Escuela Técnica Superior de Ingeniería del Diseño

Universitat Politècnica de València

Trabajo Fin de Grado

Ingeniería en Electrónica Industrial y Automática

DESARROLLO DE SENSOR WIFI PARA LA MONITORIZACIÓN DE MONÓXIDO DE CARBONO.

Documentos:

1 Memoria

2 Planos

3 Presupuesto

4 Anexos:

4-1: Documentación código

4-2: Hojas de características

Autor:

D. Carlos Calleja Chinillach

Tutor:

D. Ángel Perles Ivars

Valencia, julio de 2020

Gracias a mi tutor por su guía durante el desarrollo del proyecto. A mi familia por apoyarme durante el transcurso de la carrera.

Resumen

En los últimos años, ha habido un gran aumento de los productos que presentan conexión a internet. Esto se debe a que a la acelerada evolución de internet y ha permitido la aparición de tecnologías IoT que permiten la interconexión de dispositivos a través de una red. Además, también han aumentado las facilidades para conectar dispositivos a internet con la aparición de plataformas web pensadas para la interconexión de múltiples dispositivos.

Este trabajo pretende aprovechar estas características para realizar medidas de concentración de monóxido de carbono a treves de un dispositivo conectado a internet. Y de esta manera poder observar los datos medidos por uno o varios dispositivos en una página web ya sea utilizando un PC o cualquier dispositivo inteligente como un smartphone. En este caso se utiliza una red Wi Fi para conectar estos dispositivos a internet.

Palabras clave: IoT, Wi Fi, Internet.

Resum

En els últims anys, hi ha hagut un gran augment en els productes que tenen connexió a Internet. Això es deu al fet que Internet ha evolucionat ràpidament en els últims anys i això ha permès l'aparició de tecnologies IoT que permeten que els dispositius estiguin interconnectats a través d'una xarxa. A més, també han augment les facilitats per a connectar dispositius a internet amb l'aparició de plataformes web dissenyades per a la interconnexió de múltiples dispositius.

Aquest treball busca aprofitar aquestes característiques per realitzar mesuraments de concentració de monòxid de carboni al final d'un dispositiu connectat a Internet. I així poder observar les dades mesurades per un o més dispositius en una pàgina web, ja sigui utilitzant un PC o qualsevol smartphone. En aquest cas, s'utilitza una xarxa Wi Fi per connectar aquests dispositius a Internet.

Paraules clao: IoT, Wi Fi, Internet.

Summary

In recent years, there has been a great increase in the products that have Internet connection. This is because the Internet has evolved rapidly in recent years and this has allowed the emergence of IoT technologies that allow devices to be interconnected through a network. In addition, they have also increased the facilities to connect devices to the Internet with the emergence of Web platforms designed for the interconnection of multiple devices.

This work aims to take advantage of these features to perform carbon monoxide concentration measures at the end of an internet-connected device. And in this way to be able to observe the data measured by one or more devices on a web page either using a PC or any smart device such as a smartphone. In this case, a Wi Fi network is used to connect these devices to the internet.

Key Words: IoT, Wi Fi, Internet.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE SENSOR WIFI PARA LA MONITORIZACIÓN DE MONÓXIDO DE CARBONO.

1. MEMORIA

Autor:

D. Carlos Calleja Chinillach

Tutor:

D. Ángel Perles Ivars

Valencia, julio de 2020

INDICE DEL PROYECTO

Tabla de Figuras

Glosario de siglas y abreviaturas

IOT Internet of Things

CO Monóxido de carbono

ppm Partes por millón

BLE Bluetooth de baja energía

ADC Conversor Analógico Digital

STM STMicroelectronics

USART Transmisor-Receptor Síncrono/Asíncrono Universal

IDE Entorno de desarrollo integrado

1. OBJETO DEL PROYECTO

El objeto o finalidad última de este proyecto es el desarrollo de un dispositivo que sea capaz de medir la concentración de monóxido de carbono en el aire y transmitirla por internet a través de una red inalámbrica.

Además, también es objeto de este proyecto el aprovechamiento de diferentes herramientas online para la visualización en una página web de los datos medidos por uno varios dispositivos conectados a la red.

2. ANTECEDENTES

El monóxido de carbono (CO) es un gas inoloro e incoloro que es ligeramente menos denso que el aire. Este gas puede provocar intoxicación al ser inalado ya que se acumula en el corriente sanguíneo remplazando el oxígeno. Esto puede llegar a generar daños graves e incluso la muerte.

El monóxido de carbono se encuentra en los humos de una combustión cuando el nivel de oxígeno es especialmente bajo para mantener la reacción de combustión. Cuando este humo se inhala el cuerpo remplaza el oxígeno con el monóxido de carbono. Esto provoca que el oxígeno no llegue a los tejidos ni a los órganos.

El mayor peligro de este gas es que los síntomas que provoca no suelen ser muy obvios. Sobretodo, cuando se trata de pequeñas exposiciones. Dichos síntomas son, por ejemplo, mareo, malestar, cansancio, confusión, dolor de estómago y dificultades para respirar. Estos síntomas son muy parecidos a los de un constipado común, lo cual hace que sea difícil percatarse de que se está siendo intoxicado por monóxido de carbono.

A pesar de que la formación del monóxido de carbono requiera de una baja presencia de oxígeno durante la combustión, también pueden darse casos de intoxicación al aire libre cuando se está cerca de una combustión, por ejemplo, cerca de un generador portátil. Sin embargo, donde más problemática puede haber es en espacios cerrados, como en garajes u hogares con calefacción por cualquier tipo de combustión.

La concentración de CO es un factor del nivel intoxicación que puede sufrir un individuo. En [Silent Shadow] se describe el peligro que puede provocar el monóxido de carbono según las diferentes concentraciones medidas en partes por millón (ppm) y tiempo de exposición.

- 50ppm – Dificultades respiratorias
- 200 ppm
  - (2 a 3 horas) – Pequeños dolores de cabeza, fatiga, mareo y nauseas.
- 400 ppm

- (1 a 2 horas) – Fuerte dolor de cabeza en la frente
- (3 a 5 horas) – Perdida de la conciencia y posible muerte
- 800 ppm
- (45 minutos) – Nauseas, convulsiones y mareos.
- (2 a 3 horas) – Posible muerte
- 1600 ppm
- (20 minutos) – Nauseas, convulsiones y mareos.
- (1 hora) – Posible muerte
- 3200 ppm
- (10 minutos) – Nauseas, convulsiones y mareos.
- (1 hora) – Posible muerte
- 6400 ppm
- (10 minutos) – Nauseas, convulsiones y mareos.
- (30 minutos) – Posible muerte
- 12800 ppm
- Muerte en pocos minutos.

Por lo tanto, al ser difícil la detección del monóxido de carbono, es recomendable disponer de instrumentación capaz de detectarlo y de medir la concentración de este gas presente en el aire donde haya presencia de combustión, sobre todo, en espacios cerrados.

3. ESTUDIO DE NECESIDADES, FACTORES A CONSIDERAR: LIMITACIONES Y CONDICIONANTE

El primer paso en el desarrollo de un proyecto es definir tanto requisitos como necesidades para poder dirigir el trabajo. En el caso presente, se divide el desarrollo en diferentes apartados significativos.

3.1. Medida de CO

Dado que la peligrosidad del monóxido de carbono es una exposición prolongada a lo largo de horas, minutos en casos de una de una concentración muy alta, el dispositivo debe tener un sensor de monóxido de carbono capaz de medir la concentración presente en el ambiente en un intervalo del orden de los cinco minutos.

A parte de la frecuencia con la cual se debe de medir, también hay que definir cuál es la sensibilidad del sensor de monóxido. Como se podía ver en [Silent Shadow], incluso una concentración tan pequeña como 50 partículas por millón puede empezar a ser peligrosa, por lo tanto, una sensibilidad de al menos ±5ppm sería algo razonable.

3.2. Comunicación inalámbrica

Con el fin de poder ubicar el dispositivo atendiendo a las mínimas restricciones posibles, la transmisión de datos debe de hacerse de manera inalámbrica. Esta comunicación ha de ser constante y fiable. Además, la red inalámbrica que se use para conectar el dispositivo tiene que estar activa el máximo de tiempo posible, para así poder realizar medidas durante las 24 horas del día.

3.3. Visualización de los datos medidos

La visualización de los datos medidos ya sea por uno o varios dispositivos debe de ser sencilla y realizable desde cualquier dispositivo, ya sea un PC, smartphone, etcétera. Por lo tanto, una manera correcta de visualizar los datos puede ser desde una página web.

4. PLANTEAMIENTO DE SOLUCIONES ALTERNATIVAS Y JUSTIFICACIÓN DE LA SOLUCIÓN ADOPTADA

En este apartado se proponen distintas soluciones a diferentes problemas que se han encontrado durante el desarrollo del proyecto.

4.1. Selección del sensor

El primer problema que resolver es la elección de un sensor capaz de detectar la concentración del monóxido de carbono presente en el aire. Entre la gran variedad de sensores se encuentran los dos descritos a continuación.

4.1.1. MQ-9 Semiconductor Sensor

Este sensor se trata de un sensor de tipo semiconductor cuya conductividad crece en medida que aumenta la concentración del gas. El sensor es capaz de medir concentraciones entre 10 y 1000 ppm de monóxido de carbono.

Dentro del encapsulado del sensor, se encuentra una resistencia que se usa para calentar el sensor, esto se usa para poder medir distintos gases. En el caso de querer medir monóxido de carbono la resistencia se debe de calentar a 1,5V.

Para poder usar este sensor es necesario el desarrollo de hardware para poder hacer medidas desde un microcontrolador. Sin embargo, podemos encontrar módulos que utilicen este sensor ya desarrollados y preparados para ser usados con un microcontrolador. Como podemos ver en la Figura 1.

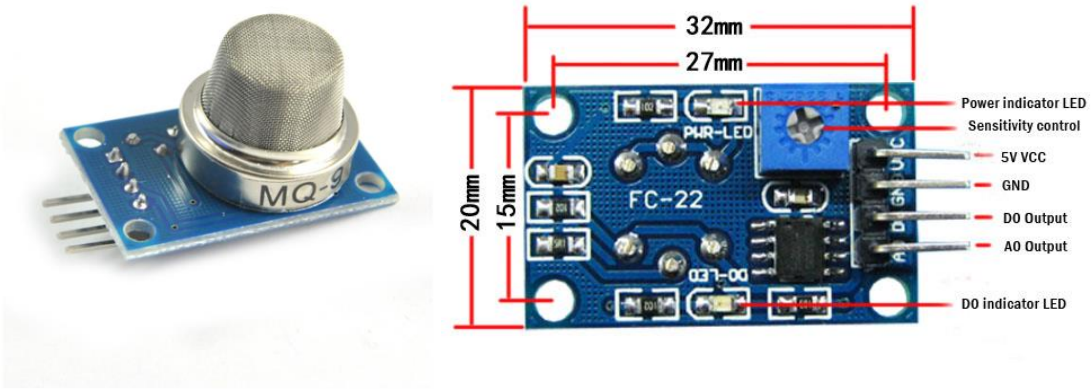


Figura 1: Módulo Sensor MQ-9

Como se puede observar este módulo se puede usar fácilmente con un microcontrolador configurando un ADC.

4.1.2. TGS 5141

El sensor TGS 5141 es un sensor electroquímico cuyo diseño está orientado a su uso en dispositivos de medición portátiles. El rango de detección de monóxido de carbono de este sensor se encuentra entre 0 y 5000 ppm.

Figura 2 TGS 5141

Al igual que el anterior, para poder utilizar este sensor junto a un microprocesador es necesario el desarrollo de hardware. Pero, también existen módulos que utilizan el sensor TGS 5141. Entre ellos se encuentra el módulo P-NUCLEO-IKA2A1 desarrollado por STMicroelectronics. En este caso se trata de una placa de expansión para la familia de placas NUCLEO de STMicroelectronics.

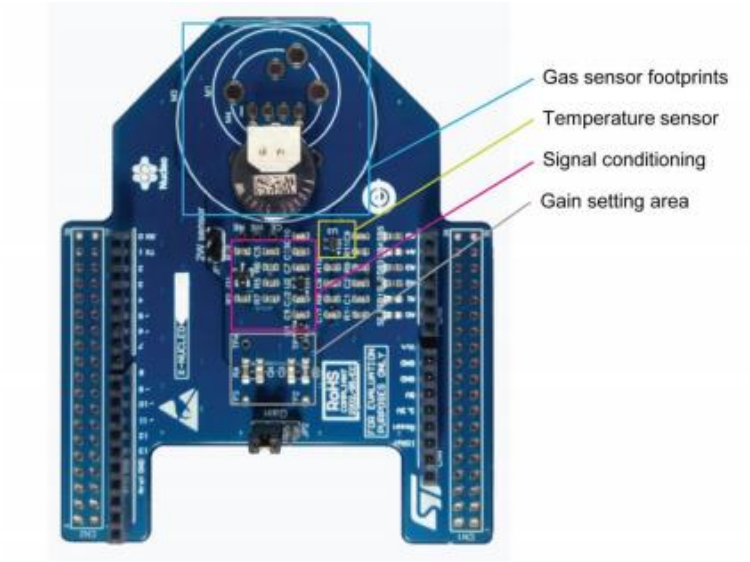


Figura 3: P-NUCLEO-IKA2A1

. Además del hardware, STM también aporta un conjunto de librerías para poder desarrollar prototipos de manera más sencilla.

4.2. Transmisión inalámbrica

Como se especificaba en el estudio de necesidad, la transmisión ha de hacerse de manera inalámbrica para ello hay que estudiar los distintos estándares de comunicación inalámbrica. En la Figura 4 se pueden ver distintos estándares de comunicación inalámbrica ordenados en función del bajo consumo y velocidad de transmisión en el eje vertical y el rango en el eje horizontal.

Figura 4: Principales tecnologías de comunicación inalámbrica.

Dada las características del proyecto, la larga distancia no es necesaria, por lo tanto, los estándares que si resultan interesantes son los siguientes:

4.2.1. Bluetooth de baja energía

Entre los estándares de comunicación inalámbrica se encuentra el BLE, bluetooth low energy (Bluetooth de baja energía).

Figura 5: Logo Bluetooth

Bluetooth en una especificación industrial para redes inalámbricas de área personal que ofrece una comunicación de corto alcance, con un rango óptimo de 10m, pero que puede ser aumentado hasta 100m haciendo uso de repetidores. En la versión 4.0 de esta especificación se integró la funcionalidad de bajo consumo.

Esta funcionalidad del bluetooth opera en 2.4GHz y tiene una tasa de transferencia de hasta 1Mbps, además, emplea el sistema de cifrado AES y esquemas de seguridad configurables.

4.2.2. Wi Fi

La red Wi Fi es una red inalámbrica de área local que permite la interconexión de varios dispositivos. Existen varios tipos de Wifi, todos ellos en un estándar IEEE 802.11. Entre estos tipos la diferencia se trata de la velocidad máxima de transmisión de datos y la banda en la que operan.

Figura 6: Logo Wi Fi

Los estándares IEEE 802.11b, IEEE 802.11g e IEEE 802.11n operan en 2.4GHz al igual que la tecnología bluetooth, lo cual puede provocar interferencia con el resto de las redes que operen en esta banda. La velocidad de transmisión de estos estándares es de hasta 11Mb/s, 54Mb/s y 150Mb/s respectivitamente.

Por último, también existe un estándar de Wifi que opera en 5GHz, se trata del estándar IEEE 802.11ac. Este estándar aparece para resolver los problemas de interferencia que puede provocar el uso de muchos dispositivos de la banda 2.4GHz.

4.3. Plataforma de procesamiento

A la hora de desarrollar un dispositivo que requiera de una unidad de procesamiento, hay que observar los distintos tipos de microcontroladores que hay en el mercado y qué diferencias hay entre ellos. Y, además, a la hora de usar un microcontrolador hay que buscar placas de desarrollo para poder realizar prototipos de manera sencilla y sin necesidad de desarrollar hardware.

4.3.1. StMicroelectronics NUCLEO-L053R8-STM32

La placa de desarrollo NUCLEO-L053R8-STM32 tiene un microcontrolador STM32. Esta placa, presenta conectividad Arduino Uno V3. Lo cual permite una fácil expansión de funcionalidad usando shields con conexión Arduino Uno V3.

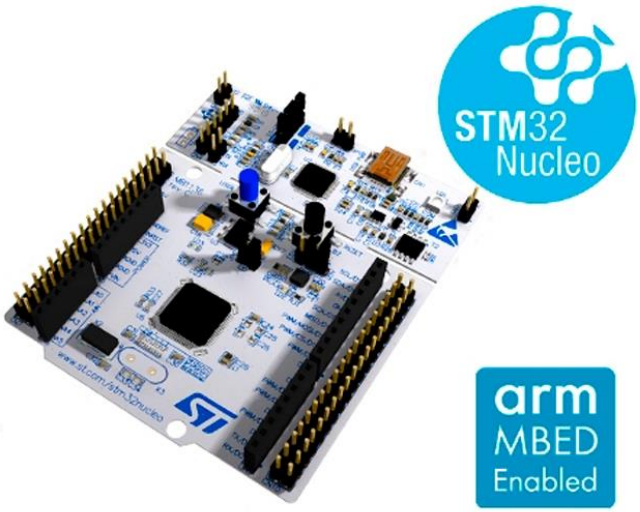


Figura 7: STM32 NUCLEO L053R8

En cuanto a la programación, esta placa tiene incorporada un ST-LINK debugger/programer, por lo tanto, solo es necesario un USB para conectarlo a un pc y poder cargar el código. El código puede ser desarrollado en diferentes plataformas distintas, entre ellas el IDE Keil-MDK-ARM.

4.3.2. Expressif ESP32 Wroom 32

La placa de desarrollo ESP32 Wroom 32 es una placa de desarrollo que dispone módulos de comunicación Wifi y Bluetooth incorporado. Por lo tanto, la creación de proyectos que requieran de alguna de estas comunicaciones se facilita bastante ya que existen librerías para hacer uso de estos módulos.

Figura 8: ESP32 Wroom 32

El módulo Wifi de esta placa soporta los estándares 802.11 b/g/n con una velocidad de hasta 150 Mb/s. Por otro lado, el módulo bluetooth soporta el protocolo v4.2 y BLE.

Por último, la placa ESP32 permite la programación usando el IDE de Arduino lo cual aporta todas las ventajas de programar un Arduino, una gran comunidad que aporta librerías para simplificar el desarrollo de prototipos.

4.4. Alimentación

Una parte importante del desarrollo del dispositivo es la alimentación eléctrica. La cual puede provenir de distintas fuentes de energía.

4.4.1. Red eléctrica

Una forma sencilla será conectar mediante adaptadores el dispositivo a la red eléctrica. De esta manera el dispositivo siempre estará alimentado mientras la red eléctrica siga en funcionamiento.

Sin embargo, el tener que conectar mediante cableado el dispositivo ocasiona restricciones en cuanto a la ubicación de este. Pero, también ofrece la posibilidad de usar conexiones inalámbricas de mayor potencia sin tener que preocuparse por el consumo energético dado que la red eléctrica es capaz de suministrar la potencia necesaria.

4.4.2. Pilas primarias

Otra forma de alimentar el dispositivo sería mediante el uso de pilas primarias. Las ventajas de usar pilas primarias son el bajo coste de estas y la sencillez a la hora de cambiarlas cuando estas se agoten.

Sin embargo, el uso de estas pilas presenta varios problemas, entre ellos la existencia de un límite de potencia que pueden suministrar al dispositivo. Además, las pilas sufren un proceso irreversible, por lo tanto, una vez agotada la energía almacenada en las mismas ya no sirven para nada y hay que deshacerse de ellas, y esta no es una tarea fácil, ya que el reciclaje de las pilas es un proceso difícil.

4.4.3. Baterías recargables

Por último, otra manera de alimentar el dispositivo es usando baterías recargables. De esta manera eliminamos el problema que presentan las pilas primarias a la hora de deshacerse de ellas, ya que estas baterías si pueden ser recargadas y reutilizables.

Sin embargo, estas baterías siguen aportando un límite en el suministro de potencia al dispositivo. Además, añade complejidad al proyecto ya que es necesario el desarrollo para detectar el nivel de batería.

4.5. Plataforma para visualización de datos

Como se especificaba en el estudio de necesidades, la visualización debe de hacerse a través de una página web para poder acceder desde cualquier dispositivo sin necesidad de instalar software.

4.5.1. Thethings.iO

Thethings.iO es una plataforma que el despliegue escalable y flexible para todo tipo de soluciones IoT. Esta plataforma permite la conexión con un gran número de dispositivos que se encuentren conectados a internet. Entre otras funcionalidades, desde la plataforma se puede administrar los dispositivos conectados y permite la visualización de datos en Dashboards. Con todas estas características, thethings.iO cumple todos los requerimientos establecidos.

