the following conditions:

2.7.1 Soldering flux: Rosin soldering flux contains least chlorine
2.7.2 Speed: 1-2 Meter/ Minute
2.7.3 Warm-up temperature：100±20℃
2.7.4 Welding temperature：250±10℃
2.7.5 1 time pass wave crest welding machine

If disobey the above using terms, sensors sensitivity will be reduced.

DATASHEET



Raspberry Pi Compute Module 3+

Raspberry Pi Compute Module 3+ Lite

Release 1, January 2019

Table 1: Release History

Release	Date	Description
1	28/01/2019	First release

The latest release of this document can be found at `https://www.raspberrypi.org`

Contents

List of Figures

List of Tables

1 Introduction

The Raspberry Pi Compute Module 3+ (CM3+) is a range of DDR2-SODIMM-mechanically-compatible System on Modules (SoMs) containing processor, memory, eMMC Flash (on non-Lite variants) and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these modules have extra IO interfaces over and above what is available on the Raspberry Pi model A/B boards, opening up more options for the designer.

The CM3+ contains a BCM2837B0 processor (as used on the Raspberry Pi 3B+), 1Gbyte LPDDR2 RAM and eMMC Flash. The CM3+ is currently available in 4 variants, CM3+/8GB, CM3+/16GB, CM3+/32GB and CM3+ Lite, which have 8, 16 and 32 Gigabytes of eMMC Flash, or no eMMC Flash, respectively.

The CM3+ Lite product is the same as CM3+ except the eMMC Flash is not fitted, and the SD/eMMC interface pins are available for the user to connect their own SD/eMMC device.

Note that the CM3+ is electrically identical and, with the exception of higher CPU z-height, physically identical to the legacy CM3 products.

CM3+ modules require a software/firmware image dated November 2018 or newer to function correctly.

2 Features

2.1 Hardware

- Low cost
- Low power
- High availability
- High reliability
  - Tested over millions of Raspberry Pis Produced to date
  - Module IO pins have 15 micro-inch hard gold plating over 2.5 micron Nickel

2.2 Peripherals

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

2.3 Software

- ARMv8 Instruction Set
- Mature and stable Linux software stack
  - Latest Linux Kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Full availability of GPU functions using standard APIs

