



# **Whitecatboard.org**

## **IoT DEVKIT MANUAL**

Rev 1.0 (ES) 2017

## Índice

1. Introducción Whitecatboard.org
2. Componentes del Kit
  - 2.1. Sensores
  - 2.2. Actuadores
  - 2.3. Placa ESP32N1
  - 2.4. Placa DEVKIT para ESP32N1
3. Entorno de Desarrollo Whitecat IDE
4. LuaRTOS Sistema Operativo del ecosistema Whitecatboard
5. Conectividad (WiFi / Bluetooth / LoRaWAN)
6. LoRaWAN Introducción y Administración en The Things Network (TTN)
7. La “Nube” (IBM Bluemix y NodeRED)
8. Casos Prácticos
  - 8.1. Caso Práctico 1
  - 8.2. Caso Práctico 2
  - 8.3. Caso Práctico 3
  - 8.4. Caso Práctico 4
  - 8.5. Caso Práctico 5
  - 8.6. Caso Práctico 6
  - 8.7. Caso Práctico 7
  - 8.8. Caso Práctico 8
9. Conclusiones
10. Anexos
  - 10.1. Esquemáticos

## 1. Introducción Whitecatboard.org

En primer lugar gracias por adquirir un kit de desarrollo de Whitecatboard.org para el ecosistema IoT, con este kit y la documentación que le acompaña trataremos de introducir al usuario en el ecosistema del "Internet de las Cosas".

EL kit ha sido diseñado teniendo en mente la formación pedagógica así como el uso en entornos más industriales , esto ha sido posible gracias a la participación de diferentes perfiles en el desarrollo del producto.

Ingenieros en el desarrollo de soluciones de software que han diseñado el sistema operativo LuaRTOS sobre el que se sustenta el entorno de desarrollo y el funcionamiento de los diferentes elementos del kit de un modo robusto y al mismo tiempo sencillo por parte del usuario.

Ingenieros en el desarrollo de soluciones de hardware que han diseñado y seleccionado los diferentes elementos del kit con el objetivo de facilitar el uso del mismo por parte de los usuarios finales.

Docentes que han revisado toda la documentación y los casos prácticos que acompañan el kit con el objetivo de maximizar el aprendizaje ....

## 2. Componentes del Kit

El kit de desarrollo de Whitecatboard.org para IoT viene organizado en una caja de plástico como se muestra en la imagen a continuación.

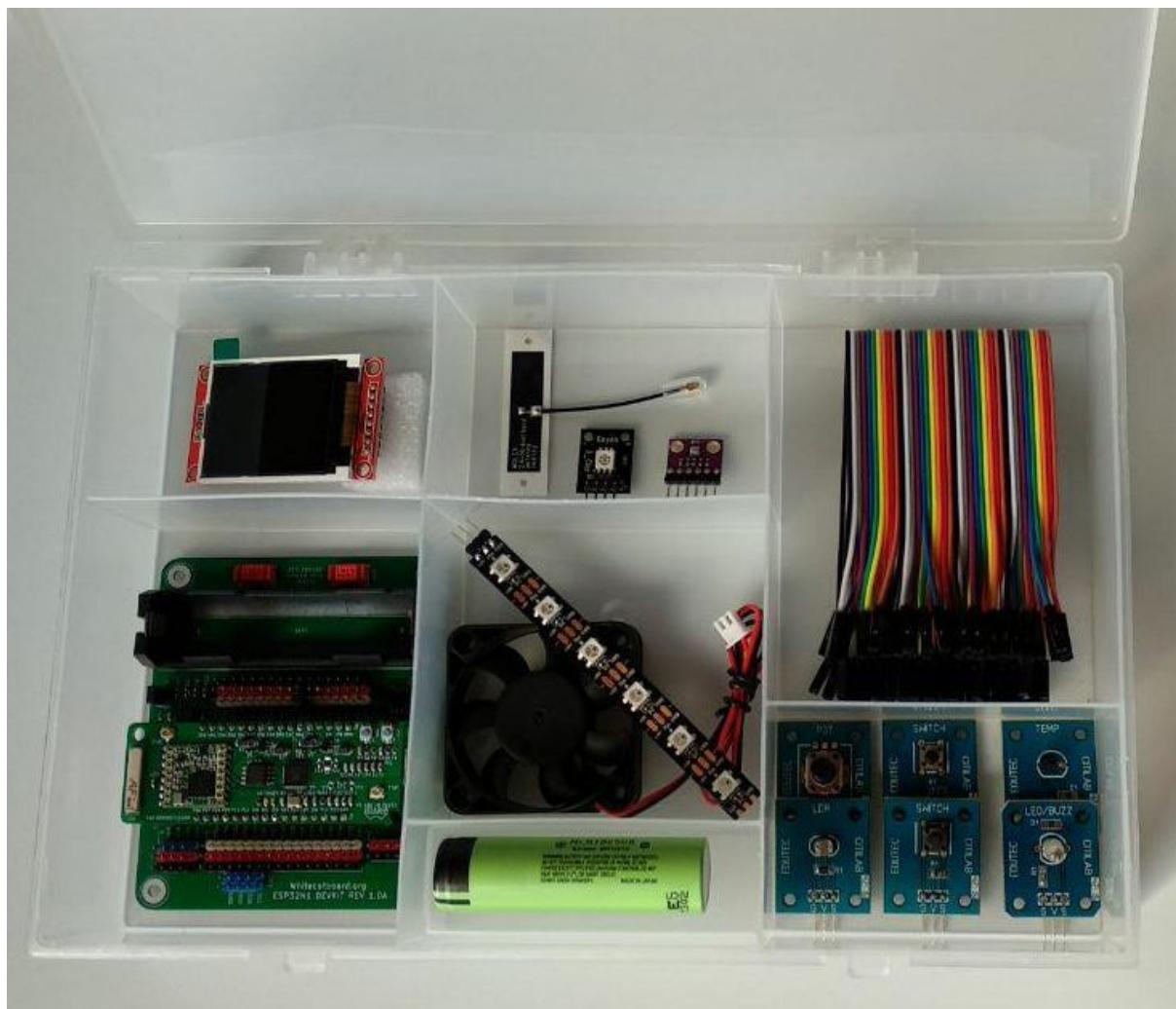


Fig-1. Caja con el kit IoT de Whitecatboard.org

A continuación se detallan algunos de los componentes del kit, estos se dividen entre sensores y actuadores.

Que es un sensor ? Un sensor es un elemento que es capaz de medir magnitudes físicas o químicas y transformarlas en un tipo de señal eléctrica que se puede medir mediante el empleo de una entrada.

Que es un actuador ? Un actuador realiza la labor contraria al sensor en este caso un actuador recibe una señal eléctrica y muestra o realiza una acción (actúa) en base a dicha señal eléctrica.

## 2.1. Sensores

El kit dispone de varios sensores que emplean señales analogicas y digitales estos tipos de señales disponen de unas entradas específicas en la placa DEVKIT ESP32N1, que se explicarán más adelante cuando se hable de dicha placa.

Los sensores que incorpora el kit actualmente son;

### - Pulsador

El pulsador es un simple interruptor momentáneo, que mientras se aprieta, permite que la corriente pase a través de él, normalmente este tipo de dispositivos se conectan por defecto a un valor positivo (pull-up) o negativo (pull-down) con una resistencia para evitar la presencia de una señal no deseada.

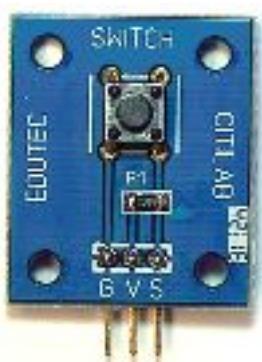


Fig-2. Pulsador momentáneo (switch)

### - Potenciómetro

El potenciómetro es una resistencia de valor variable que modifica su resistencia (dificultad al paso de la corriente) en función de la posición de su eje en la versión suministrada con el kit. El valor máximo de resistencia que tenemos en el kit es de 10K que es un valor bastante común en este tipo de dispositivos, según se conecten sus entradas y salidas el incremento o decremento de su valor de resistencia variará si el movimiento es en el sentido de las agujas del reloj o en sentido contrario.



Fig-3. Potenciómetro

### - Fotoresistencia (LDR)

Del mismo modo que el potenciómetro que hemos visto anteriormente la LDR (fotoresistor) es una resistencia de valor variable pero en vez de variar su valor por una actuación mecánica lo realiza dependiendo de la cantidad de luz que recibe (fotones), estos interactúan con la superficie de la fotoresistencia haciendo que su resistencia (dificultad al paso de la corriente) disminuya con el aumento de la intensidad de luz que incide sobre su superficie.

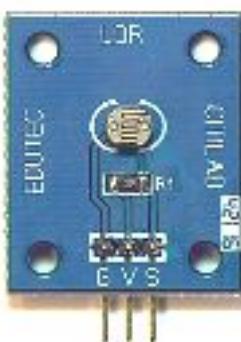


Fig-4. Fotoresistor (LDR)

### - DS180 One Wire Dallas (Sensor de Temperatura)

El sensor DS180 One Wire de Dallas Semiconductor es un sensor encapsulado en el mismo formato que un transistor, es un circuito integrado pequeño que toma la temperatura y la envía por el protocolo oneWire diseñado por Dallas Semiconductor , en este tipo de bus podríamos tener un maestro y varios esclavos (sensores) de este tipo por ejemplo para tomar medidas en diferentes lugares.

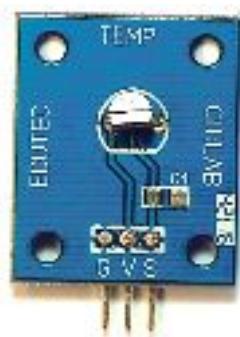


Fig-5. DS180 One Wire Dallas

### - BME280 Sensor I2C de Temperatura, Presión Atmosférica y Humedad

El sensor BME/BMP280 es un encapsulado especial con un circuito integrado interno diseñado por BOSCH, que mediante el bus de comunicaciones I2C entrega la lectura internamente procesada de Temperatura, Presión Atmosférica y Humedad.

Como en el caso anterior en el bus I2C podemos disponer de un maestro y varios sensores esclavos, en el bus i2c no se recomienda que la longitud del cable que une los diferentes esclavos sea muy larga ya que i2c se diseñó y emplea como un bus de comunicaciones entre dispositivos que se encuentran normalmente en la misma placa de circuito impreso.

Las señales del bus I2C son SDA (que es la línea de datos) y SCL (que es la línea de reloj) a diferencia del bus 1-Wire el bus I2C sincroniza las transacciones a través de la línea de reloj SCL mediante pulsos ON/OFF, para acceder a los diferentes dispositivos de un bus I2C se emplea una dirección única de cada dispositivo en el bus, esté escuchará peticiones en dicha dirección y sera el único que responderá.



Fig-6. Sensor BME280

## 2.2. Actuadores

El kit dispone de varios actuadores que emplean señales analogicas y digitales estos tipos de señales disponen de unas salidas específicas en la placa DEVKIT ESP32N1, que se explicarán más adelante cuando se hable de dicha placa.

Los actuadores que incorpora el kit actualmente son;

### - Pantalla TFT

La pantalla es uno de los actuadores más versátiles ya que permite mostrar información de una manera instantánea, la pantalla incluida en el kit dispone de 65535 colores una resolución de 128x160 píxeles y conectividad mediante puerto SPI.

El puerto SPI es más rápido que el puerto I2C ya que permite velocidades mucho más elevadas, las líneas de control de dicho puerto son SCK (pulso de reloj) MISO (entrada master) MOSI (salida master) CS (chip select).

Como se puede apreciar es similar el concepto al del I2C la diferencia radica en que para seleccionar a cada uno de los elementos en el bus SPI se hace uso de una línea CS (chip select) por dispositivo, que en estado bajo (OFF) indica a ese dispositivo que las instrucciones del bus son para él, ya que el resto de dispositivos estarán en estado alto (ON), este tipo de bus por tanto requerirá de un cable adicional al I2C.



Fig-7. Pantalla TFT 65535 colores 1.8" 128x160 píxeles

### - Led RGB

Se ha incluido con el kit un led RGB este tipo de led en realidad incorpora 3 leds en su interior, el funcionamiento del LED es idéntico al de un diodo que solo deja pasar la corriente en un sentido pero la diferencia es que ese paso de corriente genera un salto de electrones en el elemento semiconductor produciendo luz.

Las luces led actualmente están reemplazando a todos los sistemas de iluminación tradicionales ya que son mucho más eficientes a nivel de consumo así como en durabilidad.

El led RGB dispone de un cátodo (NEGATIVO -) común y tres ánodos (POSITIVOS +) donde se conectan las salidas de la placa, en total para controlar un solo LED RGB serán precisos 4 cables.



Fig-8. Led RGB

### - Led Direccionable RGB

Al igual que en el caso anterior se ha incluido otro tipo de led RGB el concepto principal de funcionamiento es idéntico al primero opera como un LED RGB normal, pero la manera de seleccionar el color y que led encender o apagar se realiza mediante un protocolo 1-Wire, de este modo es posible controlar un gran número de LEDs simplemente con 3 cables, 2 de alimentación y uno de control.

En el caso anterior sería muy elevado el número de líneas de salida para controlar el mismo número de LEDS, en el caso del kit se controlarán 6 leds solamente con 3 cables.

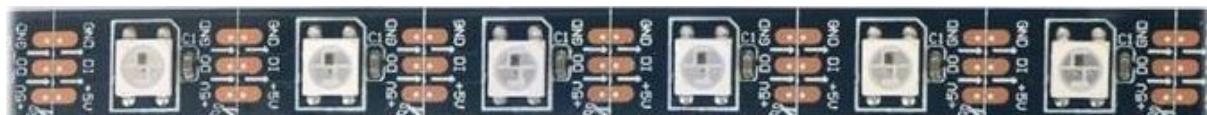


Fig-9. Tira de LED RGB direccionable

### - Micro Ventilador 3/5VDC

Con el kit se incluye también un pequeño ventilador para la realización de los casos de uso que se presentarán más adelante, este pequeño ventilador tiene un consumo de aproximadamente 120mA con lo que será preciso conectarlo a una salida especial que pueda entregar esa cantidad de corriente, cabe recordar que normalmente las salidas y entradas digitales tienen un límite de unos 25mA de corriente.

Por ese motivo el ventilador se conectará a la salida OUT de la ESP32N1, que es la salida del segundo regulador.

Lo interesante de los ventiladores construidos de este modo es que el motor principal no posee escobillas y por lo tanto no existe un desgaste en las mismas, estos ventiladores por lo tanto son diseñados para durar mucho tiempo en el equipamiento que los incorpora, en nuestro caso lo seleccionamos por su seguridad y facilidad en el uso, ya que cumple el propósito de demostrar el concepto de funcionamiento de unidades más grandes que se conectan por ejemplo mediante un relé a la red eléctrica tradicional de 220V.



Fig-10. Micro Ventilador 3/5VDC

### - Zumbador (Buzzer)

El zumbador o buzzer es un elemento formado por un material piezoelectrónico, esto es, al recibir un impulso eléctrico el material altera su estado mecánico y vibra en este caso produciendo un sonido.

Al igual que sucede en otros componentes el buzzer tiene una polaridad que hay que respetar para que funcione.

Existen dos tipos de buzzer activos y pasivos.

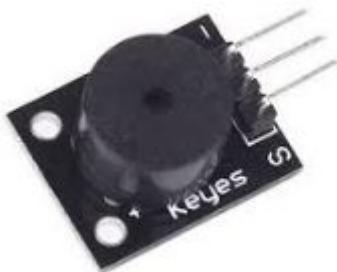


Fig-11. Zumbador (Buzzer)

### - Antena Externa WiFi / Bluetooth / BLE

En el kit se incorpora también una antena externa para ofrecer una mejora en la cobertura WIFI / Bluetooth / BLE, una antena en realidad se puede considerar como sensor cuando recibe ondas y como actuador al mismo tiempo cuando las emite.

La antena se conecta mediante un conector marcado en la placa como UFL1, de este modo se mejora la calidad de envío y recepción de la señal.



Fig-12. Antena externa con conector UFL

### - Batería de Iones de Litio (Li-Ion)

Otro de los elementos que se han añadido al kit ha sido una batería de 3200mAh de Iones de Litio , como el objetivo del kit es proporcionar una formación enfocada al IoT Internet de las Cosas es importante que se diseñen soluciones por lo tanto que puedan ser alimentadas desde una batería.

El formato de batería seleccionado es el 18650 dicho formato es bastante común y permite una relación tamaño / capacidad eléctrica bastante buena.

Este tipo de baterías son las que se emplean habitualmente en los ordenadores portátiles , vehículos eléctricos, etc en esos casos cada una de las pilas 18650 se unen en serie o paralelo para formar unidades más grandes a las que se denomina también baterías en ese caso cada una de las pilas 18650 pasa a ser denominada celda.

Uno de los principales fabricantes de batería del mundo es Panasonic y la batería incluida en el kit es la NCR18650B de dicho fabricante.



Fig-13. Bateria NCR18650B de Li-Ion de Panasonic

### - Cables Dupont de colores

El kit además incluye un conjunto de cables dupont hembra-hembra de colores para realizar las conexiones entre los diferentes dispositivos.

En el apartado de los casos prácticos en los esquemas de conexión se ha empleado la herramienta Fritzing y se usan los colores en los cables para poder estandarizar las conexiones.



Fig-14. Set de cables dupont de colores

## 2.3. Placa ESP32N1

La placa ESP32N1 es la placa principal sobre la que se desarrollan todos los casos prácticos contenidos en el kit.

Esta pieza de hardware es fruto del trabajo y desarrollo del equipo whitecatboard.org durante mas de 2 años de prototipos hasta la versión final de la placa.

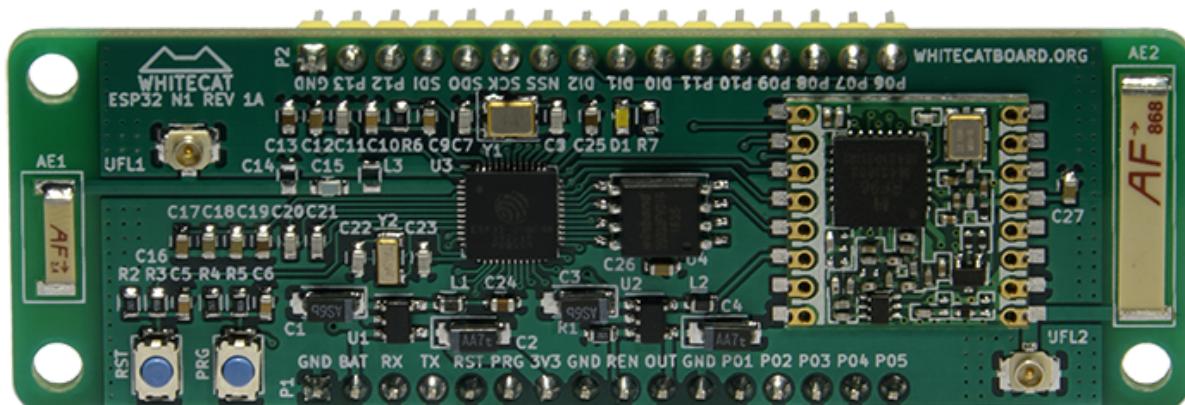


Fig-15. Placa ESP32N1

Esta placa está diseñada teniendo en mente su versatilidad y la comunidad que existe detrás del procesador principal las características principales de dicha placa son;

- Procesador ESP32 de Espressif 2 núcleos hasta 240Mhz
- Conectividad Bluetooth y BLE
- Conectividad WiFi
- Conectividad LoRa

A continuación se detallan empleando las diferentes partes del esquemático de la placa cada una de las partes que la componen.

La alimentación principal de la placa ESP32N1 se realiza a través del pin marcado como BAT dicho pin permite un voltaje de entrada de 2,7 a 5,5 voltios y está pensado para ser alimentado o bien por una fuente de 5V como podría ser un puerto USB o bien una fuente de 4,2V cómo sería una batería cargada de Iones de litio (Li-Ion).

En este caso como se verá en el apartado de la placa DEVKIT la celda de alimentación escogida es el formato estándar 18650 por la posibilidad de encontrar dichas baterías de un modo bastante extendido, y las capacidades de carga que se explicarán más adelante en los casos de uso.

La entrada de voltaje desde los 2.7V hasta los 5.5V pasa a través de un regulador DC-DC llamado step-down que rebajara el voltaje por ejemplo de 5V a 3.3V que es el voltaje al cual funcionan el resto de componentes de la placa.

Por eficiencia energética se ha empleado un step-down (comutado) en vez de un LDO (térmico), la diferencia entre los dos sistemas reside en que mientras el primero baja la tensión mediante la comutación de la salida (ON/OFF) muy rápidamente en general a frecuencias de 2Mhz (2 millones de veces por segundo) lo cual resulta muy eficiente desde el punto de vista de los consumos, el segundo sistema rebaja la tensión simplemente transformando la energía sobrante en calor y por consiguiente disipando al ambiente y por lo tanto desperdiando dicha energía en modo de calor esta circunstancia no lo hace un sistema apto para aplicaciones de bajo consumo o alimentadas por baterías.

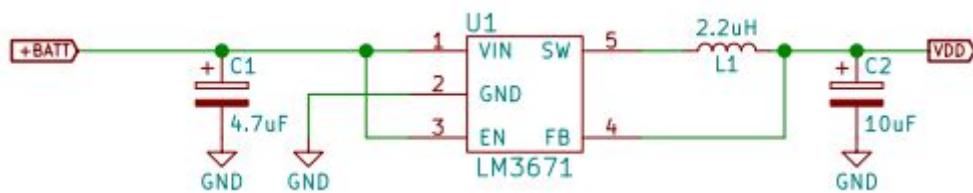


Fig-16. Esquema alimentación principal de entrada

El mismo sistema que se emplea en la alimentación principal se usa también para el regulador secundario, en este caso mediante un pin de entrada/salida del ESP32 controlamos el estado de dicho regulador , permitiendo así que este esté operativo o no en función de las necesidades de comunicación, sensorica, etc. este segundo regulador está conectado a su salida en el pin marcado como OUT y al mismo tiempo también alimenta el módulo HopeRF de LoRaWAN, el pin de control REG\_EN para (ON/OFF) es el GPIO27.