Figura 9: Logo thethings.iO

Sin embargo, si bien es cierto que se puede crear una cuenta gratuita junto a un mes de prueba, si se quiere usar a largo plazo esta opción requiere de pagos temporales.

4.5.2. Thinger.io

Al igual que la opción anterior, Thinger.io es una plataforma que ofrece tanto la conexión de múltiples dispositivos, la administración de estos y la visualización de datos mediante dashboards configurables.

Figura 10: Logo thinger.io

Al igual que thethings.iO la plataforma Thinger.io también requiere de pagos mensuales para poder usar sus servicios. Sin embargo, esta plataforma sí que dispone

de un plan gratuito sin tiempo limitado, aunque las funcionalidades si que se ven altamente restringidas. A pesar de todo, el plan gratuito sigue permitiendo la conexión de múltiples dispositivos y la visualización de datos en dashboards configurables. Siendo el mayor inconveniente, el escaso número de dispositivos conectados simultáneamente, tratándose de 6 dispositivos totales

5. DESCRIPCIÓN DETALLADA DE LA SOLUCIÓN ADOPTADA

Una vez exploradas las diferentes alternativas propuestas para solucionar cada problema comienza el desarrollo del proyecto. El desarrollo se divide en varios apartados que se describen a continuación.

5.1. Selección de componentes

El primer paso en el desarrollo se trata de la elección de los componentes entre las diferentes alternativas propuestas en el apartado anterior.

5.1.1. Sensor de CO

En primer lugar, se decide el sensor dado que en el reside la principal funcionalidad del dispositivo. Además, la elección de un sensor u otro determina también otras elecciones, como la placa de desarrollo a emplear.

Se decide la utilización del sensor TGS 5141 al existir una placa de expansión que simplifica en gran medida el desarrollo del dispositivo (P-NUCLEO-IKA02A1). Esto se debe a que la placa está pensada para ser insertada directamente en placas de desarrollo de la familia NUCLEO de STMicroelectronics sin necesidad de cablear ni desarrollar ningún hardware. además, STMicroelectronics deja a disposición una librería que simplifica el desarrollo de software.

5.1.2. Red de comunicación inalámbrica

La siguiente elección se trata del estándar de comunicación a utilizar. Al igual que en la selección del sensor, la red elegida también guiará las siguientes elecciones y desarrollo del proyecto.

En este caso, se decide por utilizar la red Wi Fi por la sencillez de conectar el dispositivo a internet.

5.1.3. Plataforma web para la visualización de datos

Una vez elegidos el sensor y la red de comunicación, la siguiente elección recae en la plataforma web para visualizar los datos en línea. Entre las dos alternativas se ha decidido por la plataforma thinger.io.

Se ha elegido esta plataforma debido al plan gratuito para poder hacer pruebas de desarrollo. Además, esta plataforma ofrece librerías para una gran cantidad de placas de desarrollo que simplifican en gran medida el desarrollo de software para poder utilizar esta plataforma.

5.1.4. Plataforma de procesamiento

En cuanto a la plataforma de procesamiento, dado que se ha elegido una placa de expansión para placas NUCLEO de STMicroelectronics, la placa a utilizar debe de pertenecer a esa familia. En este caso se ha elegido la placa de desarrollo NUCLEO-L053R8-STM32.

Sin embargo, esta placa no ofrece el hardware necesario para establecer comunicación a través de una red Wi Fi. Por lo tanto, además de utilizar la placa de STMicroelectronics, también se decide utilizar otra placa de desarrollo, la placa se trata de una ESP32. Esta placa tiene hardware incorporado para la comunicación vía Wi Fi.

Por lo tanto, habrá dos placas comunicadas, una dedicada a tomar medidas usando la placa de expansión P-NUCLEO-IKA02A1 y otra dedicada a la comunicación inalámbrica a través de la red Wi Fi.

En cuanto a la comunicación entre ambas placas, se decide usar el estándar USART, y además, para que la placa que se dedica a medir no se encuentre haciendo medidas constantemente, se configura una interrupción externa a través de un pin de la placa de STMicroelectronics.

5.1.5. Alimentación eléctrica

Dada la elección de comunicar el dispositivo haciendo uso de redes Wi Fi, la posibilidad de alimentar el dispositivo usando cualquier tipo de baterías portátiles queda descartada ya que la red Wi Fi necesita de una gran potencia para ser usada. Por lo tanto, el dispositivo debe de ser conectado a la red eléctrica para obtener alimentación eléctrica.

5.2. Desarrollo

Una vez se ha elegido todos los componentes necesarios solo queda el desarrollo del proyecto. En este caso, el desarrollo se puede dividir en varios apartados diferenciados.

5.2.1. Conexión hardware

La primera parte de desarrollo consiste en decidir de qué manera han de ser conectadas todas las piezas de hardware del dispositivo. Para ello hay que tener en cuenta que necesidades de entradas o salidas tiene cada placa y que pines de estas pueden ser utilizados. Además de configurar los diferentes jumpers en función de las necesidades.

En primer lugar, la placa de expansión P-NUCLEO-IKA02A1, como podemos ver en la documentación (Figura 11) tiene un jumper, JMP2, que define la ganancia que va a utilizar el sensor en la fase de adecuación de la señal. En este caso se quiere una ganancia de 470k. Por lo tanto, hay que cortocircuitar los pines 2 y 3 del JMP2.

Position	Gain	Capacitor value	Max. sensor current
Pins 1 and 2 shorted	47 k	100 n	60 µA
Pins 2 and 3 shorted	470 k	1 µ	6 µA

Figura 11: Configuracion JMP2 P-NUCLEO-IKA02A1

Una vez terminada la configuración de hardware de la placa de expansión queda conectarla a la placa de desarrollo NUCLEO-L053R8, en este caso se trata de introducir la placa de expansión en el conector Arduino v3 del que dispone la placa de desarrollo.

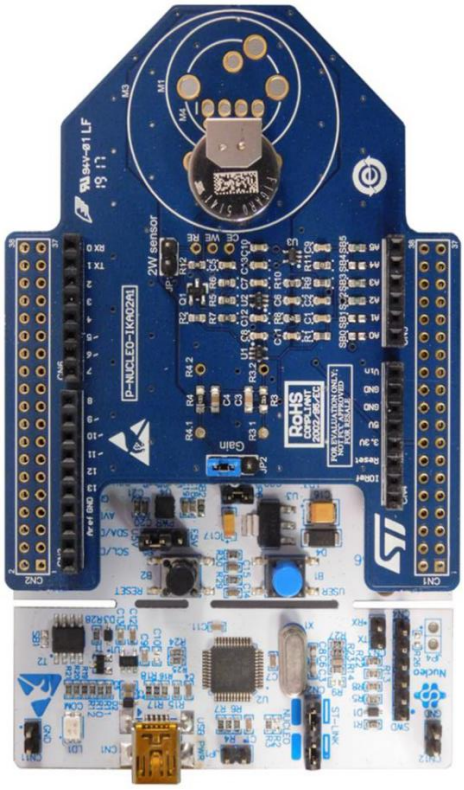


Figura 12: P-NUCLEO-IKA02A1 intalada en NUCLEO-L053R8

A continuación, el siguiente paso se trata de elegir los pines para conectar las placas NUCLEO-L053R8 y ESP32. Como se había decidido usar una comunicación USART entre ellas, es necesario la conexión de 3 pines para establecer la comunicación (GND, RX, TX). Pero, además, también se ha decidido la utilización de una interrupción externa en la placa NUCLEO-L053R8 cuando un pin de entrada pasa de estado LOW a estado HIGH, controlado desde el ESP32. Por lo tanto, hará falta de la utilización de 3 pines de cada placa, además de conectar las tierras de ambas.

Una vez decididos cuantos pines se van a utilizar de cada placa, solo queda localizar en la documentación de cada placa que pines se pueden utilizar para cada función. En [os.mbed.com] se encuentras las imágenes que describen la configuración de los distintos pines de la placa NUCLEO-L053R89 y que pueden servir para orientarse en la selección de pines. En este caso es interesante la siguiente imagen que representa las funciones de lo pines que se encuentran en la parte derecha de la placa.

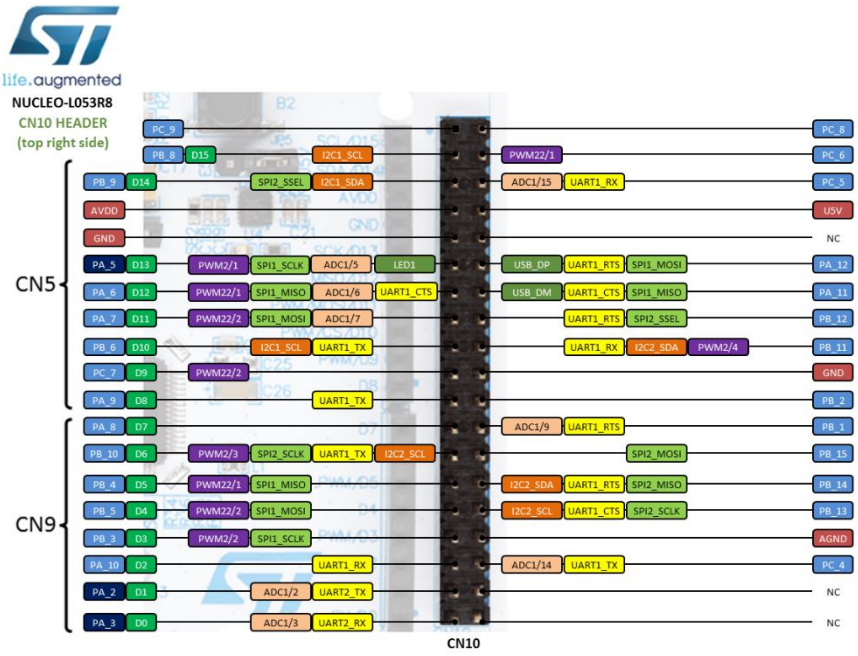


Figura 13: NUCLEO-L053R8 PINOUT

Como se puede observar en la Figura 13, los pines PA9 Y PB10 pueden ser utilizados para establecer comunicación USART. Por otra parte, queda decidir qué pin se va a usar para crear la interrupción externa, como cualquier pin de entrada y salida de propósito general se puede usar como fuente de interrupción, se escoge el pin PB4 por encontrarse junto a los pines elegidos y que todos queden juntos para simplificar a la hora de conectar.

Ya elegidos los pines a utilizar por la placa de STMicroelectronics, toca decidir qué pines usar en la placa ESP32. Al igual que en la anterior hará falta 3 pines, 2 para la comunicación USART y de salida para generar la interrupción en la otra placa.

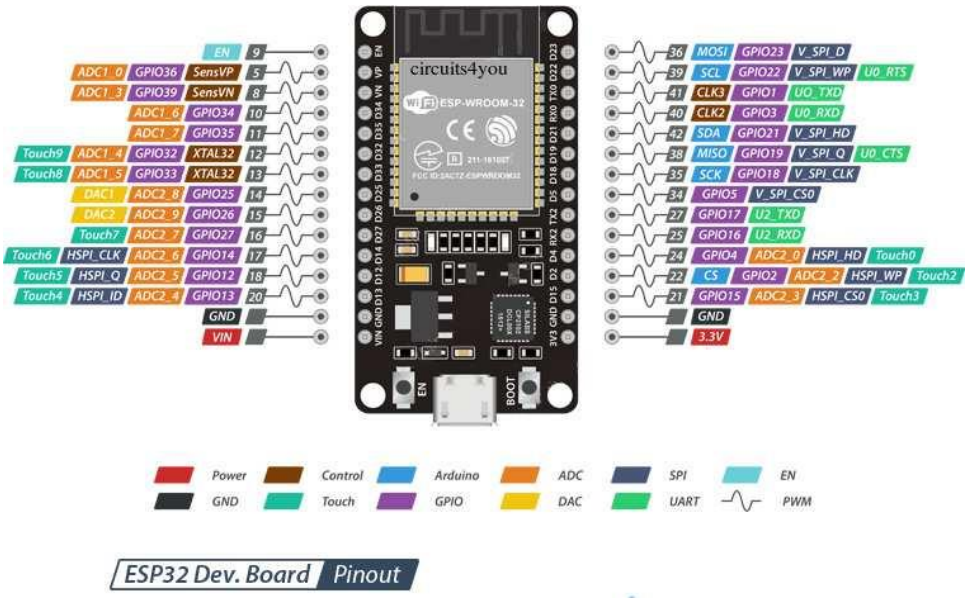


Figura 14: ESP32 PINOUT

Como se puede encontrar en la documentación, la placa tiene 3 interfaces seriales compatibles con hardware (USART0, USART1, USART2). Sin embargo, de estas tres opciones, USART0 se utiliza para la programación y para comunicación cuando se enciendo o reinicia la placa y USART1 se usa en ciertas placas para acceder a la memoria flash mediante la interfaz SPI. Por tanto, solo queda libre la interfaz USART2, cuyos pines son 16(Rx) y 17(Tx).

En cuanto al pin de salida para generar la interrupción en la otra placa, cualquier pin de salida sirve para ello. Por tanto, se escoge utilizar el pin 5.

5.2.2. Desarrollo de software

El siguiente paso es la programación de las dos placas de desarrollo, por tanto, se ha dividido este apartado en dos subapartados correspondientes a cada una de las placas.

5.2.2.1. NUCLEO-L053R8

Para el desarrollo de software de la placa NUCLEO-L053R8 se ha hecho uso de dos programas, uno para generar un proyecto que inicialice los módulos necesarios de la placa y otro para hacer la programación sobre el dicho proyecto base. Estos programas en cuestión se tratan de:

- STM32CubeMX
- Keil uVision5

5.2.2.1.1. STM32CubeMX

Una vez instalados los programas necesarios para poder realizar la programación, el primer paso es generar el proyecto base. Para ello se usa el programa STM32CubeMX, en la ventana iniciar del programa, se puede crear un proyecto eligiendo una placa como se puede ver en la Figura 15.

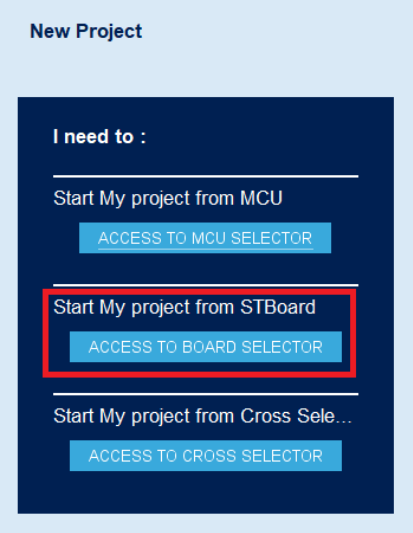


Figura 15: Creación de nuevo proyecto en STM32CubeMX

Seleccionada la placa deseada, en este caso la placa NUCLEO-L053R8, el programa cargará una ventana donde se puede configurar los diferentes módulos de la placa. En la Figura 16 se puede observar un proyecto vacío con el microcontrolador seleccionado.

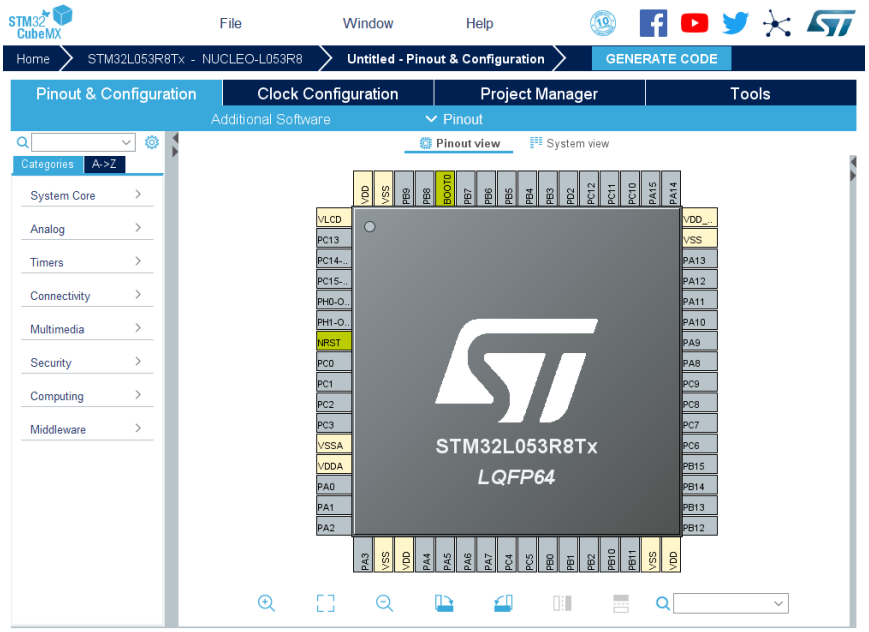


Figura 16: STM32CubeMX proyecto vacío

Desde esta ventana se puede configurar los diferentes módulos necesarios, en este caso se trata del modulo USART y de la interrupción externa. Para configurar la comunicación USART, hay que desplegar la categoría “Connectivity”, que se encuentra en lado izquierdo en la Figura 16: STM32CubeMX proyecto vacío, una vez desplegado se selecciona USART1 (Figura 17), esta se trata de la interfaz cuyos pines se habían decidido usar para la comunicación.

Figura 17: Selección USART1 STM32CubeMX

Una vez mostrado el menú de configuración de la interfaz USART1 queda seleccionar y definir los diferentes apartados de la configuración. En la Figura 18 se puede observar los diferentes campos necesarios para definir el funcionamiento de la interfaz.

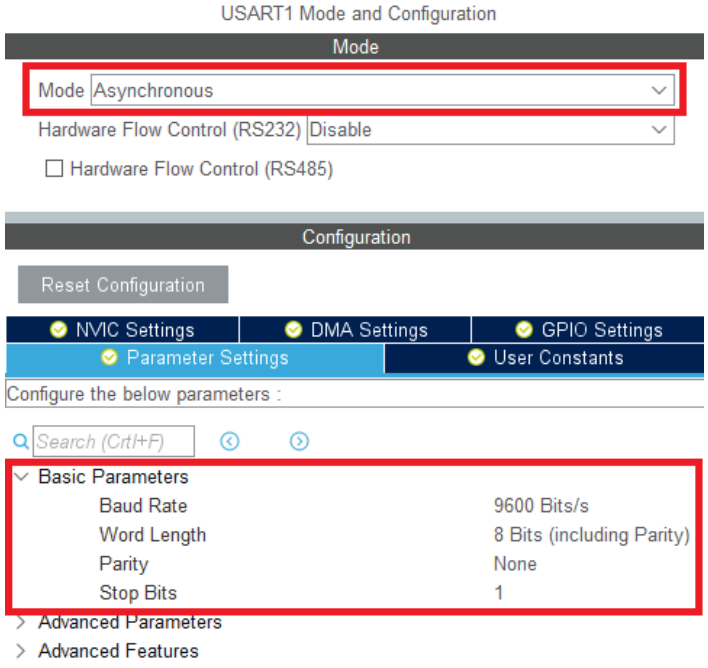


Figura 18: Menú de configuración USART1 de STM32CubeMX

Tras configurar la interfaz de comunicación USART1, el siguiente paso es la configuración de la interrupción externa. En este caso, para configurar la interrupción, hay que hacer click sobre el pin deseado y aparecerá un menú desplegable (

Figura 19) donde se selecciona “GPIO_EXTI4”.

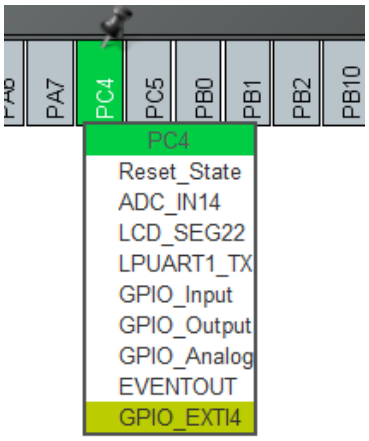


Figura 19: Funcionalidades pin PC4

Después de haber creado la interrupción queda configurar sus propiedades. Para ello, al igual que con la configuración de la interfaz USART1, a la izquierda de la ventana principal del proyecto (Figura 16) se despliega el apartado “System Core” y dentro de

este apartado se abre el menú GPIO. En este menú aparecerá el pin PC4 que anteriormente se había definido su funcionamiento como fuente de interrupción externa (Figura 19). En este menú se selecciona el pin PC4 y aparecerá un menú, como se puede ver en la Figura 20, en ese menú se indica que la interrupción ha de ocurrir cuando el estado del pin pasa de estar en LOW a estar en HIGH, es decir, en el flanco de subida.