3 Block Diagram

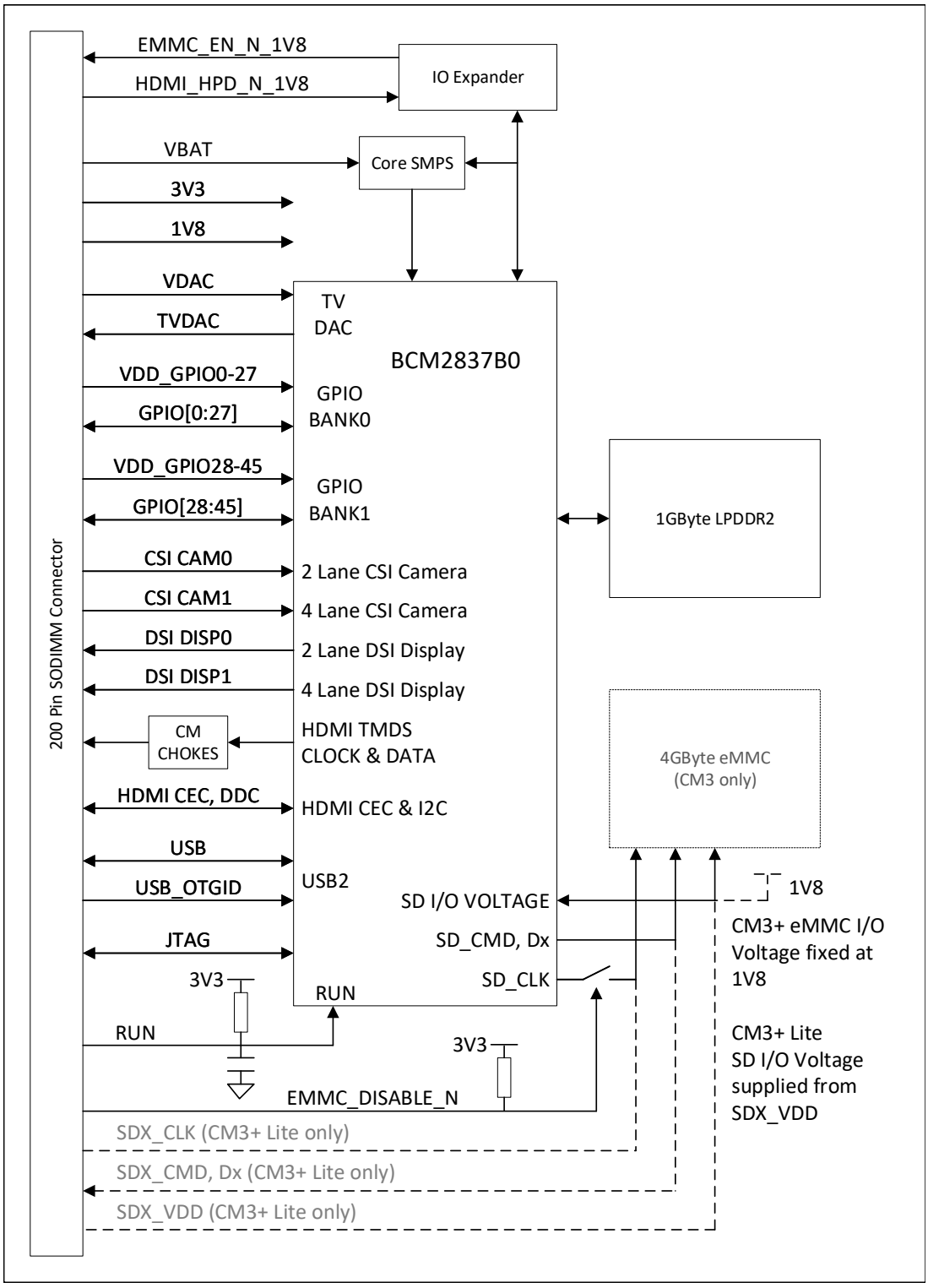


Figure 1: CM3+ Block Diagram

4 Mechanical Specification

The CM3+ modules conform to JEDEC MO-224 mechanical specification for 200 pin DDR2 (1.8V) SODIMM modules and therefore should work with the many DDR2 SODIMM sockets available on the market. **(Please note that the pinout of the Compute Module is not the same as a DDR2 SODIMM module; they are not electrically compatible.)**

The SODIMM form factor was chosen as a way to provide the 200 pin connections using a standard, readily available and low cost connector compatible with low cost PCB manufacture.

The maximum component height on the underside of the Compute Module is 1.2mm.

The maximum component height on the top side of the Compute Module is 2.5mm.

The Compute Module PCB thickness is 1.0mm +/- 0.1mm.

Note that the location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however, maximum component heights and PCB thickness will be kept as specified.

Figure 2 gives the CM3+ mechanical dimensions.