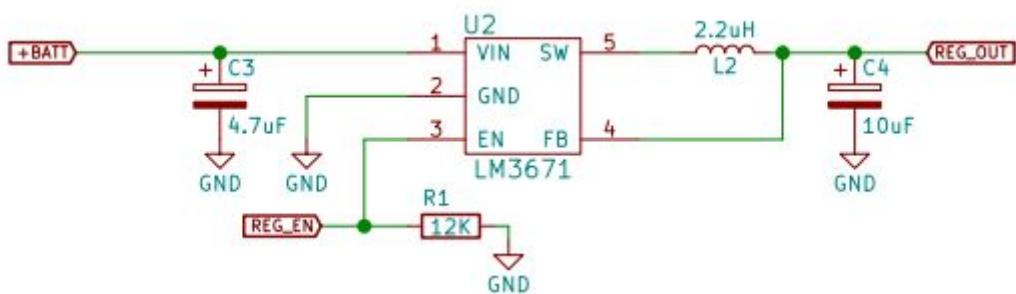


Fig-17. Esquema alimentación secundaria de salida

La placa incorpora una memoria de almacenamiento externa para poder guardar el programa y los datos, en este caso el sistema operativo LuRTOS y el sistema de ficheros que acompaña a dicho sistema operativo incluyendo los ejemplos que vienen preinstalados de serie.

La memoria flash está alimentada desde un pin especial del procesador principal ESP32 , cuando este entra en modo de reposo la memoria deja de estar alimentada para el consiguiente ahorro de energía.

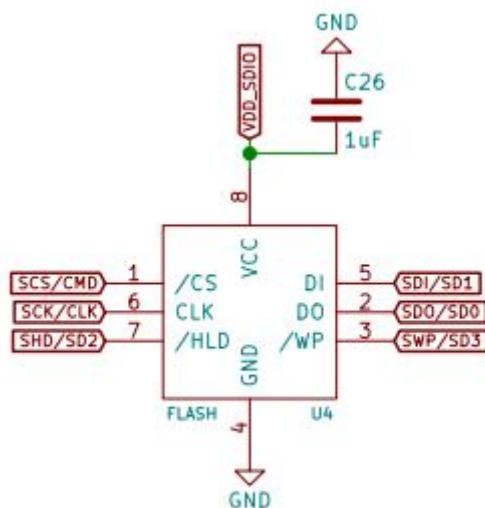


Fig-18. Esquema memoria de almacenamiento flash

Para su conexionado con elementos externos o bien con la placa de desarrollo DEVKIT la ESP32N1 cuenta con 2 tiras de pines una a cada lado con 16 conexiones cada una en la siguiente figura se puede observar cada uno de esos pines en el esquemático y en la figura que le sigue se detallan cada una de las funciones de dichos pines.

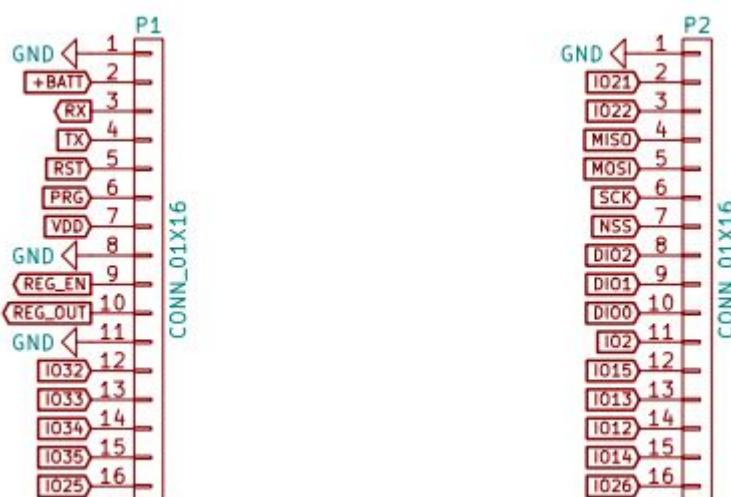


Fig-19. Esquema pines de conexiones ESP32N1

En la siguiente figura se describen cada una de las funciones que incorporan los pines laterales de la placa ESP32N1.

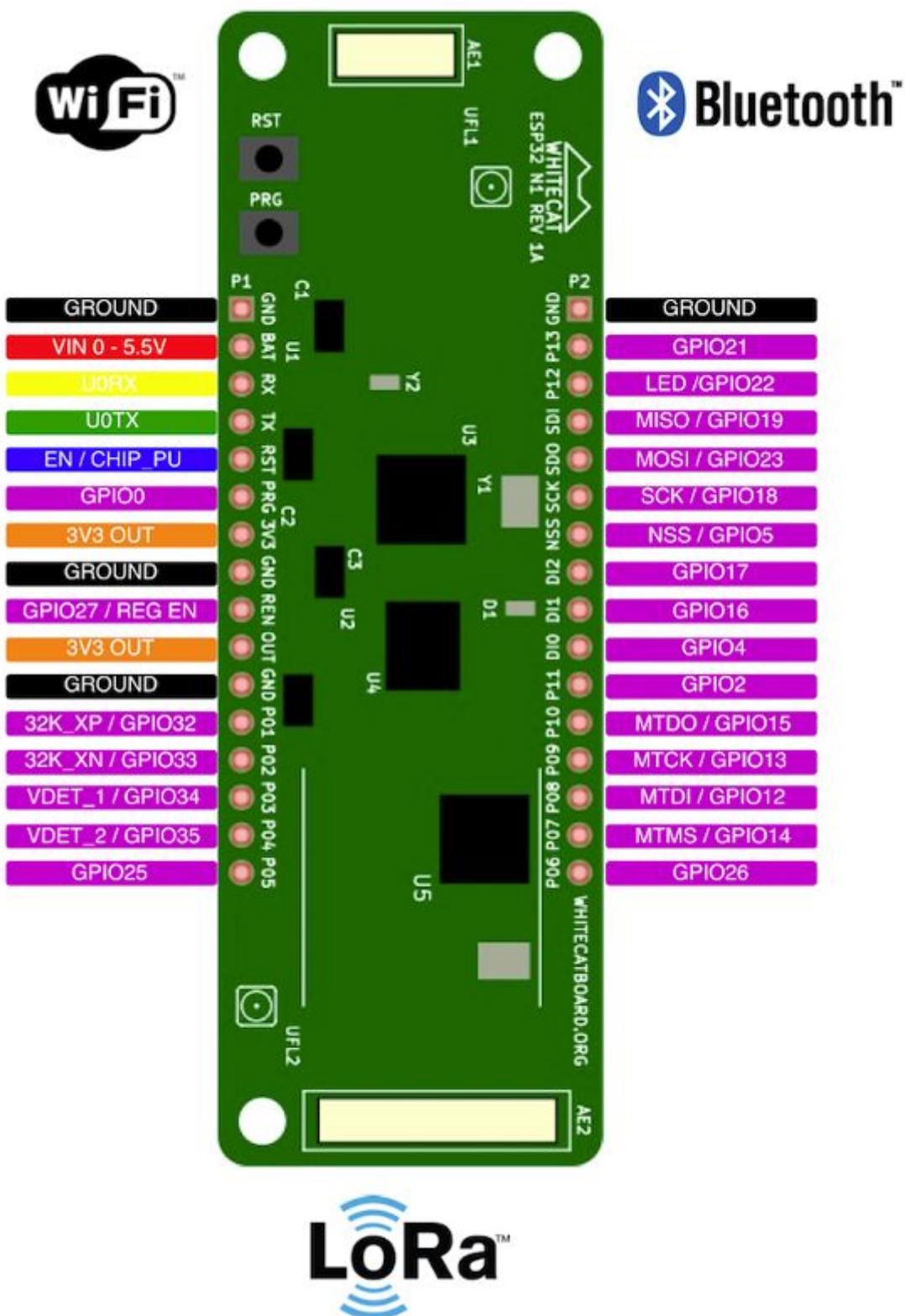


Fig-20. Resumen pines placa ESP32N1

La placa cuenta con un botón para producir un reinicio por hardware, dicho tipo de reinicio se emplea en el caso de un fallo inesperado que dejase la placa en un estado bloqueado y principalmente es empleado para entrar en el modo bootloader conjuntamente con el botón de programación.

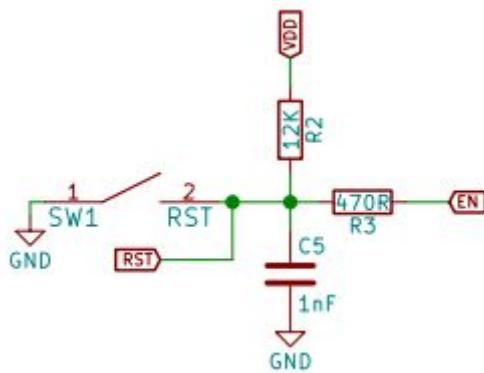


Fig-21. Esquema botón de reset

Mientras se reinicia la placa ESP32N1 si se mantiene pulsado el botón de programación PRG se entra en un modo especial que permite subir el programa desarrollado a la memoria flash, en el caso de empleo de la placa con el IDE de bloques de LuaRTOS esto no será necesario usando la placa DEVKIT ya que esta posee un sistema “autoprogram”

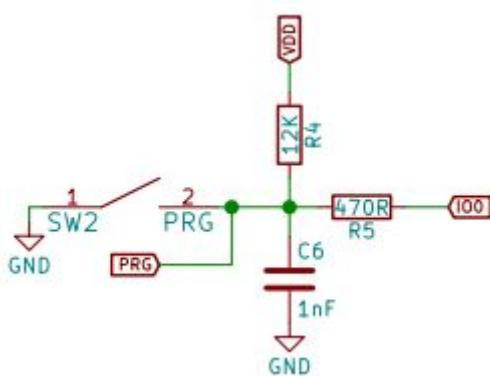


Fig-22. Esquema botón de programación (modo bootloader)

El módulo HopeRF de radio, es el encargado de realizar las comunicaciones en la banda LoRaWAN de cada país por defecto en la Unión Europea se monta el módulo en su versión de 868Mhz y la antena cerámica correspondiente de la misma frecuencia, el uso de antena cerámica simplifica y acorta los tiempos de diseño de la placa, y permite mayor versatilidad ya que se pueden adaptar en el mismo formato diferentes frecuencias sin necesidad entonces de cambiar el diseño.

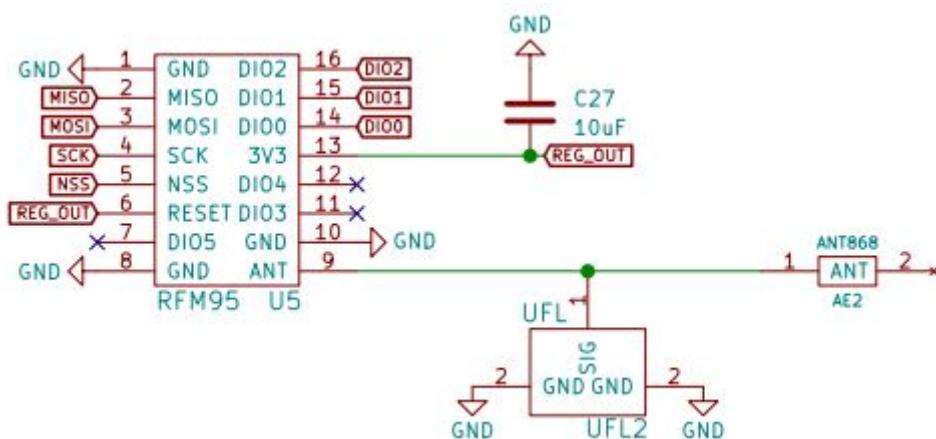


Fig-23. Esquema radio LoRaWAN (transceiver HopeRF)

Existe un led de estado conectado al IO22 , este led con LuaRTOS cargado en memoria y ejecutándose parpadea una vez por segundo.

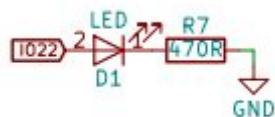


Fig-24. Led de estado

Por último la parte principal del módulo ESP32N1 es la CPU a continuación se muestra un esquemático con las diferentes partes que integran el circuito y hacen posible su funcionamiento.

En la línea LNA\_IN se encuentra la antena de WiFi y bluetooth así como un conector UFL para la antena externa, se ha provisionado el pcb para poder realizar un match network y optimizar los parámetros de la antena cerámica en placa.

El componente marcado como Y1 es el cristal de cuarzo principal que hace funcionar la CPU este opera a 40Mhz internamente mediante el uso de PLL se incrementa la frecuencia de funcionamiento hasta los 240Mhz.

El resto de componentes son condensadores de desacoplo, condensadores en las salidas / entradas SENSOR y condensadores en CAP1 y CAP2.

El oscilador de 32.768Khz no está operativo en la revisión actual de la CPU ya que forma parte de una de las erratas, por tal motivo es una de las partes que no están soldadas en la placa físicamente.

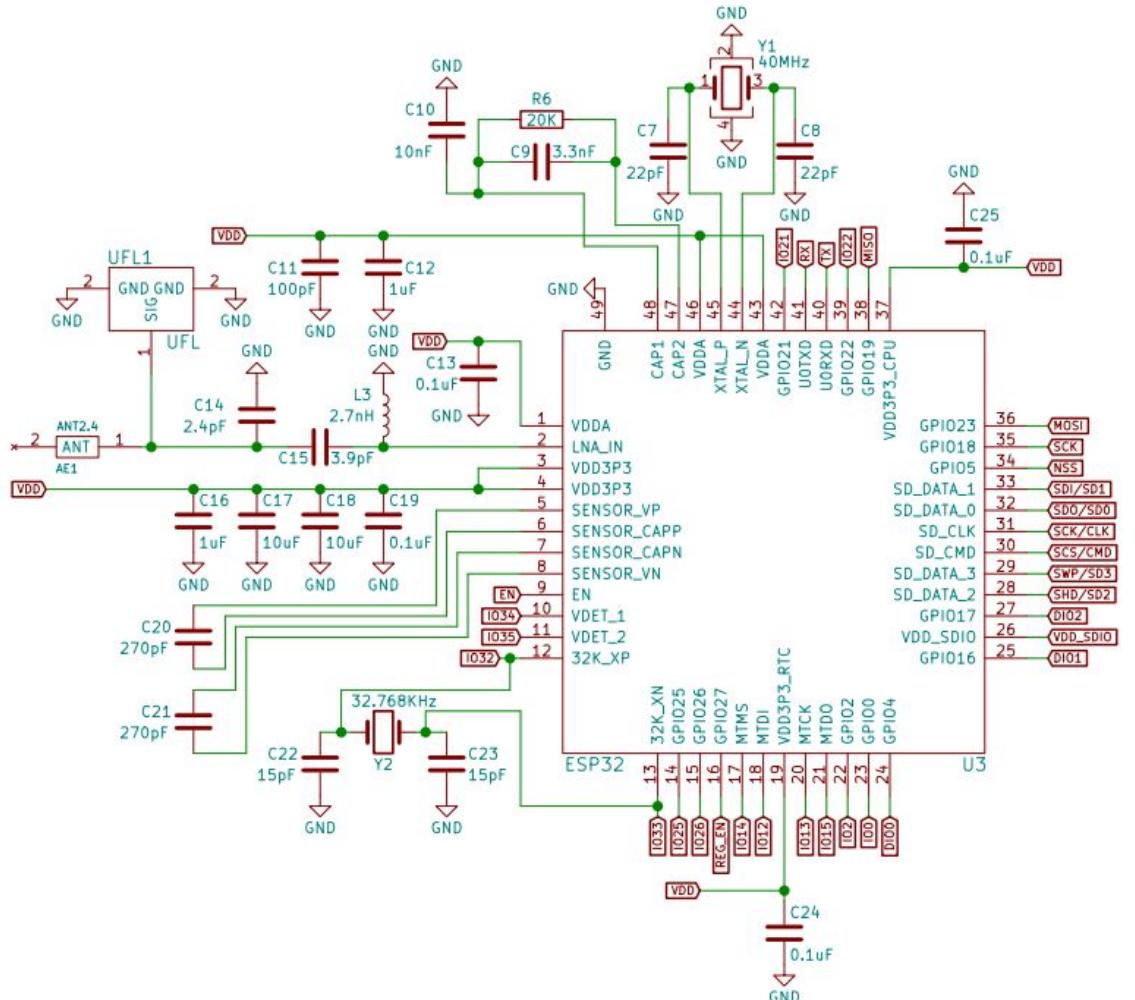


Fig-25. Esquemático de la CPU ESP32

Como última nota sobre la placa ESP32N1 se muestra una imagen con la antena externa conectada en el puerto UFL1, dicha antena mejora las prestaciones de la WiFi y el Bluetooth en aquellas aplicaciones donde sea preciso y no sea suficiente con la antena cerámica incluida.

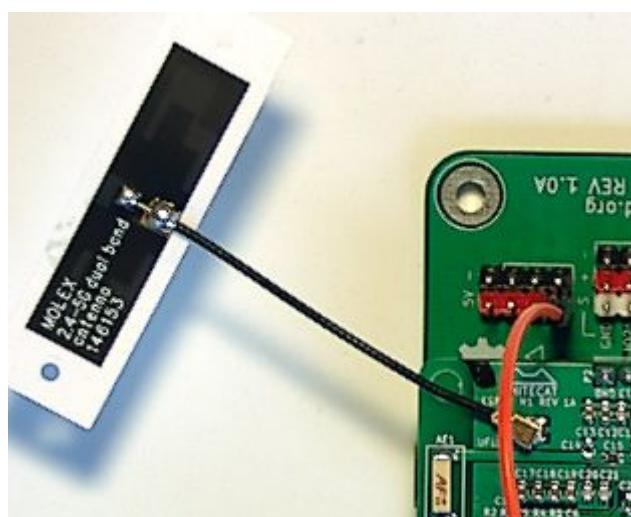


Fig-26. Detalle conexión antena externa WiFi / Bluetooth en conector UFL1

## 2.2. Placa DEVKIT para ESP32N1

La placa ESP32N1 DEVKIT es un componente de hardware que mejora las prestaciones de la placa ESP32N1 incrementando sus funcionalidades así como facilitando al mismo tiempo la interconexión de otros dispositivos, la comunicación con el PC y la programación automática desde el entorno de desarrollo WhitecatIDE.

A continuación se muestra una imagen de la placa ESP32N1 DEVKIT.

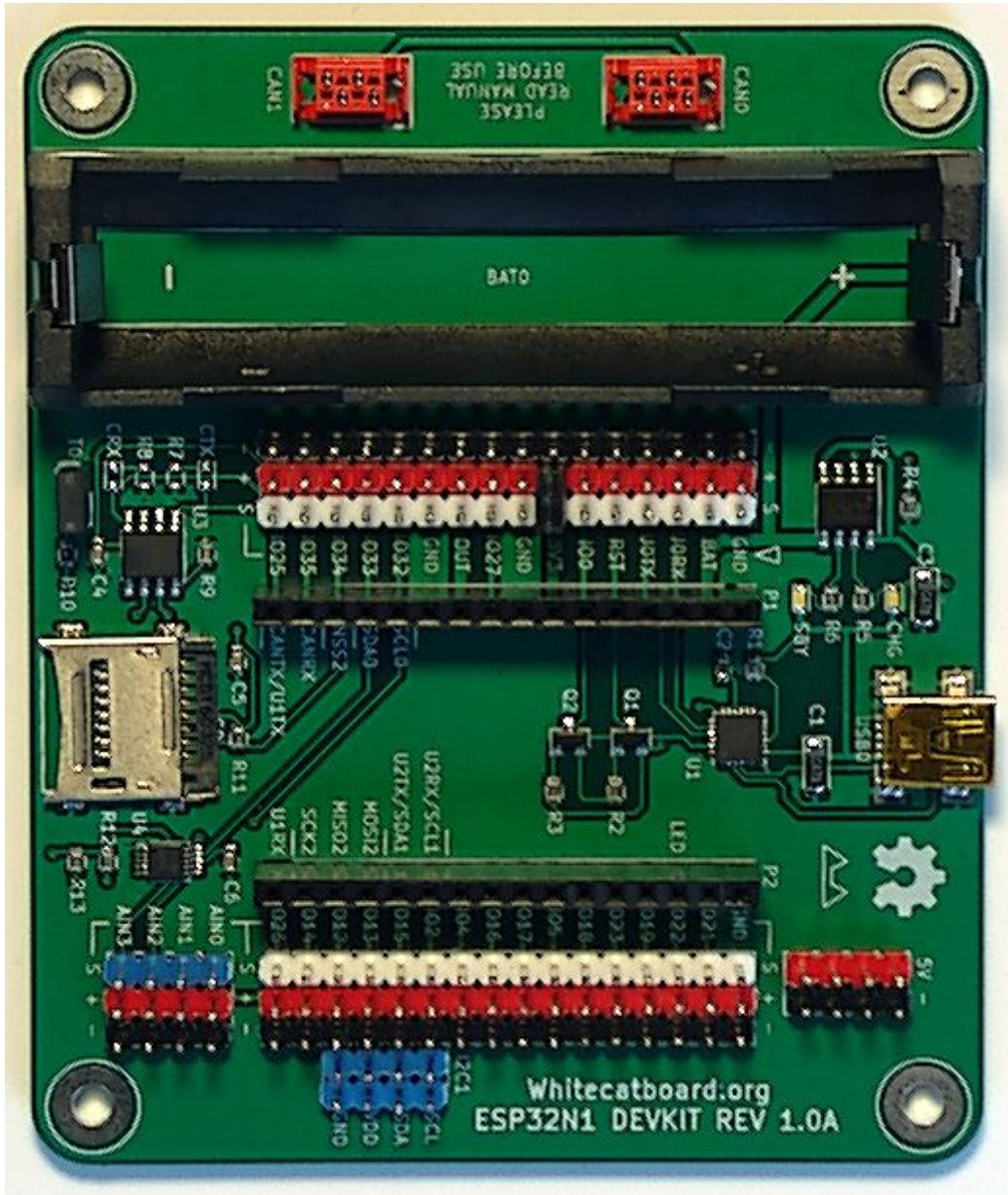


Fig-27. Placa de desarrollo ESP32N1 DEVKIT Revisión 1.0A

La placa ESP32N1 se inserta en los pines de la ESP32N1 DEVKIT como se muestra a continuación en la siguiente imagen.