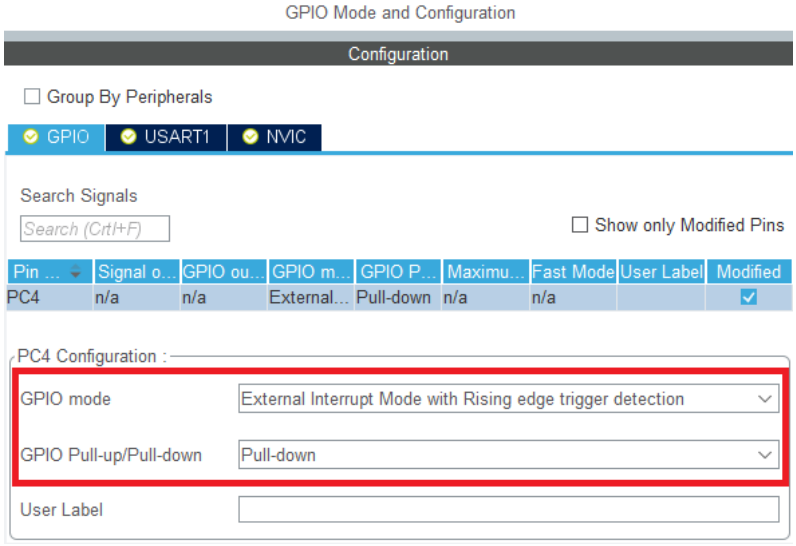


Figura 20: Configuración pin PC4

De esta manera queda completada la configuración de los módulos tanto de interrupción como de comunicación de la placa NUCLEO-L053R8

Además, el mismo programa también permite configurar el reloj de la placa, para ello hay que abrir la ventana de configuración del reloj. Dentro de esta ventana se encuentra un botón que sirve para poner la configuración del reloj en el modo por defecto. Como en el caso de este proyecto no es necesaria la configuración de una velocidad de reloj específica se puede hacer uso de esta utilidad y colocar la configuración del reloj en el modo por defecto.

El último paso en este programa consiste en generar el proyecto para continuar con la programación en el entorno de desarrollo Keil uVision 5. Para ello se abre la venta "Project Manager". Donde se puede elegir el nombre del proyecto, la ruta en la cual se va a generar y la IDE para la cual se va a generar el proyecto (**¡Error! No se encuentra el origen de la referencia.**).

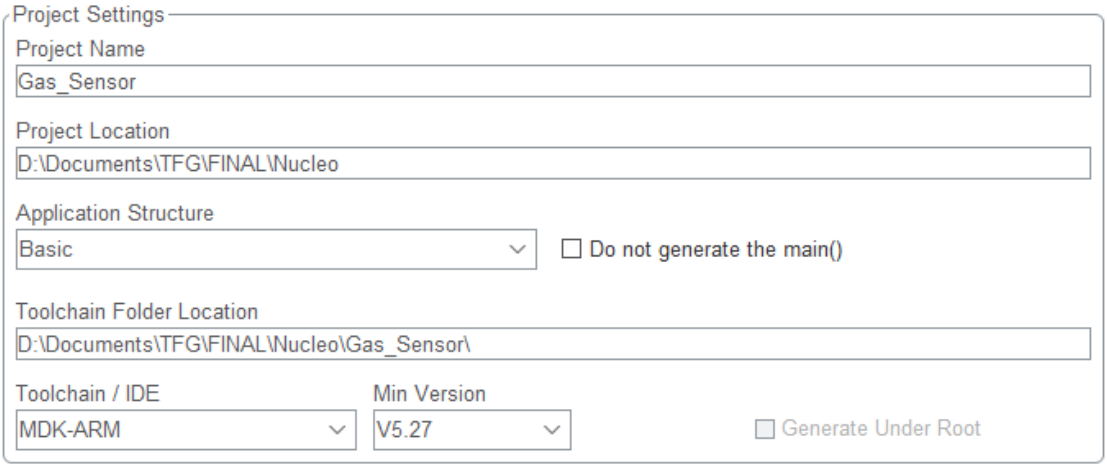


Figura 21: Configuración generación de proyecto

Una vez configurada la generación del proyecto solo queda generarlo. Para ello se pulsa el botón que se encuentra en la parte superior derecha del programa (Figura 16) en el que pone “GENERATE CODE”. Esto generara el código con todas las librerías HAL necesarias para el funcionamiento tanto de la comunicación como de la interrupción externa.

5.2.2.1.2. Programación

Una vez generado el proyecto por el software STM32CubeMX, el siguiente paso es la programación del microcontrolador usando para ello el IDE Keil uVision 5. Para la programación se han usado las siguientes funciones de sus respectivas librerías.

Librería stm32l0xx_hal

Estas son las funciones de la librería HAL necesarias para completar el proyecto generado.

- `void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)`: El código programado en el cuerpo de esta función es ejecutado cuando ocurre una interrupción externa.
- `HAL_StatusTypeDef HAL_UART_Transmit(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)`: Esta función envía datos mediante el puerto USART configurado.

Librería x_nucleo_ika02a1.h

En cuanto a las funciones utilizadas en el código correspondientes a la librería x_nucleo_ika02a1:

- `float Get_Gas_concentration(float gain, float sens)`: Lee el valor analógico del pin al que se conecta el sensor de gas y devuelve la concentración de gas en ppm.
- `float Get_Temperature(void)`: Lee el valor analógico del pin en el que se conecta el sensor y devuelve el valor de la temperatura en grados.
- `float Get_comp_gas_concentration(float gas_ppm, int8_t temp)`: Esta función devuelve el valor de la concentración de gas compensado con la temperatura en ppm

5.2.3. ESP32

5.2.3.1. Programación ESP32

A continuación, se procede a programar el código necesario para conectar la placa ESP32 con la plataforma thinger.io. Para ello lo primero es configurar un IDE para poder desarrollar código y cargarlo en la placa. En este caso se decide usar la IDE de Arduino al ser compatible con las placas ESP32.

Para poder utilizar el IDE de Arduino con la plataforma ESP32, hay que configurar ciertos aspectos del IDE. Desplegando el menú "Archivo" que se encuentra en la barra de herramientas superior y pulsando sobre preferencias se abre la ventana de preferencias del IDE. En la parte inferior de esta ventana se enuentra un campo llamado "Gestor de URLs Adicionales de Targetas" en el cual hay que escribir la siguiente URL: https://dl.espressif.com/dl/package_esp32_index.json (Figura 22).

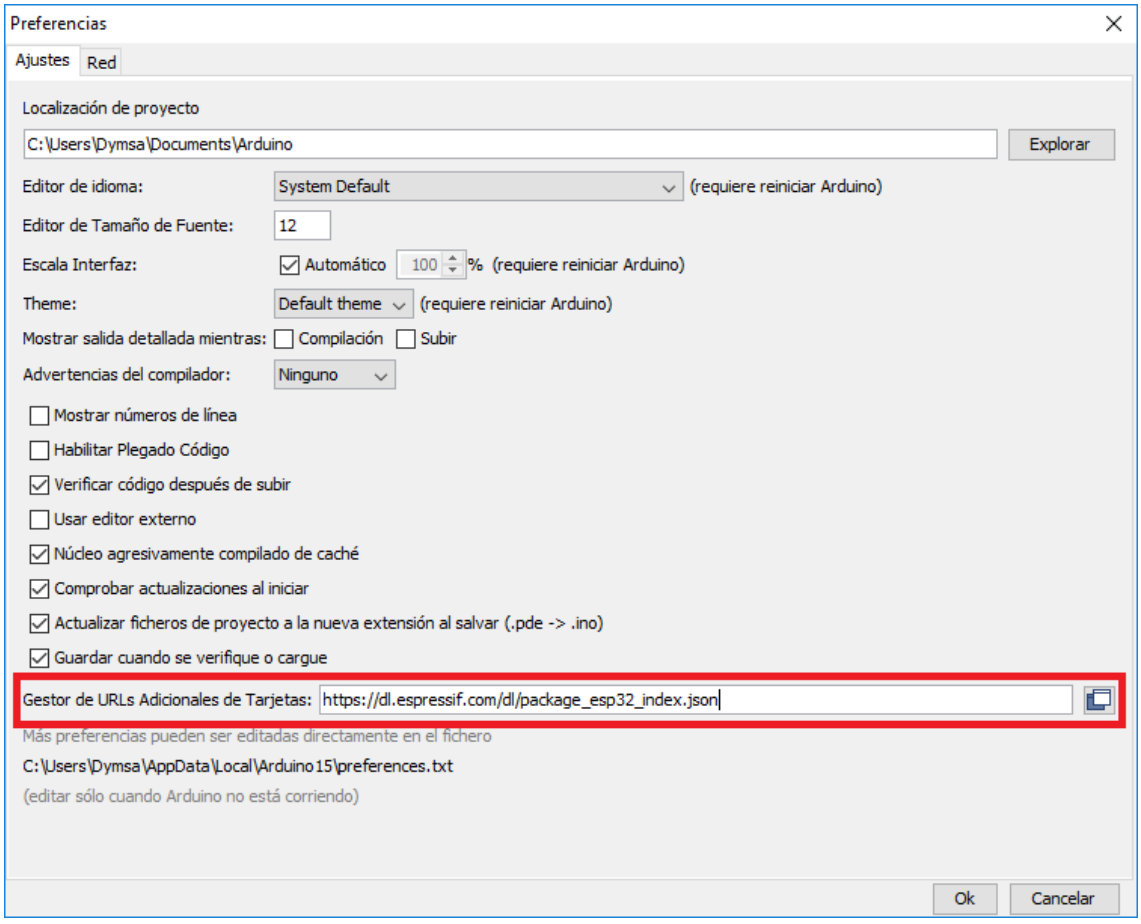


Figura 22: Preferencias IDE Arduino

Una vez añadida esta URL aparecerán las diferentes placas de ESP como objetivo de desarrollo, se selecciona la placa ESP32-Wroom32 como objetivo del proyecto desplegando el menú de herramientas.

Por último, antes de empezar a programar, se importa la librería de thinger.io para ESP32. Para ello se abre el gestor de librerías, que se encuentra en el menú desplegable de herramientas, y se busca “thinger.io”. Entre las librerías se selecciona la llamada “thinger.io” y se pulsa sobre el botón instalar (Figura 23).

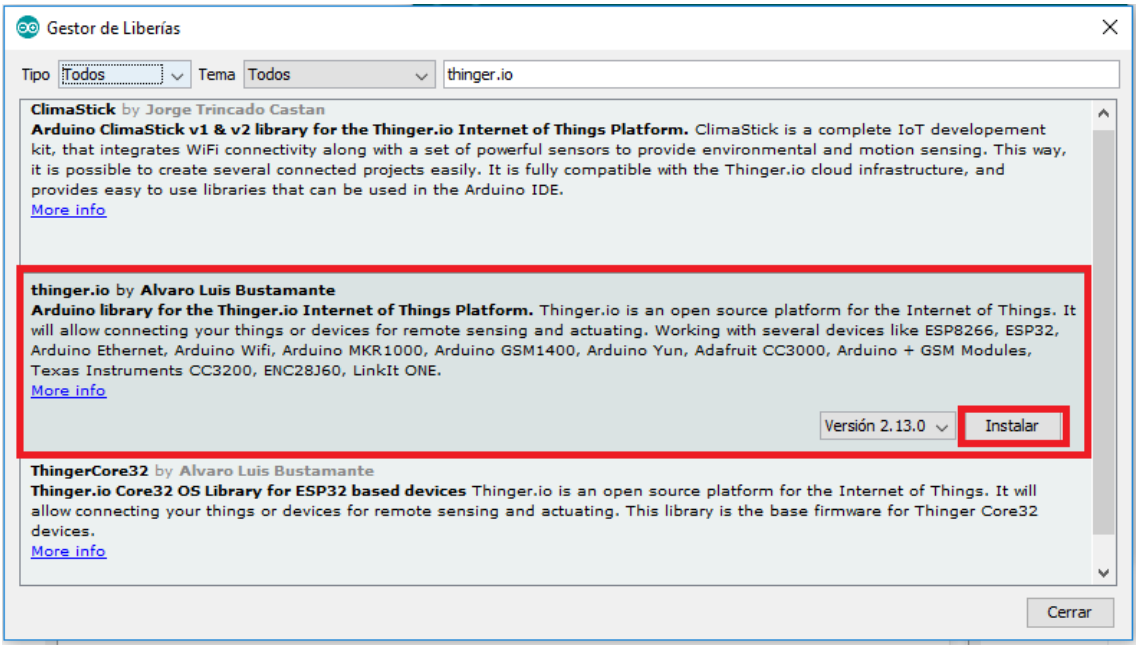


Figura 23: Gestor de librerías de Arduino

Una vez terminada la configuración del IDE de Arduino y la instalación de las librerias de thinger.io ya se puede empezar a programar el codigo de la placa ESP32. En [docs.thinger.io], en el apartado “Quick Start” se puede encontrar el siguiente código de ejemplo que utiliza conexión Wi Fi con esta placa (Figura 24), a partir del cual se puede programar las funcionalidades deseadas.

```
#include <ThingerESP32.h>

#define USERNAME "your_user_name"
#define DEVICE_ID "your_device_id"
#define DEVICE_CREDENTIAL "your_device_credential"

#define SSID "your_wifi_ssid"
#define SSID_PASSWORD "your_wifi_ssid_password"

ThingerESP32 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);

  thing.add_wifi(SSID, SSID_PASSWORD);

  // digital pin control example (i.e. turning on/off a light, a relay, configuring a parameter, etc)
  thing["led"] << digitalPin(LED_BUILTIN);

  // resource output example (i.e. reading a sensor value)
  thing["millis"] >> outputValue(millis());
}

void loop() {
  thing.handle();
}
```

Figura 24: Programa ejemplo ESP32 de Thinger.io

5.2.3.2. Configuración de la plataforma Thinger.io

El siguiente paso es la configuración de la plataforma Thinger.io para poder conectar el dispositivo a internet y poder observar las medidas que realiza. El primer paso consiste en configurar la plataforma web para conectar el dispositivo a la misma. Para ello, es necesario crear una cuenta a la que se asociaran los dispositivos que se conecten. Al iniciar sesión en la plataforma aparecerá la ventana principal en la cual se puede observar a la izquierda una barra de navegación (Figura 25).

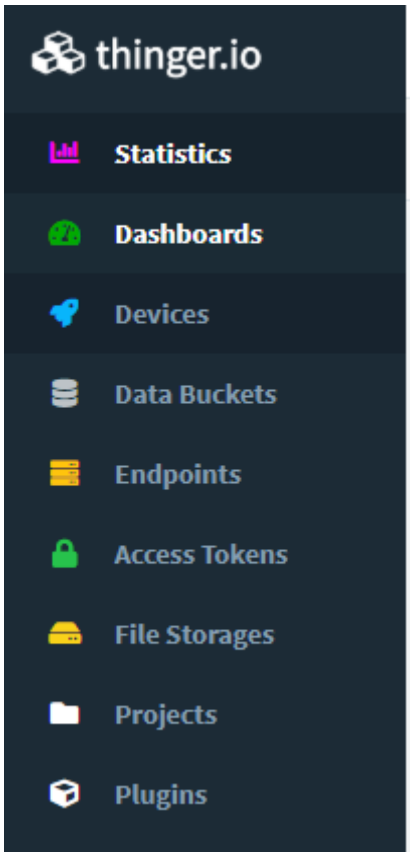


Figura 25: Barra de navegación Thinger.io

Tras entrar en la plataforma e iniciar sesión, lo siguiente es crear un nuevo dispositivo en la plataforma, para ello se pulsa en el apartado “Devices” en la barra de navegación (Figura 25). Dentro de esta página, aparecerá una lista de los dispositivos dados de alta por el usuario. Para dar de alta un nuevo dispositivo se pulsa sobre el botón “Add device”, al pulsar esto aparece un diálogo donde se ha de cumplimentar información para configurar el dispositivo (Figura 26).

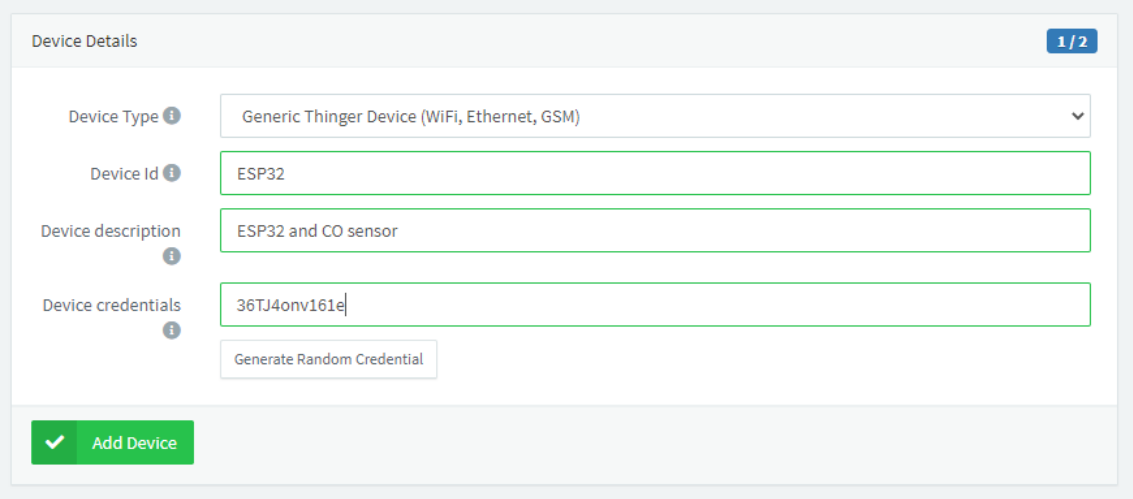


Figura 26: Configuración nuevo dispositivo Thinger.io

Una vez dado de alta el dispositivo, hay que crear un data bucket, que es donde se almacenarán las medidas realizadas por el dispositivo, para ello se pulsa sobre el apartado "Data buckets" en la barra de navegación. Al pulsar aparecerá la lista de data buckets creados por el usuario. En eta página se encuentra un botón para crear un nuevo data bucket, al pulsar sobre el aparecerá un dialogo que hay que rellenar con la información necesaria para configurar como se va a aportar las medidas al data bucket (Figura 27).

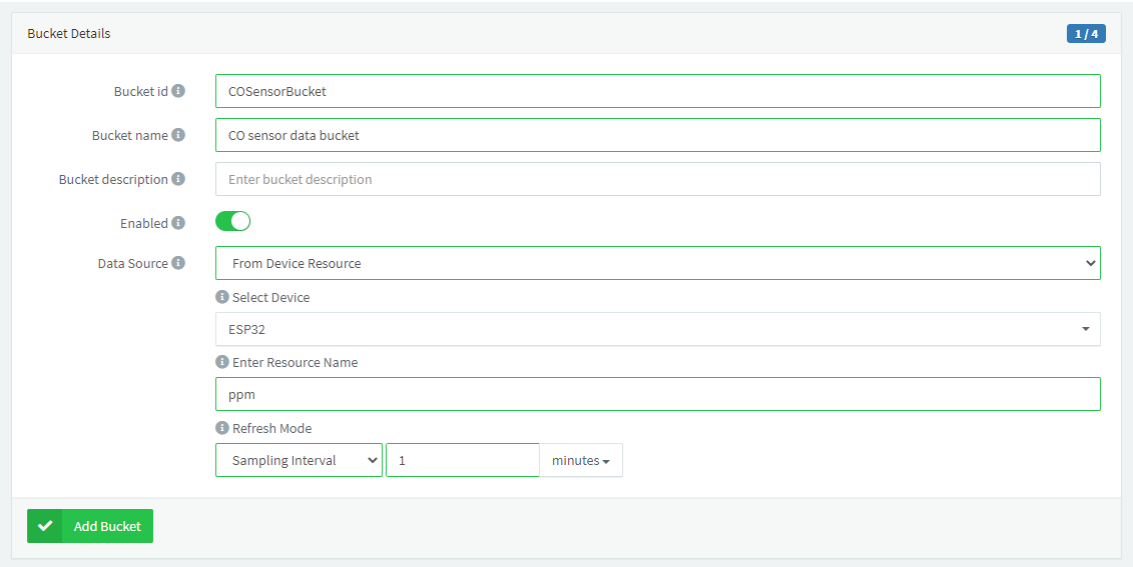


Figura 27: Configuración nuevo Data Bucket

Entre las diferentes posibilidades para obtener las medidas, se elige la opción "From Device Resource", al elegir esta opción la página pedirá desde que dispositivo se ha de coger la información, dando a elegir entre los dispositivos dados de alta. Una vez elegido

el dispositivo, la plataforma también preguntará de qué modo ha de refrescarse la información. En este caso se escoge la opción “Sampling Interval” y se pone a un minuto. Esto hará que la propia plataforma sea la que se encargue de hacer que el dispositivo haga la medición y transmita la información.

Terminada la configuración del data bucket, solo queda configurar un dashboard para poder ver los datos medidos de una manera cómoda. Para ello se pulsa sobre el apartado “Dashboards” en la barra de navegación (Figura 25). Al pulsar aparecerán los dashboards creados por el usuario y, al igual que en los apartados anteriores, un botón para crear un nuevo dashboard. Al pulsar la plataforma pedirá cierta información (Figura 28).