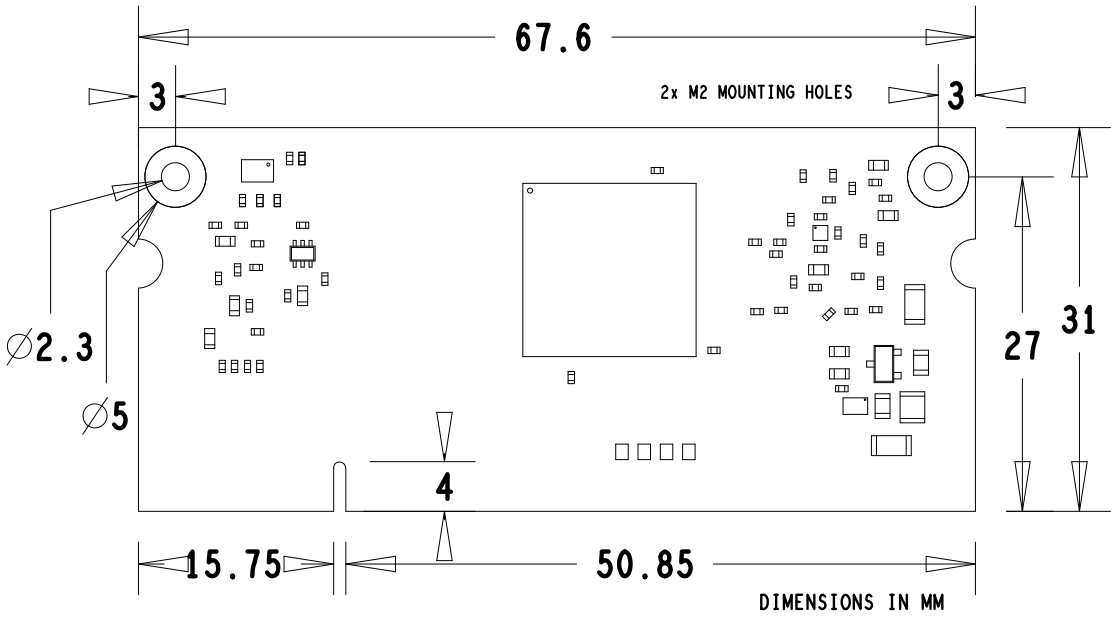


Figure 2: CM3+ Mechanical Dimensions

5 Pin Assignments

CM3+	CM3+ Lite	PIN		PIN	CM3+	CM3+ Lite
GND		1		2	EMMC_DISABLE_N	
GPIO0		3		4	NC	SDX_VDD
GPIO1		5		6	NC	SDX_VDD
GND		7		8	GND	
GPIO2		9		10	NC	SDX_CLK
GPIO3		11		12	NC	SDX_CMD
GND		13		14	GND	
GPIO4		15		16	NC	SDX_D0
GPIO5		17		18	NC	SDX_D1
GND		19		20	GND	
GPIO6		21		22	NC	SDX_D2
GPIO7		23		24	NC	SDX_D3
GND		25		26	GND	
GPIO8		27		28	GPIO28	
GPIO9		29		30	GPIO29	
GND		31		32	GND	
GPIO10		33		34	GPIO30	
GPIO11		35		36	GPIO31	
GND		37		38	GND	
GPIO0-27_VDD		39		40	GPIO0-27_VDD	
		KEY				
GPIO28-45_VDD		41		42	GPIO28-45_VDD	
GND		43		44	GND	
GPIO12		45		46	GPIO32	
GPIO13		47		48	GPIO33	
GND		49		50	GND	
GPIO14		51		52	GPIO34	
GPIO15		53		54	GPIO35	
GND		55		56	GND	
GPIO16		57		58	GPIO36	
GPIO17		59		60	GPIO37	
GND		61		62	GND	
GPIO18		63		64	GPIO38	
GPIO19		65		66	GPIO39	
GND		67		68	GND	
GPIO20		69		70	GPIO40	
GPIO21		71		72	GPIO41	
GND		73		74	GND	
GPIO22		75		76	GPIO42	
GPIO23		77		78	GPIO43	
GND		79		80	GND	
GPIO24		81		82	GPIO44	
GPIO25		83		84	GPIO45	
GND		85		86	GND	
GPIO26		87		88	HDMI_HPD_N_1V8	
GPIO27		89		90	EMMC_EN_N_1V8	
GND		91		92	GND	
DSI0_DN1		93		94	DSI1_DP0	
DSI0_DP1		95		96	DSI1_DN0	
GND		97		98	GND	
DSI0_DN0		99		100	DSI1_CP	
DSI0_DP0		101		102	DSI1_CN	
GND		103		104	GND	
DSI0_CN		105		106	DSI1_DP3	
DSI0_CP		107		108	DSI1_DN3	
GND		109		110	GND	
HDMI_CLK_N		111		112	DSI1_DP2	
HDMI_CLK_P		113		114	DSI1_DN2	
GND		115		116	GND	
HDMI_D0_N		117		118	DSI1_DP1	
HDMI_D0_P		119		120	DSI1_DN1	
GND		121		122	GND	
HDMI_D1_N		123		124	NC	
HDMI_D1_P		125		126	NC	
GND		127		128	NC	
HDMI_D2_N		129		130	NC	
HDMI_D2_P		131		132	NC	
GND		133		134	GND	
CAM1_DP3		135		136	CAM0_DP0	
CAM1_DN3		137		138	CAM0_DN0	
GND		139		140	GND	
CAM1_DP2		141		142	CAM0_CP	
CAM1_DN2		143		144	CAM0_CN	
GND		145		146	GND	
CAM1_CP		147		148	CAM0_DP1	
CAM1_CN		149		150	CAM0_DN1	
GND		151		152	GND	
CAM1_DP1		153		154	NC	
CAM1_DN1		155		156	NC	
GND		157		158	NC	
CAM1_DP0		159		160	NC	
CAM1_DN0		161		162	NC	
GND		163		164	GND	
USB_DP		165		166	TVDAC	
USB_DM		167		168	USB_OTGID	
GND		169		170	GND	
HDMI_CEC		171		172	VC_TRST_N	
HDMI_SDA		173		174	VC_TDI	
HDMI_SCL		175		176	VC_TMS	
RUN		177		178	VC_TDO	
DD_CORE (DO NOT CONNEC		179		180	VC_TCK	
GND		181		182	GND	
1V8		183		184	1V8	
1V8		185		186	1V8	
GND		187		188	GND	
VDAC		189		190	VDAC	
3V3		191		192	3V3	
3V3		193		194	3V3	
GND		195		196	GND	
VBAT		197		198	VBAT	
VBAT		199		200	VBAT	

Table 2: Compute Module 3+ SODIMM Connector Pinout

Table 2 gives the Compute Module 3+ pinout and Table 3 gives the pin functions.

Pin Name	DIR	Voltage Ref	PDN[a] State	If Unused	Description/Notes
RUN and Boot Control (see text for usage guide)					
RUN	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
GPIO					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 1
Primary SD Interface[d,e]					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
USB Interface					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
HDMI Interface					
HDMI_SCL	I/O	3V3[b]	Z[f]	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3[b]	Z[f]	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
CAM0 (CSI0) 2-lane Interface					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
CAM1 (CSI1) 4-lane Interface					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
DSI0 (Display 0) 2-lane Interface					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
DSI1 (Display 1) 4-lane Interface					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
TV Out					
TVDAC	O	-	Z	Leave open	Composite video DAC output
JTAG Interface					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

[a] The PDN column indicates power-down state (when RUN pin LOW)
[b] Must be driven by an open-collector driver
[c] GPIO have software enabled pulls which keep state over power-down
[d] Only available on Lite variants
[e] The CM will always try to boot from this interface first
[f] Requires external pull-up resistor to 5V as per HDMI spec

Table 3: Pin Functions

6 Electrical Specification

Caution! Stresses above those listed in Table 4 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VBAT	Core SMPS Supply	-0.5	6.0	V
3V3	3V3 Supply Voltage	-0.5	4.10	V
1V8	1V8 Supply Voltage	-0.5	2.10	V
VDAC	TV DAC Supply	-0.5	4.10	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	-0.5	4.10	V
GPIO28-45_VDD	GPIO28-45 I/O Supply Voltage	-0.5	4.10	V
SDX_VDD	Primary SD/eMMC Supply Voltage	-0.5	4.10	V

Table 4: Absolute Maximum Ratings

DC Characteristics are defined in Table 5

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 1.8V	-	-	0.6	V
		VDD_IO = 2.7V	-	-	0.8	V
		VDD_IO = 3.3V	-	-	0.9	V
V_{IH}	Input high voltage[a]	VDD_IO = 1.8V	1.0	-	-	V
		VDD_IO = 2.7V	1.3	-	-	V
		VDD_IO = 3.3V	1.6	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	5	μA
C_{IN}	Input capacitance	-	-	5	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD_IO = 2.7V, IOL = -2mA	-	-	0.15	V
		VDD_IO = 3.3V, IOL = -2mA	-	-	0.14	V
V_{OH}	Output high voltage[b]	VDD_IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD_IO = 2.7V, IOH = 2mA	2.5	-	-	V
		VDD_IO = 3.3V, IOH = 2mA	3.0	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 1.8V, VO = 0.4V	12	-	-	mA
		VDD_IO = 2.7V, VO = 0.4V	17	-	-	mA
		VDD_IO = 3.3V, VO = 0.4V	18	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 1.8V, VO = 1.4V	10	-	-	mA
		VDD_IO = 2.7V, VO = 2.3V	16	-	-	mA
		VDD_IO = 3.3V, VO = 2.3V	17	-	-	mA
R_{PU}	Pullup resistor	-	50	-	65	kΩ
R_{PD}	Pulldown resistor	-	50	-	65	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 5: DC Characteristics

AC Characteristics are defined in Table 6 and Fig. 3.