**ES IMPORTANTE TENER EN CUENTA LA MANERA DE INSERTAR LA PLACA ESP32N1 SOBRE LA PLACA DE DESARROLLO YA QUE UNA INSERCIÓN INCORRECTA PODRÍA DAÑAR LAS 2 PLACAS.**

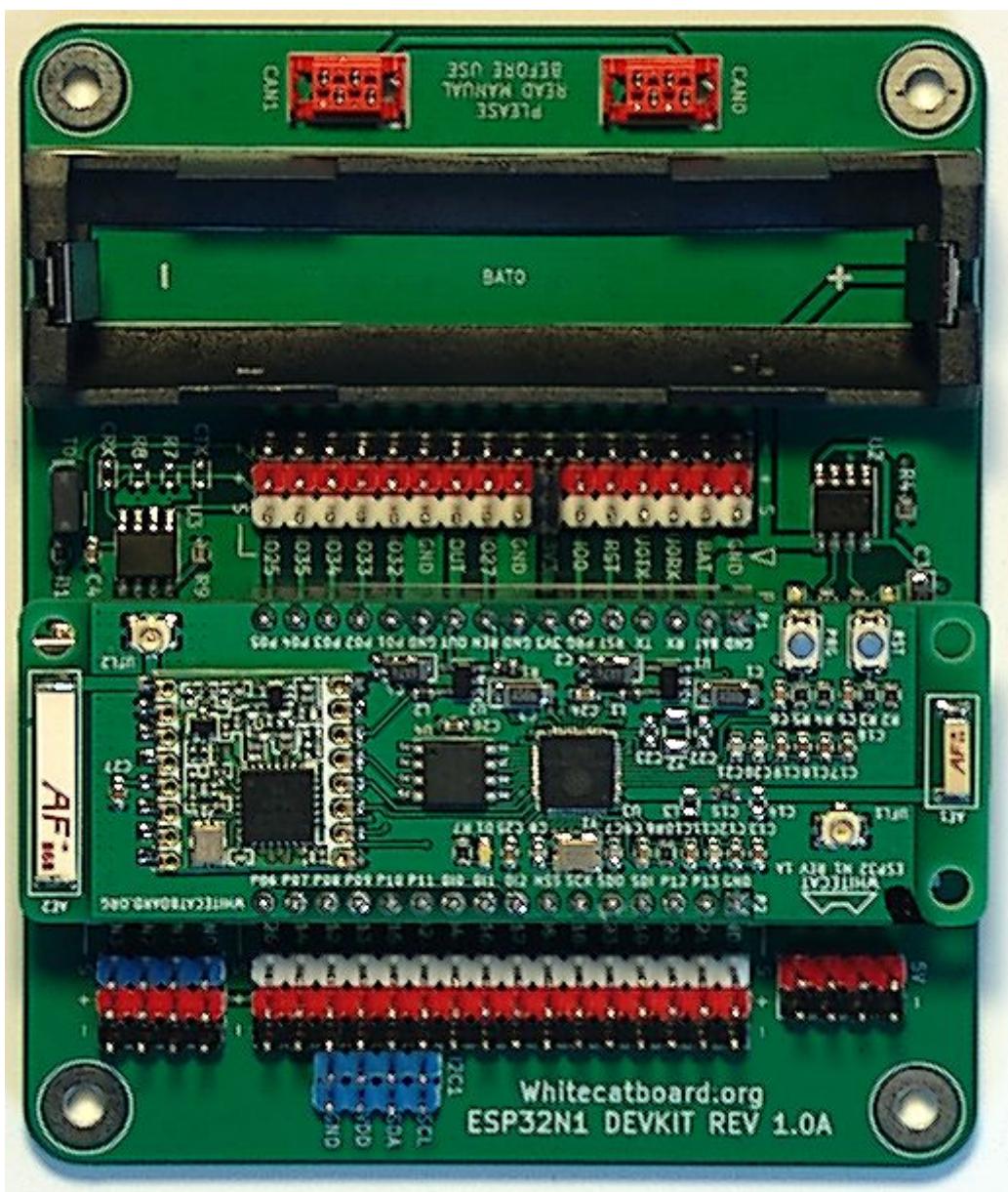


Fig-28. Placa ESP32N1 Insertada en la placa de desarrollo DEVKIT, recordar que es importante la manera en cómo se coloca la placa para evitar dañarla.



La revisión 1.0A de la placa cuenta con un cargador de baterías de Ion de Litio (Li-Ion) incorporado, ya que es posible que la batería no esté presente al no ser fija en la configuración de la placa es preciso usar un cable desde cualquier pin marcado como 5V hasta el pin de señal marcado como BAT.



**ATENCIÓN !! SOLAMENTE ES PRECISO USAR ESTE CABLE SI LA BATERIA NO ESTA PRESENTE EN EL ZÓCALO MARCADO COMO BAT0.**

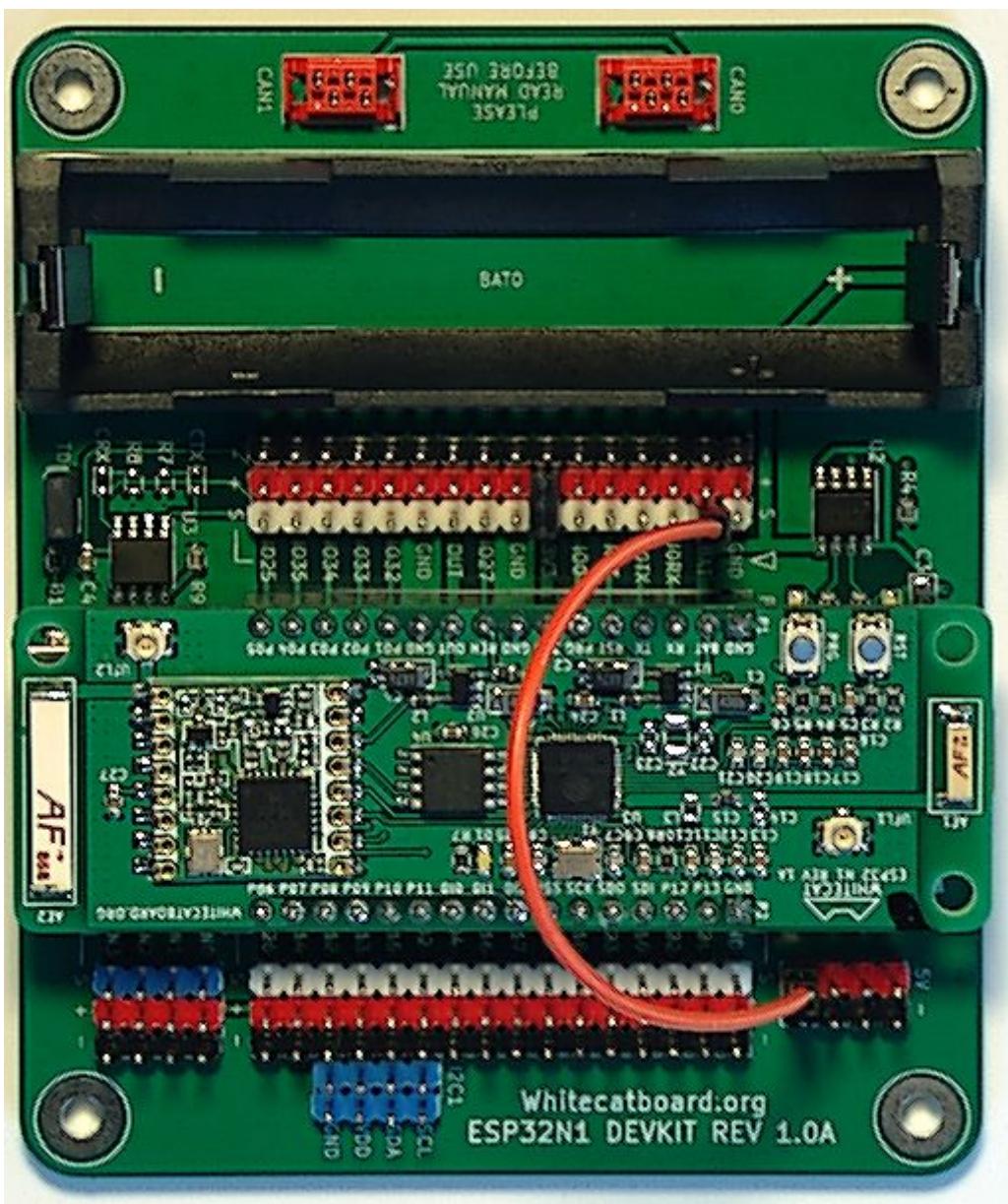


Fig-29. Cable usado cuando la batería no está insertada en el zócalo marcado como BAT0

Si por el contrario hemos insertado la batería en el zócalo marcado como BAT0 tal y como muestra la imagen es importante que no esté presente el cable desde el pin 5V hasta el pin de señal BAT.



**QUITAR EL CABLE ENTRE 5V Y BAT SI TENEMOS INSERTADA LA BATERÍA PARA UN CORRECTO FUNCIONAMIENTO, ES IMPORTANTE EXTREMAR LAS**

**PRECAUCIONES SIEMPRE QUE SE TRABAJAN CON BATERÍAS , SI POR ALGÚN MOTIVO NOTASE EXCESO DE TEMPERATURA EN LA BATERÍA DESCONECTE LA PLACA INMEDIATAMENTE.**

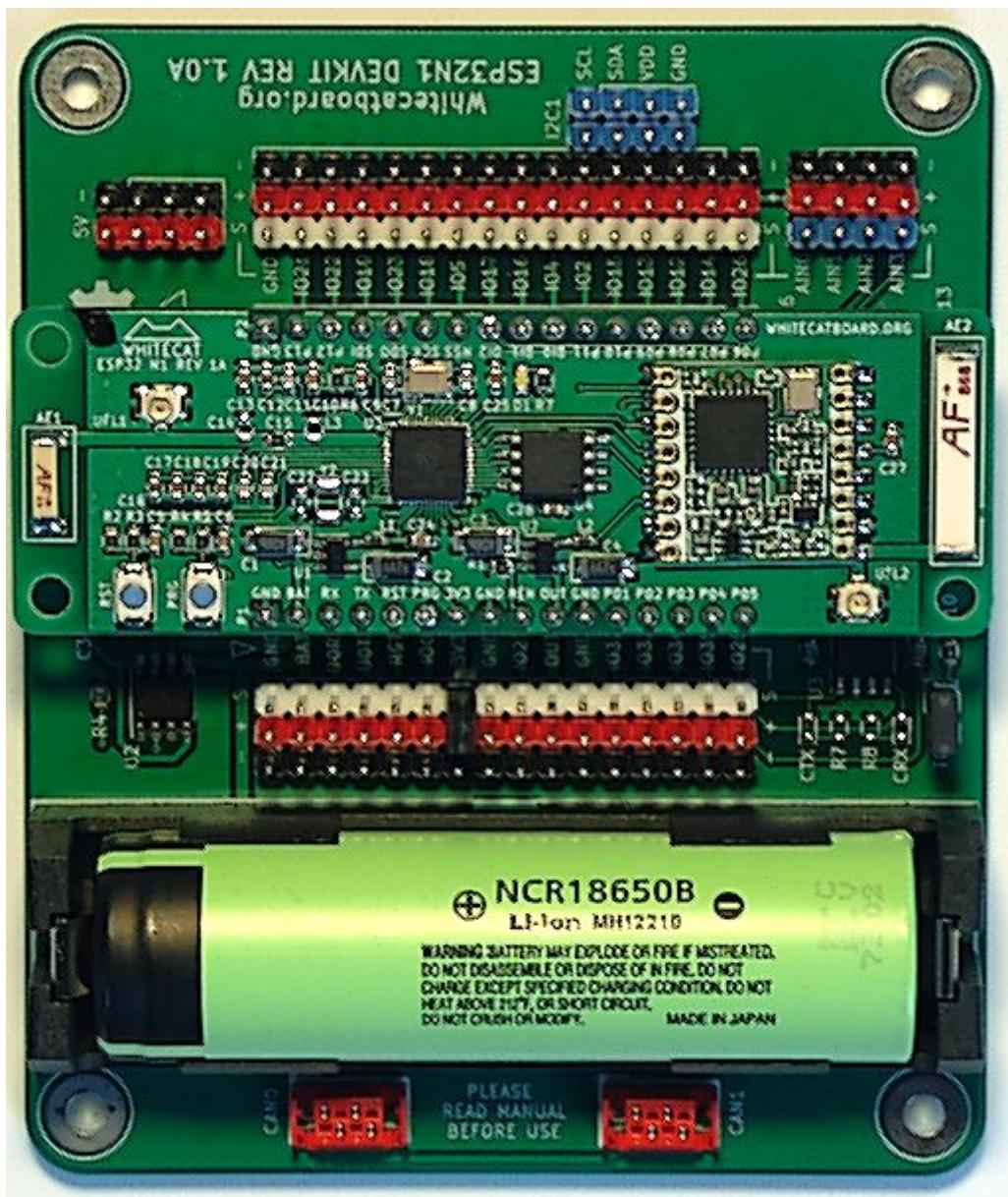


Fig-30. Placa DEVKIT con la batería insertada en el zócalo BAT0, es importante quitar el cable mencionado anteriormente cuando esta sea la configuración.



**Una vez repasado el punto inicial de la batería, por favor tenga en cuenta que las baterías no son un juguete si el kit va ha ser usado por un menor recomendamos o bien que no se use la batería o bien que exista supervisión por parte de un adulto, si se hace uso de ella.**

A continuación procederemos a detallar como en el caso de la ESP32N1 las diferentes partes de la placa de desarrollo siguiendo el esquemático.

La parte inicial es la que se conecta al ordenador mediante un conector USB tipo Mini-B esta conexión aportará alimentación a la placa así como una conexión serie sobre el puerto usb necesaria para la comunicación con la placa.

La conexión serie a USB la realiza el integrado Silabs CP2104, que además soporta el uso de las líneas TX / RX / RTS y DTR que serán precisas además para la programación automática como veremos más adelante.

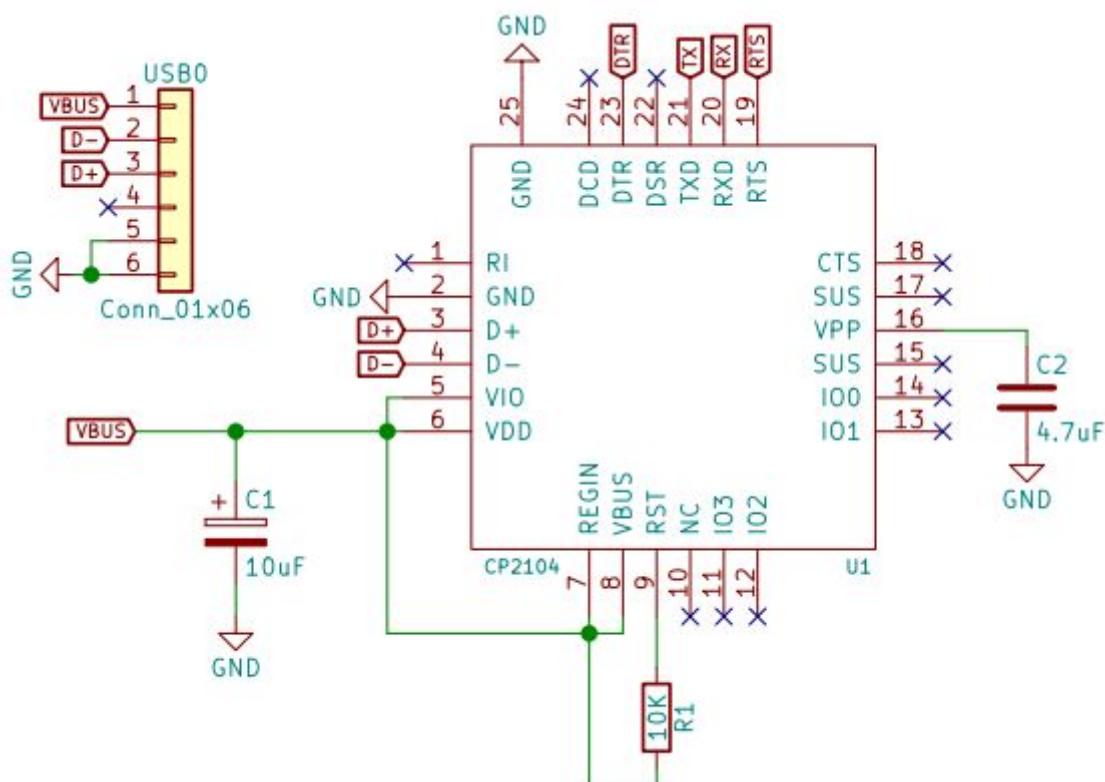


Fig-31. Esquemático del Integrado Silabs CP2104

Como hemos comentado a parte de la conectividad la conexión USB proporciona a la placa energía eléctrica, en este caso a través del cargador de baterías TP4056 se suministra un volumen de tensión máximo de 4.2V y un volumen de corriente máximo de 580mA.

El cargador de baterías TP4056 es un cargador del tipo voltaje/corriente es un circuito integrado para la carga de baterías bastante popular y extendido, en su caso es capaz solamente de cargar y gestionar una sola celda como es el caso en la placa DEVKIT.



**ATENCIÓN, RECUERDE EL USO DE BATERÍAS DEBE ESTAR SUPERVISADO SI APRECIA CUALQUIER ANOMALÍA O EXCESO DE TEMPERATURA POR FAVOR DESCONECTE LA PLACA.**

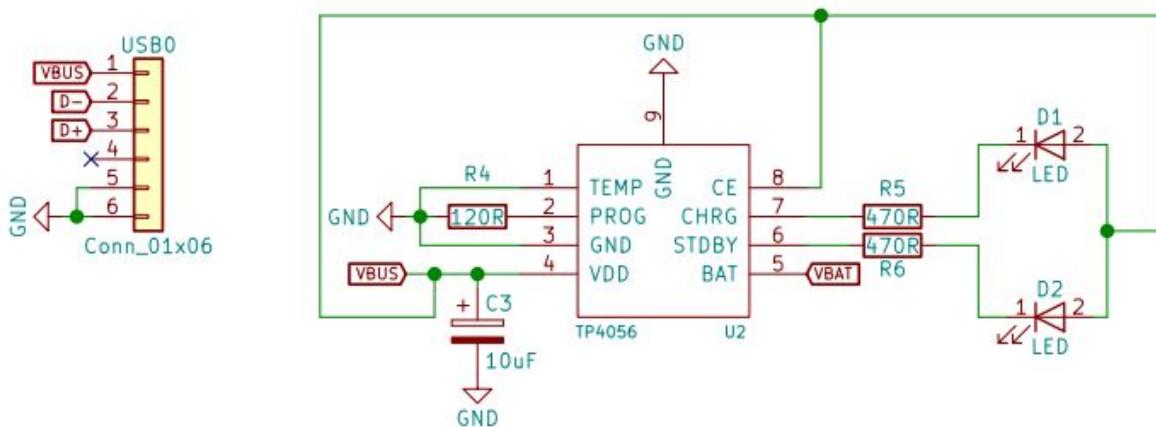


Fig-32. Esquemático del circuito integrado TP4056 encargado de cargar la batería y suministrar energía a la ESP32N1.

La siguiente ampliación que presenta la placa de desarrollo es la implementación del transceiver para comunicaciones CANBus este tipo de red es una red con 2 cables CANH y CANL desarrollada por BOSCH principalmente para su uso como bus principal de comunicaciones en vehículos pero que también es ampliamente usado en entornos industriales.

A bajas velocidades la longitud de los cables empleados en CANBus puede sobrepasar los 200 metros, por tal motivo es un tipo de red interesante para instalaciones industriales, domóticas, etc.

La ventaja del CANBus radica aparte de en la inmunidad de la señal permitiendo su uso en entornos con grandes interferencias, en el concepto de que cada elemento del bus puede ser al mismo tiempo maestro o esclavo no teniendo las limitaciones que tenían los buses 1-WIRE , i2c y SPI.

A continuación se muestra el esquemático del transceiver de Texas Instruments empleado para dar conectividad canbus a la placa ESP32N1.

El Jumper (puente) situado en T0 es para añadir una terminación de 120OHM al bus CANBus.

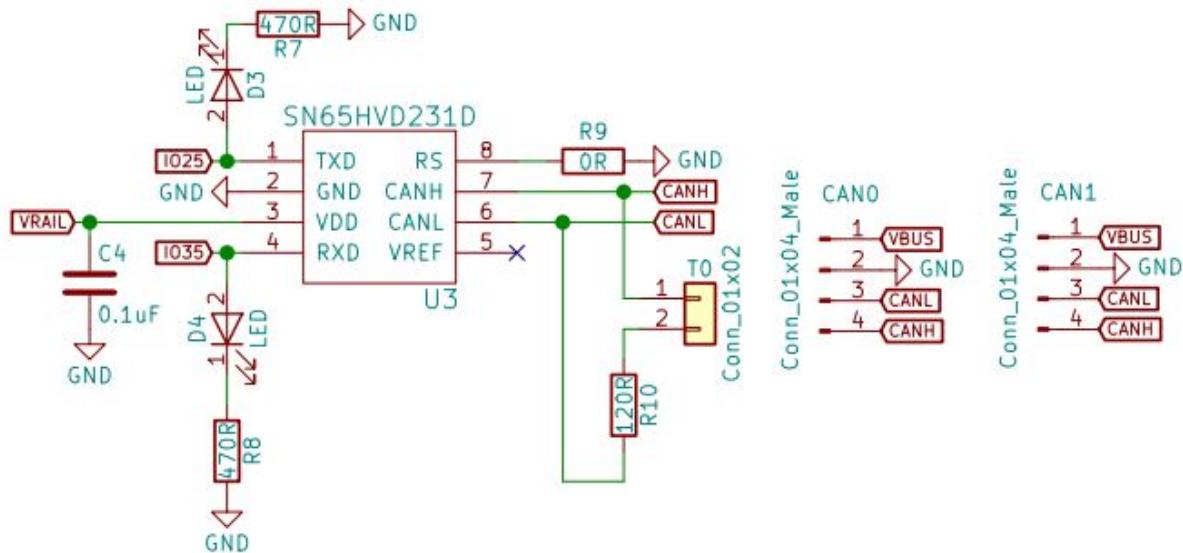


Fig-33. Esquemático del transceiver SN65HVD231D para conectividad CANBus.