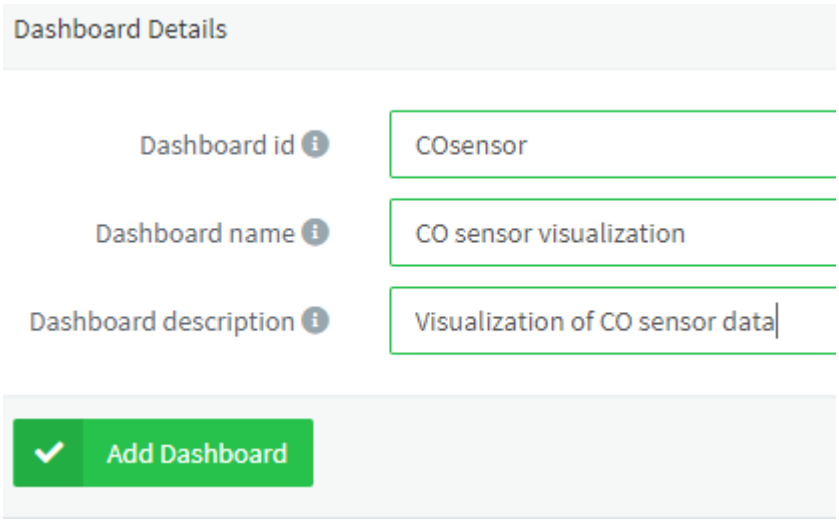


Figura 28: Configuración nuevo Dashborad en Thinger.io

Al crear el dashboard, se puede configurar la informacion que aparece haciendo usos de widgets. En este caso se ha colocado un widget que muestra una gráfica donde se podrá ver las medidas a lo largo del tiempo. Este widget permite configurar el periodo a mostrar. Además, se ha colocado un widget para mostrar la hora. Así pues, el dasboard tendrá la siguiente forma (Figura 29).

Figura 29: Dashboard sensor de monóxido

6. RESULTADOS

Terminado el desarrollo, se procede a realizar unas pruebas de funcionamiento. Así pues, se enciende el dispositivo y se conecta al Wi Fi para el cual se ha programado. En este la red Wi Fi procede de un smartphone Android usando la característica de zona Wi Fi.

En cuanto el dispositivo se conecta a internet, empiezan a aparecer las medias cada minuto en el dashboard de thinger.io (Figura 30). Para comprobar que el sensor funciona correctamente se utiliza un mechero para generar monóxido de carbono. Como se puede observar en la gráfica, la concentración de monóxido crece momentáneamente mientras el mechero se encuentra encendido.

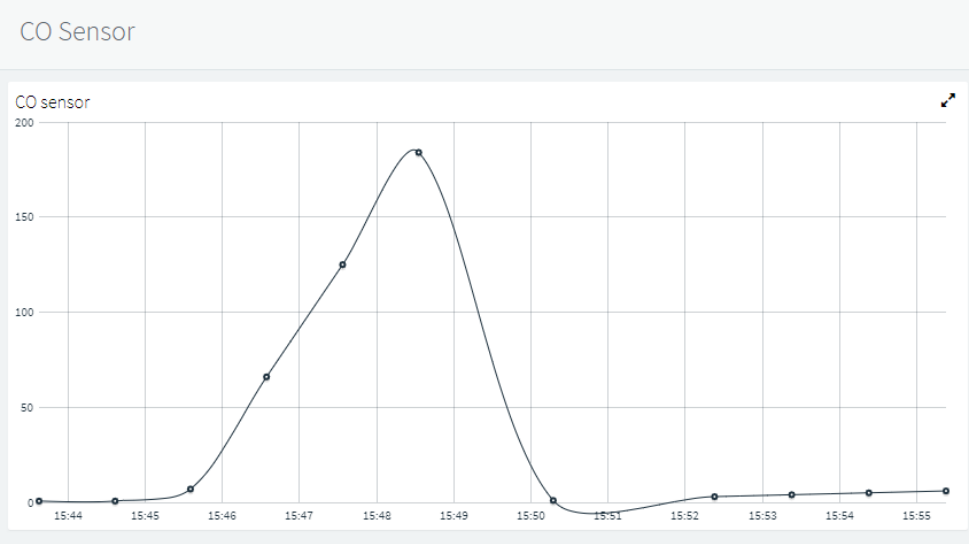


Figura 30: Prueba funcionamiento dashboard Thinger.io

7. CONCLUSIONES

El proyecto es una oportunidad para aplicar lo aprendido durante el transcurso del grado, además de ampliar el conocimiento en ciertas áreas. También ha servido para aprender sobre tecnologías actuales.

En cuanto a la realización del proyecto, se decidió utilizar productos existentes en el mercado que no requieras del desarrollo de hardware para el funcionamiento del dispositivo y así simplificar el desarrollo.

Esta misma estrategia se ha seguido también a la hora del desarrollo de software del proyecto, haciendo uso siempre que se podía de librerías ya desarrolladas para conseguir un desarrollo más rápido de un prototipo.

Como se puede observar a lo largo del proyecto, hoy en día es muy fácil la conexión de cualquier dispositivo a la red gracias a las facilidades que ofrecen distintas plataformas como la utilizada en este trabajo (Thinger.io). Por lo tanto, no es de extrañar que dentro de unos años veamos muchos dispositivos conectados a internet.

Además, la plataforma de programación Arduino tiene una gran comunidad que ofrece librerías de código abierto, lo cual puede ayudar también a realizar proyectos de manera más sencilla sin tener que desarrollar uno mismo estas librerías.

Respecto a la utilidad de este dispositivo, opino que presenta un gran potencial por la sencillez de desarrollo y el poder usarlo en hogares que usen calefacción mediante combustión o en garajes tanto particulares como públicos donde se puede dar una gran concentración de monóxido de carbono debida a una falta de ventilación adecuada.

8. BIBLIOGRAFÍA

[Carbon monoxide 2017] Portable Generator Danger Even Outside. Disponible en: https://carbonmonoxide.com/2017/06/portable-generator-danger.html

[CDC 2009] Carbon Monoxide Poisoning. Disponible en: https://www.cdc.gov/co/es/faqs.htm

[NHZ 2019] Carbon monoxide poisoning. Disponible en: https://www.nhs.uk/conditions/carbon-monoxide-poisoning/

[Silent shadow] The Dangers of Carbon Monoxide Disponible en: https://www.silentshadow.org/the-dangers-of-carbon-monoxide.html

[Thethings.iO] Plataform features, Disponible en: https://thethings.io/iot-platform-features/

[os.mbed.com] Nucleo-lr53r8 pinout, Disponible en: https://os.mbed.com/platforms/ST-Nucleo-L053R8/

[docs.thinger.io] Documentación de thinger.io. Disponible en: https://docs.thinger.io/

DESARROLLO DE SENSOR WIFI PARA LA MONITORIZACIÓN DE MONÓXIDO DE CARBONO.

2.PLANOS

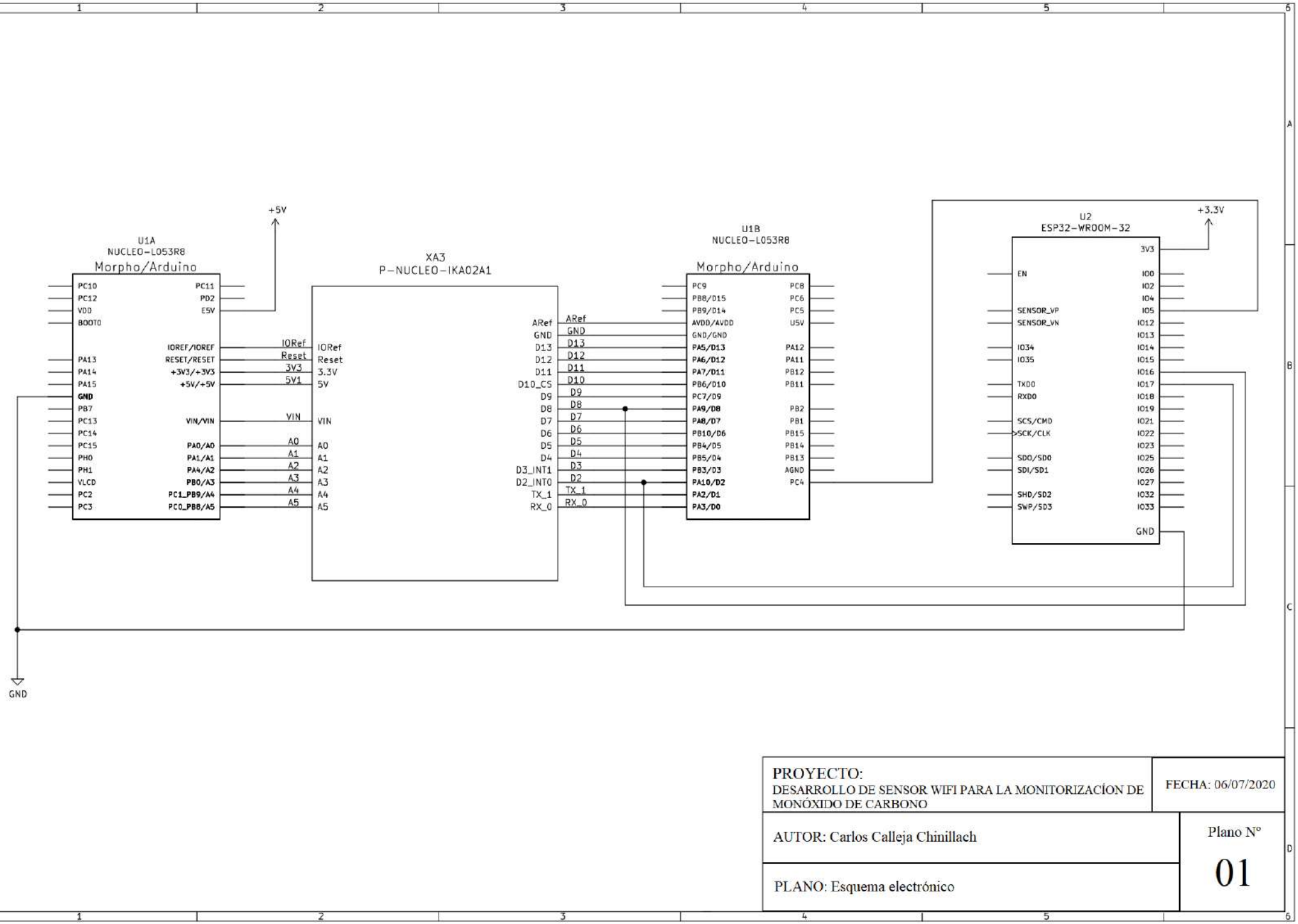
Autor:

D. Carlos Calleja Chinillach

Tutor:

D. Ángel Perles Ivars

Valencia, julio de 2020



PROYECTO: DESARROLLO DE SENSOR WIFI PARA LA MONITORIZACÍON DE MONÓXIDO DE CARBONO	FECHA: 06/07/2020
AUTOR: Carlos Calleja Chinillach	Plano Nº 01
PLANO: Esquema electrónico	

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE SENSOR WIFI PARA LA MONITORIZACIÓN DE MONÓXIDO DE CARBONO.

3. PRESUPUESTO

Autor:

D. Carlos Calleja Chinillach

Tutor:

D. Ángel Perles Ivars

Valencia, julio de 2020

Contenido Presupuesto

1. Presupuesto

El presente presupuesto abarca tanto los costes de los materiales y componentes como los costes de la mano de obra necesaria tanto para el diseño, la programación y el montaje del proyecto.

Los costes de componentes y materiales han sido obtenidos comparando entre distribuidores autorizados como Mouser Electronics y Digikey a fecha de junio de 2020. El precio de estos puede variar en función del distribuidor, fecha de compra, gastos de envío y compras al por mayor.

En este presupuesto no se incluyen los precios del software empleado ni el precio del ordenador necesario para el desarrollo del proyecto.

1.1. Coste de componentes y materiales

A continuación, la lista de componentes y materiales junto con su correspondiente coste. El subtotal de este apartado corresponde al precio por dispositivo.

Producto	Unidades	Precio
NUCLEO-L053R8	1	12,20 €
P-NUCLEO-IKA02A1	1	37,91 €
ESP32-Wroom32	1	13,27 €
10 cables de puenteo WIRE JUMPER FEMALE TO FEMALE 15CM	1	2,7
Subtotal:		66,08 €

1.2. Coste de mano de obra

A continuación, los costes de mano de obra con respecto a las horas empleadas.

Concepto	Unidades	Precio (€/h)	Total
Horas de diseño, programación y montaje	300	45	13.500,00 €
		Subtotal:	13.500,00 €

1.3.Resumen

Por último, la suma total del coste teniendo en cuenta componentes, materiales y mano de obra.

Concepto	Precio
Coste componentes y material	66,08 €
Coste mano de obra	13.500,00 €
Total sin I.V.A.	13.566,08 €
I.V.A.(21%)	2.848,88 €
TOTAL	16.414,96 €

El coste total del proyecto asciende a DIECISEIS MIL CUATROCIENTOS CATORCE EUROS CON NOVENTA Y SEIS CENTIMOS (**16.414,96**)

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

DESARROLLO DE SENSOR WIFI PARA LA MONITORIZACIÓN DE MONÓXIDO DE CARBONO.

4. ANEXOS

Autor:

D. Carlos Calleja Chinillach

Tutor:

D. Ángel Perles Ivars

Valencia, julio de 2020

ANEXO 1. Código

Contenido Anexo 1

1. NUCLEO-L053R8

1.1.stm32l0xx_hal_conf.h

```
/**
  ******************************************************************************
  * @file    stm32l0xx_hal_conf.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32l0xx_hal_conf.h.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2016 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32L0xx_HAL_CONF_H
#define __STM32L0xx_HAL_CONF_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/

/* ########################## Module Selection ############################## */
/**
  * @brief This is the list of modules to be used in the HAL driver
  */

#define HAL_MODULE_ENABLED
#define HAL_ADC_MODULE_ENABLED
/*#define HAL_CRYP_MODULE_ENABLED   */
/*#define HAL_COMP_MODULE_ENABLED   */
/*#define HAL_CRC_MODULE_ENABLED   */
/*#define HAL_CRYP_MODULE_ENABLED   */
/*#define HAL_DAC_MODULE_ENABLED   */
/*#define HAL_FIREWALL_MODULE_ENABLED   */
/*#define HAL_I2S_MODULE_ENABLED   */
/*#define HAL_IWDG_MODULE_ENABLED   */
/*#define HAL_LCD_MODULE_ENABLED   */
/*#define HAL_LPTIM_MODULE_ENABLED   */
/*#define HAL_RNG_MODULE_ENABLED   */
/*#define HAL_RTC_MODULE_ENABLED   */
/*#define HAL_SPI_MODULE_ENABLED   */
/*#define HAL_TIM_MODULE_ENABLED   */
/*#define HAL_TSC_MODULE_ENABLED   */
#define HAL_UART_MODULE_ENABLED
/*#define HAL_USART_MODULE_ENABLED   */
/*#define HAL_IRDA_MODULE_ENABLED   */
/*#define HAL_SMARTCARD_MODULE_ENABLED   */
/*#define HAL_SMBUS_MODULE_ENABLED   */
/*#define HAL_WWDG_MODULE_ENABLED   */
/*#define HAL_PCD_MODULE_ENABLED   */
/*#define HAL_EXTI_MODULE_ENABLED   */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
```

```
#define HAL I2C MODULE ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED

/* ########################## Oscillator Values adaptation ####################*/
/**
  * @brief Adjust the value of External High Speed oscillator (HSE) used in your
application.
  *        This value is used by the RCC HAL module to compute the system frequency
  *        (when HSE is used as system clock source, directly or through the PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)8000000U) /*!< Value of the External oscillator in Hz
*/
#endif /* HSE_VALUE */

#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    ((uint32_t)100U)   /*!< Time out for HSE start up, in
ms */
#endif /* HSE_STARTUP_TIMEOUT */

/**
  * @brief Internal Multiple Speed oscillator (MSI) default value.
  *        This value is the default MSI range value after Reset.
  */
#if !defined  (MSI_VALUE)
  #define MSI_VALUE    ((uint32_t)2097000U) /*!< Value of the Internal oscillator in
Hz*/
#endif /* MSI_VALUE */

/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the system frequency
  *        (when HSI is used as system clock source, directly or through the PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the Internal oscillator in
Hz*/
#endif /* HSI_VALUE */

/**
  * @brief Internal High Speed oscillator for USB (HSI48) value.
  */
#if !defined  (HSI48_VALUE)
#define HSI48_VALUE ((uint32_t)48000000U) /*!< Value of the Internal High Speed
oscillator for USB in Hz.
                                             The real value may vary depending on the
variations
                                             in voltage and temperature.  */
#endif /* HSI48_VALUE */

/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI VALUE)
 #define LSI_VALUE  ((uint32_t)37000U)       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                      /*!< Value of the Internal Low Speed
oscillator in Hz
                                             The real value may vary depending on the
variations
                                             in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  *        This value is used by the UART, RTC HAL module to compute the system
frequency
  */
#if !defined  (LSE_VALUE)
  #define LSE_VALUE    ((uint32_t)32768U) /*!< Value of the External oscillator in Hz*/
#endif /* LSE_VALUE */
```

```
#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT  ((uint32_t)5000U)   /*!< Time out for LSE start up, in ms
*/
#endif /* LSE_STARTUP_TIMEOUT */

/* Tip: To avoid modifying this file each time you need to use different HSE,
   ===  you can define the HSE value in your toolchain compiler preprocessor. */

/* ############################ System Configuration ######################### */
/**
  * @brief This is the HAL system configuration section
  */
#define  VDD_VALUE                    ((uint32_t)3300U) /*!< Value of VDD in mv */
#define  TICK_INT_PRIORITY            ((uint32_t)0U)    /*!< tick interrupt priority */
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              0U
#define  PREREAD_ENABLE               1U
#define  BUFFER_CACHE_DISABLE         0U

/* ########################## Assert Selection ############################## */
/**
  * @brief Uncomment the line below to expanse the "assert_param" macro in the
  *        HAL drivers code
  */
/* #define USE_FULL_ASSERT    1U */

/* Includes ------------------------------------------------------------------*/
/**
  * @brief Include module's header file
  */

#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32l0xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32l0xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32l0xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32l0xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */

#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32l0xx_hal_cortex.h"
#endif /* HAL CORTEX MODULE ENABLED */

#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32l0xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */

#ifdef HAL COMP MODULE ENABLED
  #include "stm32l0xx_hal_comp.h"
#endif /* HAL_COMP_MODULE_ENABLED */

#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32l0xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */

#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32l0xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */

#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32l0xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */
```

```
#ifdef HAL_FIREWALL_MODULE_ENABLED
  #include "stm32l0xx_hal_firewall.h"
#endif /* HAL_FIREWALL_MODULE_ENABLED */

#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32l0xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */

#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32l0xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */

#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32l0xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */

#ifdef HAL_IWDG_MODULE_ENABLED
 #include "stm32l0xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */

#ifdef HAL_LCD_MODULE_ENABLED
 #include "stm32l0xx_hal_lcd.h"
#endif /* HAL_LCD_MODULE_ENABLED */

#ifdef HAL_LPTIM_MODULE_ENABLED
#include "stm32l0xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */

#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32l0xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */

#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32l0xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */

#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32l0xx_hal_rtc.h"

#endif /* HAL_RTC_MODULE_ENABLED */

#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32l0xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */

#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32l0xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */

#ifdef HAL TSC MODULE ENABLED
 #include "stm32l0xx_hal_tsc.h"
#endif /* HAL_TSC_MODULE_ENABLED */

#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32l0xx_hal_uart.h"
#endif /* HAL UART MODULE ENABLED */

#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32l0xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */

#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32l0xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */

#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32l0xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */

#ifdef HAL_SMBUS_MODULE_ENABLED
```

```
 #include "stm32l0xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */

#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32l0xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */

#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32l0xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */

/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr: If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t *)__FILE__,
__LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */

#ifdef __cplusplus
}
#endif

#endif /* __STM32L0xx_HAL_CONF_H */


/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

1.2.stm32l0xx_it.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32l0xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2020 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
 ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32L0xx_IT_H
#define __STM32L0xx_IT_H

#ifdef __cplusplus
 extern "C" {
#endif
```

```
/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
void SVC_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
void EXTI4_15_IRQHandler(void);

#ifdef __cplusplus
}
#endif

#endif /* __STM32L0xx_IT_H */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

1.3.main.h

```
/**
  ******************************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
  ******************************************************************************
  */


/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32l0xx_hal.h"
#include "stm32l0xx_nucleo.h"
#include "stm32l0xx_hal_conf.h"
#include "x_nucleo_ika02a1.h"
#include <string.h>
#include <stdio.h>
#include <math.h>


/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);