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	1.6	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	1.7	-	ns
GPCLK	t_{JOSC}	Oscillator-derived GPCLK cycle-cycle jitter (RMS)	-	-	20	ps
GPCLK	t_{JPLL}	PLL-derived GPCLK cycle-cycle jitter (RMS)	-	-	48	ps

[a] Default drive strength, CL = 5pF, VDD_IOx = 3.3V

Table 6: Digital I/O Pin AC Characteristics

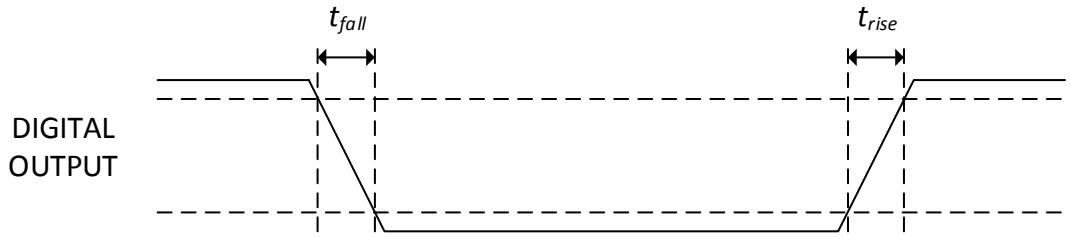


Figure 3: Digital IO Characteristics

7 Power Supplies

The Compute Module 3+ has six separate supplies that must be present and powered at all times; you cannot leave any of them unpowered, even if a specific interface or GPIO bank is unused. The six supplies are as follows:

1. VBAT is used to power the BCM2837 processor core. It feeds the SMPS that generates the chip core voltage.
2. 3V3 powers various BCM2837 PHYs, IO and the eMMC Flash.
3. 1V8 powers various BCM2837 PHYs, IO and SDRAM.
4. VDAC powers the composite (TV-out) DAC.
5. GPIO0-27_VREF powers the GPIO 0-27 IO bank.
6. GPIO28-45_VREF powers the GPIO 28-45 IO bank.

Supply	Descripion	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply[a]	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-45 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

[a] Requires a clean 2.5-2.8V supply if TV DAC is used, else connect to 3V3

Table 7: Power Supply Operating Ranges

7.1 Supply Sequencing

Supplies should be staggered so that the highest voltage comes up first, then the remaining voltages in descending order. This is to avoid forward biasing internal (on-chip) diodes between supplies, and causing latch-up. Alternatively supplies can be synchronised to come up at exactly the same time as long as at no point a lower voltage supply rail voltage exceeds a higher voltage supply rail voltage.

7.2 Power Requirements

Exact power requirements will be heavily dependent upon the individual use case. If an on-chip subsystem is unused, it is usually in a low power state or completely turned off. For instance, if your application does not use 3D graphics then a large part of the core digital logic will never turn on and need power. This is also the case for camera and display interfaces, HDMI, USB interfaces, video encoders and decoders, and so on.

Powerchain design is critical for stable and reliable operation of the Compute Module 3+. We strongly recommend that designers spend time measuring and verifying power requirements for their particular use case and application, as well as paying careful attention to power supply sequencing and maximum supply voltage tolerance.

Table 8 specifies the recommended minimum power supply outputs required to power the Compute Module 3+.

Supply	Minimum Requirement	Unit
VBAT (CM1)	2000[a]	mW
VBAT (CM3,3L)	3500[a]	mW
3V3	250	mA
1V8	250	mA
VDAC	25	mA
GPIO0-27_VDD	50[b]	mA
GPIO28-45_VDD	50[b]	mA
SDX_VDD	50[b]	mA

[a] Recommended minimum. Actual power drawn is very dependent on use-case
[b] Each GPIO can supply up to 16mA, aggregate current per bank must not exceed 50mA

Table 8: Mimimum Power Supply Requirements

8 Booting

The eMMC Flash device on CM3+ is directly connected to the primary BCM2837 SD/eMMC interface. These connections are not accessible on the module pins. On CM3+ Lite this SD interface is available on the SDX_ pins.

When initially powered on, or after the RUN pin has been held low and then released, the BCM2837 will try to access the primary SD/eMMC interface. It will then look for a file called bootcode.bin on the primary partition (which must be FAT) to start booting the system. If it cannot access the SD/eMMC device or the boot code cannot be found, it will fall back to waiting for boot code to be written to it over USB; in other words, its USB port is in slave mode waiting to accept boot code from a suitable host.

A USB boot tool is available on Github which allows a host PC running Linux to write the BCM2837 boot code over USB to the module. That boot code then runs and provides access to the SD/eMMC as a USB mass storage device, which can then be read and written using the host PC. Note that a Raspberry Pi can be used as the host machine. For those using Windows a precompiled and packeged tool is available. For more information see here.

The Compute Module has a pin called EMMC_DISABLE_N which when shorted to GND will disable the SD/eMMC interface (by physically disconnecting the SD_CMD pin), forcing BCM2837 to boot from USB. Note that when the eMMC is disabled in this way, it takes a couple of seconds from powering up for the processor to stop attempting to talk to the SD/eMMC device and fall back to booting from USB.