Por cierto , que es un transceiver (transceptor) ?

Pues es un elemento que contiene un emisor y un receptor en el mismo encapsulado, en este caso orientado a un propósito como es la comunicación en CANBus.

Debido a las limitaciones en el apartado de conversores analogicos a digital que posee la cpu ESP32 se ha optado también por añadir en la placa de desarrollo un circuito integrado específico para dicho propósito, presentando así una flexibilidad y valores mucho más precisos.

El mundo fisico es analogico por eso cuando se quieren medir cosas que suceden mediante un sensor en muchas ocasiones es preciso el uso de conversores analogicos a digital ADC

, estos reciben una señal analógica una magnitud y la codifican a unos y ceros (10011001) la cantidad de unos y ceros que pueden emplear para dar un valor de salida se denomina la resolución, en el caso de la revisión 1.0A de la placa de desarrollo hemos incorporado el conversor ADC ADS1015 de 12 bits de resolución es decir doce unos o ceros para indicar el valor medido.

Existe la contrapartida a la entrada ADC o conversor Analogico Digital , que es el conversor Digital Analogico DAC el propósito como su propio nombre indica es exactamente el opuesto al ADC y convierte un valor digital de unos y ceros en un valor analógico, en este caso nosotros lo llamaremos PWM y lo implementaremos en los pines de la ESP32N1 directamente mediante las funciones de LuaRTOS.

A continuación se muestra el esquemático del conversor analógico - digital ADC.

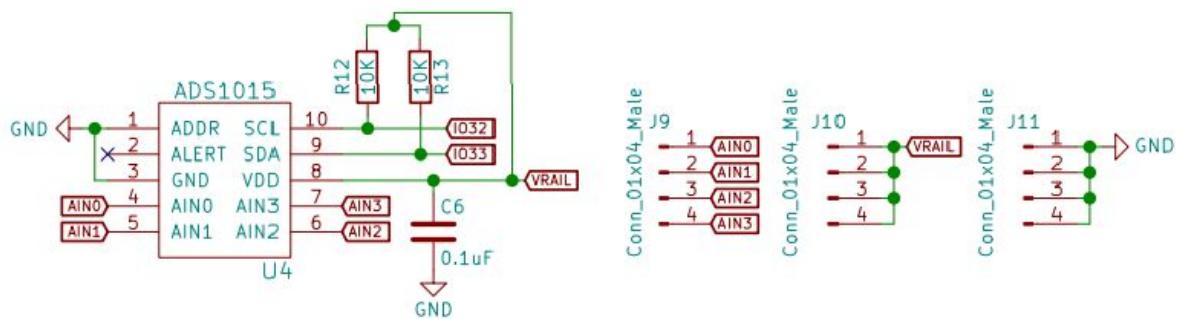


Fig-34. Esquematico conversor analogico-digital ADC 12/16 bit

En este caso además comentar que el circuito integrado que realiza las mediciones es el mencionado ADS1015 de 12 bit se puede emplear un ADS1115 en futuras revisiones que ofrecería 16bit.

El circuito integrado ADS1X15 funciona mediante bus i2c y en el caso de la placa de desarrollo este está conectado al bus i2c (0) que incluye los pines SCL (IO32) y SDA (IO33) así mismo la dirección en el bus del chip ADS1X15 es la 0.

Por último se muestran los diferentes pines de señal que están directamente vinculados a la placa ESP32N1 a estos pines se les ha añadido un pin de alimentación y un pin de masa común.

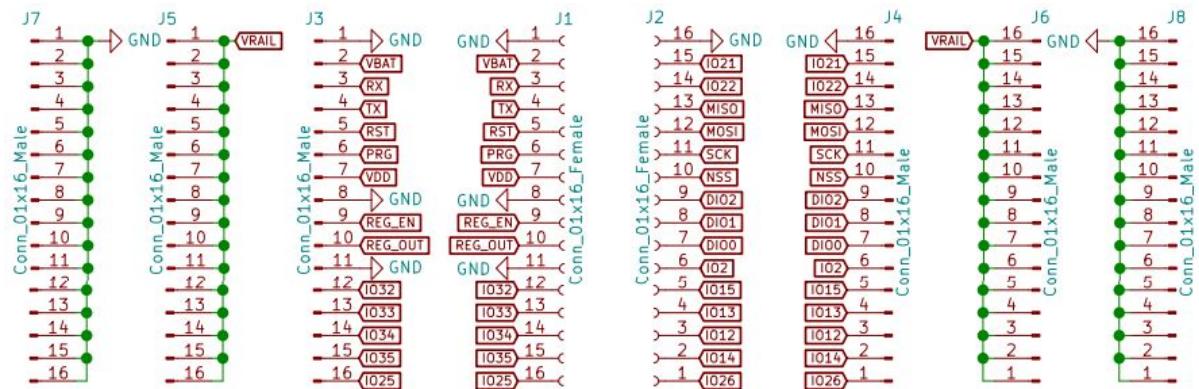


Fig-35. Esquemático de los pines de la placa de desarrollo

Para facilitar la identificación de los diferentes pines se ha ampliado la serigrafía en la placa de desarrollo indicando a qué pin GPIO corresponde cada uno de ellos.

Por otro lado también se han marcado con diferentes colores los pines en la placa siendo de color rojo los pines de alimentación, negro los pines de masa y blancos o azules los pines de señal.

Se han incorporado también para facilitar su conexiónado varios pines al puerto I2C 1 como muestra la imagen siguiente.

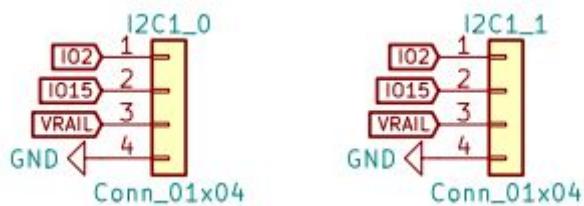


Fig-36. Esquemático pines I2C-1

### 3. Entorno de Desarrollo Whitecat IDE

El entorno de desarrollo del ecosistema Whitecatboard se llama WhitecatIDE, dicho entorno está basado en tecnología HTML5 y se ejecuta desde dentro de un navegador de tal

manera que el usuario siempre dispondrá de la versión más actualizada de dicho entorno de desarrollo sin necesidad de instalarlo en su PC.

El único elemento que se deberá instalar en el PC el usuario es el agente que es el encargado de comunicar el navegador con la placa en local en el PC.

El entorno de desarrollo de WhitecatIDE permite el trabajo principalmente con programación mediante bloques pero también permite una consola para la edición de texto.

A continuación se definen las funciones de los elementos principales que aparecen en la pantalla principal del IDE, es preciso acceder mediante una cuenta de Google ya que los ficheros de ejemplo se guardarán en una carpeta de drive del usuario.

La dirección para acceder al entorno de desarrollo es <https://ide.whitecatboard.org>

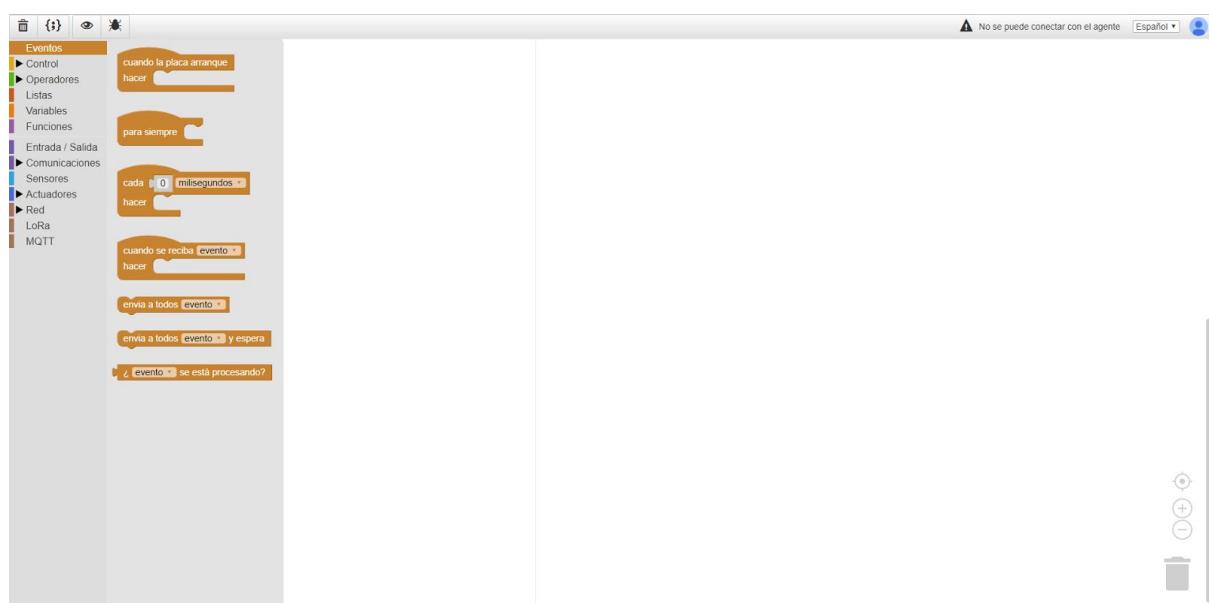


Fig-37. Pantalla principal del entorno de desarrollo en modo bloques



Fig-38. Pantalla principal del entorno de desarrollo en modo código

#### Barra superior

En la barra superior encontramos diferentes botones en función del estado , estado de comunicación con la placa, selector de idioma y perfil de usuario.



Botón para eliminar los bloques del área de trabajo.



Botón para alternar entre el modo de Bloques y el modo de Código.



Botón para abrir un programa existente, estos se pueden abrir desde la placa o desde el PC o desde la nube.



Botones para guardar el programa y guardar como ..., se pueden almacenar los programas tanto en la placa como en la nube.



Botón para reiniciar la placa, este botón provoca un reinicio de la placa por hardware.



Botón para parar la ejecución del programa en curso, si existe algun programa ejecutándose en la placa lo detiene.



Botón para la ejecución de un programa, este botón ejecuta los bloques o el código que tengamos en el área de trabajo de manera instantánea después de copiar el código a la placa.



Botón que muestra el código generado mediante los bloques se puede utilizar a modo de aprendizaje o revisión del código generado, no confundir con el botón que alterna entre modo codigo o bloques.



Botón que habilita el modo de depuración para la búsqueda de errores.

Whitecat N1 ESP32 DEVKIT ▾ |  test\_neopixel.xml | Español ▾ 

```

Lua RTOS beta 0.1. Copyright (C) 2015 – 2017 whitecatboard.org

build 1511857188
commit b8545c9a4ce242147bb5dbe92de9bcdedda5bf7
Running from factory partition
board type N1ESP32-DEVKIT
cpu ESP32 rev 0 at 240 Mhz
flash EUI d66634415a4e8010
spiffs0 start address at 0x310000, size 512 Kb, partition spiffs
spiffs0 mounted
sd0 is at spi2, cs=GPIO21
fat0 can't mounted
can't redirect console messages to file system, an SDCARD is needed

Lua RTOS beta 0.1 powered by Lua 5.3.4
Lua RTOS-boot-scripts-aborted-ESP32
/ > █

```

Mediante el desplegable es posible acceder a la consola de texto de la placa.



Los últimos elementos que se muestran en la parte superior son;

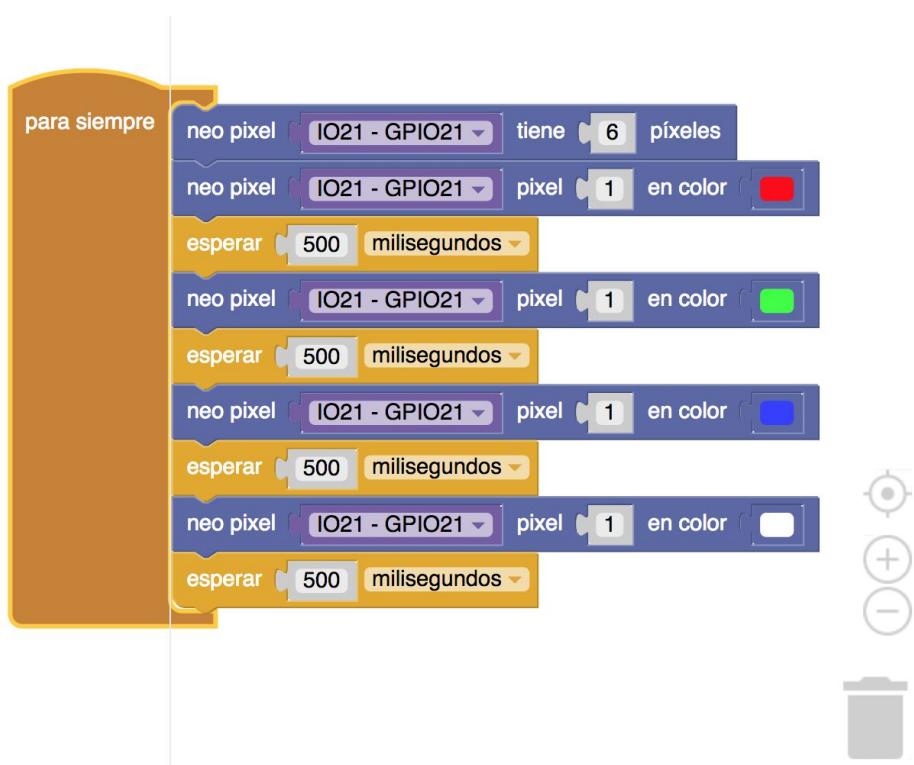
- Nombre del fichero con el que estamos trabajando.
- Selección de Idioma.
- Perfil de usuario para cerrar la sesión, etc.

## Área de bloques

A la izquierda encontramos el área de trabajo en ella están presentes todos los bloques que se pueden utilizar en el entorno de desarrollo, organizados por categorías y tipos.



## Área de trabajo

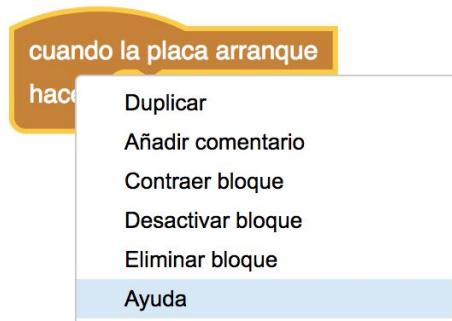


## Los Bloques

Los bloques son elementos que generan código a partir de unos procesos ya predefinidos por el equipo de desarrollo de whitecatboard.org, de este modo facilita enormemente la labor al usuario a la hora de desarrollar proyectos utilizando los bloques.



**IMPORTANTE, TODOS LOS BLOQUES ESTÁN DOCUMENTADOS MEDIANTE UN MENÚ CONTEXTUAL, EN EL PROPIO IDE UTILIZANDO EL BOTÓN DERECHO DEL RATÓN SOBRE UN BLOQUE PODREMOS ACCEDER A SU AYUDA EN LÍNEA.**



**Ayuda**

Este **bloque** es un **bloque sombrero** que se activa cuando se la placa arranca. Cuando se activa el bloque ejecuta su **script**. Cualquier otro bloque sombrero no se activará hasta que finalice este bloque.

Como se ha comentado antes, cualquier otro bloque sombrero no se activará hasta que finalice este bloque, por tanto, no es posible utilizar el bloque "[Envía a todos \(\) y espera](#)" Dentro de este bloque, o el programa se colgará.

Normalmente, este bloque se utiliza para:

- Inicializar las variables globales y las estructuras de datos.
- Reiniciar el hardware.
- Sincronizar la activación de otros bloques sombrero.

**Forma**

## 4. LuaRTOS Sistema Operativo del ecosistema Whitecatboard

Lua RTOS es un sistema operativo en tiempo real diseñado para funcionar en sistemas embebidos, con requisitos mínimos de FLASH y memoria RAM.

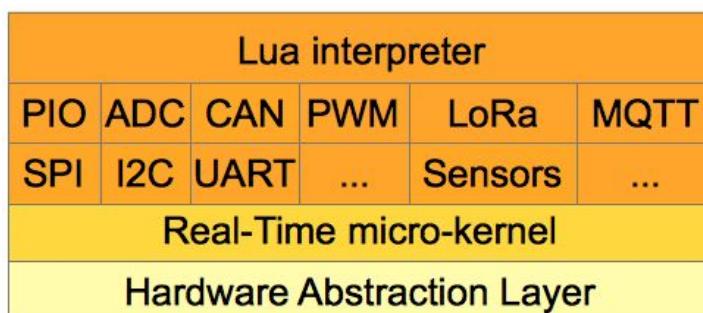
Actualmente, Lua RTOS está disponible para las plataformas ESP32, ESP8266 y PIC32MZ.

Los dispositivos que funcionan con Lua RTOS pueden ser programados directamente desde la consola del propio dispositivo, o mediante el entorno de desarrollo integrado The Whitecat IDE utilizando bloques, o Lua.

Lua RTOS ha sido desarrollado íntegramente en CitiLab Cornellà, Barcelona.

### Diseño en 3 capas

- En la capa superior corre un intérprete Lua 5.3.4 que ofrece al programador todos los recursos proporcionados por el lenguaje de programación Lua, además de módulos especiales para acceder al hardware (PIO, ADC, I2C, RTC, etc ...), y servicios de middleware (Lua Threads, LoRa WAN, MQTT, ...).
- En la capa media hay un micro-kernel en tiempo real, alimentado por FreeRTOS. Este es el responsable de que las cosas sucedan en el tiempo esperado.
- En la capa inferior hay una capa de abstracción de hardware, que habla directamente con el hardware de la plataforma.



## Funciones principales

- Lua Threads. Permite ejecutar funciones de Lua en paralelo.
- Shell integrado que permite ejecutar ciertos comandos de una manera similar a linux.
- Editor integrado que permite editar archivos.
- Módulos de acceso a sensores. Dispone de una API unificada de acceso a los sensores más utilizados en la industria.
- Módulos de acceso al hardware: GPIO, ADC, CAN, PWM, SPI, I2C, UART, TIMER.
- Módulos middleware: LoRaWAN, MQTT, HTTP server.
- Sistema de archivos: SPIFFS / FAT.
- Almacenamiento: FLASH interna / SD Card.
- Comunicaciones: Ethernet, Bluetooth, WiFi, LoRaWAN.
- Otros módulos: TFT display, NeoPixel, LCD, Stepper, Servos.

## 5. Conectividad (WiFi / Bluetooth / LoRaWAN)

Uno de los aspectos más importantes del kit de desarrollo IoT se basa en las capacidades de comunicación del entorno y el hardware.

Por defecto la placa ya trae de serie conectividad en WiFi, Bluetooth y LoRaWAN, las 2 primeras dependen de infraestructura estándar.

Los estándares de WiFi / Bluetooth y BLE (Bluetooth de bajo consumo) están soportados por defecto en la placa en la misma CPU ESP32.

El estándar LoRaWAN está soportado mediante el transceiver HopeRF como se comentó en el apartado de hardware de la placa ESP32N1

Ahora bien ya que para el uso de WiFi o Bluetooth son necesarios o bien un Punto de Acceso a la red WiFi o un dispositivo cercano que soporte bluetooth, para el uso del estándar LoRaWAN será preciso el uso de un Gateway LoRaWAN que permite conectar nuestro dispositivo con Internet mediante la red de Radio LoRaWAN.

Como el desarrollo del Kit se realizó en Cornellà del Llobregat , en dicho municipio por defecto contamos con acceso a 3 gateways LoRaWAN con una cobertura muy amplia sobre el municipio y alrededores, como se puede ver en la imagen.



Fig-39. Cobertura y ubicación de los Gateways LoRaWAN en Cornellà del Llobregat.

Como se abordará en el siguiente capítulo no solamente es necesario la presencia de la infraestructura de red suficiente para comunicar vía radio con los dispositivos , sino que también necesitaremos de una cuenta en un proveedor de servicios LoRaWAN en nuestro caso explicaremos como realizar dicho proceso con The Things Network (TTN).

En el apartado de WiFi la conectividad, la búsqueda de redes y el servicio de Access Point AP están implementados en el propio sistema operativo LuaRTOS y soportado por tanto en el entorno de desarrollo mediante sus bloques y ejemplos correspondientes.

En el apartado de Bluetooth y BLE debido a su dificultad de implementación actualmente está soportado en hardware pero no está implementado a nivel de sistema operativo ni de bloques.

## 6. LoRaWAN Introducción y Administración en The Things Network (TTN)

Como se ha comentado en el capítulo anterior para poder desarrollar comunicaciones en el ámbito de una red LoRaWAN debemos contar con los siguientes elementos.



Fig-40. Elementos de una red LoRaWAN

Un Nodo, no hay problema porque nuestro kit de desarrollo DEVKIT ESP32N1 es un nodo con soporte para LoRaWAN.

Un Gateway, actualmente en Cornellà del Llobregat hay desplegados 3 gateways pero este número está creciendo en diferentes municipios y la tendencia es que cada vez existirán más Gateways disponibles ya que son los encargados de conectar nuestros nodos con Internet.

Un espacio de monitorización y trabajo, aquí es donde interviene el objeto de este capítulo que tratará de explicar los elementos que precisamos para poder empezar a trabajar con el sistema LoRaWAN

En primer lugar debemos conocer el proyecto The Things Network , dicho proyecto pretende democratizar el uso de los gateways entre los participantes de dicha red de tal modo que yo podré emplear los gateways y la infraestructura de otros y otros podrán utilizar mi gateway o gateways en caso de que los tenga conectados a dicha red, de este modo se pretende dinamizar el crecimiento de una red LoRaWAN pública.

Después de lo expuesto con anterioridad recomendamos buscar información en la web sobre el proyecto ya que es un proyecto dinámico que va cambiando con el paso del tiempo y van surgiendo novedades.

<https://www.thethingsnetwork.org/>

Una vez visto el proyecto de The Things Network, en adelante lo llamaremos TTN para abreviar, procederemos a registrarnos como usuario para poder utilizar su entorno web.