#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

1.4.main.c

```
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
```

```
  */

#include "main.h"

/* Private variables ---------------------------------------------------------*/
UART_HandleTypeDef huart1;

float sens = 2417.0;
float gain = 470000.0;

float gas_value_ppm = 0;
float gas_value_ppm_comp = 0;
float temperature = 0;

char uartOutBuffer[256];


/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();

  /* Infinite loop */
  while (1)
  {
    __NOP();
  }
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the CPU, AHB and APB busses clocks
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_5;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB busses clocks
```

```
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_MSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1;
  PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief USART1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART1_UART_Init(void)
{
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin : PC4 */
  GPIO_InitStruct.Pin = GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /* EXTI interrupt init*/
  HAL_NVIC_SetPriority(EXTI4_15_IRQn, 1, 0);
  HAL_NVIC_EnableIRQ(EXTI4_15_IRQn);

}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_4){
```

```
        gas value ppm = Get Gas concentration(gain,sens/1000);
        temperature = Get_Temperature();
        gas_value_ppm_comp =
Get_comp_gas_concentration(gas_value_ppm,(int8_t)round(temperature));

        sprintf(uartOutBuffer, "%.2f\r\n",gas_value_ppm_comp);
        HAL_UART_Transmit(&huart1, (uint8_t*)uartOutBuffer, strlen(uartOutBuffer),
5000);
    }
}


#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

2. ESP32

```
/**
  ******************************************************************************
  * @file           : ESP32.in
  * @brief          : This program is used to comunicate a STM32 board with the
  *                                  plataform thinger.io
  ******************************************************************************
  */
#include <ThingerESP32.h>

#define DEBUG_ENABLE false

#define USERNAME "carcalch"
#define DEVICE_ID "ESP32"
#define DEVICE_CREDENTIAL "36TJ4onv161e"

#define SSID "tfgtest"
#define SSID_PASSWORD "passtfgtest"

ThingerESP32 thing(USERNAME, DEVICE_ID, DEVICE_CREDENTIAL);

String uartInBuffer = "";
float gas_value_ppm = 0;
unsigned long previousMillis  = 0;


/**
  * @brief  Comunicates with the Nucleo-L053R8 and gets the sensor value in ppm using
USART
  * @return float value of the gas concentration in ppm
  */
float get_data(){
  uartInBuffer = "";
  digitalWrite(5, HIGH);
  delay(100);
```

```
  digitalWrite(5, LOW);
  previousMillis = millis();
  while(!Serial2.available() && previousMillis + 1000 < millis());
  if(!Serial2.available()){
    if(DEBUG_ENABLE) Serial.println("STM32 TIMEOUT");
    return 0;
  }
  while(Serial2.available()){
    uartInBuffer += ((char)Serial2.read());
  }
  gas_value_ppm = uartInBuffer.toFloat();
  if(DEBUG_ENABLE) Serial.println(gas_value_ppm);
  return gas_value_ppm;
}


/**
  * @brief  This function is executed whenever the program starts
  */
void setup() {
  thing.add_wifi(SSID, SSID_PASSWORD);
  thing["ppm"] >> outputValue(get_data());
  Serial2.begin(9600, SERIAL_8N1, 16, 17);
  pinMode(5, OUTPUT);
  while(!Serial2);
  if(DEBUG_ENABLE){
    Serial.begin(115200);
    while(!Serial);
    Serial.println("SYSTEM INIT");
  }
}