Note that once booted over USB, BCM2837 needs to re-enable the SD/eMMC device (by releasing EMMC_DISABLE_N) to allow access to it as mass storage. It expects to be able to do this by driving the EMMC_EN_N_1V8 pin LOW, which at boot is initially an input with a pull up to 1V8. If an end user wishes to add the ability to access the SD/eMMC over USB in their product, similar circuitry to that used on the Compute Module IO Board to enable/disable the USB boot and SD/eMMC must be used; that is, EMMC_DISABLE_N pulled low via MOSFET(s) and released again by MOSFET, with the gate controlled by EMMC_EN_N_1V8. **Ensure you use MOSFETs suitable for switching at 1.8V (i.e. use a device with gate threshold voltage, Vt, suitable for 1.8V switching).**

9 Peripherals

9.1 GPIO

BCM2837 has in total 54 GPIO lines in 3 separate voltage banks. All GPIO pins have at least two alternative functions within the SoC. When not used for the alternate peripheral function, each GPIO pin may be set as an input (optionally as an interrupt) or an output. The alternate functions are usually peripheral I/Os, and most peripherals appear twice to allow flexibility on the choice of I/O voltage.

GPIO bank2 is used on the module to connect to the eMMC device and for an on-board I2C bus (to talk to the core SMPS and control the special function pins). On CM3+ Lite most of bank2 is exposed to allow a user to connect their choice of SD card or eMMC device (if required).

Bank0 and 1 GPIOs are available for general use. GPIO0 to GPIO27 are bank0 and GPIO28-45 make up bank1. GPIO0-27_VDD is the power supply for bank0 and GPIO28-45_VDD is the power supply for bank1. SDX_VDD is the supply for bank2 on CM3+ Lite. These supplies can be in the range 1.8V-3.3V (see Table 7) and are not optional; each bank must be powered, even when none of the GPIOs for that bank are used.

Note that the HDMI_HPD_N_1V8 and EMMC_EN_N_1V8 pins are 1.8V IO and are used for special functions (HDMI hot plug detect and boot control respectively). Please do not use these pins for any other purpose, as the software for the module will always expect these pins to have these special functions. If they are unused please leave them unconnected.

All GPIOs except GPIO28, 29, 44 and 45 have weak in-pad pull-ups or pull-downs enabled when the device is powered on. It is recommended to add off-chip pulls to GPIO28, 29, 44 and 45 to make sure they never float during power on and initial boot.

9.1.1 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	-	-	-
1	High	SCL0	SA4	DE	-	-	-
2	High	SDA1	SA3	LCD_VSYNC	-	-	-
3	High	SCL1	SA2	LCD_HSYNC	-	-	-
4	High	GPCLK0	SA1	DPI_D0	-	-	ARM_TDI
5	High	GPCLK1	SA0	DPI_D1	-	-	ARM_TDO
6	High	GPCLK2	SOE_N	DPI_D2	-	-	ARM_RTCK
7	High	SPI0_CE1_N	SWE_N	DPI_D3	-	-	-
8	High	SPI0_CE0_N	SD0	DPI_D4	-	-	-
9	Low	SPI0_MISO	SD1	DPI_D5	-	-	-
10	Low	SPI0_MOSI	SD2	DPI_D6	-	-	-
11	Low	SPI0_SCLK	SD3	DPI_D7	-	-	-
12	Low	PWM0	SD4	DPI_D8	-	-	ARM_TMS
13	Low	PWM1	SD5	DPI_D9	-	-	ARM_TCK
14	Low	TXD0	SD6	DPI_D10	-	-	TXD1
15	Low	RXD0	SD7	DPI_D11	-	-	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	-	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	-	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	-	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	-	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	-
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	-
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	-
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	-
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	-
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	-

Table 9: GPIO Bank0 Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
28	None	SDA0	SA5	PCM_CLK	FL0	-	-
29	None	SCL0	SA4	PCM_FS	FL1	-	-
30	Low	TE0	SA3	PCM_DIN	CTS0	-	CTS1
31	Low	FL0	SA2	PCM_DOUT	RTS0	-	RTS1
32	Low	GPCLK0	SA1	RING_OCLK	TXD0	-	TXD1
33	Low	FL1	SA0	TE1	RXD0	-	RXD1
34	High	GPCLK0	SOE_N	TE2	SD1_CLK	-	-
35	High	SPI0_CE1_N	SWE_N	-	SD1_CMD	-	-
36	High	SPI0_CE0_N	SD0	TXD0	SD1_DAT0	-	-
37	Low	SPI0_MISO	SD1	RXD0	SD1_DAT1	-	-
38	Low	SPI0_MOSI	SD2	RTS0	SD1_DAT2	-	-
39	Low	SPI0_SCLK	SD3	CTS0	SD1_DAT3	-	-
40	Low	PWM0	SD4	-	SD1_DAT4	SPI2_MISO	TXD1
41	Low	PWM1	SD5	TE0	SD1_DAT5	SPI2_MOSI	RXD1
42	Low	GPCLK1	SD6	TE1	SD1_DAT6	SPI2_SCLK	RTS1
43	Low	GPCLK2	SD7	TE2	SD1_DAT7	SPI2_CE0_N	CTS1
44	None	GPCLK1	SDA0	SDA1	TE0	SPI2_CE1_N	-
45	None	PWM1	SCL0	SCL1	TE1	SPI2_CE2_N	-

Table 10: GPIO Bank1 Alternate Functions

Table 9 and Table 10 detail the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the Broadcom Peripherals Specification document and have Linux drivers available.

9.1.2 Secondary Memory Interface (SMI)

The SMI peripheral is an asynchronous NAND type bus supporting Intel mode80 type transfers at 8 or 16 bit widths and available in the ALT1 positions on GPIO banks 0 and 1 (see Table 9 and Table 10). It is not publicly documented in the Broadcom Peripherals Specification but a Linux driver is available in the Raspberry Pi Github Linux repository (bcm2835_smi.c in linux/drivers/misc).

9.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available on bank 0 GPIOs. This up-to-24-bit parallel interface can support a secondary display. Again this interface is not documented in the Broadcom Peripherals Specification but documentation can be found here.