## CREATE AN ACCOUNT

Create an account for The Things Network and start exploring the world of Internet of Things with us.

### USERNAME

This will be your username — pick a good one because you will **not** be able to change it.



ACME

### EMAIL ADDRESS

You will occasionally receive account related emails. This email address is not public.



acme@acme.com

### PASSWORD

Use at least 6 characters.



••••••|

[Create account](#)

By registering an account you agree to our [Terms and Conditions](#) and [Privacy Policy](#).

Already have an account? [Log in](#)

Fig-41. El proceso de registro es muy sencillo solamente necesitamos los datos de la imagen.

Una vez registrados accederemos a la consola, desde la consola se administran los dos elementos principales de una red LoRaWAN que son los Nodos y Los Gateways (en el caso que dispusiéramos de gateways propios), el objetivo de este manual se centra más en el apartado de aplicaciones.

Las aplicaciones son agrupaciones de nodos dentro de un mismo contexto, estas agrupaciones permiten administrar todos los nodos dentro de una aplicación.

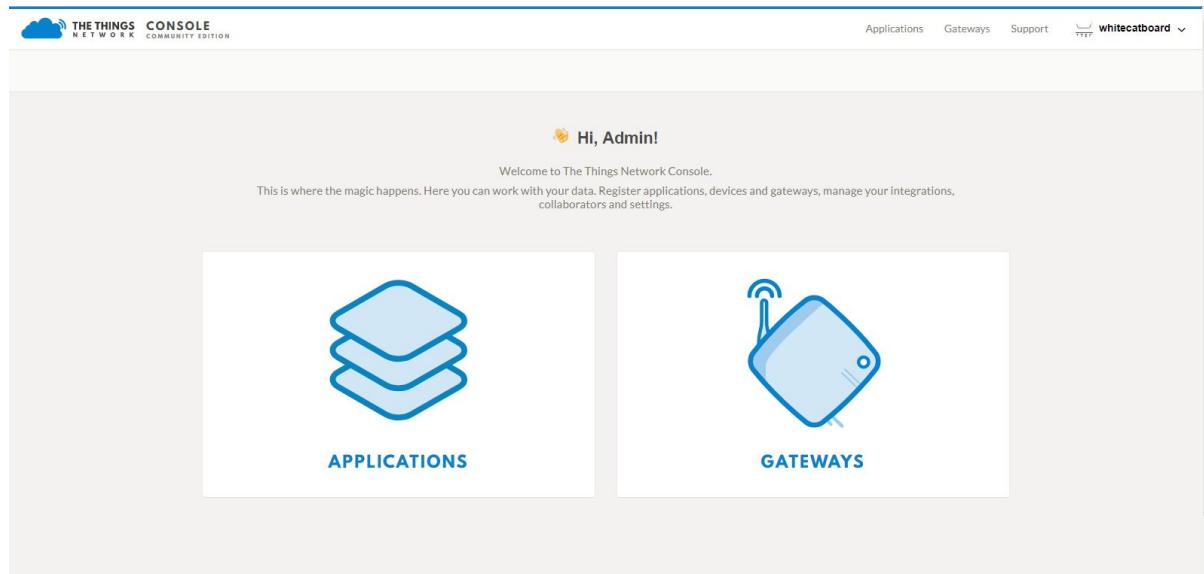


Fig-42. Pantalla principal de la consola de TTN

En la siguiente pantalla una vez seleccionada las aplicaciones podremos seleccionar, crear o eliminar más aplicaciones.

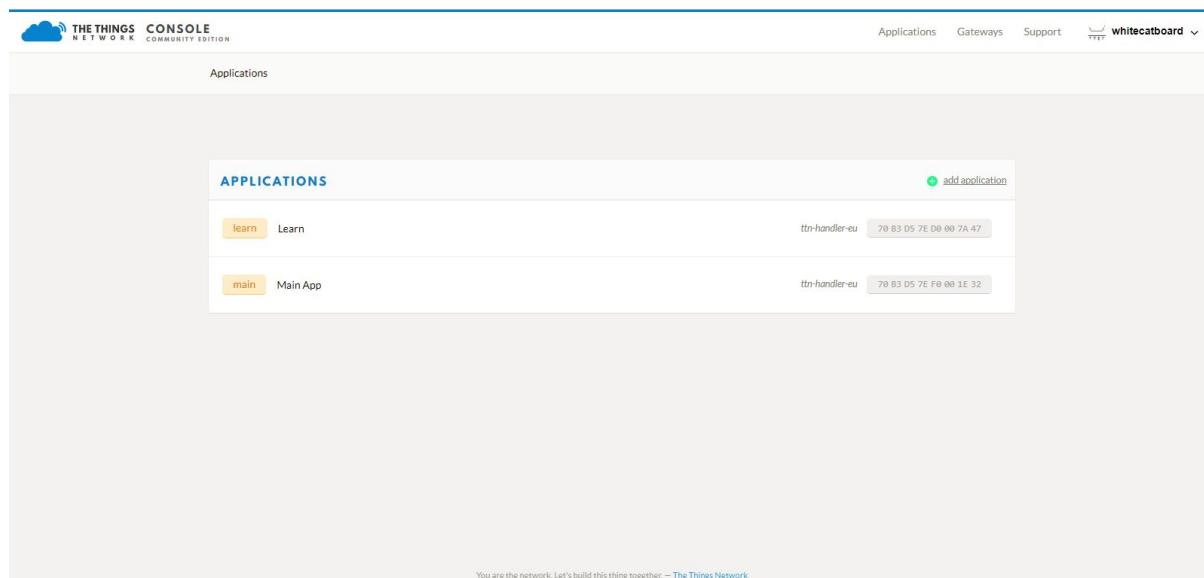
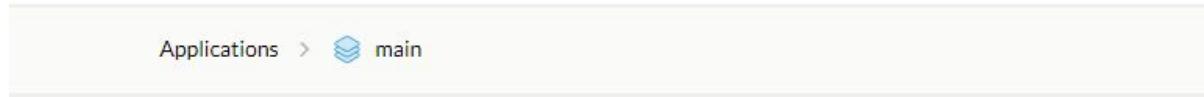
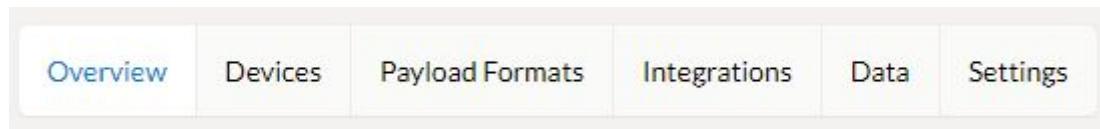


Fig-43. Pantalla de aplicaciones de la consola TTN

Al seleccionar una aplicación vamos a ver a continuación cada uno de los apartados que nos aparecen en la pantalla de la aplicación.



En primer lugar tenemos la barra de navegación que nos indica en qué lugar estamos, en este caso en aplicaciones y dentro de la aplicación main.



Overview, la primera pestaña , es el resumen de los parámetros principales de la aplicación que veremos a continuación.

Devices, la segunda pestaña, es donde realizaremos la gestión de los nodos que enviarán y recibirán información desde la aplicación.

Payload Formats, la tercera pestaña, permite que definamos un código en javascript , para la transformación de la información , recibida / enviada, etc. En nuestro caso como la información se empaqueta y desempaquetan con un proceso especial usaremos esta opción para facilitar el procesamiento de la información.

Integrations, la cuarta pestaña, permite que agreguemos aplicaciones corriendo también en la nube dentro del ecosistema de TTN , como por ejemplo podría ser la aplicación de Data Storage que permite almacenar los datos recibidos y enviados en una base de datos en la nube de TTN durante un periodo de 7 días y puede ser consultada mediante el uso de una API específica.

Data, la quinta pestaña, permite visualizar en tiempo real la información que está siendo recibida desde los nodos y enviada hacia los nodos, si desplegamos nos mostrará además el número de gateways con la información asociada por donde han sido recibidos los paquetes de información

Settings, la sexta pestaña, permite especificar las diferentes opciones de la aplicación.

A continuación en las siguientes páginas veremos más en profundidad cada uno de los apartados.

**APPLICATION OVERVIEW**

Application ID **main**

Description Main App

Created 11 months ago

Handler ttn-handler-eu (*current handler*)

[documentation](#)

Application Overview, Desde este apartado se muestra el application ID que será preciso para conectar más adelante mediante MQTT para la comunicación de los datos.

Description, que es simplemente un texto para identificar la aplicación.

Created, el tiempo que hace que se creó la aplicación.

Handler, el sistema en la nube que procesa y gestiona nuestra aplicación, en este caso lo gestionaremos desde ttn-handler-eu, que hace referencia a los servidores ubicados en Europa, existen también en EEUU, Asia, etc....

En el link de documentation arriba a la derecha , accederemos a la documentación que explica cómo conectar con la API de TTN para poder explotar los datos.

**APPLICATION EUIS**

manage euis

70 B3 D5 7E F0 00 1E 32

Application EUIS, Desde este apartado veremos el application EUI y podremos asignar más de uno EUI es el acrónimo de (Extended Unique Identifier) Identificador Único Extendido, este tipo de identificador está creado para no repetirse e identificar de manera inequívoca las aplicaciones , también se emplea el mismo concepto en los nodos y sumando el Application EUI y el EUI del nodo se asegura que la información solo llegue al lugar que sea preciso.

**DEVICES**

register device manage devices

7 registered devices

Devices, como comentamos al principio las aplicaciones son agrupaciones de dispositivos y en este apartado gestionaremos los dispositivos que queramos conectar a nuestra aplicación.

Antes de poder empezar a enviar y recibir información es importante que dispongamos de los nodos (dispositivos) registrados.

A Continuación se muestra una pantalla con todos los dispositivos de una aplicación de ejemplo.

| DEVICES   |                         | register device |
|-----------|-------------------------|-----------------|
| < >       |                         | 1 – 7 / 7       |
| node_0001 | 4E 4F 44 45 00 00 00 01 | •               |
| node_0002 | 4E 4F 44 45 00 00 00 02 | •               |
| node_0003 | 4E 4F 44 45 00 00 00 03 | •               |
| node_0004 | 4E 4F 44 45 00 00 00 04 | •               |
| node_0005 | 4E 4F 44 45 00 00 00 05 | •               |
| node_0006 | C4 60 C8 20 97 8E 2F 2D | •               |
| nodo_otaa | CB 62 54 18 5B 43 0E 2E | •               |

Cada dispositivo como sucede con las aplicaciones consta de un nombre para facilitar su identificación y un EUI que en este caso se llama Device EUI.

Los nodos de Whitecatboard.org obtienen el Device EUI desde el arranque del LuRTOS ya que la memoria flash empleada en el dispositivo tiene un identificador único, en otros casos se emplea la MAC Address o dirección de red del dispositivo que es un número que se registra en fábrica.

Aquí podemos ver en el arranque de LuRTOS desde consola la indicación de la Flash EUI

```

Lua RTOS beta 0.1. Copyright (C) 2015 - 2017 whitecatboard.org

build 1512648088
commit e5f1fd94d481f8a10f536d9179b5b03a8c946570
Running from factory partition
board type N1ESP32-DEVKIT
cpu ESP32 rev 0 at 240 Mhz
flash EUI d66634415a653030
spiffs0 start address at 0x310000, size 512 Kb, partition spiffs
spiffs0 mounted
sd0 is at spi2, cs=GPIO21
fat0 can't mounted
can't redirect console messages to file system, an SDCARD is needed

Lua RTOS beta 0.1 powered by Lua 5.3.4
Lua RTOS-boot-scripts-aborted-ESP32
/ > █

```

Los dispositivos deben autenticarse y codificar los datos que envía y recibe en base a unas claves públicas y privadas, que son las claves de aplicación y de sesión más los EUI's conforman el espacio de direcciones y las variables suficientes para poder ofrecer garantías de seguridad a la comunicación en red.

Existen dos tipos de dispositivos en función de cómo gestionan esas claves público/privadas que comentamos para codificar/decodificar la información, los tipos son;

OTA, (On the Air) se refiere a dispositivos que se registran solamente con una AppKey y un Device EUI mediante esos dos valores cuando el dispositivo se conecta , negocia con el servidor la obtención de una clave de sesión para cifrar las comunicaciones en los 2 puntos.

Es un sistema más rápido a la hora de desplegar nodos ya que solamente es preciso emplear el AppKey, en cada sesión puede cambiar la clave.

Por otro lado a la hora de realizar comunicaciones es más lento ya que se establece un protocolo de comunicación envío / recepción entre la red y el nodo para negociar precisamente las claves.

ABP, (Address By Personalization) En este caso las claves para codificar la información se programan desde fábrica y no se cambian durante toda la vida del dispositivo, es más rápido en términos de comunicación ya que de antemano TTN y el nodo conocen las claves de codificación y por lo tanto no es preciso que se obtengan en ninguna comunicación.

A continuación vemos los parámetros de un dispositivo.

The screenshot shows a 'DEVICE OVERVIEW' page with the following details:

- Application ID:** main
- Device ID:** node\_0001
- Activation Method:** ABP
- Device EUI:** 4E 4F 44 45 00 00 00 01
- Application EUI:** 70 B3 D5 7E F0 00 1E 32
- Device Address:** 26 01 13 12
- Network Session Key:** (redacted)
- App Session Key:** (redacted)
- Status:** 13 days ago (green dot)
- Frames up:** 0 ([reset frame counters](#))
- Frames down:** 0

Application ID, el identificador de la aplicación.

Device ID, el identificador del dispositivo

Activation Method, ABP / OTA el método de activación del dispositivo como hemos visto anteriormente.

Device EUI, el identificador único de dispositivo.

Application EUI, el identificador único de aplicación.

Device Address, La dirección única del dispositivo.

Network Session Key, La clave de sesión en el caso de ABP se genera en la creación del dispositivo, en el caso de OTA es dinámica.

App Session Key, del mismo modo que el Network Session Key en ABP se genera a la creación del dispositivo y en OTA es dinámica.

Status , color más tiempo que indican si el dispositivo está activo o no (verde y rojo) y la última vez que envió o recibió un mensaje desde la aplicación en minutos, días, etc.

Frames Up, El número de paquetes de información que ha enviado el dispositivo hacia el servidor, es posible resetear el contador, hay que tener en cuenta que por motivos de seguridad se pretende que el contador sea único para que no se hagan reenvío de paquetes de un modo malintencionado

Frames Down, El número de paquetes que el servidor a enviado al Nodo.

**DOWLINK**

Scheduling

FPort

Confirmed

Payload

bytes fields 0 bytes

Es posible desde la consola del dispositivo enviar un downlink, que es un paquete de información desde el servidor al dispositivo, es importante como se remarca en el siguiente párrafo saber que actualmente, en una red LoRaWAN siempre es el nodo el que inicia la comunicación, por ese motivo los Downlink caducan por tiempo.



**IMPORTANTE, EN UNA RED LoRaWAN SIEMPRE ES EL NODO EL QUE INICIA LA COMUNICACIÓN.**

**SIMULATE UP LINK**

FPort

Payload

0 bytes

También es posible desde la consola del dispositivo simular un uplink, información como si llegara del dispositivo al servidor para probar por ejemplo los scripts en el apartado de Payload Formats.

**COLLABORATORS**

manage collaborators

whitecatboard

collaborators delete devices settings

En TTN además mediante la opción de Collaborators, es posible colaborar con otros usuarios en la gestión de la red de este modo podemos asignar permisos para que otros usuarios accedan a la información por ejemplo de diferentes nodos, o puedan gestionar gateways, etc....

Por defecto solo el usuario registrado que ha creado la aplicación tendrá permisos para gestionarla.

The screenshot shows the 'ACCESS KEYS' section of a web interface. It displays a single 'default key' entry. Below the key name are three buttons: 'devices', 'messages', and a third one which is partially visible. To the right of the key name is a text input field containing 'base64' and a small icon. At the top right of the section is a link labeled 'manage keys'.

Por último referente a la aplicación tenemos el Access Keys, con estas claves es posible como veremos más adelante, por ejemplo acceder a la cola MQTT de la aplicación para obtener los datos de los dispositivos.

The screenshot shows the 'PAYLOAD FORMATS' section. Under the 'Payload Format' heading, it says 'The payload format sent by your devices' and 'Custom'. Below this is a code editor window containing a Javascript script for a 'decoder'. The script is titled 'Bytes2Float32' and handles the conversion of bytes to floating-point numbers. The code editor includes line numbers from 1 to 13. A status message 'decoder has no changes' is shown at the bottom right of the editor area. Navigation tabs for 'decoder', 'converter', 'validator', and 'encoder' are located below the editor. A red link 'remove decoder' is visible on the right side.

```

1 function Bytes2Float32(bytes) {
2     bytes = parseInt(bytes);
3
4     var sign = (bytes & 0x80000000) ? -1 : 1;
5     var exponent = ((bytes >> 23) & 0xFF) - 127;
6     var significand = (bytes & ~(-1 << 23));
7
8     if (exponent == 128) {
9         return sign * ((significand) ? Number.NaN : Number.POSITIVE_INFINITY);
10    }
11    if (exponent == -127) {
12        if (significand == 0) return sign * 0.0;
13        exponent = -126;
}

```

Payload Formats, como se comentó con anterioridad es posible definir una serie de scripts en Javascript que permiten realizar tareas con la información “payload” que se envía y se recibe en la red, en esta sección es donde se definen y prueban dichos scripts.

Podemos definir un script por cada tipo de acción , decoder (para la recepción de datos), converter (para convertir de un formato a otro), validator (para validar que la información es correcta), encoder (para el envío de datos).

The screenshot shows the 'INTEGRATION OVERVIEW' section. It displays the following information:

- Status:** Running (indicated by a green dot)
- Integration info:** [go to platform](#)
- Platform:** Data Storage (v2.0.1) (represented by a cloud icon)
- Author:** The Things Industries B.V.
- Description:** Stores data and makes it available through an API. Your data is stored for seven days.

Integration Overview, como se comentó con anterioridad también es posible integrar aplicaciones de terceros dentro de la aplicación de TTN, de este modo es posible realizar otro tipo de tareas con la información que recibe o envía nuestra aplicación, en este caso como se puede observar la integración que tenemos es una integración de almacenamiento en una base de datos la url que nos devuelve es externa y con la información de la API es posible que accedemos a los datos.

En este caso en concreto de storage los datos se conservarán durante 7 días una vez pasado ese tiempo serán eliminados.

The screenshot shows a web-based application interface titled "APPLICATION DATA". At the top, there are filter buttons for "uplink", "downlink", "activation", "ack", and "error". Below these filters, there are three columns labeled "time", "counter", and "port". A large, empty rectangular area below these columns is likely where the data list would appear.

Application Data, desde esta pantalla podremos ver la información que se envía y se recibe a través de la red.

Los paquetes se muestran con la información del payload procesado mediante el payload format y además incluyen información del nodo que lo envía, el gateway por donde llega, tiempo, etc.

Los tipos de paquetes que podemos monitorizar son;

recibidos (uplink) , enviados (downlink), activación (para dispositivos OTA), ack (paquetes con confirmación), error (frames con error)

The screenshot displays two configuration pages side-by-side. On the left is the "APP SETTINGS" page with sections for "General", "EUIs", "Collaborators", and "Access Keys". On the right is the "GENERAL" page, which contains fields for "Description" (set to "Main App") and "Handler" (set to "ttn-handler-eu"). Both pages feature a "Save" button with a green checkmark icon.

Por último la pestaña de app settings nos permite definir las opciones de la aplicación que vimos con anterioridad.

Por último y con el objetivo de poder realizar los casos de uso del capítulo 8, vamos a ver como crear un nodo ABP y un nodo OTA.

#### - Creación Nodo OTA

Para la creación de un nodo OTA es muy sencillo, solamente necesitaremos especificar los siguientes parámetros, como muestra la imagen.

**REGISTER DEVICE**

**Device ID**  
This is the unique identifier for the device in this app. The device ID will be immutable.  
nodo\_ota

**Device EUI**  
The device EUI is the unique identifier for this device on the network. You can change the EUI later.  
01 02 03 04 05 06 07 08 8 bytes

**App Key**  
The App Key will be used to secure the communication between your device and the network.  
this field will be generated

**App EUI**  
70 B3 D5 7E F0 00 1E 32

Cancel Register

Device ID (nombre que queremos darle), DeviceEUI (devuelto por LuaRTOS al arranque) , AppKey será generado y AppEUI es el de la Aplicación con esos parámetros ya podemos generar el dispositivo.

Para generar un nodo ABP el proceso es un poco más complicado ya que necesitaremos la herramienta ttncctl, que es una herramienta de consola para la gestión de los parámetros de TTN como hemos visto en formato WEB, pero permite realizar ciertas tareas como en este caso que no están disponibles en la web como es la creación de un nodo ABP.

En primer lugar descargamos la herramienta ttncctl desde la web de TTN, en;

<https://www.thethingsnetwork.org/docs/network/cli/quick-start.html>

Lo descomprimimos y lo copiamos en alguna ruta dentro del path y lo renombramos a ttnctl, por ejemplo en Windows lo ideal es copiarlo a la carpeta Windows con el nombre ttnctl.exe.