/**
  * @brief  Infinite loop
  */
void loop() {
  thing.handle();
}
```

ANEXO 2. Hojas de características

Contenido Anexo 2. Hojas de características

MQ-9 Semiconductor Sensor for CO/Combustible Gas

Sensitive material of MQ-9 gas sensor is SnO_2, which with lower conductivity in clean air. It make detection by method of cycle high and low temperature, and detect CO when low temperature (heated by 1.5V). The sensor's conductivity is more higher along with the gas concentration rising. When high temperature (heated by 5.0V), it detects Methane, Propane etc combustible gas and cleans the other gases adsorbed under low temperature. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ-9 gas sensor has high sensitity to Carbon Monoxide, Methane and LPG. The sensor could be used to detect different gases contains CO and combustible gases, it is with low cost and suitable for different application.

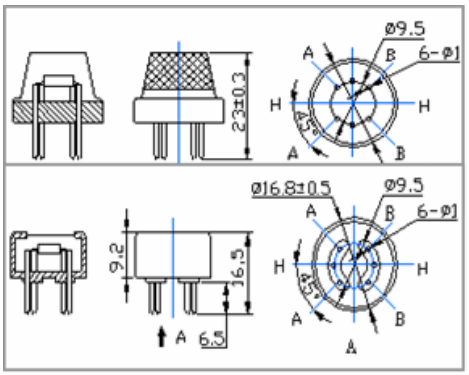
Character

* Good sensitivity to CO/Combustible gas
* High sensitivity to Methane, Propane and CO
* Long life and low cost
* Simple drive circuit

Application

* Domestic gas leakage detector
* Industrial gas detector
* Portable gas detector

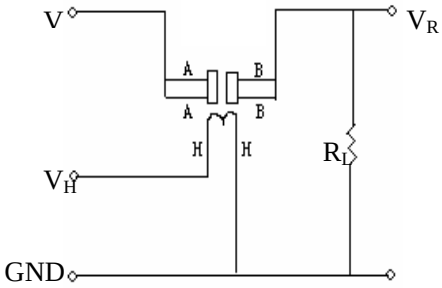
Configuration



Technical Data

Model No.			MQ-9
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite
Detection Gas			CO and combustible gas
Concentration			10-1000ppm CO 100-10000ppm combustible gas
Circuit	Loop Voltage	V_C	≤10V DC
	Heater Voltage	V_H	5.0V±0.2V ACorDC（High） 1.5V±0.1V ACorDC（Low）
	Heater Time	T_L	60±1S（High）90±1S（Low）
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤350mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 100ppm CO)
	Sensitivity	S	Rs(in air)/Rs(100ppm CO)≥5
	Slope	α	≤0.6(R_{300ppm}/R_{100ppm} CO)
Condition	Tem. Humidity		20℃±2℃; 65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V; V_H（High）: 5.0V±0.1V; V_H（Low）: 1.5V±0.1V
	Preheat time		Over 48 hours

Basic test loop



The above is basic test circuit of the sensor. The sensor need to be put 2 voltage, heater voltage（VH） and test voltage（VC）. VH used to supply certified working temperature to the sensor, while VC used to detect voltage (VRL) on load resistance（RL）whom is in series with sensor. The sensor has light polarity, Vc need DC power. VC and VH could use same power circuit with precondition to assure performance of sensor. In order to make the sensor with better performance, suitable RL value is needed:

Power of Sensitivity body(Ps): $Ps=Vc^2\times Rs/(Rs+RL)^2$

Tel: 86-371-67169070/80 Fax: 86-371-67169090 Email: sales@hwsensor.com

Resistance of sensor(Rs): Rs=(Vc/VRL-1)×RL

Sensitivity Characteristics

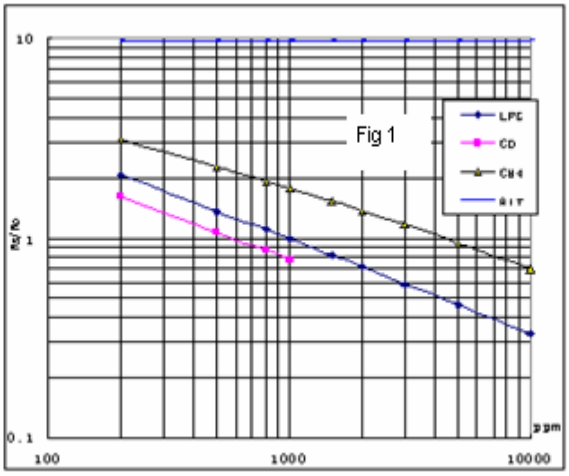


Fig.1 shows the typical sensitivity characteristics of the MQ-9, ordinate means resistance ratio of the sensor (Rs/Ro), abscissa is concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 1000ppm LPG. All test are under standard test conditions.

Influence of Temperature/Humidity

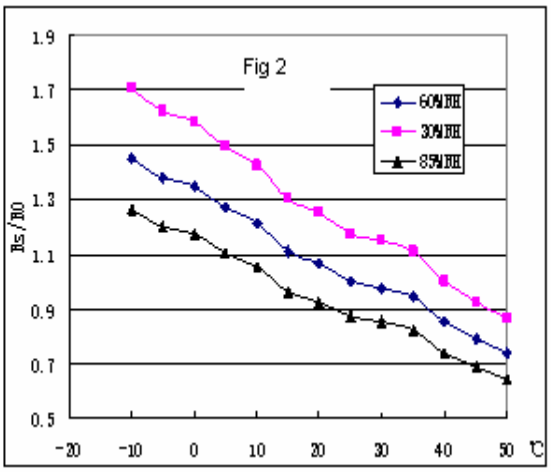
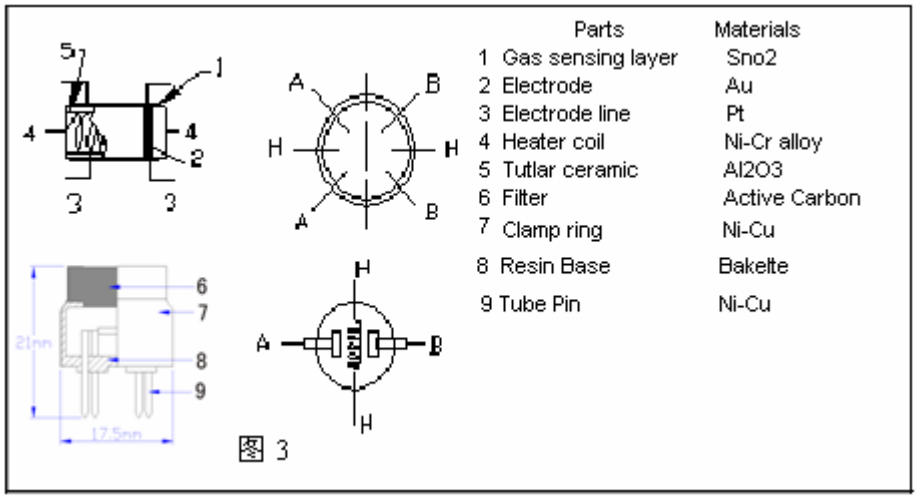


Fig.2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor (Rs/Ro), Rs means resistance of sensor in 1000ppm Propane under different tem. and humidity. Ro means resistance of the sensor in environment of 1000ppm Propane, 20℃/65%RH

Structure and configuration



Structure and configuration of MQ-9 gas sensor is shown as Fig. 3, sensor composed by micro AL2O3 ceramic tube, Tin Dioxide (SnO2) sensitive layer, measuring electrode and heater are fixed into a crust made by plastic and stainless steel net. The heater provides necessary work conditions for work of sensitive components. The enveloped MQ-7 have 6 pin, 4 of them are used to fetch signals, and other 2 are used for providing heating current.

Notification

1 Following conditions must be prohibited

1.1 Exposed to organic silicon steam

Organic silicon steam cause sensors invalid, sensors must be avoid exposing to silicon bond, fixature, silicon latex, putty or plastic contain silicon environment

1.2 High Corrosive gas

If the sensors exposed to high concentration corrosive gas (such as H_2Sz, SO_X, Cl_2, HCl etc), it will not only result in corrosion of sensors structure, also it cause sincere sensitivity attenuation.

1.3 Alkali, Alkali metals salt, halogen pollution

The sensors performance will be changed badly if sensors be sprayed polluted by alkali metals salt especially brine, or be exposed to halogen such as fluorin.

1.4 Touch water

Sensitivity of the sensors will be reduced when spattered or dipped in water.

1.5 Freezing

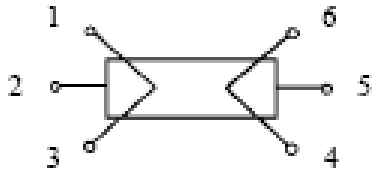
Do avoid icing on sensor'surface, otherwise sensor would lose sensitivity.

1.6 Applied voltage higher

Applied voltage on sensor should not be higher than stipulated value, otherwise it cause down-line or heater damaged, and bring on sensors' sensitivity characteristic changed badly.

1.7 Voltage on wrong pins

For 6 pins sensor, if apply voltage on 1、3 pins or 4、6 pins, it will make lead broken, and without signal when apply on 2、4 pins



2 Following conditions must be avoided

2.1 Water Condensation

Indoor conditions, slight water condensation will effect sensors performance lightly. However, if water condensation on sensors surface and keep a certain period, sensor' sensitivity will be decreased.

2.2 Used in high gas concentration

No matter the sensor is electrified or not, if long time placed in high gas concentration, if will affect sensors characteristic.

2.3 Long time storage

The sensors resistance produce reversible drift if it's stored for long time without electrify, this drift is related with storage conditions. Sensors should be stored in airproof without silicon gel bag with clean air. For the sensors with long time storage but no electrify, they need long aging time for stbility before using.

2.4 Long time exposed to adverse environment

No matter the sensors electrified or not, if exposed to adverse environment for long time, such as high humidity, high temperature, or high pollution etc, it will effect the sensors performance badly.

2.5 Vibration

Continual vibration will result in sensors down-lead response then repture. In transportation or assembling line, pneumatic screwdriver/ultrasonic welding machine can lead this vibration.

2.6 Concussion

If sensors meet strong concussion, it may lead its lead wire disconnected.

2.7 Usage

For sensor, handmade welding is optimal way. If use wave crest welding should meet the following conditions:

2.7.1 Soldering flux: Rosin soldering flux contains least chlorine
2.7.2 Speed: 1-2 Meter/ Minute
2.7.3 Warm-up temperature：100±20℃
2.7.4 Welding temperature：250±10℃
2.7.5 1 time pass wave crest welding machine

If disobey the above using terms, sensors sensitivity will be reduced.

FIGARO

PRODUCT INFORMATION

TGS 5141-P00 - for the detection of Carbon Monoxide

Features:

* Ultra compact
* Battery operable
* High repeatability/selectivity to CO
* Linear relationship between CO gas concentration and sensor output
* Simple calibration
* Long life
* UL recognized component
* Meets UL2034, EN50291, and EN54-31 requirements

Applications:

* Residential and commercial CO detectors
* Fire detection

Figaro's **TGS5141** is a battery operable electrochemical sensor which uses a unique electrolyte that eliminates the need for a water reservoir. By eliminating the water reservoir used in TGS5042, the comparative size of TGS5141 is reduced to just 10% of TGS5042. With its ultra compact size, this sensor is the ideal choice for size oriented applications such as portable CO detectors, small residential CO detectors, and multi-sensor fire detectors. OEM customers will find individual sensors data printed on each sensor in bar code form, enabling users to skip the costly gas calibration process and allowing for individual sensor tracking.

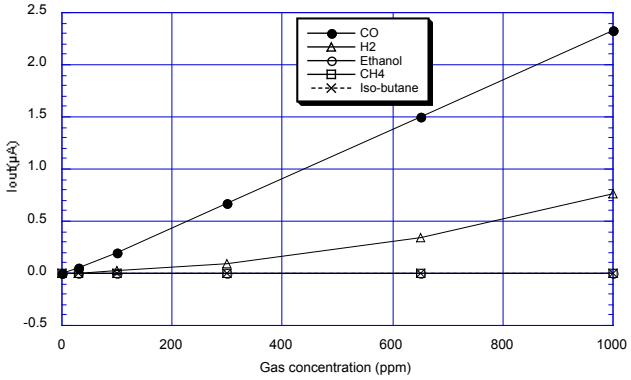
The figure below represents typical sensitivity characteristics, all data having been gathered at standard test conditions (see reverse side of this sheet). The Y-axis shows theoutput current of the sensor (Iout/μA) in each gas. Output current is linear to CO concentration, with a deviation of less than ±5% in the range of 0~500ppm.

The figure below represents typical temperature dependency characteristics. The Y-axis shows the sensor output ratio (I/Io) as defined below. The linear relationship between I/Io and CO concentration is constant regardless of the CO concentration range.

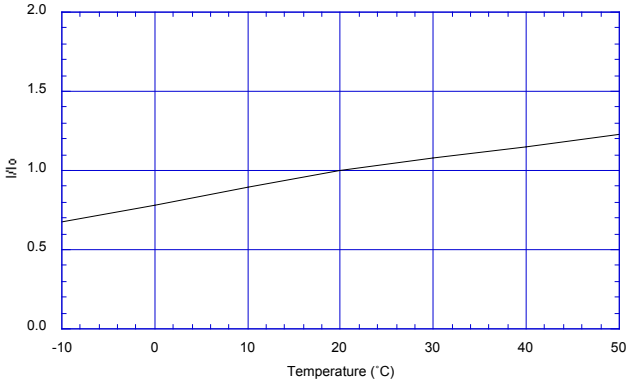
I = Sensor output current in 400ppm of CO at various temperatures

Io = Sensor output current in 400ppm at 20˚C/50%RH

Sensitivity Characteristics:



Temperature Dependency:



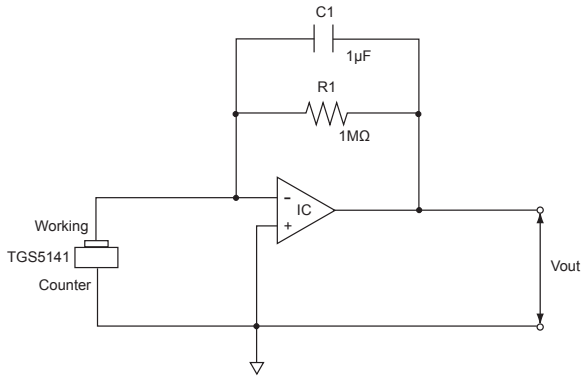
Basic Measuring Circuit:

The diagram at the right shows the basic measuring circuit of TGS5141. The sensor generates a minute electric current which is converted into sensor output voltage (Vout) by an op-amp/resistor (R1) combination.

Figaro recommends the following electrical parts:

R1 : 1MΩ
C1 : 1μF
IC : AD708

NOTE: When voltage is applied to the sensor output terminal, the sensor may be damaged. Voltage applied to the sensor should be strictly limited to less than ±10mV. An additional resistor or FET is required to prevent polarization of the sensor when Vc is off.



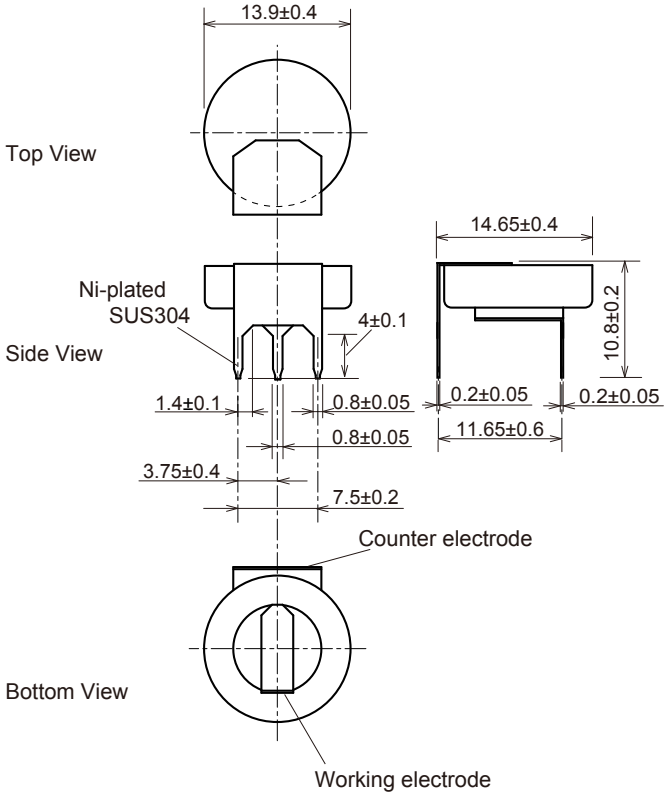
Basic measuring circuit of TGS5141

Specifications:

Item	Specification
Model number	TGS5141-P00
Target gases	Carbon monoxide
Typical detection range	0 ~ 5,000ppm
Output current in CO	1.2~3.2nA/ppm
Baseline offset(*1)	<±10ppm equivalent
Operating temperature	-10˚C ~ +50˚C (continuous) -20˚C ~ +60˚C (intermittent)
Operating humidity	10 ~ 95%RH (no condensation)
Response time (T90)	within 60 seconds
Storage conditions	-10˚C ~ +50˚C (continuous) -20˚C ~ +60˚C (intermittent)
Weight	approx. 2.5g
Standard test conditions	20±2˚C, 40±10%RH

(*1) *Represents sensor output in air under operating conditions*

Structure and Dimensions:



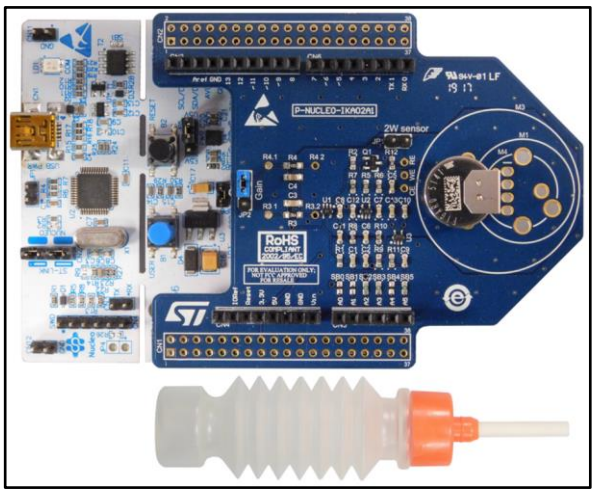
Unit : mm

FIGARO ENGINEERING INC.
1-5-11 Senba-nishi
Mino, Osaka 562-8505 JAPAN
Tel: 81-72-728-2567
Fax: 81-72-728-0467
email: figaro@figaro.co.jp
www.figaro.co.jp

P-NUCLEO-IKA02A1

STM32 Nucleo pack: electrochemical toxic gas sensor expansion board with CO sensor

Data brief



Features

- STM32 Nucleo gas expansion board
  - compatible with most electrochemical sensors
  - four different footprints for sensors (PCD13,5, PCD17, Mini and TGS5141)
  - two-, three- and four-electrode sensors
  - signal conditioning with TSU111
  - STLM20 temperature sensor
  - changeable gain
- NUCLEO-L053R8
  - Ultra-low-power ARM® Cortex®-M0+ MCU (32 MHz max.) with 64 Kbytes Flash and 8 Kbytes of SRAM
- Carbon monoxide sensor
  - Figaro TGS5141
  - coin-cell sensor
  - expected life time > 10 yrs
  - can pass 5000 ppm EN50291
- Low power design for long battery life
- RoHS compliant

Description

The P-NUCLEO-IKA02A1 evaluation pack provides a reference design for various electrochemical sensors.

The STM32 Nucleo gas expansion board interfaces electrochemical sensors with the MCU on the STM32 Nucleo development board. Two TSU111 operational amplifiers provide signal conditioning; they are ideal for electrochemical sensing thanks to their high precision and low power consumption. The expansion board includes an ultra-low current precision analog temperature sensor STLM20 used for compensation of gas readings.

STM32 Nucleo boards provide an affordable and flexible way for users to experiment with new ideas and build prototypes with any STM32 microcontroller line. The NUCLEO-L053R8 is designed for low power applications.

The design and componentry are optimised for battery operation and maximum battery life time.

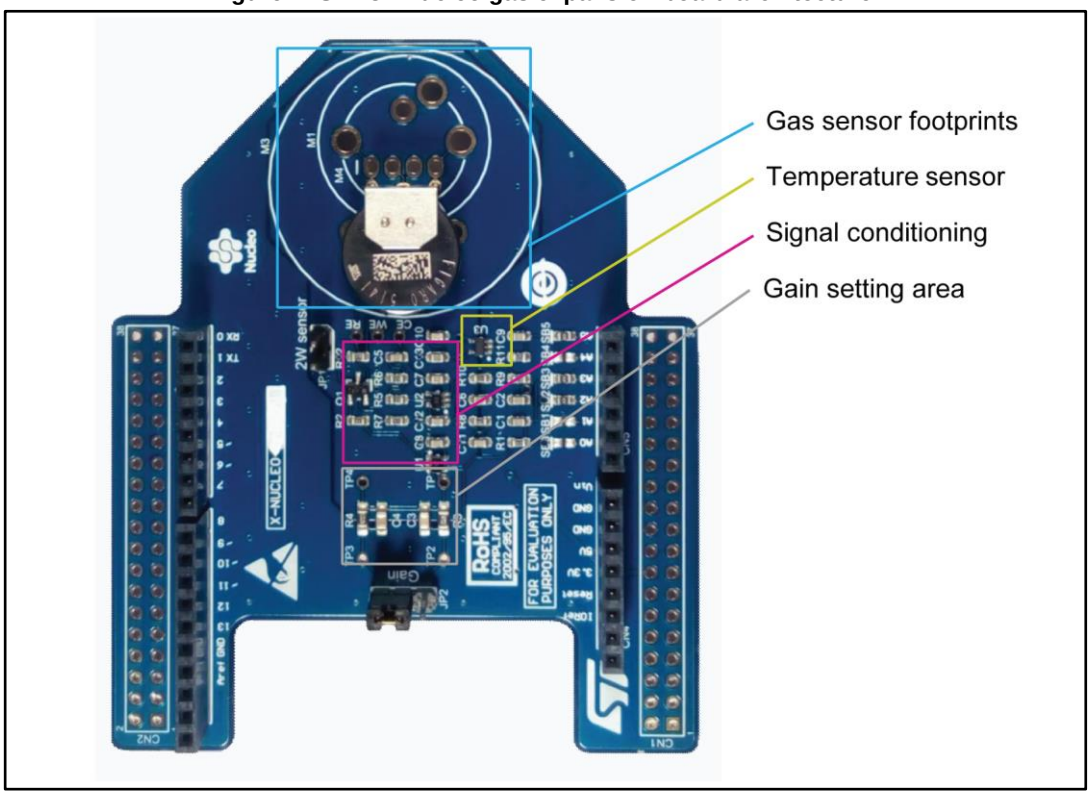
1 Board description

STM32 Nucleo gas expansion board consist of three main blocks

- gas sensor footprints – attached carbon monoxide sensor TGS5141
- operational amplifiers TSU111 for signal conditioning
- precision analog temperature sensor STLM20 for temperature compensation

Information regarding the NUCLEO-L053R8 and NUCLEO-F401RE development boards can be found at http://www.st.com/stm32nucleo

Figure 1: STM32 Nucleo gas expansion board architecture



2 STM32 Nucleo gas expansion board

The STM32 Nucleo gas expansion board is compatible with most X-NUCLEO expansion boards. See the following figure and table for configuration and compatibility information, respectively.

Figure 2: Configuration of solder bridges

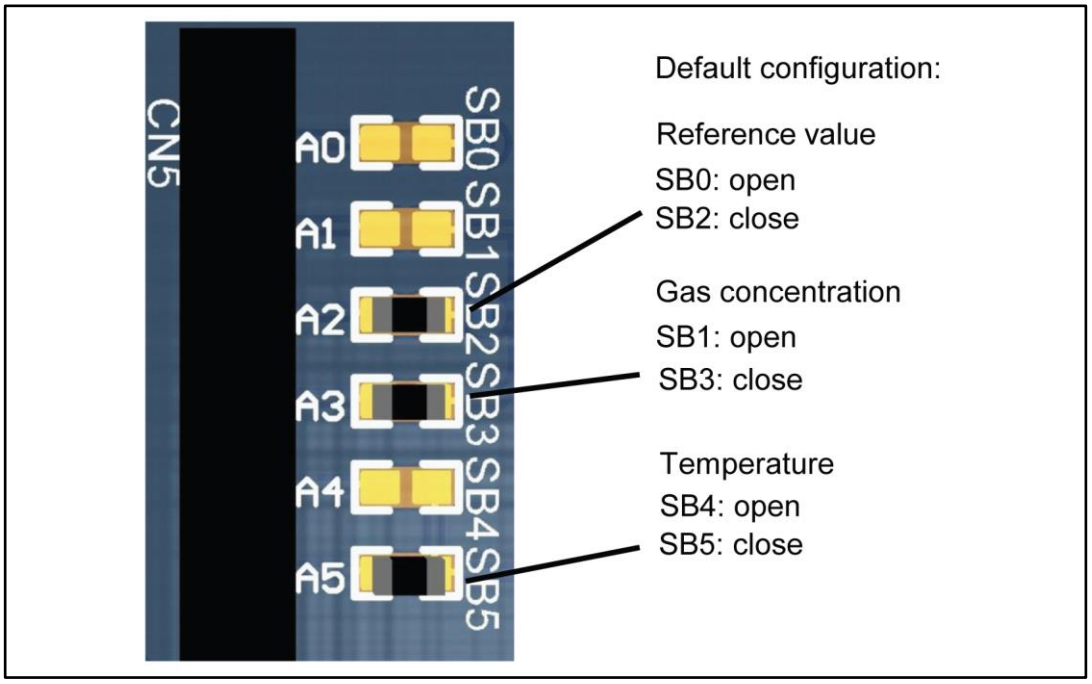


Table 1: Compatibility table

Extension board	Reference	Gas reading	Temperature
X-NUCLEO-IDB0xA1	Default	Default	Default
X-NUCLEO-IDW01M1	Default [a]	Default	Default
X-NUCLEO-IDS01Ax	Default [b]	Default [b]	Default [b]
X-NUCLEO-IKS01Ax	Default [c]	Default [c]	Default [c]
X-NUCLEO-IKA01A1	Alternative	Default	Default

[a] Alternative connection of GPIO13 of Wi-Fi module cannot be used

[b] Optional SPI connection and GPIO pins of SPSGRF module cannot be used

[c] Limited usage of INT on DIL24 and DRDY – see schematic pack and used alternative configuration. It is possible to use humidity sensor on board to provide temperature and humidity compensation

3 Ordering information

Table 2: Order code

Order code	Description
P-NUCLEO-IKA02A1	STM32 Nucleo pack: Electrochemical toxic gas sensor expansion board

4 Revision history

Table 3: Document revision history

Date	Version	Changes
19-Jun-2017	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

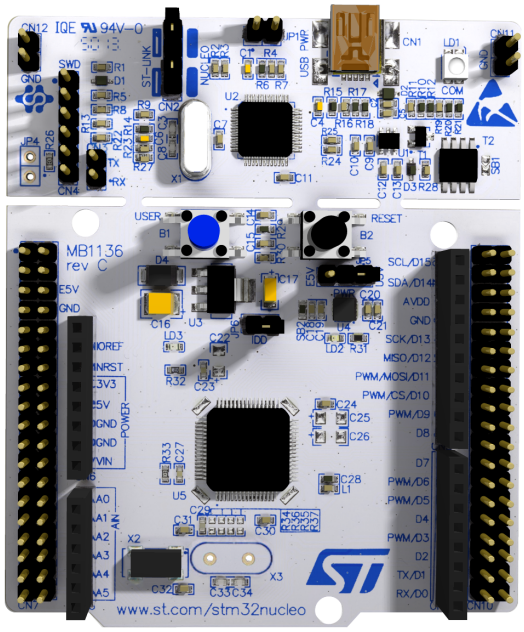
UM1724
User manual

STM32 Nucleo-64 boards (MB1136)

Introduction