9.1.4 SD/SDIO Interface

The BCM283x supports two SD card interfaces, SD0 and SD1.

The first (SD0) is a proprietary Broadcom controller that does not support SDIO and is the primary interface used to boot and talk to the eMMC or SDX_x signals.

The second interface (SD1) is standards compliant and can interface to SD, SDIO and eMMC devices; for example on a Raspberry Pi 3 B+ it is used to talk to the on-board CYW43455 WiFi device in SDIO mode.

Both interfaces can support speeds up to 50MHz single ended (SD High Speed Mode).

9.2 CSI (MIPI Serial Camera)

Currently the CSI interface is not openly documented and only CSI camera sensors supported by the official Raspberry Pi firmware will work with this interface. Supported sensors are the OmniVision OV5647 and Sony IMX219.

It is recommended to attach other cameras via USB.

9.3 DSI (MIPI Serial Display)

Currently the DSI interface is not openly documented and only DSI displays supported by the official Raspberry Pi firmware will work with this interface.

Displays can also be added via the parallel DPI interface which is available as a GPIO alternate function - see Table 9 and Section 9.1.3

9.4 USB

The BCM2837 USB port is On-The-Go (OTG) capable. If using either as a fixed slave or fixed master, please tie the USB_OTGID pin to ground.

The USB port (Pins USB_DP and USB_DM) must be routed as 90 ohm differential PCB traces.

Note that the port is capable of being used as a true OTG port however there is no official documentation. Some users have had success making this work.

9.5 HDMI

BCM283x supports HDMI V1.3a.

It is recommended that users follow a similar arrangement to the Compute Module IO Board circuitry for HDMI output.

The HDMI CK_P/N (clock) and D0-D2_P/N (data) pins must each be routed as matched length 100 ohm differential PCB traces. It is also important to make sure that each differential pair is closely phase matched. Finally, keep HDMI traces well away from other noise sources and as short as possible.

Failure to observe these design rules is likely to result in EMC failure.

9.6 Composite (TV Out)

The TVDAC pin can be used to output composite video (PAL or NTSC). Please route this signal away from noise sources and use a 75 ohm PCB trace.

Note that the TV DAC is powered from the VDAC supply which must be a clean supply of 2.5-2.8V. It is recommended users generate this supply from 3V3 using a low noise LDO.

If the TVDAC output is not used VDAC can be connected to 3V3, but it must be powered even if the TV-out functionality is unused.

10 Thermals

The BCM2837 SoC employs DVFS (Dynamic Voltage and Frequency Scaling) on the core voltage. When the processor is idle (low CPU utilisation), it will reduce the core frequency and voltage to reduce current draw and heat output. When the core utilisation exceeds a certain threshold the core votlage is increased and the core frequency is boosted to the maximum working frerquency of 1.2GHz. The voltage and frequency are throttled back when the CPU load reduces back to an 'idle' level OR when the silicon temperature as mesured by the on-chip temperature sensor exceeds 80C (thermal throttling).

A designer must pay careful attention to the thermal design of products using the CM3+ so that performance is not artificially curtailed due to the processor thermal throttling, as the Quad ARM complex in the BCM2837 can generate significant heat output under load.

10.1 Temperature Range

The operating temperature range of the module is set by the lowest maximum and highest minimum of any of the components used.

The eMMC and LPDDR2 have the narrowest range, these are rated for -25 to +80 degrees Celsius. Therefore the nominal range for the CM3+ and CM3+ Lite is -25C to +80C.

However, this range is the maximum for the silicon die; therefore, users would have to take into account the heat generated when in use and make sure this does not cause the temperature to exceed 80 degrees Celsius.

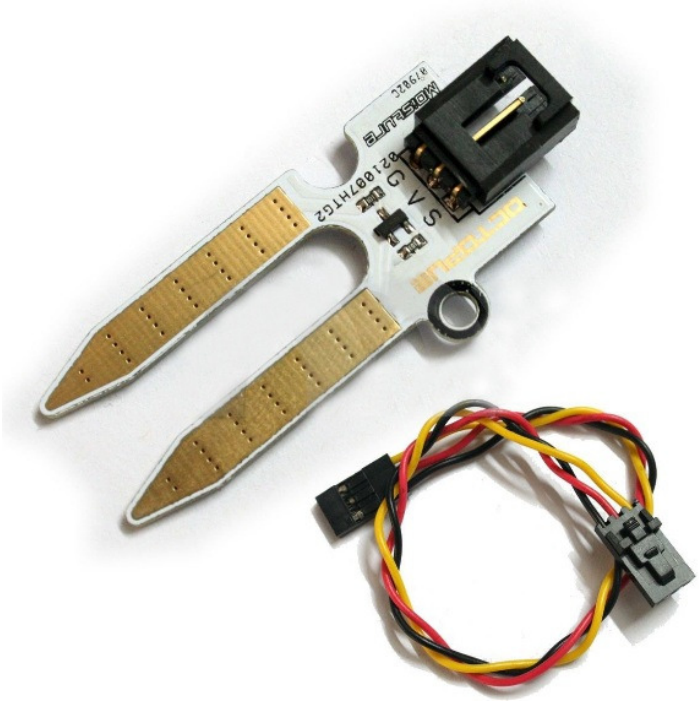
11 Availability

Raspberry Pi guarantee availability of CM3+ and CM3+ Lite until at least January 2026.

12 Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

OBSoil-01 Octopus Soil Moisture Sensor Brick



Octopus Soil Moisture Sensor Brick can read the amount of moisture present in the soil surrounding it. Ideal for monitoring an urban garden, or your pet plant's water level. This is a must have tool for a connected garden!