Una vez tengamos acceso lo ejecutamos y vemos una salida como esta

```

Símbolo del sistema
Available Commands:
  applications      Manage applications
  config            Print the used configuration
  devices           Manage devices
  downlink          Send a downlink message to a device
  gateways          Manage gateways
  help              Help about any command
  selfupdate        Update ttnctl to the latest version
  subscribe         Subscribe to events for this application
  user              Show the current user
  version           Get build and version information

Flags:
  --allow-insecure      Allow insecure fallback if TLS unavailable
  --auth-server string   The address of the OAuth 2.0 server (default "https://account.thethingsnetwork.org")
  --config string        config file (default is ${HOME}/.ttnctl.yml)
  --data string          directory where ttnctl stores data (default is ${HOME}/.ttnctl)
  --discovery-address string The address of the Discovery server (default "discover.thethingsnetwork.org:1900")
  --handler-id string    The ID of the TTN Handler as announced in the Discovery server (default "ttn-handler-eu")
  -h, --help             help for ttnctl
  --mqtt-address string  The address of the MQTT broker (default "eu.thethings.network:1883")
  --mqtt-password string The password for the MQTT broker
  --mqtt-username string The username for the MQTT broker
  --router-id string     The ID of the TTN Router as announced in the Discovery server (default "ttn-router-eu")

)
Use "ttnctl [command] --help" for more information about a command.

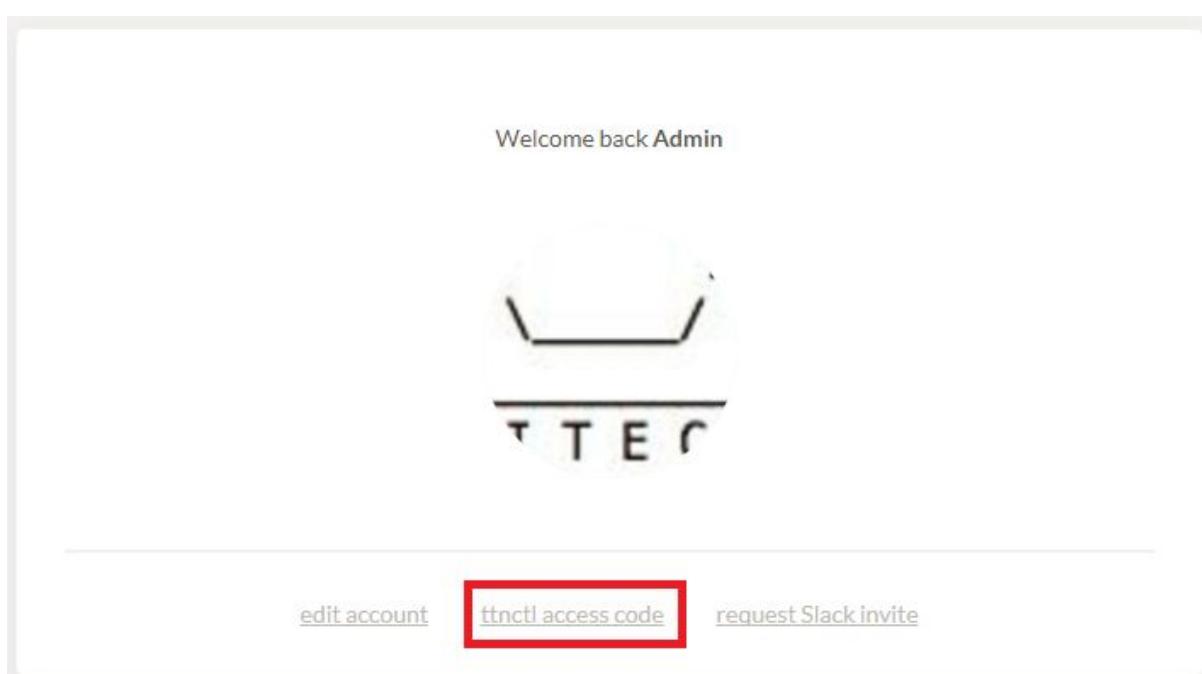
C:\Users\mferrera>

```

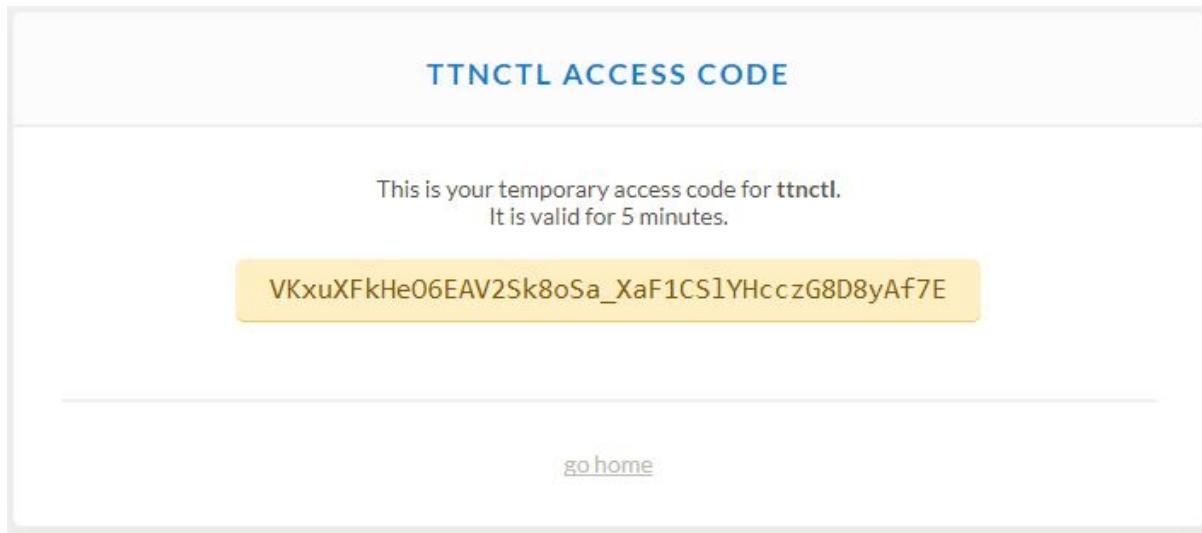
En primer lugar necesitaremos acceder a la cuenta del mismo modo que hacíamos en formato web para hacer esto iremos a la siguiente dirección

<https://account.thethingsnetwork.org/>

Desde allí copiaremos el código de acceso que nos facilitaran en la web.



Una vez pinchamos en el enlace de ttectl access code nos facilitarán un código válido para acceder mediante la herramienta ttectl durante 5 minutos.



Una vez tenemos el código accedemos mediante, el siguiente comando.

***ttectl user login VKxuXFkHe06EAV2Sk8oSa\_XaF1CS1YHcczG8D8yAf7E***

```
Simbolo del sistema
help      Help about any command
selfupdate Update ttectl to the latest version
subscribe Subscribe to events for this application
user      Show the current user
version   Get build and version information

Flags:
--allow-insecure          Allow insecure fallback if TLS unavailable
--auth-server string       The address of the OAuth 2.0 server (default "https://account.thethingsnetwork.org")
--config string            config file (default is $HOME/.ttectl.yml)
--data string              directory where ttectl stores data (default is $HOME/.ttectl)
--discovery-address string The address of the Discovery server (default "discover.thethingsnetwork.org:1900")
--handler-id string        The ID of the TTN Handler as announced in the Discovery server (default "ttn-handler-eu")
-h, --help                 help for ttectl
--mqtt-address string     The address of the MQTT broker (default "eu.thethings.network:1883")
--mqtt-password string    The password for the MQTT broker
--mqtt-username string    The username for the MQTT broker
--router-id string         The ID of the TTN Router as announced in the Discovery server (default "ttn-router-eu")

Use "ttectl [command] --help" for more information about a command.

C:\Users\mferrera>ttectl user login VKxuXFkHe06EAV2Sk8oSa_XaF1CS1YHcczG8D8yAf7E
INFO Successfully logged in as whitecatboard (admin@whitecatboard.org)

C:\Users\mferrera>
```

Y obtendremos la siguiente pantalla que nos indica que ya hemos accedido a nuestra cuenta de usuario

El siguiente paso es seleccionar la aplicación donde queremos crear el nodo, para ello listamos las aplicaciones para seleccionar una de ellas, mediante el comando;

***ttnctl applications list***

```

Símbolo del sistema
--auth-server string      The address of the OAuth 2.0 server (default "https://account.thethingsnetwork.org")
--config string            config file (default is $HOME/.ttnctl.yml)
--data string              directory where ttnctl stores data (default is $HOME/.ttnctl)
--discovery-address string The address of the Discovery server (default "discover.thethingsnetwork.org:1900")
--handler-id string        The ID of the TTN Handler as announced in the Discovery server (default "ttn-handler-eu")
--help
--mqtt-address string     The address of the MQTT broker (default "eu.thethings.network:1883")
--mqtt-password string    The password for the MQTT broker
--mqtt-username string   The username for the MQTT broker
--router-id string        The ID of the TTN Router as announced in the Discovery server (default "ttn-router-eu")

Use "ttnctl [command] --help" for more information about a command.

C:\Users\mferrera>ttnctl user login VKxuXFkHe06EAV2Sk8oSa_XaF1CSlYHcczG8D8yAf7E
INFO Successfully logged in as whitecatboard (admin@whitecatboard.org)

C:\Users\mferrera>ttnctl applications list
INFO Found 2 applications:

ID      Description      EUIs      Access Keys      Collaborators
1      learn      Learn      1          1
2      main      Main App      1          1

C:\Users\mferrera>

```

Una vez que tengamos la lista de aplicaciones seleccionaremos la que nos interesa con el comando;

***ttnctl applications select 2***

```

Símbolo del sistema
C:\Users\mferrera>ttnctl applications select
INFO Found 2 applications:

ID      Description
1      learn      Learn
2      main      Main App

Which one do you want to use?
Enter the number (1 - 2) > 2
INFO Found one EUI "70B3D57EF0001E32", selecting that one.
INFO Updated configuration           AppEUI=70B3D57EF0001E32 AppID=main

C:\Users\mferrera>

```

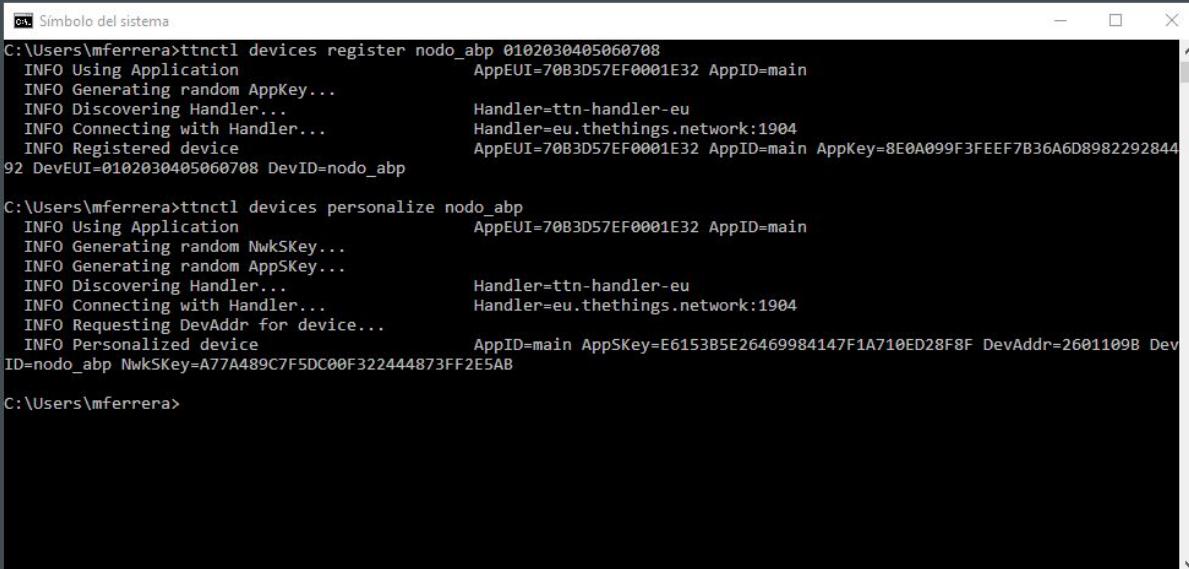
Nos aparecerá la siguiente pantalla confirmando la selección de aplicación.

El siguiente paso será crear el nodo mediante el siguiente comando.

***ttnctl devices register nodo\_abp 0102030405060708***

y acto seguido lo personalizamos para indicar a TTN que será un nodo ABP

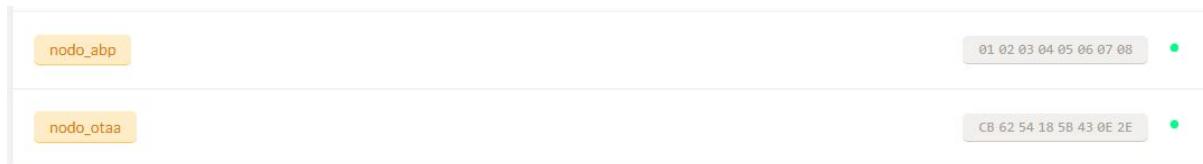
### ***ttnctl devices personalize nodo\_abp***



```
C:\Users\mferrera>ttnctl devices register nodo_abp 0102030405060708
INFO Using Application AppEUI=70B3D57EF0001E32 AppID=main
INFO Generating random AppKey...
INFO Discovering Handler... Handler=ttn-handler-eu
INFO Connecting with Handler... Handler=eu.thethings.network:1904
INFO Registered device 92 DevEUI=0102030405060708 DevID=nodo_abp
AppEUI=70B3D57EF0001E32 AppID=main AppKey=8E0A099F3FEEF7B36A6D8982292844

C:\Users\mferrera>ttnctl devices personalize nodo_abp
INFO Using Application AppEUI=70B3D57EF0001E32 AppID=main
INFO Generating random NwkSKey...
INFO Generating random AppSKey...
INFO Discovering Handler... Handler=ttn-handler-eu
INFO Connecting with Handler... Handler=eu.thethings.network:1904
INFO Requesting DevAddr for device...
INFO Personalized device AppID=main AppSKey=E6153B5E26469984147F1A710ED28F8F DevAddr=2601109B DevID=nodo_abp NwkSKey=A77A489C7F5DC00F322444873FF2E5AB
C:\Users\mferrera>
```

De este modo ya tendriamos creado el nodo en formato ABP para poder realizar las pruebas desde los siguientes casos de uso en el capítulo 8, podemos comprobar desde la web que el nodo se ha creado correctamente y el formato en el que se ha creado para confirmar que TTN lo ha creado como ABP.



En la imagen podemos confirmar que TTN ha creado los 2 nodos con las opciones que hemos especificado, uno en método OTA y otro en método ABP.

Ya los tendriamos preparados para las prácticas que desarrollaremos en el capítulo 8.

El último punto es acceder a la información , este punto se abordará en el siguiente Capítulo y se desarrolla cuando se explique el objeto MQTT del editor de Bloques de NodeRED.

Ya que el método preferido para acceder a la información en tiempo real en TTN será a través de su servidor MQTT.

MQTT es el acrónimo de Message Queuing Transport Telemetry, Y es un protocolo desarrollado por los laboratorios I+D de IBM y que tiene una estructura de publicación suscripción que lo hacen muy interesante para el desarrollo de aplicaciones IoT como es el caso que nos ocupa, en el siguiente capítulo abordaremos más en profundidad su uso.

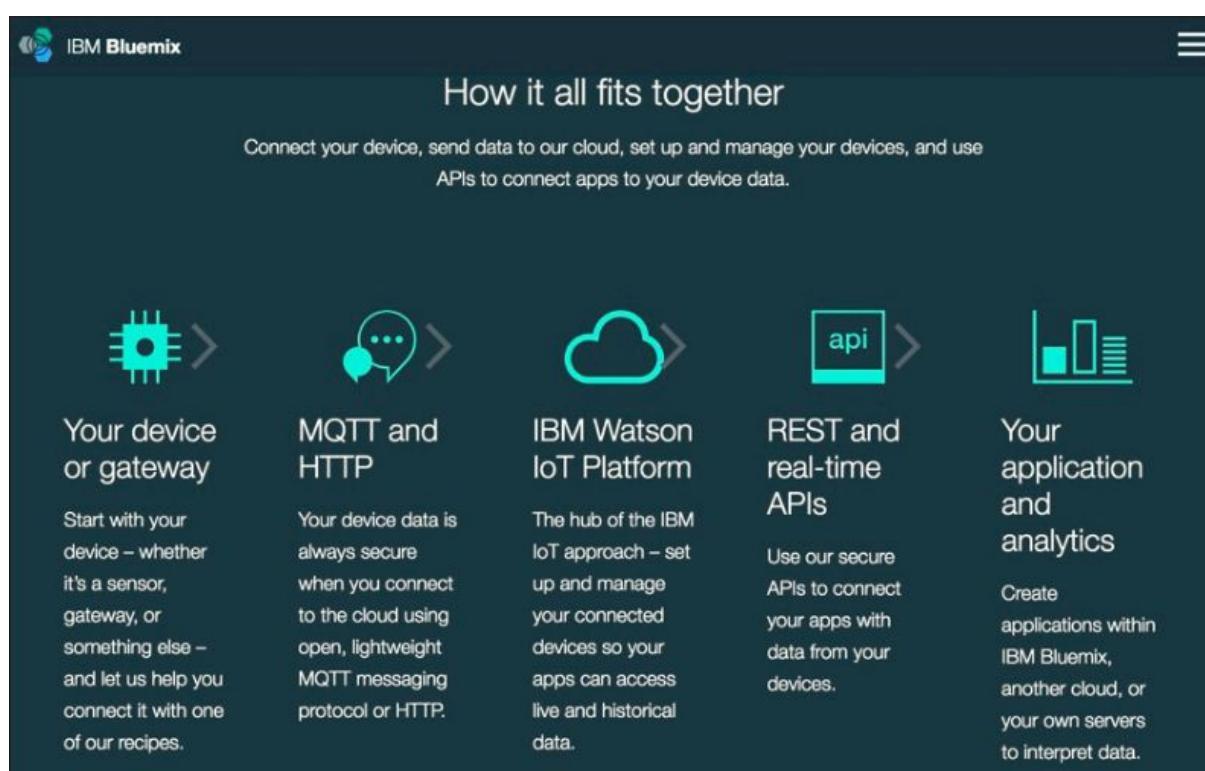
## 7. La “Nube” (IBM Bluemix y NodeRED)

Una vez que hemos visto en el capítulo anterior cómo enviar y recibir información desde un espacio en la nube hasta los nodos y desde estos últimos a la nube , el siguiente paso es que hacemos con dicha información.

Para ello en el entorno whitecatboard.org proponemos el uso de tecnologías de servidor consolidadas que permitan disponer de un sistema en la nube robusto y eficaz para el desarrollo de la parte de servidor, aquella que aporta la visión global de todos los dispositivos que están enviando información y va a permitir desarrollar las soluciones que trabajen con los datos de dichos dispositivos y sensores.

La solución en la nube consta de dos partes una parte de infraestructura es decir allí donde va a residir la lógica y la herramienta con la que vamos a generar dicha logica que llamaremos generica IBM Bluemix y otra parte que es la herramienta con la que desarrollaremos e implementaremos la lógica desde el servidor, que llamaremos Node-RED.

### Qué es IBM Bluemix ?



Como comentamos en el párrafo anterior desde nuestro punto de vista IBM Bluemix es para nosotros un conjunto de servicios que van a permitir que nuestra solución funcione en un entorno robusto que nos permita despreocuparse de la infraestructura.

Eso es a groso modo bluemix para nosotros, un conjunto de productos y servicios en la nube que permiten desplegar soluciones basadas en máquinas virtuales y que corren y solicitan recursos bajo demanda, de tal modo es el equivalente a disponer de un sistema SAS pero en hardware y software para el despliegue de soluciones IoT.

Además toda la tecnología de IA por ejemplo de IBM con Watson puede formar parte de dicha solución , permitiendo una escalabilidad en el futuro muy interesante.

Actualmente nuestro uso de bluemix se centra en el servicio de máquina virtual y sobre ella hacemos correr el sistema de NodeRED.

A continuación mostramos los rasgos más importantes en diferentes pantallas de la consola de administración de Bluemix.

En primer lugar la pantalla donde estan nuestros servicios (servidores) corriendo.

<https://idaas.iam.ibm.com/idaas/mtfim/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:basicldapuser>



Pantalla de login a la URL de IBM Cloud donde accederemos a la gestion de nuestros productos y servicios.

Panel de control

GRUPO DE RECURSOS: Todos los recursos | REGIÓN: Reino Unido | ORGANIZACIÓN DE CLOUD FOUNDRY: WHITECATBOARD | ESPACIO DE CLOUD FOUNDRY: WORKSPACE | Filtrar por nombre de recurso... | Crear recurso

**Apps de Cloud Foundry** 512 MB/512 GB en uso

| Nombre        | Ruta                             | Memoria (MB) | Estado             |
|---------------|----------------------------------|--------------|--------------------|
| whitecatboard | whitecatboard.eu-gb.mybluemix... | 512          | En Ejecución (1/1) |

**Servicios** 1/1500 en uso

| Nombre                        | Oferta de servicios | Plan |
|-------------------------------|---------------------|------|
| whitecatboard-cloudantNoSQLDB | Cloudant NoSQL DB   | Lite |

Catálogo de recursos del IBM Cloud una vez que accedemos, en este caso podemos ver operando una app de cloud foundry que en realidad es un servicio de máquina virtual.

También podemos ver corriendo un servicio que se instala por defecto de una base de datos en el momento que ponemos en marcha el servidor virtual.

Lo importante de IBM Cloud y Bluemix es que el servicio o producto se modifica bajo demanda de tal modo la facturación por el servicio o producto también es variable en función del uso.

Por otra parte uno de los elementos interesantes es que IBM Cloud permite un periodo de evaluación que comprende un mes de uso sin cargo alguno, este tipo de servicio es el que se propone en las prácticas que se verán en el capítulo siguiente.

Iniciación | Visión general | Tiempo de ejecución | Conexiones | Registros | Supervisión | Gestión de API

whitecatboard | En ejecución | Visitar URL de app | Rutas | C | O | :

Org: WHITECATBOARD | Ubicación: United Kingdom | Espacio: WORKSPACE

**Tiempo de ejecución**

- PAQUETE DE COMPILACIÓN: Node-RED Starter
- INSTANCIAS: 1
- MB DE MEMORIA POR INSTANCIA: 512
- ASIGNACIÓN DE MB TOTAL: 512

Todas las instancias se están ejecutando  
El estado de salud es del 100%

**Conexiones (1)**

- whitecatboard-cloudantNoSQLDB

**Coste de tiempo de ejecución**

- 0,00 \$
- Cargos actuales para el periodo de facturación  
(1 de dic. de 2017 - 31 de dic. de 2017)
- 0,00 \$
- Total estimado para el periodo de facturación  
(1 de dic. de 2017 - 31 de dic. de 2017)

Ejemplo del dashboard de IBM Cloud donde vemos el servicio corriendo y el resumen de facturación, etc. Desde esta pantalla se pueden configurar diferentes opciones de nuestro servicio. en nuestro caso lo que nos interesa es la aplicación NodeRED, que veremos a continuación.

## NodeRED

NodeRED es una herramienta diseñada por IBM para la programación visual mediante bloques y en entorno Web de aplicaciones que corren sobre un servidor Node.js.