The STM32 Nucleo-64 boards based on the MB1136 reference board (NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F103RB, NUCLEO-F302R8, NUCLEO-F303RE, NUCLEO-F334R8, NUCLEO-F401RE, NUCLEO-F410RB, NUCLEO-F411RE, NUCLEO-F446RE, NUCLEO-L010RB, NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE, NUCLEO-L452RE, NUCLEO-L476RG) provide an affordable and flexible way for users to try out new concepts and build prototypes with the STM32 microcontrollers in LQFP64 package, choosing from the various combinations of performance, power consumption and features. The Arduino™ Uno V3 connectivity support and the ST morpho headers provide an easy means of expanding the functionality of the Nucleo open development platform with a wide choice of specialized shields. The STM32 Nucleo boards do not require any separate probe as they integrate the ST-LINK/V2-1 debugger and programmer. The STM32 Nucleo boards come with the comprehensive free software libraries and examples available with the STM32Cube MCU Packages, as well as direct access to the Arm® Mbed™ online resources at http://mbed.org/.

Figure 1. STM32 Nucleo-64 board



Picture is not contractual.

Content

List of Tables

List of Figures

1 Features

The STM32 Nucleo board offers the following features:

- STM32 microcontroller in LQFP64 package
- Three LEDs:
  - USB communication (LD1), user LED (LD2), power LED (LD3)
- Two push-buttons: USER and RESET
- Two types of extension resources
  - Arduino™ Uno V3 connectivity
  - ST morpho extension pin headers for full access to all STM32 I/Os
- Flexible board power supply:
  - USB VBUS or external source (3.3 V, 5 V, 7-12 V)
  - Power management access point
- On-board ST-LINK/V2-1 debugger and programmer with SWD connector
  - Selection-mode switch to use the kit as a standalone ST-LINK/V2-1
- USB re-enumeration capability. Three different interfaces supported on USB:
  - Virtual COM port
  - Mass storage
  - Debug port
- Comprehensive free software libraries and examples available with the STM32Cube MCU Package
- Arm® Mbed™(a) (see http://mbed.org)

a. Arm and Mbed are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

2 Ordering information

To order an STM32 Nucleo-64 board, refer to *Table 1*. Additional information is available from the datasheet and reference manual of the target STM32.

Table 1. Ordering information

Order code	Board reference	Targeted STM32	Differentiating feature
NUCLEO-F030R8	MB1136	STM32F030R8T6	Arm® Mbed Enabled™
NUCLEO-F070RB		STM32F070RBT6	
NUCLEO-F072RB		STM32F072RBT6	
NUCLEO-F091RC		STM32F091RCT6U	
NUCLEO-F103RB		STM32F103RBT6	
NUCLEO-F302R8		STM32F302R8T6	
NUCLEO-F303RE		STM32F303RET6	
NUCLEO-F334R8		STM32F334R8T6	
NUCLEO-F401RE		STM32F401RET6U	
NUCLEO-F410RB		STM32F410RBT6U	
NUCLEO-F411RE		STM32F411RET6U	
NUCLEO-F446RE		STM32F446RET6U	
NUCLEO-L010RB		STM32L010RBT6	-
NUCLEO-L053R8		STM32L053R8T6	Arm® Mbed Enabled™
NUCLEO-L073RZ		STM32L073RZT6U	
NUCLEO-L152RE		STM32L152RET6	
NUCLEO-L452RE		STM32L452RET6U	-
NUCLEO-L476RG		STM32L476RGT6U	Arm® Mbed Enabled™

2.1 Product marking

Evaluation tools marked as "ES" or "E" are not yet qualified and therefore they are not ready to be used as reference design or in production. Any consequences deriving from such usage will not be at ST charge. In no event, ST will be liable for any customer usage of these engineering sample tools as reference design or in production.

"E" or "ES" marking examples of location:

- On the targeted STM32 that is soldered on the board (for illustration of STM32 marking, refer to the section “Package information” of the STM32 datasheet available at *www.st.com*).
- Next to the evaluation tool ordering part number, that is stuck or silk-screen printed on the board.

Some boards feature a specific STM32 device version, which allows the operation of any bundled commercial stack/library available. This STM32 device shows a "U" marking option at the end of the standard part number and is not available for sales.

In order to use the same commercial stack in his application, a developer may need to purchase a part number specific to this stack/library. The price of those part numbers includes the stack/library royalties.

2.2 Codification

The meaning of the codification is explained in *Table 2*.

Table 2. Codification explanation

NUCLEO-XXYYRT	Description	Example: NUCLEO-L452RE
XX	MCU series in STM32 Arm Cortex MCUs	STM32L4 Series
YY	STM32 product line in the series	STM32L452
R	STM32 package pin count	64 pins
T	STM32 Flash memory size: – 8 for 64 Kbytes – B for 128 Kbytes – C for 256 Kbytes – E for 512 Kbytes – G for 1 Mbyte – Z for 192 Kbytes	512 Kbytes

The order code is printed on a sticker placed at the top or bottom side of the board.

3 Development environment

3.1 System requirements

- Windows® OS (7, 8 and 10), Linux® 64-bit, or macOS®
- USB Type-A to Mini-B cable

3.2 Development toolchains

- Arm® Keil®: MDK-ARM(a)
- IAR™: EWARM(a)
- GCC-based IDEs
- Arm® Mbed™ online(b) (see http://mbed.org)

3.3 Demonstration software

The demonstration software, included in the STM32Cube MCU Package corresponding to the on-board microcontroller, is preloaded in the STM32 Flash memory for easy demonstration of the device peripherals in standalone mode. The latest versions of the demonstration source code and associated documentation can be downloaded from *www.st.com*.

4 Conventions

Table 3 provides the conventions used for the ON and OFF settings in the present document.

Table 3. ON/OFF conventions

Convention	Definition
Jumper JP1 ON	Jumper fitted
Jumper JP1 OFF	Jumper not fitted
Solder bridge SBx ON	SBx connections closed by solder or 0 ohm resistor
Solder bridge SBx OFF	SBx connections left open

In this document the references are “STM32 Nucleo board” and “STM32 Nucleo boards” for all information that is common to all sale types.

a. On Windows® only.

b. Refer to the http://mbed.com website and to *Table 1: Ordering information*, to determine which order codes are supported.

5 Quick start

The STM32 Nucleo board is a low-cost and easy-to-use development platform used to quickly evaluate and start a development with an STM32 microcontroller in LQFP64 package.

Before installing and using the product, accept the Evaluation Product License Agreement from the www.st.com/epla webpage.

For more information on the STM32 Nucleo board and to access the demonstration software, visit www.st.com/stm32nucleo website.

5.1 Getting started

Follow the sequence below to configure the STM32 Nucleo board and launch the demo software:

1. Check the jumper position on the board, JP1 off, JP5 (PWR) on U5V, JP6 on (IDD), CN2 on (NUCLEO) selected.
2. For correct identification of all device interfaces from the host PC, install the Nucleo USB driver available from the www.st.com/stm32nucleo webpage, prior to connecting the board.
3. Connect the STM32 Nucleo board to a PC with a USB cable ‘Type-A to Mini-B’ through USB connector CN1 to power the board. The red LED LD3 (PWR) and LD1 (COM) should light up. LD1 (COM) and green LED LD2 should blink.
4. Press button B1 (left button).
5. Observe the blinking frequency of LED LD2 at different frequencies, by clicking on the button B1.
6. The demonstration software and several software examples on how to use the STM32 Nucleo board features are available at the www.st.com/stm32nucleo webpage.
7. Develop the application using the available examples.

5.2 NUCLEO-L476RG bootloader limitations

Boot from system Flash memory results in executing **bootloader** code stored in the system Flash memory, protected against write and erase. This allows in-system programming (ISP), that is, flashing the STM32 user Flash memory. It also allows writing data into RAM. The data come in via one of the communication interfaces such as USART, SPI, I^2C bus, USB or CAN.

Bootloader version can be identified by reading Bootloader ID at the address 0x1FFF6FFE.

The STM32L476RGT6 part soldered on the NUCLEO-L476RG main board is marked with a date code, corresponding to its date of manufacturing. STM32L476RGT6 parts with the date code prior or equal to week 22 of 2015 are fitted with **bootloader V 9.0,** affected by the limitations to be worked around, as described hereunder. Parts with the date code starting from week 23 of 2015 contain bootloader V 9.2 in which the limitations no longer exist.

To locate the visual date code information on the STM32L476RGT6 package, refer to the section “Package information” of the datasheet (DS10198) available at *www.st.com.* Date code related portion of the package marking, takes Y WW format, where Y is the last digit of

the year and WW is the week. For example, a part manufactured in week 23 of 2015 bares the date code 5 23.

Bootloader ID of the bootloader V 9.0 is 0x90.

The following limitations exist in the bootloader V 9.0:

1. RAM data get corrupted when written via USART/SPI/I^2C/USB interface

Description:

Data write operation into RAM space via USART, SPI, I^2C bus or USB results in wrong or no data written.

Workaround:

To correct the issue of wrong write into RAM, download STSW-STM32158 bootloader V 9.0 patch package from the *www.st.com* website and load "Bootloader V9.0 SRAM patch" to the STM32, following the information in readme.txt file available in the package.

2. User Flash memory data get corrupted when written via CAN interface

Description:

Data write operation into user Flash memory space via CAN interface results in wrong or no data written.

Workaround:

To correct the issue of wrong write into Flash memory, download STSW-STM32158 bootloader V 0.9 patch package from the *www.st.com* website and load "Bootloader V9.0 CAN patch" to the STM32, following the information in readme.txt file available in the package.

5.3 Hardware configuration variants

The board can be delivered with different configurations of the oscillator of the target STM32. For all the details concerning high-speed configurations of the oscillator refer to *Section 6.7.1*. For all the details concerning low-speed configurations of the oscillator refer to *Section 6.7.2*.

6 Hardware layout and configuration

The STM32 Nucleo board is designed around the STM32 microcontrollers in a 64-pin LQFP package.

Figure 2 shows the connections between the STM32 and its peripherals (ST-LINK/V2-1, push-button, LED, Arduino connectors and ST morpho connector).

Figure 3 and *Figure 4* show the location of these features on the STM32 Nucleo board. *Figure 5* shows the mechanical dimension of the STM32 Nucleo board.

Figure 2. Hardware block diagram

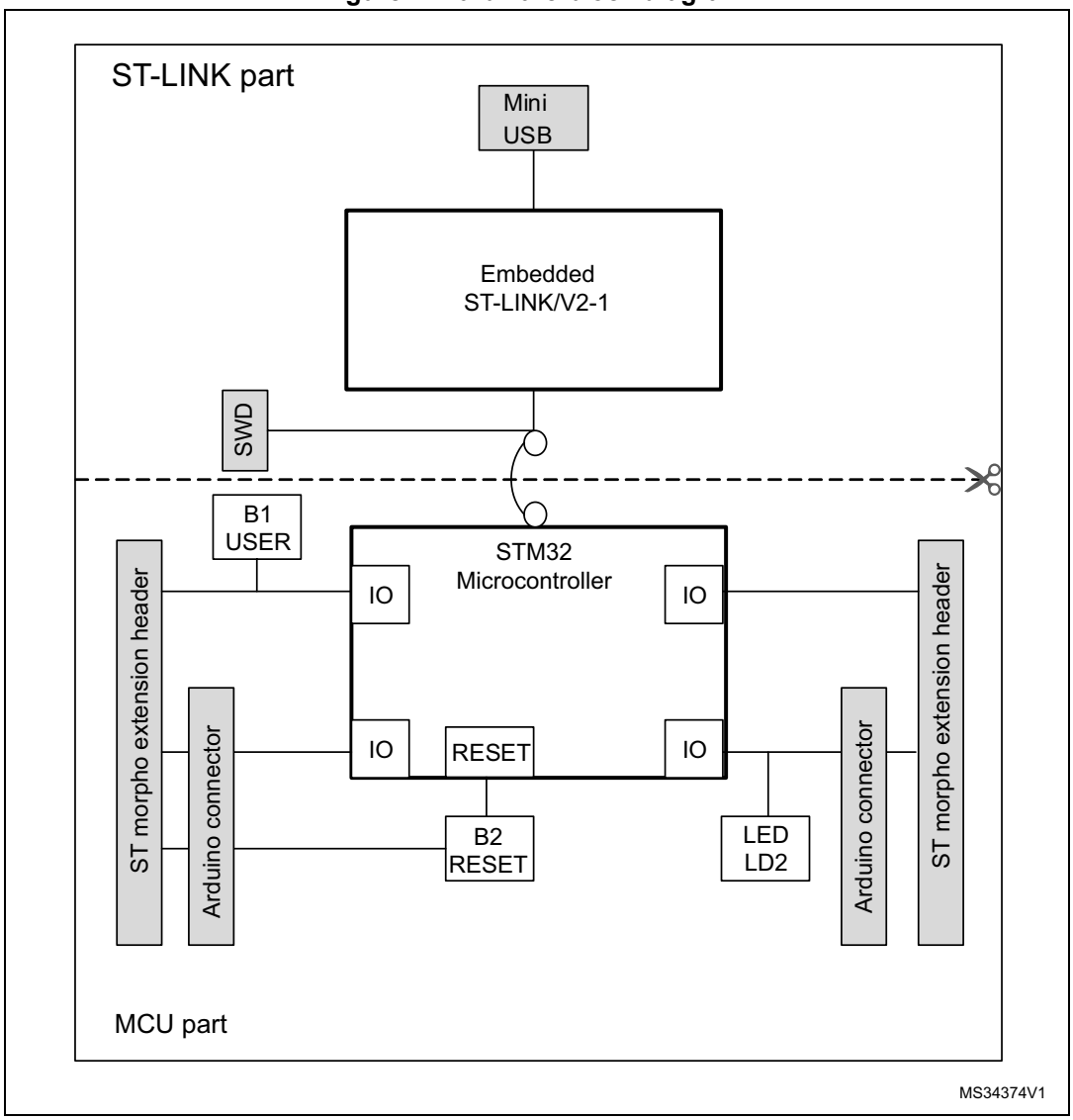
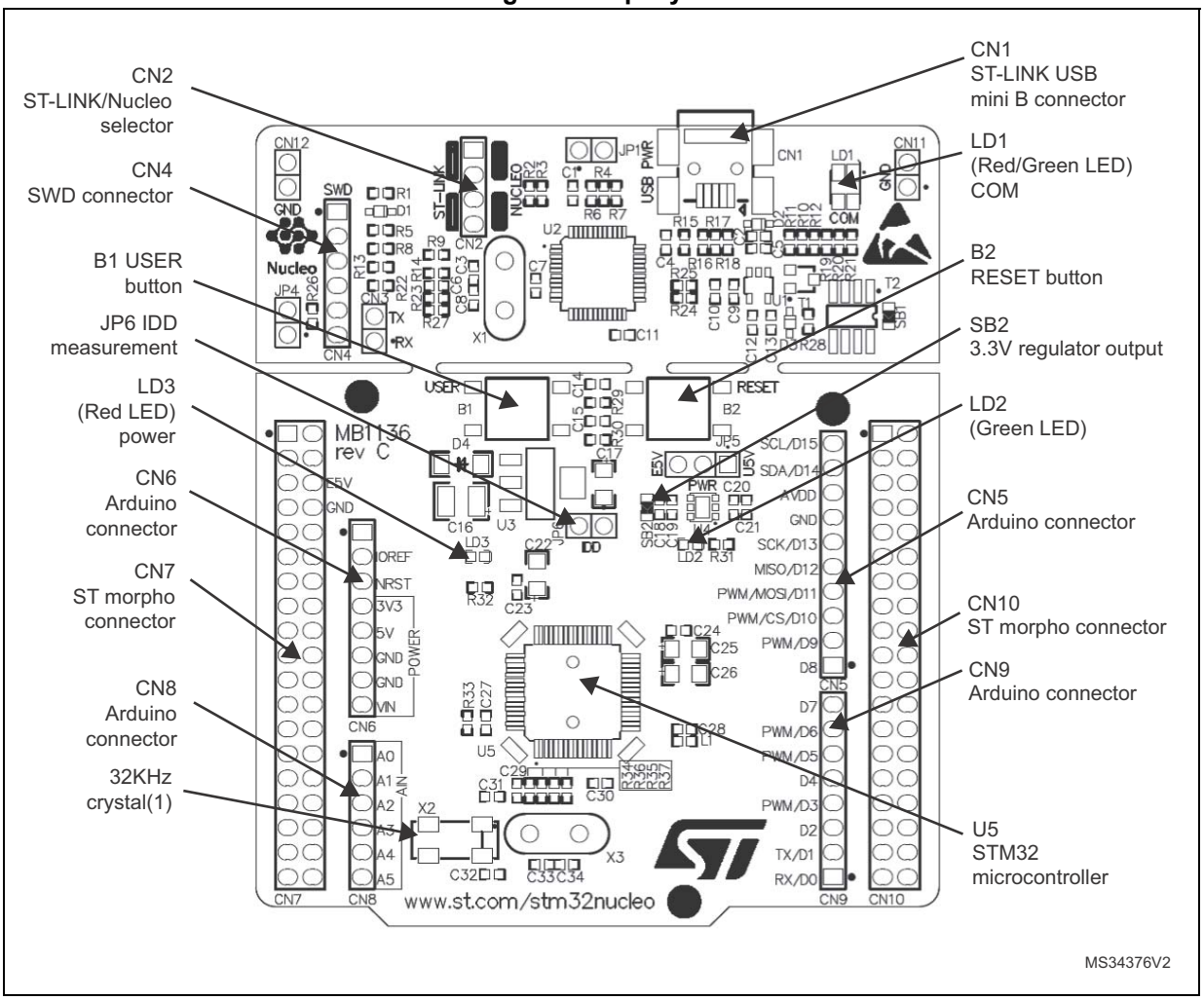


Figure 3. Top layout



1. Crystal may be present or not depending on board version, refer to *Section 6.7.2*.

Figure 4. Bottom layout

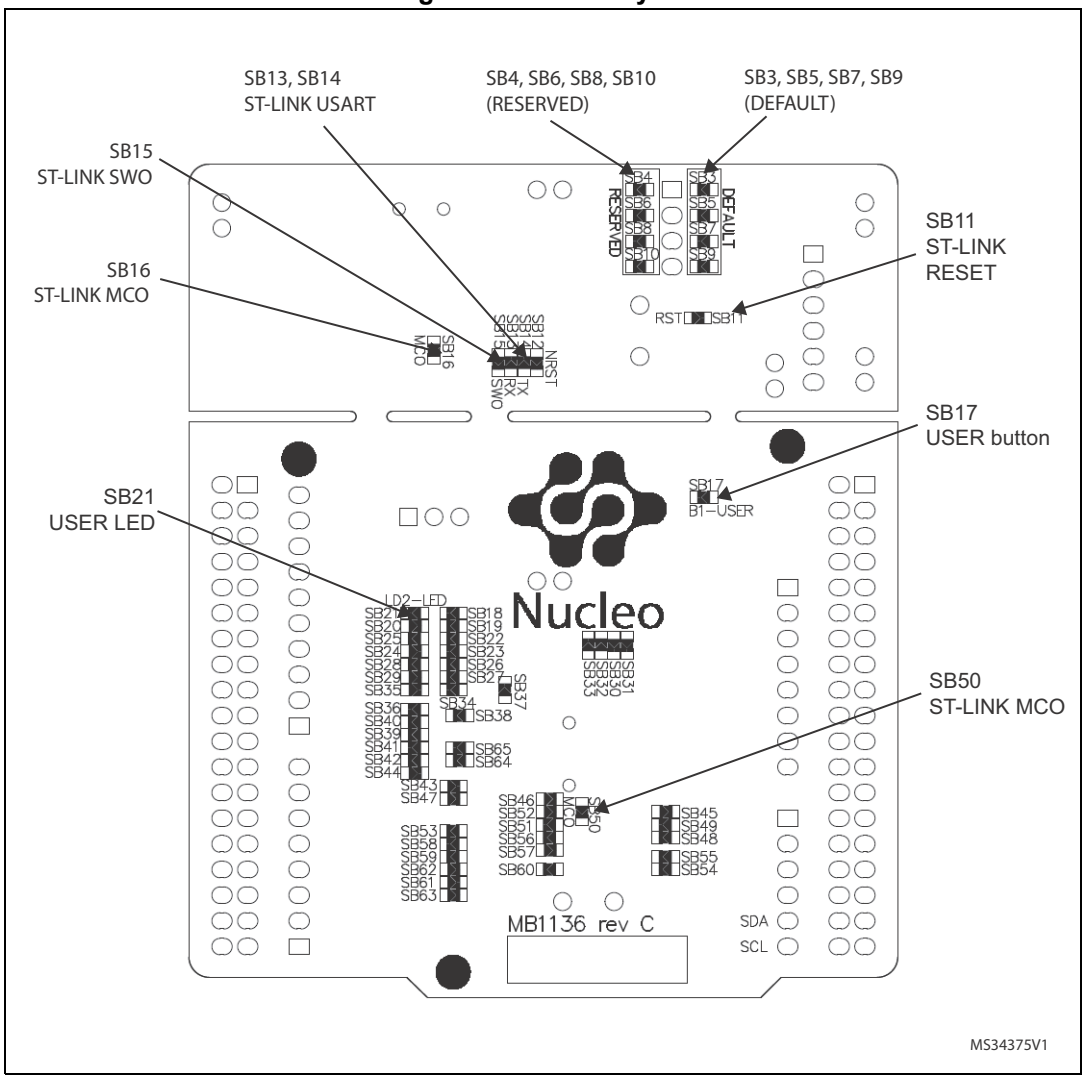
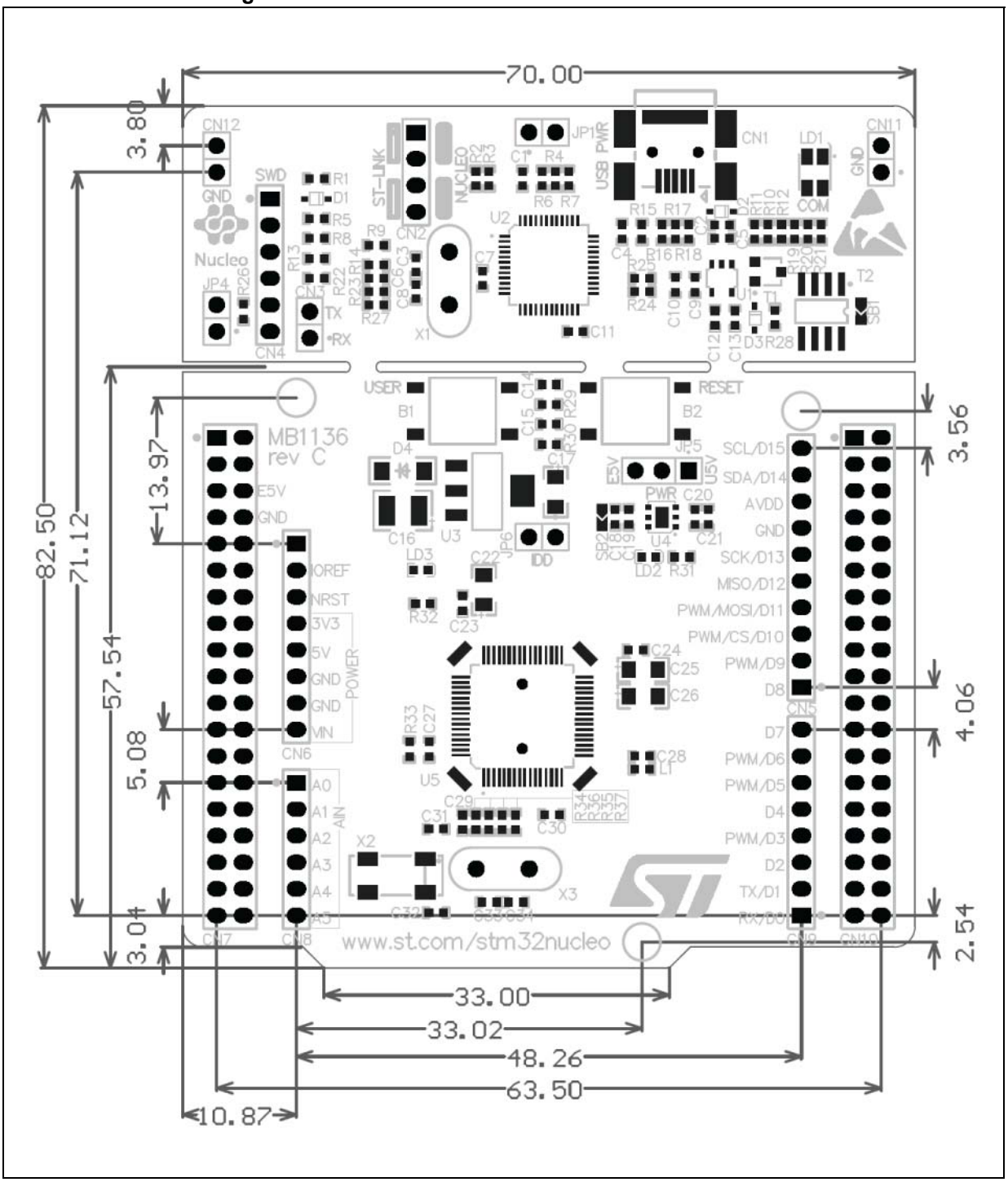


Figure 5. STM32 Nucleo board mechanical dimensions



6.1 Cuttable PCB

The STM32 Nucleo board is divided into two parts: ST-LINK part and target STM32 part. The ST-LINK part of the PCB can be cut out to reduce the board size. In this case the remaining target STM32 part can only be powered by VIN, E5V and 3.3V on ST morpho connector CN7 or VIN and 3.3V on Arduino connector CN6. It is still possible to use the ST-LINK part to program the main STM32 using wires between CN4 and SWD signals available on ST morpho connector (SWCLK CN7 pin 15 and SWDIO CN7 pin 13).

6.2 Embedded ST-LINK/V2-1

The ST-LINK/V2-1 programming and debugging tool is integrated in the STM32 Nucleo board.

The ST-LINK/V2-1 makes the STM32 Nucleo boards Mbed Enabled™.

The embedded ST-LINK/V2-1 supports only SWD for STM32 devices. For information about debugging and programming features refer to *ST-LINK/V2 in-circuit debugger/programmer for STM8 and STM32* User manual (UM1075), which describes in details all the ST-LINK/V2 features.

The changes versus ST-LINK/V2 version are listed below.

- New features supported on ST-LINK/V2-1:
  - USB software re-enumeration
  - Virtual COM port interface on USB
  - Mass storage interface on USB
  - USB power management request for more than 100 mA power on USB
- Features not supported on ST-LINK/V2-1:
  - SWIM interface
  - Minimum supported application voltage limited to 3 V
- Known limitation:
  - Activating the readout protection on ST-LINK/V2-1 target prevents the target application from running afterwards. The target readout protection must be kept disabled on ST-LINK/V2-1 boards.

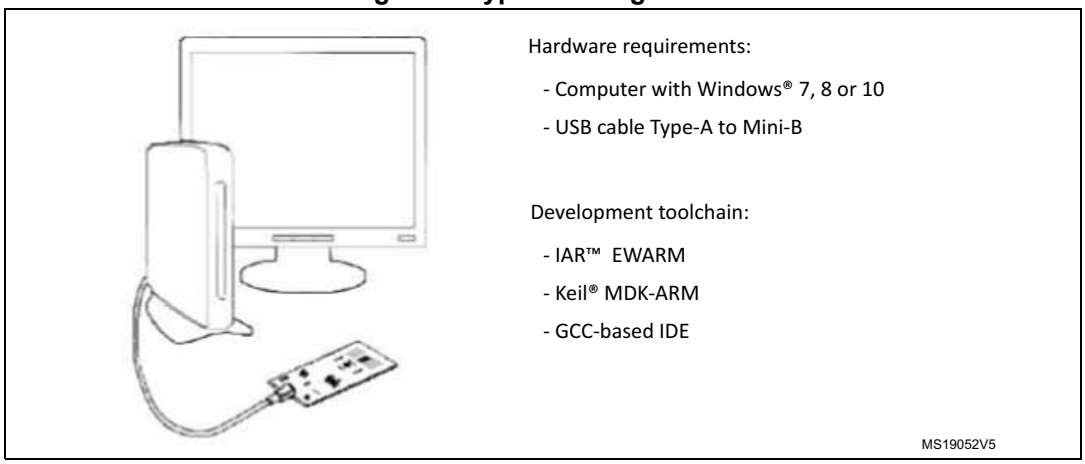
There are two different ways to use the embedded ST-LINK/V2-1 depending on the jumper states (see *Table 4* and *Figure 6*):

- Program/debug the on-board STM32 (*Section 6.2.2*)
- Program/debug an MCU in an external application board using a cable connected to SWD connector CN4 (*Section 6.2.4*).

Table 4. Jumper states

Jumper state	Description
Both CN2 jumpers ON	ST-LINK/V2-1 functions enabled for on board programming (default)
Both CN2 jumpers OFF	ST-LINK/V2-1 functions enabled for external CN4 connector (SWD supported)

Figure 6. Typical configuration



6.2.1 Driver

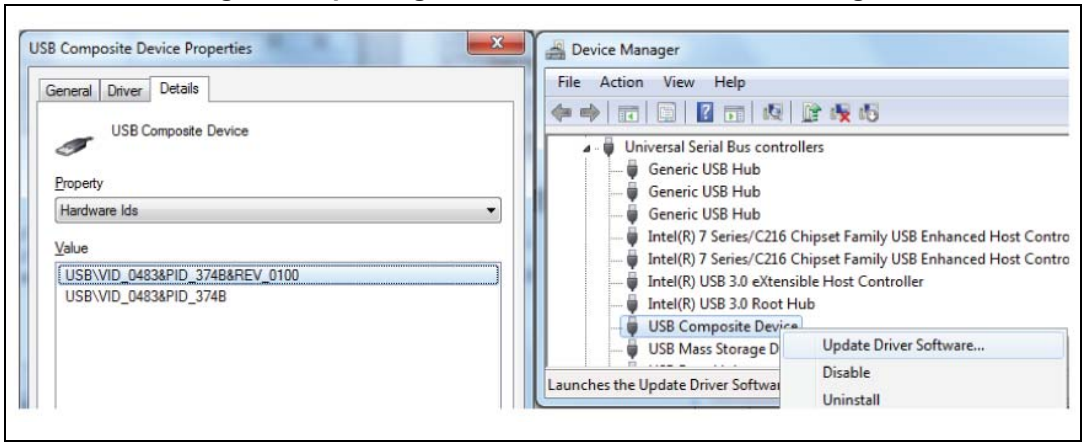
Before connecting the Nucleo-64 board to a Windows 7, Windows 8 or Windows XP PC via USB, a driver for ST-LINK/V2-1 must be installed. It can be downloaded from the *www.st.com* website.

In case the STM32 Nucleo-64 board is connected to the PC before installing the driver, the PC device manager may report some Nucleo interfaces as “Unknown”.

To recover from this situation, after installing the dedicated driver, the association of “Unknown” USB devices found on the STM32 Nucleo-64 board to this dedicated driver, must be updated in the device manager manually.

Note: It is recommended to proceed using USB Composite Device, as shown in Figure 7.

Figure 7. Updating the list of drivers in Device Manager



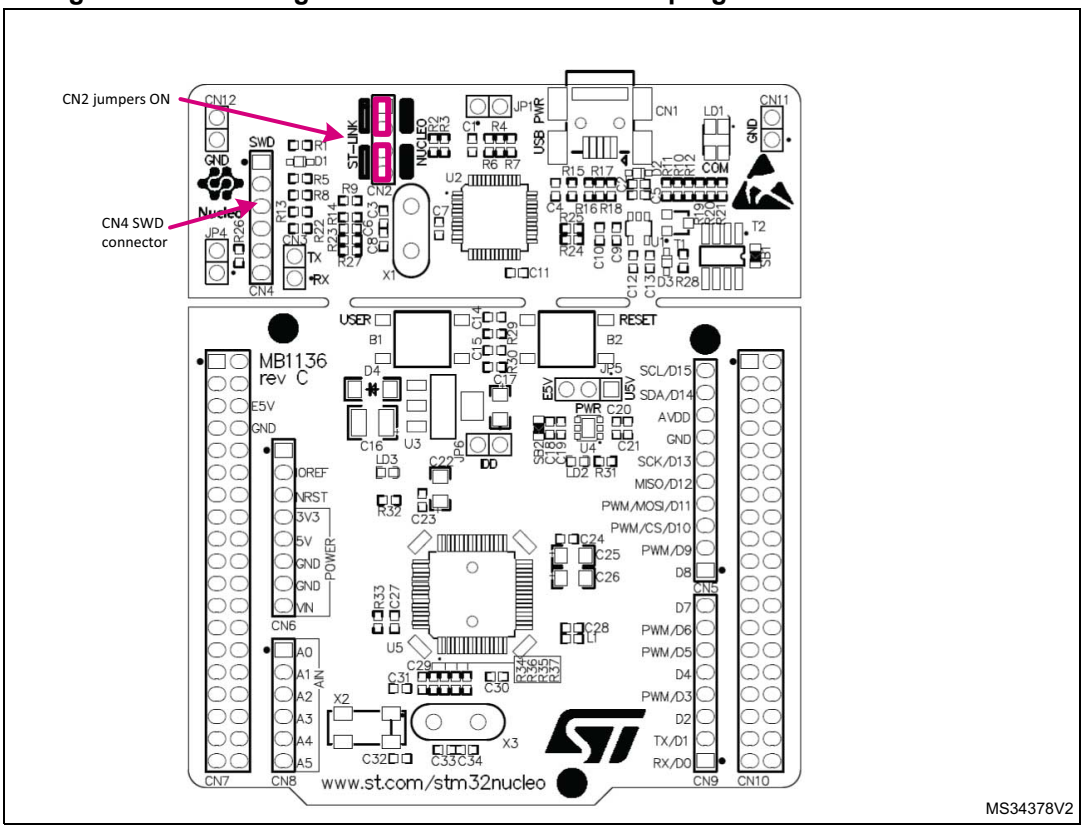
6.2.2 ST-LINK/V2-1 firmware upgrade

The ST-LINK/V2-1 embeds a firmware upgrade mechanism for in-situ upgrade through the USB port. As the firmware may evolve during the life time of the ST-LINK/V2-1 product (for example new functionality, bug fixes, support for new microcontroller families), it is recommended to visit *www.st.com* website before starting to use the STM32 Nucleo board and periodically, in order to stay up-to-date with the latest firmware version.

6.2.3 Using the ST-LINK/V2-1 to program and debug the STM32 on board

To program the STM32 on the board, plug in the two jumpers on CN2, as shown in red in *Figure 8*. Do not use the CN4 connector as this could disturb the communication with the STM32 microcontroller of the STM32 Nucleo board.