This sensor uses the two probes to pass current through the soil, and then it reads that resistance to get the moisture level. More water makes the soil conduct electricity more easily (less resistance), while dry soil conducts electricity poorly (more resistance).

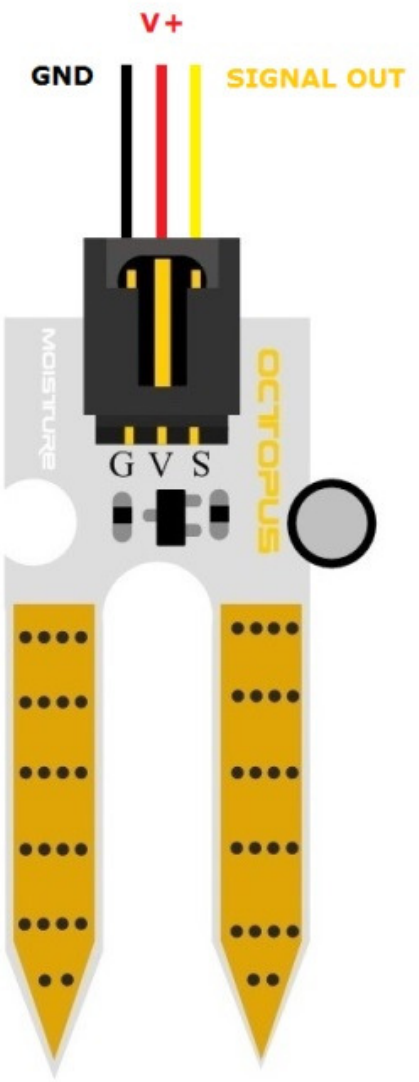
This sensor isn't hardened against contamination or exposure of the control circuitry to water and may be prone to electrolytic corrosion across the probes (Also it can be switched on, take the reading and switched off to minimize electrolytic corrosion), so it isn't well suited to being left in place or used outdoors.

Specification

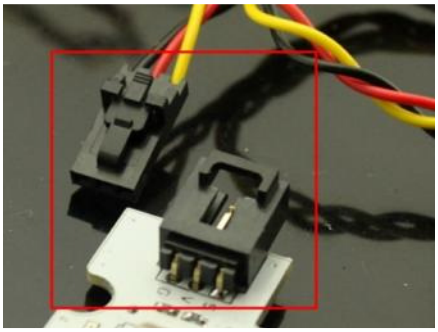
Item	Condition	Min	Typical	Max	Unit
Voltage	-	3.3	/	5	V
Current	-	0	/	35	mA
Output Voltage	Supply Voltage 5 V	0	~	4.2	V
Output Value	Sensor in dry soil	0	~	300	/
	Sensor in humid soil	300	~	700	/
	Sensor in water	700	~	950	/

Pinout:

- **G: GND (Black wire)**
- **V: Voltage Supply (+) (Red wire)**
- **S: Signal out (Yellow wire)**



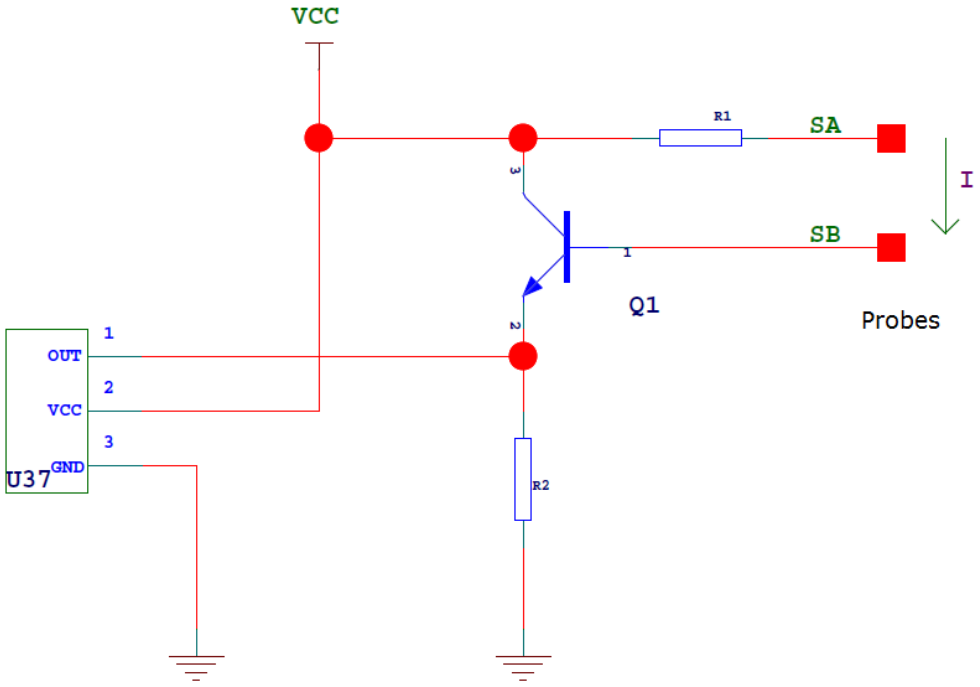
3P buckled wires connector and cable included:



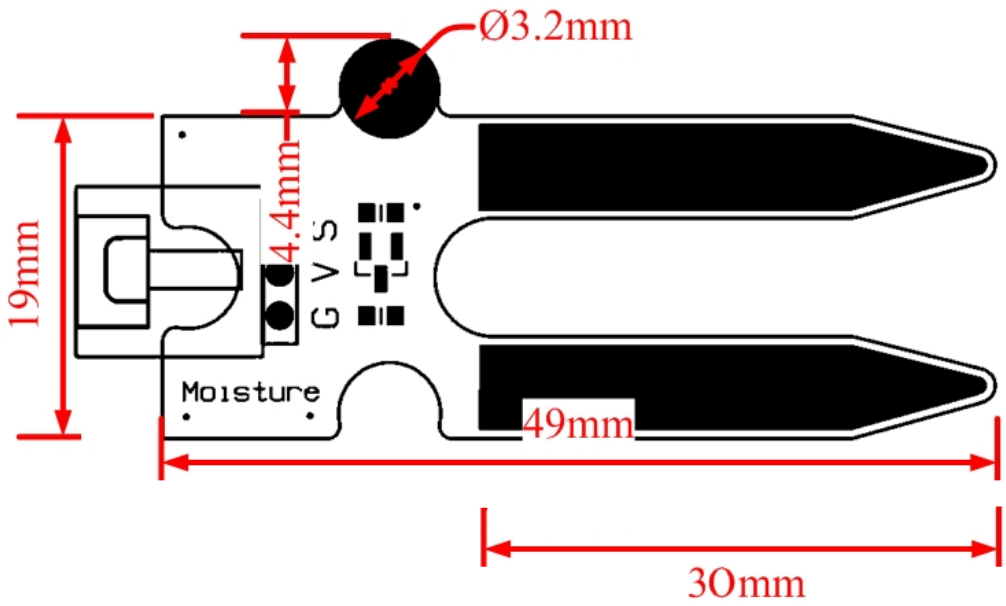
Applications

- Moisture sensoring
- Botanical gardening
- Flood detection
- Liquid level detection

Internal Schematic

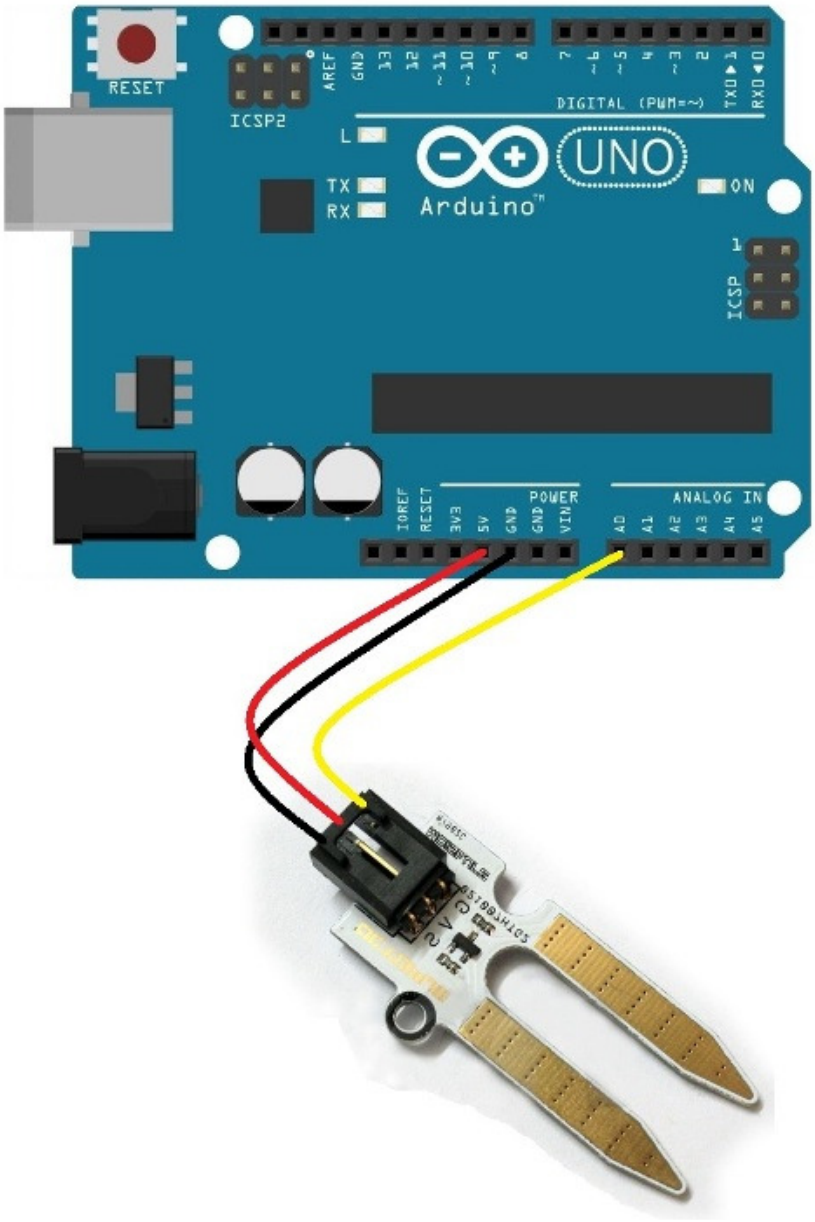


Mechanic Dimensions



Arduino Example

Connection Diagram



Sample Code

```
/*
  # Example code for the moisture sensor
  # Editor     : Lauren
  # Date       : 13.01.2012
  # Version    : 1.0
  # Connect the sensor to the A0 (Analog 0) pin on the Arduino board

  # the sensor value description
  # 0  ~300     dry soil
  # 300~700     humid soil
  # 700~950     in water
*/

void setup(){
  Serial.begin(57600);
}

void loop(){
  Serial.print("Moisture Sensor Value:");
  Serial.println(analogRead(0));
  delay(100);
}
```

The Result in different condition after open the “serial monitor” of Arduino IDE (Under tools menu):