Node.js es un servidor que ejecuta Javascript en una máquina virtual propia sin necesidad de que los programas escritos en ese lenguaje se ejecuten en la parte cliente normalmente un navegador.

Históricamente Javascript siempre se ha ejecutado dentro de un navegador web por lo tanto siempre se ha ejecutado en la parte cliente, con Node.js se aprovechó el desarrollo de aplicaciones en Javascript para que estas se pudieran ejecutar también por ejemplo en un servidor.

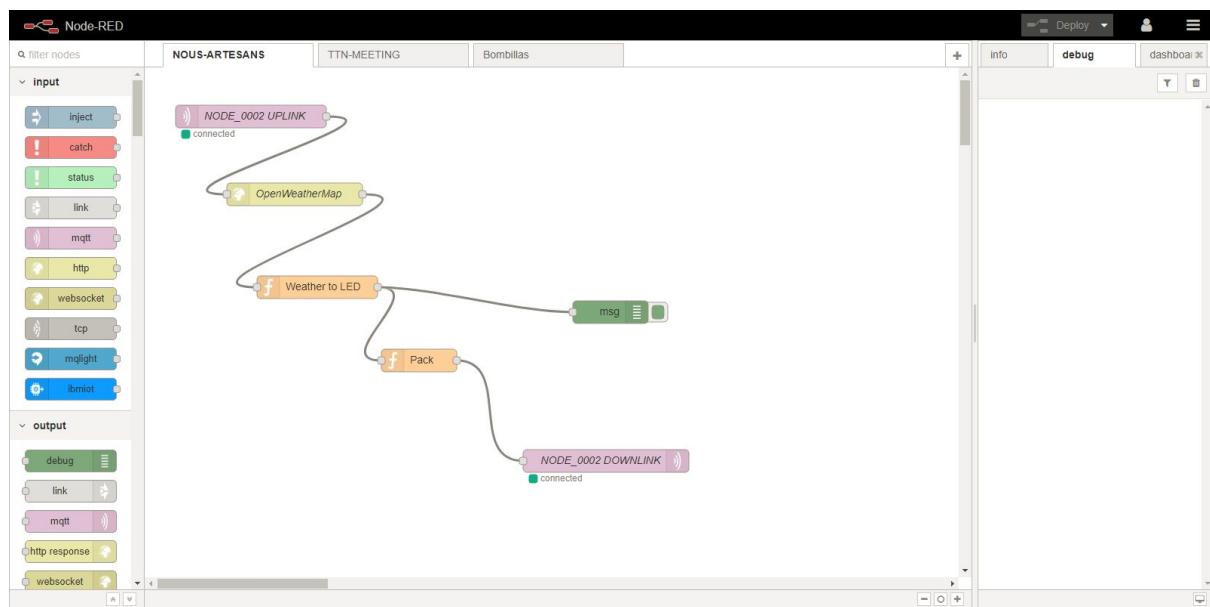
Esto permite aprovechar todo el potencial del lenguaje de programación Javascript en entornos de servidor para el despliegue de aplicaciones como en nuestro caso , orientadas al manejo de la información en el servidor de datos y procesos en la nube.

NodeRed por tanto como se comentaba al inicio es un conjunto de módulos desarrollados en Javascript que permiten de un modo sencillo e intuitivo la programación de eventos mas complejos en un ámbito de red distribuido.

Aquí podemos ver la pantalla de inicio de sesión para poder acceder al workspace o entorno de programación de NodeRED.



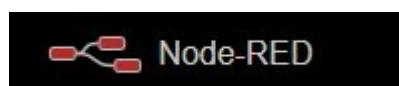
Mediante esta pantalla accederemos al área de trabajo de NodeRed que pasamos a describir a continuacion.



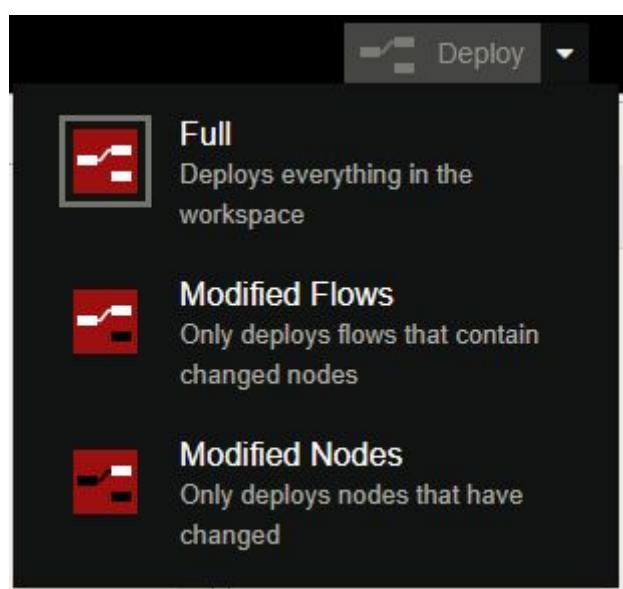
Pantalla principal , área de trabajo de NodeRED.

A continuacion veremos cada uno de los apartados del area de trabajo.

Barra superior

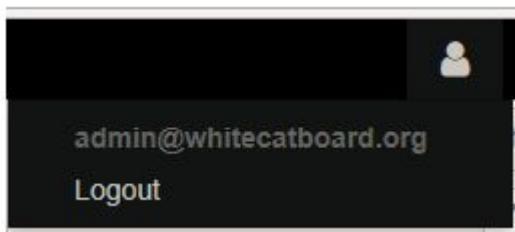


Logotipo de Node-RED , en según qué instalaciones este logotipo se puede personalizar para adecuarlo a las necesidades estéticas del proyecto / cliente.

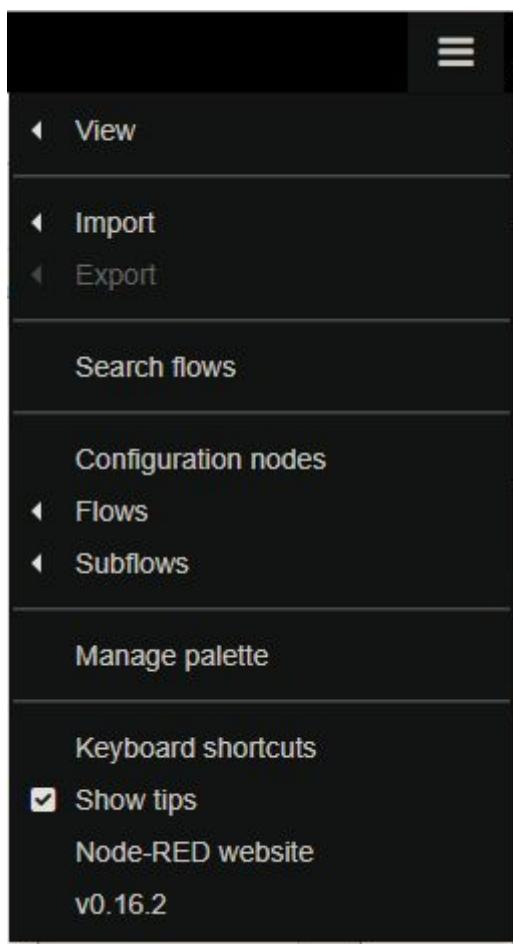


Cada vez que se modifique algo en las áreas de trabajo se puede desplegar en el servidor para que empiecen a funcionar los nuevos bloques y su código, en este momento si hubiera

cualquier error en nuestros bloques o código NodeRED nos lo notifica antes de proseguir con el despliegue.



Icono de usuario , identifica la cuenta con la que hemos accedido y permite que hagamos un logout para cerrar la sesión.



Menú de opciones de NodeRED, desde este menú se permite cambiar aspectos de la configuración a nivel de las vistas, importar / exportar o eliminar “flows” que es como se llaman los programas en NodeRED.

Se pueden crear subflows que son pequeñas partes de bloques / código dentro de partes mayores.

Se pueden modificar los parámetros de los nodos (bloques) de configuración para aquellos nodos que requieran dichos parámetros como por ejemplo la conexión a servidores, etc.

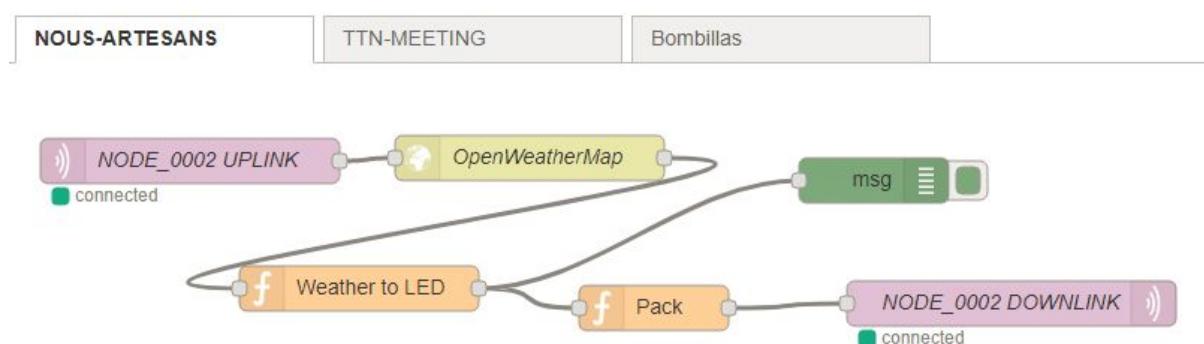


En la parte izquierda tenemos los nodos que son los bloques que podemos arrastrar al área de trabajo están divididos como podemos apreciar en la imagen en diferentes secciones , cada una de ellas tiene diferentes nodos relacionados para hacer tareas específicas dentro de cada sección a continuación vemos unos cuantos nodos de ejemplo dentro de la categoría input.



Estos son los nodos que aparecen en la categoría de entradas, todos ellos hacen referencia a maneras en las que podemos recibir información desde una petición web desde mqtt, desde la plataforma IBM IoT desde un socket TCP, etc.

En nuestro caso cuando trabajemos con LoRa desde TTN en los casos prácticos emplearemos principalmente los nodos de MQTT, también los emplearemos en los ejemplos que se realicen empleando la conectividad WiFi.

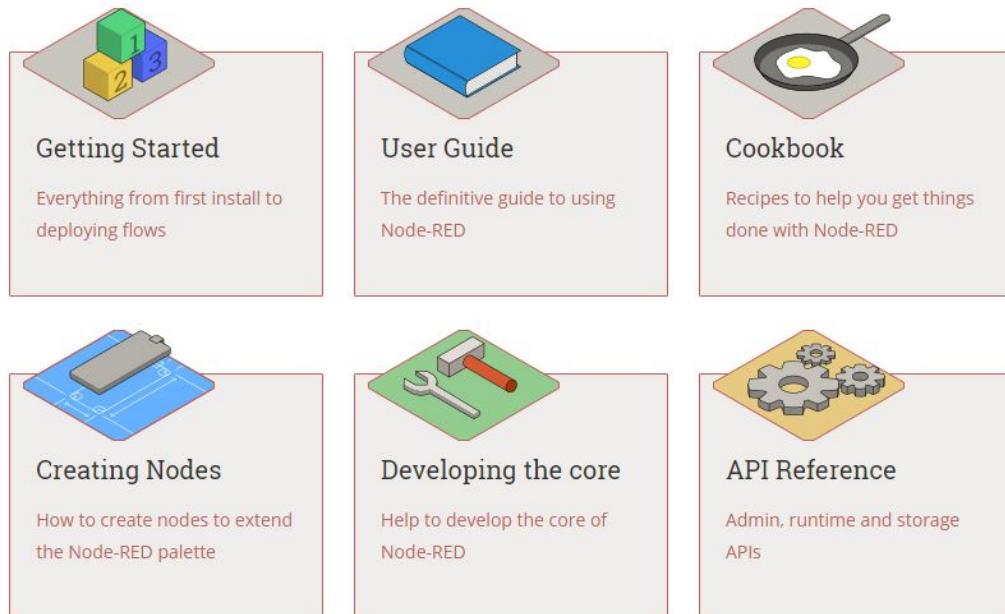


En la imagen podemos ver los diferentes flows en las diferentes pestañas y los nodos que componen uno de los flows, como se puede observar el modelo de programación en bloques se basa en entradas y salidas en cada bloque.

Para profundizar en el uso de la herramienta de NodeRED se puede acceder al sitio de desarrollo y documentación en la siguiente URL;

<https://nodered.org/docs/>

## Documentation



Poseen una documentación muy bien elaborada y muy extensa sobre cada una de las funcionalidades de NodeRED.

A parte de la gran potencia que posee NodeRED para el desarrollo de aplicaciones de manera visual , uno de sus valores clave es que es Software Libre y su desarrollador principal es IBM.

Es importante tener conocimientos de Javascript para poder aprovechar al máximo las posibilidades de NodeRED y como veíamos con anterioridad lo mismo sucede con los scripts que podemos configurar en los payload functions de TTN.

## 8. Casos Prácticos

A continuación presentamos las “fichas” de varios casos prácticos que se han diseñado con el objetivo de empoderar al usuario desde un ejemplo muy simple , hasta comprender los diferentes tipos de elementos que pueden conformar un dispositivo considerado IoT hasta el ultimo caso práctico donde se pondrá en valor todo lo aprendido en los primeros casos.

Cada caso práctico esta explicado de una forma simple y sencilla para facilitar su comprensión.

En cada ficha se explicará el objetivo del caso, los elementos necesarios así como los el esquema de conexión mediante un diagrama, el código en bloques y el código en Lua.

También se explicará la parte local y la parte que se ejecuta en el servidor, en el caso práctico que lo precise.

## 8.1. Caso 1

### - Descripción

En este primer caso vamos a conectar un sensor BME280 de presión, temperatura y humedad para mostrar los datos recogidos por el sensor en la consola.

### - Objetivos

El objetivo de este primer caso práctico básico, es familiarizarnos con los diferentes elementos del kit así como con el entorno de desarrollo y empezar con los primeros bloques.

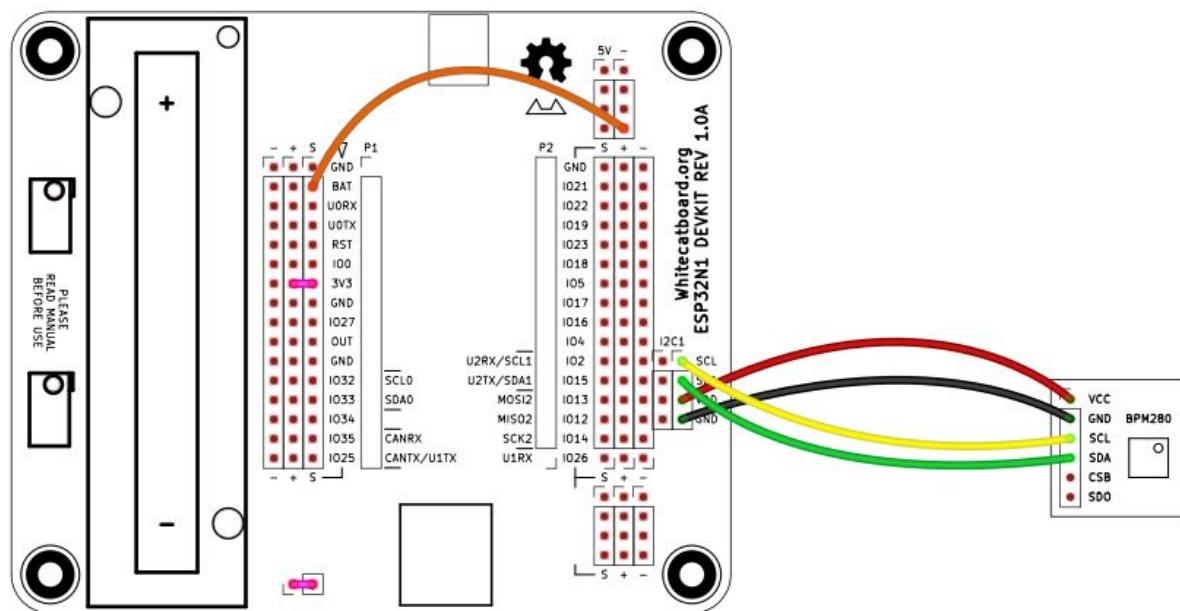
### - Materiales

Para la realización de este caso práctico necesitaremos;

- 1x ESP32N1
- 1x ESP32N1 DEVKIT
- 1x SENSOR BOSCH BME280
- 1x SET DE CABLES

### - Diagrama de Conexión

Este es el diagrama de conexión (recordar si se emplea batería usar el cable naranja desde el pin marcado como bat a un pin +5V)



fritzing



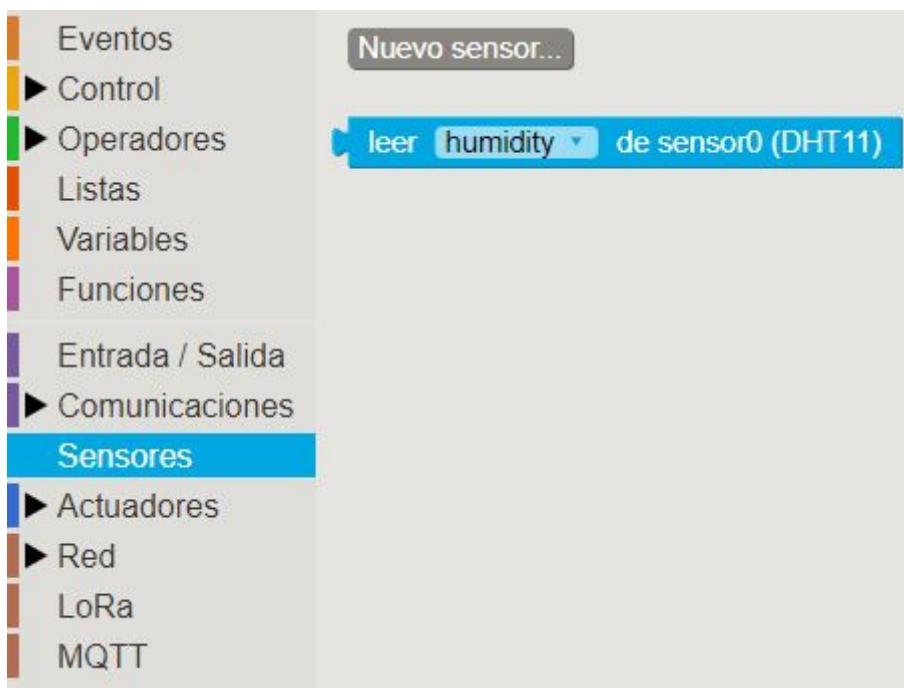
**SE RECOMIENDA DESCONECTAR LA PLACA MIENTRAS SE REALIZA EL CONEXIONADO DE LOS ELEMENTOS DE LA PRÁCTICA.**

Este sensor opera por el puerto I2C en nuestro caso como muestra el diagrama lo conectaremos al puerto I2C número 1, para ello será preciso conectar VCC, GND, SCL y SDA.

Para estandarizar las prácticas y las líneas de datos en los cables, etc, se recomienda seguir también el patrón de colores.

Una vez conectado todo como se indica en el diagrama procederemos a usar los bloques necesarios para la lectura de los valores que aporta el sensor y su impresión por consola.

#### - Programa en bloques



Para poder emplear el sensro BME280 necesitaremos configurarlo en el entorno para ello, seleccionamos la opción sensores y ahí dentro nuevo sensor.

Acto seguido seleccionaremos por ejemplo la temperatura como magnitud a medir ya que es una de las que dispone el sensor y para finalizar seleccionaremos el sensor llamado BME280 de la lista y el puerto I2C el número 1, y le asignaremos un nombre al sensor en este caso el que aparece por defecto como sensor1.



Una vez creado el sensor procederemos a añadir el resto de los bloques para la práctica , para que queden del siguiente modo.



Con estos bloques se realizará un bucle infinito en la placa en el cual se imprimirá por la consola los 3 valores obtenidos del mismo sensor , se hará una pausa de 5 segundos y se repetirá la operación, en la consola podemos ver el valor de los bloques para imprimir.

Whitecat N1 ESP32 DEVKIT |  caso1.xml | Español | 

```
21.63152
***** FIN LECTURA VALORES SENSOR *****
***** INICIO LECTURA VALORES SENSOR *****
48.37318
993.1817
21.63776
***** FIN LECTURA VALORES SENSOR *****
***** INICIO LECTURA VALORES SENSOR *****
48.34554
993.1424
21.62653
***** FIN LECTURA VALORES SENSOR *****
***** INICIO LECTURA VALORES SENSOR *****
48.27376
993.1638
21.63402
***** FIN LECTURA VALORES SENSOR *****
```

## 8.2. Caso 2

### - Descripción

En este segundo caso vamos a proseguir tal y como teníamos en el caso primero conectado el sensor BME280 de presión, temperatura y humedad pero en esta ocasión para mostrar los datos recogidos por el sensor en la consola de administración de TTN.

### - Objetivos

El objetivo de este segundo caso práctico es una vez asimilados los conocimientos del primer caso práctico, realizar el envío de los datos recogidos por el sensor a la nube mediante el uso de la red LoRa, al final de la práctica el usuario habrá aprendido cómo enviar los datos a la nube para su procesamiento mediante TTN.

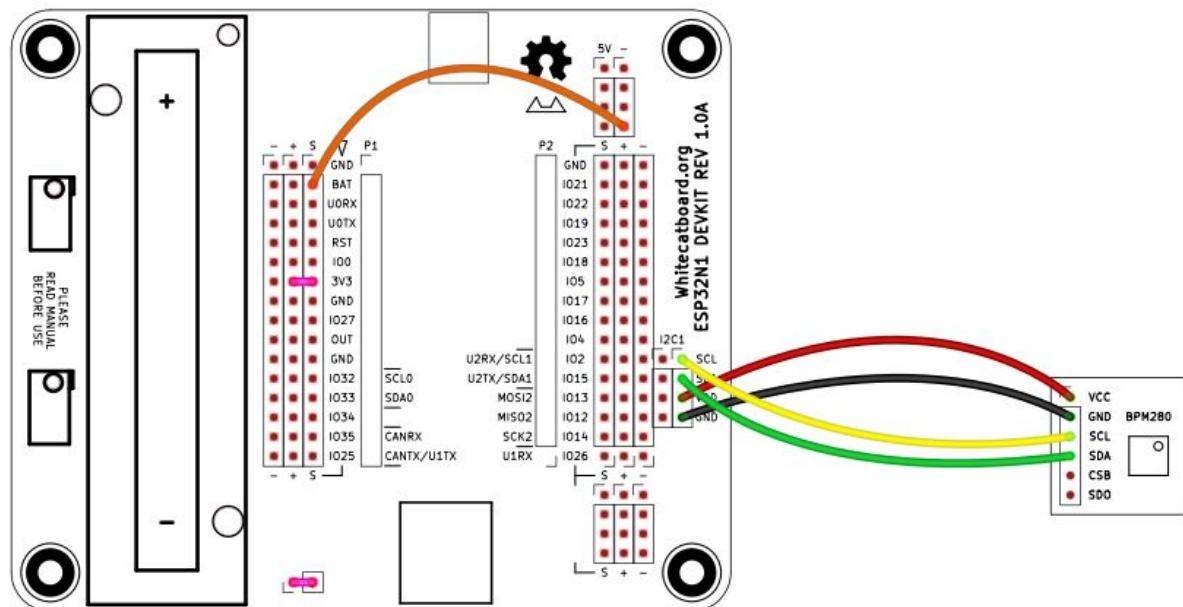
### - Materiales

Para la realización de este caso práctico necesitaremos;

- 1x ESP32N1
- 1x ESP32N1 DEVKIT
- 1x SENSOR BOSCH BME280
- 1x SET DE CABLES
- 1x CONECTIVIDAD LoRaWAN (GATEWAY O ÁREA CON COBERTURA LoRaWAN)

### - Diagrama de Conexión

Este es el diagrama de conexión (recordar si se emplea batería usar el cable naranja desde el pin marcado como bat a un pin +5V)



fritzing

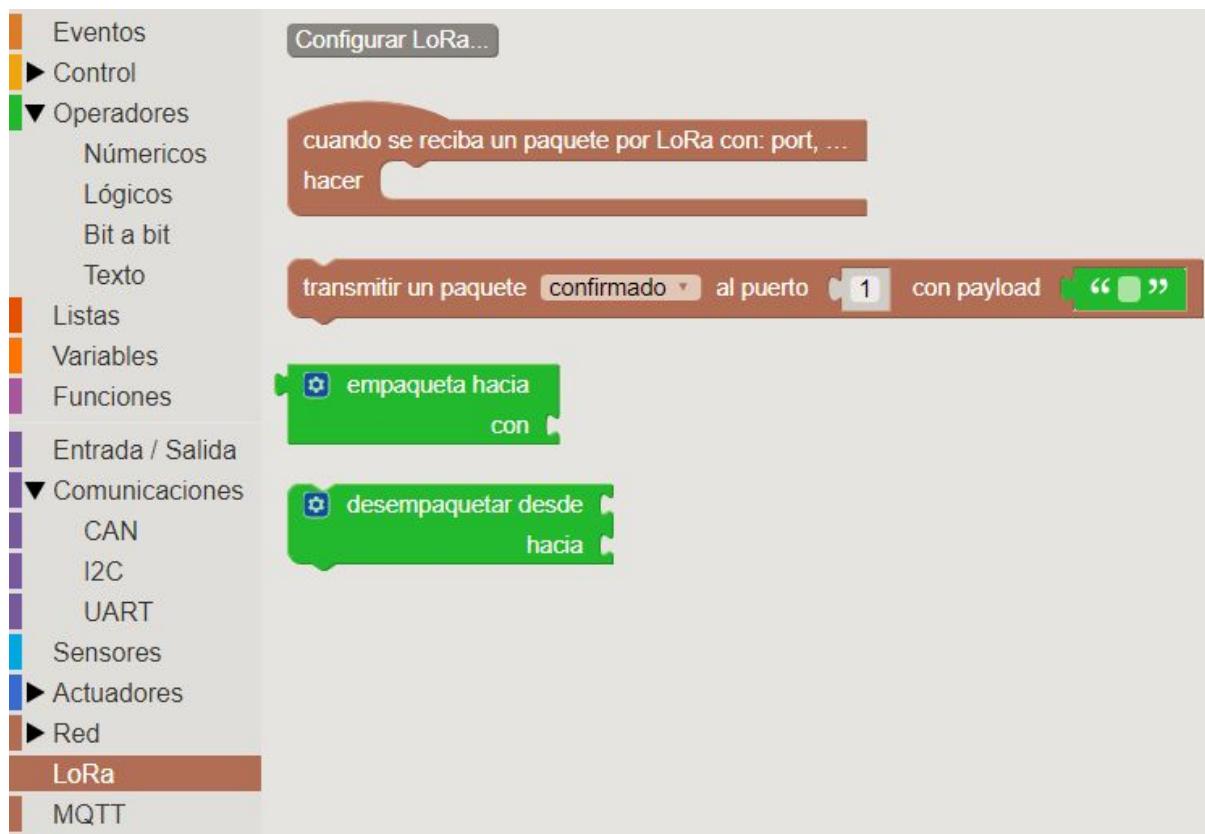


**SE RECOMIENDA DESCONECTAR LA PLACA MIENTRAS SE REALIZA EL CONEXIONADO DE LOS ELEMENTOS DE LA PRÁCTICA.**

Las conexiones la mantendremos ya que son iguales a las empleadas en el primer caso práctico.

## - Programa en bloques

En este segundo caso ya dispondremos del sensor como se creo en el primer caso , pero deberemos configurar la red LoRaWAN en el dispositivo , para realizar esta tarea procederemos de un modo similar a como hicimos con el sensor, en este caso desde el apartado de LoRa.



Seleccionar el apartado de LoRa y seleccionar la opción Configurar LoRa....

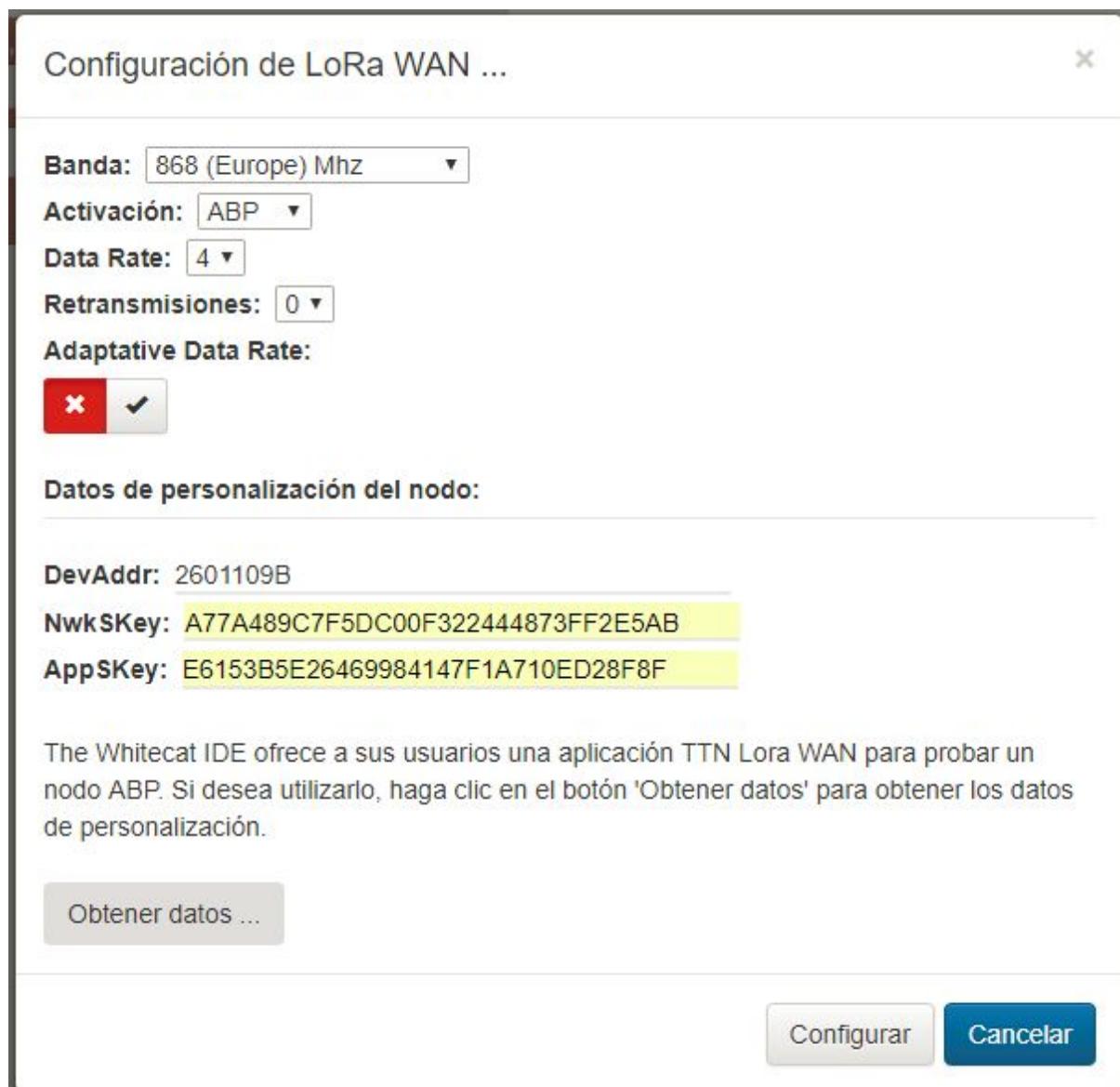
Antes de configurar LoRa será preciso que tengamos los datos desde la consola TTN como vimos en el ejemplo del capítulo 6 de TTN, usaremos el nodo de ejemplo de ABP que creamos que se llamaba nodo\_abp, vamos a consultar sus datos....

The screenshot shows the TTN Device Overview page for a device named 'nodo\_abp'. The top navigation bar includes 'Applications > main > Devices > nodo\_abp'. Below the navigation is a header with 'Overview' (selected), 'Data', and 'Settings' tabs. The main section is titled 'DEVICE OVERVIEW' and contains the following fields:

- Application ID:** main
- Device ID:** nodo\_abp
- Activation Method:** ABP
- Device EUI:** 01 02 03 04 05 06 07 08
- Application EUI:** 70 B3 D5 7E F0 00 1E 32
- Device Address:** 26 01 10 9B
- Network Session Key:** (hex values: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00)
- App Session Key:** (hex values: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00)

En la consola de administración de TTN seleccionamos el nodo\_abp y desde la opción de device overview copiamos los datos precisos para la conexión.

Volvemos a la configuración en bloques y los introducimos, tienen exactamente el mismo nombre de tal manera nos quedaría como muestra la siguiente imagen.



Así nos quedaría la configuración de la red LoRa copiando los parámetros de la consola de TTN.

Una vez completado los parámetros de conexión ya podemos añadir el resto de bloques para que la práctica quede como se muestra en la siguiente figura.



En la consola veremos el siguiente texto



The screenshot shows the Whitecat N1 ESP32 DEVKIT software interface. At the top, there is a header bar with the text "Whitecat N1 ESP32 DEVKIT" and a dropdown menu, followed by a file icon labeled "caso2.xml" and a language dropdown set to "Español". On the far right of the header is a user profile icon. Below the header is a large white terminal window containing the following text:

```
Lua RTOS beta 0.1 powered by Lua 5.3.4
Lua RTOS-boot-scripts-aborted-ESP32
/ > require("block");wcBlock.delevepMode=true;ofile("/_run.lua")
/ > <blockStart,5>
<blockStart,4>
<blockEnd,4>
Paquete LoRa enviado
```

Y si todo es correcto en la consola de TTN veremos la información tal y como llega desde el Nodo.

The screenshot shows the TTN Application Data interface. The navigation path is Applications > main > Devices > nodo\_abp > Data. The main area is titled "APPLICATION DATA" and has tabs for uplink, downlink, activation, ack, and error. The "uplink" tab is selected. There are filters for time, counter, and port. A single message is listed: a timestamp of 00:13:35, a counter of 6, and a port of 1. The payload is shown as hex: 03 00 00 D9 8E 3E 42 1B 2E 78 44 53 DB AF 41. Below this, under "Payload", the raw hex value 03 00 00 D9 8E 3E 42 1B 2E 78 44 53 DB AF 41 is displayed. Under "Fields", a JSON object is shown:

```
{
  "port": 1,
  "values": [
    47.6394966430664,
    992.7203979492188,
    21.982091903686523
  ]
}
```

**ENHORABUENA !!**, Si esto es lo que observas en la consola de TTN es que tu NODO está enviando la información del sensor cada 30 segundos de una manera correcta, como se puede apreciar en la imagen.

Con este resultado hemos aprendido ya a enviar información desde el nodo a la nube empleando para ello la red de comunicaciones IoT de LoRaWAN en la siguiente práctica veremos cómo recibir un valor ahora desde la nube hacia el nodo y operar en consecuencia en la placa.

## 8.3. Caso 3

### - Descripción

En este tercer caso vamos a conectar la tira de leds direccionables NeoPixel y vamos a encender y apagar leds de dicha tira en base a un valor enviado desde TTN.

### - Objetivos

El objetivo de este tercer caso práctico es una vez asimilados los conocimientos del primer y segundo caso práctico, ver como se envían ahora los datos desde la nube al nodo, de este modo completamos las opciones de comunicación de un nodo con el servidor TTN.

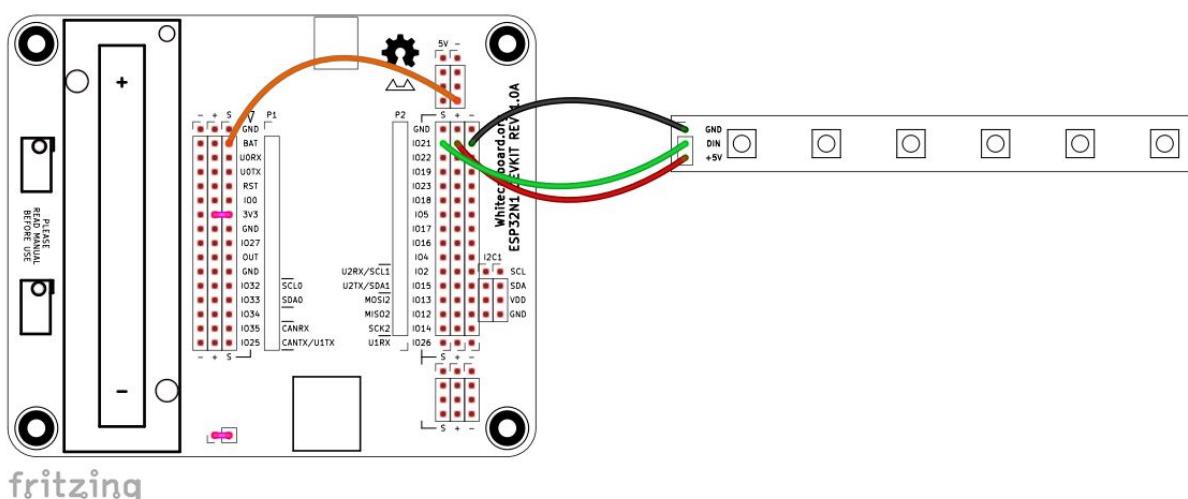
### - Materiales

Para la realización de este caso práctico necesitaremos;

- 1x ESP32N1
- 1x ESP32N1 DEVKIT
- 1x TIRA 6 LEDS NEOPIXEL
- 1x SET DE CABLES
- 1x CONECTIVIDAD LoRaWAN (GATEWAY O ÁREA CON COBERTURA LoRaWAN)

### - Diagrama de Conexión

Este es el diagrama de conexión (recordar si se emplea batería usar el cable naranja desde el pin marcado como bat a un pin +5V)



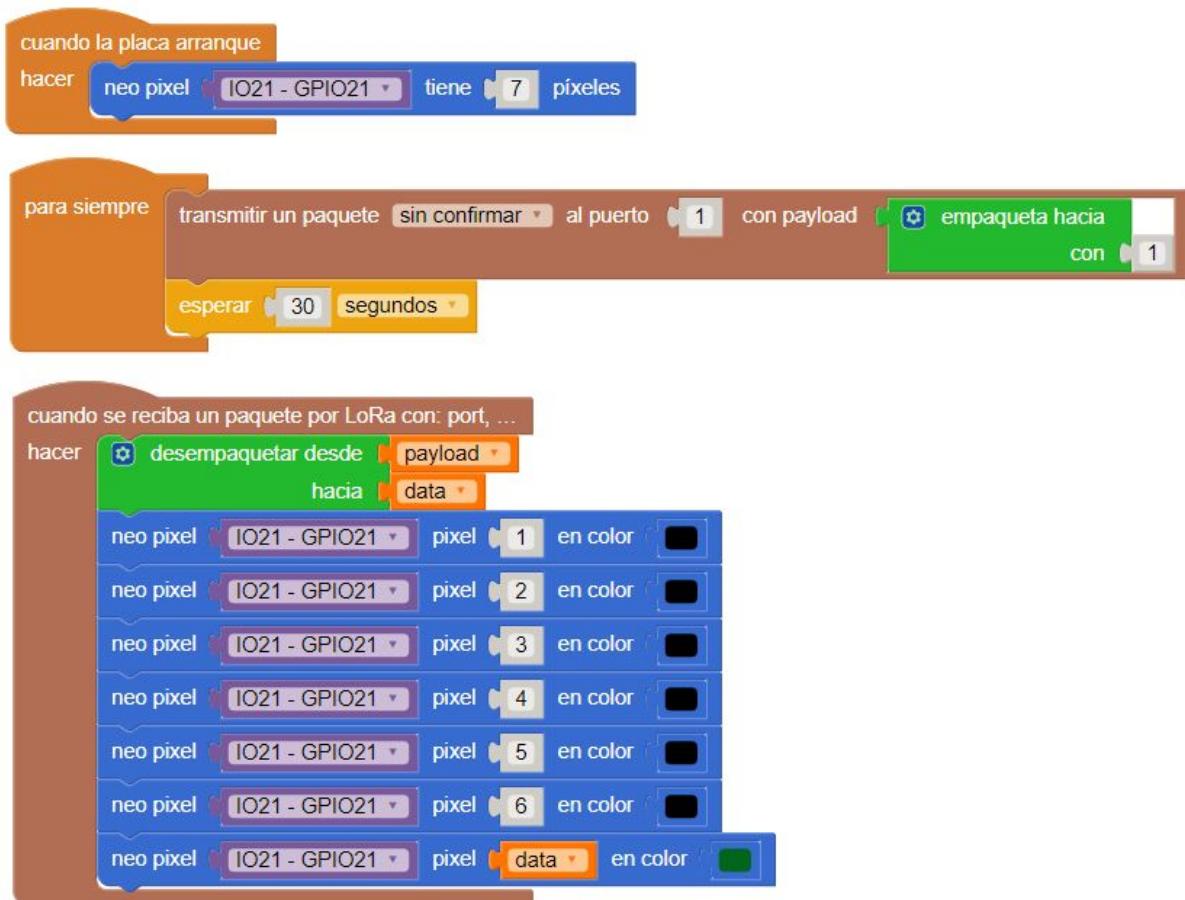


**SE RECOMIENDA DESCONECTAR LA PLACA MIENTRAS SE REALIZA EL CONEXIONADO DE LOS ELEMENTOS DE LA PRÁCTICA.**

Como se comentó en el capítulo de actuadores los leds NeoPixel usan un protocolo parecido al 1-Wire, de tal manera solo necesitaremos 3 cables conectados a la placa 2 para alimentación y un cable para datos en el IO21 en el caso del ejemplo, como muestra el diagrama de conexiones.

### - Programa en bloques

En este caso mantenemos la configuración de Lora como la teníamos en el caso anterior pero agregaremos la opción de recibir datos.



Cuando se inicia el programa se definen el número de leds que hay en la tira en este caso ponemos uno más ya que el 7 lo usaremos para apagar todos los leds.

Acto seguido , es importante recordar que los nodos tipo A de LoRaWAN como es nuestro caso son siempre los que inician la comunicación por eso mantenemos la transmisión de un paquete en este caso con el valor 1 fijo, simplemente a efectos de iniciar la comunicación con el servidor.

Acto seguido escuchamos para ver si recibimos un paquete LoRaWAN desde el servidor si es así el paquete esperado es un número entre 1 y 7 que indicara que NeoPixel encender en color Verde.

Recordemos que en las funciones de Format Payload teníamos el método de downlink para empaquetar por ese motivo usamos el bloque desempaquetar hacia una variable que es la que empleamos después.

La manera de transmitir el dato hacia el nodo se realiza mediante la opción downlink del device como se muestra en la siguiente imagen, por ejemplo para encender el led número 3 en la tira de NeoPixel pondremos algo como esto, dentro del device overview.



The screenshot shows the 'Data' tab of the 'nodo\_abp' device's application data. The interface includes a breadcrumb navigation bar: Applications > main > Devices > nodo\_abp > Data. Below this is a header with tabs for Overview, Data (which is selected), and Settings. A toolbar with pause and clear buttons is at the top right. The main area is titled 'APPLICATION DATA' and contains a table with columns: time, counter, port, payload, port, and values. The table shows three entries:

| time       | counter | port | payload           | port    | values |
|------------|---------|------|-------------------|---------|--------|
| ▼ 12:09:39 |         | 1    | 01 10 03 00 00 00 | port: 1 | [3]    |
| ▲ 12:09:42 | 1       | 1    | 01 10 01 00 00 00 | port: 1 | [1]    |
| ▼ 12:08:51 |         | 1    | scheduled         | port: 1 | [3]    |

Proceso de envío del downlink desde la consola de datos, primero aparece como scheduled y posteriormente cuando recibimos el payload para iniciar la comunicación el valor es enviado.

Simplemente queda confirmar visualmente que el led número 3 se ha encendido en color verde y podemos dar la práctica por finalizada, ya somos capaces de enviar órdenes por LoRaWAN también hacia los nodos.

En el siguiente caso de uso emplearemos la red WiFi para realizar una comunicación de datos vía MQTT contra un servidor mqtt público.

## **9. Conclusiones**

## **10. Anexos**

### **10.1. Esquemáticos**