Figure 8. Connecting the STM32 Nucleo board to program the on-board STM32



6.2.4 Using ST-LINK/V2-1 to program and debug an external STM32 application

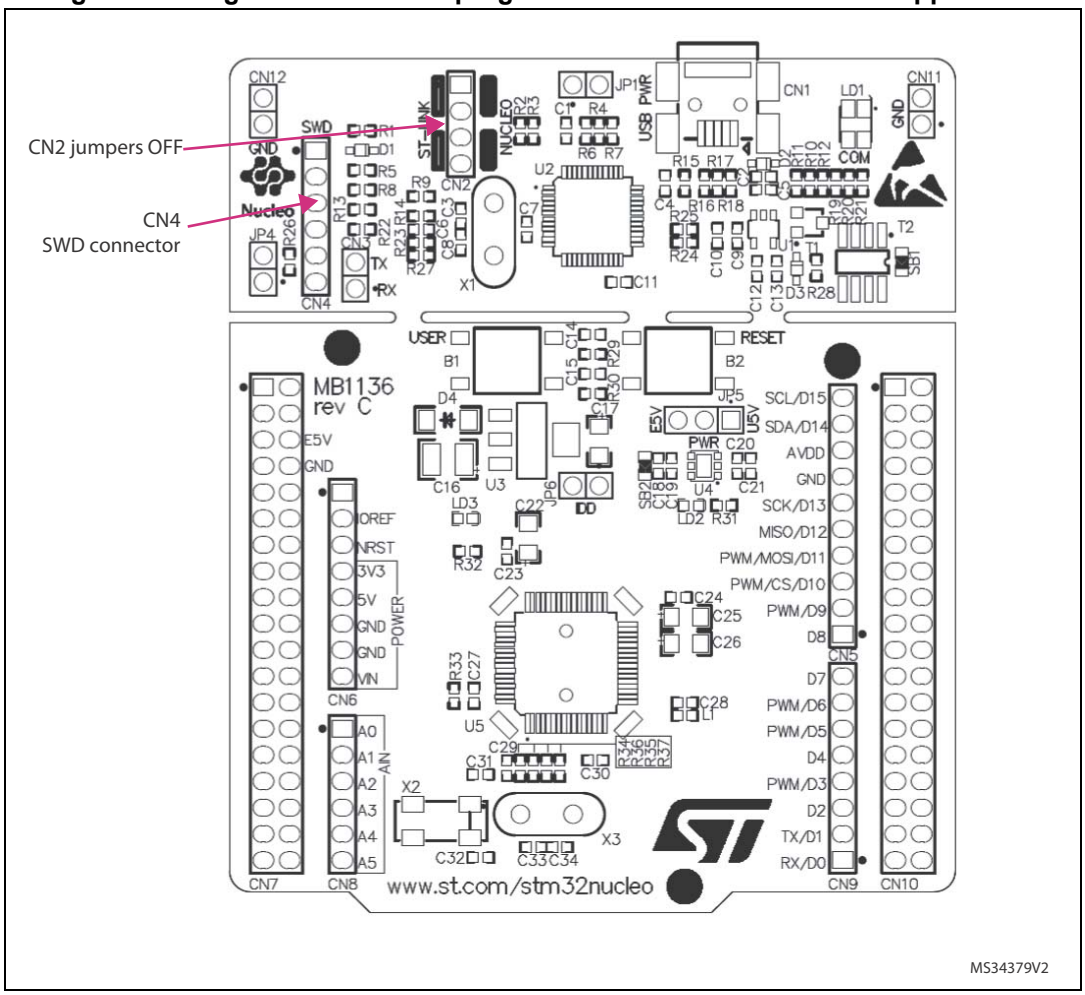
It is very easy to use the ST-LINK/V2-1 to program the STM32 on an external application. Simply remove the two jumpers from CN2 as illustrated in *Figure 9: Using ST-LINK/V2-1 to program the STM32 on an external application*, and connect the application to the CN4 debug connector according to *Table 5*.

Note: SB12 NRST (target STM32 RESET) must be OFF if CN4 pin 5 is used in the external application.

Table 5. Debug connector CN4 (SWD)

Pin	CN4	Designation
1	VDD_TARGET	VDD from application
2	SWCLK	SWD clock
3	GND	ground
4	SWDIO	SWD data input/output
5	NRST	RESET of target STM32
6	SWO	Reserved

Figure 9. Using ST-LINK/V2-1 to program the STM32 on an external application



6.3 Power supply and power selection

The power supply is provided either by the host PC through the USB cable, or by an external source: VIN (7V-12V), E5V (5V) or +3.3V power supply pins on CN6 or CN7. In case VIN, E5V or +3.3V is used to power the STM32 Nucleo board, using an external power supply unit or an auxiliary equipment, this power source must comply with the standard EN-60950-1: 2006+A11/2009, and must be Safety Extra Low Voltage (SELV) with limited power capability.

6.3.1 Power supply input from the USB connector

The ST-LINK/V2-1 supports USB power management allowing to request more than 100 mA current to the host PC.

All parts of the STM32 Nucleo board and shield can be powered from the ST-LINK USB connector CN1 (U5V or VBUS). Note that only the ST-LINK part is power supplied before the USB enumeration as the host PC only provides 100 mA to the board at that time. During the USB enumeration, the STM32 Nucleo board requires 300 mA of current to the host PC. If the host is able to provide the required power, the targeted STM32 microcontroller is powered and the red LED LD3 is turned ON, thus the STM32 Nucleo board and its shield can consume a maximum of 300 mA current, not more. If the host is not able to provide the required current, the targeted STM32 microcontroller and the MCU part including the extension board are not power supplied. As a consequence the red LED LD3 remains turned OFF. In such case it is mandatory to use an external power supply as explained in the next *Section 6.3.2: External power supply inputs: VIN and E5V*.

When the board is power supplied by USB (U5V) a jumper must be connected between pin 1 and pin 2 of JP5 as shown in *Table 8*.

JP1 is configured according to the maximum current consumption of the board when powered by USB (U5V). JP1 jumper can be set in case the board is powered by USB and maximum current consumption on U5V does not exceed 100 mA (including an eventual extension board or Arduino shield). In such condition USB enumeration will always succeed since no more than 100mA is requested to the PC. Possible configurations of JP1 are summarized in *Table 6*.

Table 6. JP1 configuration table

Jumper state	Power supply	Allowed current
JP1 jumper OFF	USB power through CN1	300 mA max
JP1 jumper ON		100 mA max

Warning: If the maximum current consumption of the NUCLEO and its extension boards exceeds 300 mA, it is mandatory to power the NUCLEO using an external power supply connected to E5V or VIN.

Note: In case the board is powered by an USB charger, there is no USB enumeration, so the led LD3 remains set to OFF permanently and the target STM32 is not powered. In this specific case the jumper JP1 needs to be set to ON, to allow target STM32 to be powered anyway.

6.3.2 External power supply inputs: VIN and E5V

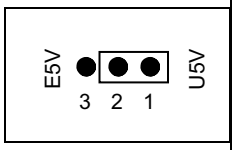
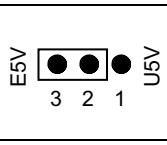
The external power sources VIN and E5V are summarized in the *Table 7.* When the board is power supplied by VIN or E5V, the jumpers configuration must be the following:

- Jumper on JP5 pin 2 and pin 3
- Jumper removed on JP1

Table 7. External power sources

Input power name	Connectors pins	Voltage range	Max current	Limitation
VIN	CN6 pin 8 CN7 pin 24	7 V to 12 V	800 mA	From 7 V to 12 V only and input current capability is linked to input voltage: 800 mA input current when Vin=7 V 450 mA input current when 7 V<Vin (< or =) 9 V 250 mA input current when 9 V<Vin (< or =) 12 V
E5V	CN7 pin 6	4.75 V to 5.25 V	500 mA	-

Table 8. Power-related jumper

Jumper	Description
JP5	U5V (ST-LINK VBUS) is used as power source when JP5 is set as shown below (Default setting) 
	VIN or E5V is used as power source when JP5 is set as shown below. 

Using VIN or E5V as external power supply

VIN or E5V can be used as external power supply in case the current consumption of the STM32 Nucleo and extensions boards exceeds the allowed current on USB. In this condition it is still possible to use the USB for communication, for programming or debugging only, but it is mandatory to power supply the board first using VIN or E5V then connect the USB cable to the PC. Proceeding this way ensures that the enumeration occurs thanks to the external power source.

The following power sequence procedure must be respected:

1. Connect the jumper between pin 2 and pin 3 of JP5
2. Check that JP1 is removed
3. Connect the external power source to VIN or E5V
4. Power on the external power supply 7 V< VIN < 12 V to VIN, or 5 V for E5V
5. Check that LD3 is turned ON
6. Connect the PC to USB connector CN1

If this order is not respected, the board may be supplied by VBUS first then by VIN or E5V, and the following risks may be encountered:

1. If more than 300 mA current is needed by the board, the PC may be damaged or the current supply can be limited by the PC. As a consequence the board is not powered correctly.
2. 300 mA is requested at enumeration (since JP1 must be OFF) so there is risk that the request is rejected and the enumeration does not succeed if the PC cannot provide such current. Consequently the board is not power supplied (LED LD3 remains OFF).

6.3.3 External power supply input: + 3.3V

It can be of interest to use the +3.3V (CN6 pin 4 or CN7 pin 12 and pin 16) directly as power input for instance in case the 3.3V is provided by an extension board. When the STM32 Nucleo board is power supplied by +3.3V, the ST-LINK is not powered thus the programming and debug features are unavailable. The external power sources +3.3V is summarized in the *Table 9.*

Table 9. +3.3 V external power source

Input power name	Connectors pins	Voltage range	Limitation
+3.3V	CN6 pin 4 CN7 pin 12 and pin 16	3 V to 3.6 V	Used when ST-LINK part of PCB is cut or SB2 and SB12 OFF

Two different configurations are possible when using +3.3V to power the board:

- ST-LINK is removed (PCB cut) or
- SB2 (3.3V regulator) and SB12 (NRST) are OFF.

6.3.4 External power supply output

When powered by USB, VIN or E5V, the +5V (CN6 pin 5 or CN7 pin 18) can be used as output power supply for an Arduino shield or an extension board. In this case, the maximum current of the power source specified in *Table 7* must be respected.

The +3.3V (CN6 pin 4 or CN7 pin 12 and 16) can be used also as power supply output. The current is limited by the maximum current capability of the regulator U4 (500 mA max).

6.4 LEDs

The tricolor LED (green, orange, red) LD1 (COM) provides information about ST-LINK communication status. LD1 default color is red. LD1 turns to green to indicate that communication is in progress between the PC and the ST-LINK/V2-1, with the following setup:

- Slow blinking Red/Off: at power-on before USB initialization
- Fast blinking Red/Off: after the first correct communication between the PC and ST-LINK/V2-1 (enumeration)
- Red LED On: when the initialization between the PC and ST-LINK/V2-1 is complete
- Green LED On: after a successful target communication initialization
- Blinking Red/Green: during communication with target
- Green On: communication finished and successful
- Orange On: Communication failure

User LD2: the green LED is a user LED connected to Arduino signal D13 corresponding to STM32 I/O PA5 (pin 21) or PB13 (pin 34) depending on the STM32 target. Refer to *Table 11* to *Table 23* when:

- the I/O is HIGH value, the LED is on
- the I/O is LOW, the LED is off

LD3 PWR: the red LED indicates that the STM32 part is powered and +5V power is available.

6.5 Push-buttons

B1 USER: the user button is connected to the I/O PC13 (pin 2) of the STM32 microcontroller.

B2 RESET: this push-button is connected to NRST, and is used to RESET the STM32 microcontroller.

Note: The blue and black plastic hats that are placed on the push buttons can be removed if necessary, for example when a shield or when an application board is plugged on top of the Nucleo board. This will avoid pressure on the buttons and consequently a possible permanent target STM32 RESET.

6.6 JP6 (IDD)

Jumper JP6, labeled IDD, is used to measure the STM32 microcontroller consumption by removing the jumper and by connecting an ammeter:

- Jumper ON: STM32 microcontroller is powered (default).
- Jumper OFF: an ammeter must be connected to measure the STM32 microcontroller current. If there is no ammeter, STM32 microcontroller is not powered.

6.7 OSC clock

6.7.1 OSC clock supply

There are four ways to configure the pins corresponding to external high-speed clock (HSE):

- **MCO from ST-LINK**: MCO output of ST-LINK MCU is used as input clock. This frequency cannot be changed, it is fixed at 8 MHz and connected to PF0/PD0/PH0-OSC_IN of the STM32 microcontroller.

  The following configuration is needed:

  – SB55 OFF and SB54 ON
  – SB16 and SB50 ON
  – R35 and R37 removed
- **HSE oscillator on-board from X3 crystal (not provided):** for typical frequencies and its capacitors and resistors, refer to STM32 microcontroller datasheet. Refer to the AN2867 Application note for oscillator design guide for STM32 microcontrollers.The X3 crystal has the following characteristics: 8 MHz, 16 pF, 20 ppm, and DIP footprint. It is recommended to use 9SL8000016AFXHF0 manufactured by Hong Kong X'tals Limited.

  The following configuration is needed:

  – SB54 and SB55 OFF
  – R35 and R37 soldered
  – C33 and C34 soldered with 20 pF capacitors
  – SB16 and SB50 OFF
- **Oscillator from external PF0/PD0/PH0**: from an external oscillator through pin 29 of the CN7 connector.

  The following configuration is needed:

  – SB55 ON
  – SB50 OFF
  – R35 and R37 removed
- **HSE not used**: PF0/PD0/PH0 and PF1/PD1/PH1 are used as GPIO instead of clock

  The following configuration is needed:

  – SB54 and SB55 ON
  – SB16 and SB50 (MCO) OFF
  – R35 and R37 removed

There are two possible default configurations of the HSE pins, depending on the version of the STM32 Nucleo board hardware.

The board version MB1136 C-01 or MB1136 C-02 is mentioned on the sticker, placed on the bottom side of the PCB.

The board marking MB1136 C-01 corresponds to a board, configured as HSE not used.

The board marking MB1136 C-02 (or higher) corresponds to a board, configured to use ST-LINK MCO as clock input.

Note: For NUCLEO-L476RG and NUCLEO-L452RE the ST-LINK MCO output is not connected to OSCIN to reduce power consumption in low power mode. Consequently NUCLEO-L476RG and NUCLEO-L452RE configuration corresponds to HSE not used.

6.7.2 OSC 32 kHz clock supply

There are three ways to configure the pins corresponding to low-speed clock (LSE):

- **On-board oscillator:** X2 crystal. Refer to the *Oscillator design guide for STM8S, STM8A and STM32 microcontrollers* application note (AN2867) for oscillator design guide for STM32 microcontrollers.It is recommended to use ABS25-32.768KHZ-6-T, manufactured by Abracon corporation.
- **Oscillator from external PC14**: from external oscillator through the pin 25 of CN7 connector.

  The following configuration is needed:
  - SB48 and SB49 ON
  - R34 and R36 removed
- **LSE not used**: PC14 and PC15 are used as GPIOs instead of low speed clock.

  The following configuration is needed:
  - SB48 and SB49 ON
  - R34 and R36 removed

There are three possible default configurations of the LSE depending on the version of the STM32 Nucleo board hardware.

The board version MB1136 C-01 or MB1136 C-02 is mentioned on the sticker placed on the bottom side of the PCB.

The board marking MB1136 C-01 corresponds to a board configured as LSE not used.

The board marking MB1136 C-02 (or higher) corresponds to a board configured with on-board 32 kHz oscillator.

The board marking MB1136 C-03 (or higher) corresponds to a board using new LSE crystal (ABS25) and C26, C31 and C32 value update.

6.8 USART communication

The USART2 interface available on PA2 and PA3 of the STM32 microcontroller can be connected to ST-LINK MCU, ST morpho connector or to Arduino connector. The choice can be changed by setting the related solder bridges. By default the USART2 communication between the target STM32 and ST-LINK MCU is enabled, in order to support virtual COM port for Mbed™ (SB13 and SB14 ON, SB62 and SB63 OFF). If the communication between the target STM32 PA2 (D1) or PA3 (D0) and shield or extension board is required, SB62 and SB63 should be ON, SB13 and SB14 should be OFF. In such case it is possible to connect another USART to ST-LINK MCU using flying wires between ST morpho connector and CN3. For instance on NUCLEO-F103RB it is possible to use USART3 available on PC10 (TX) and PC11 (RX). Two flying wires need to be connected as follow:

- PC10 (USART3_TX) available on CN7 pin 1 to CN3 pin RX
- PC11 (USART3_RX) available on CN7 pin 2 to CN3 pin TX

6.9 Solder bridges

Table 10. Solder bridges

Bridge	State(1)	Description
SB54, SB55 (X3 crystal)(2)	OFF	X3, C33, C34, R35 and R37 provide a clock as shown in *Section Appendix A: Electrical schematics* PF0/PD0/PH0, PF1/PD1/PH1 are disconnected from CN7.
	ON	PF0/PD0/PH0, PF1/PD1/PH1 are connected to CN12. (R35, R37 and SB50 must not be fitted).
SB3,5,7,9 (DEFAULT)	**ON**	Reserved, do not modify.
SB4,6,8,10 (RESERVED)	OFF	Reserved, do not modify.
SB48,49 (X2 crystal)(3)	OFF	X2, C31, C32, R34 and R36 deliver a 32 kHz clock. PC14, PC15 are not connected to CN7.
	ON	PC14, PC15 are only connected to CN7. Remove only R34, R36.
SB17 (B1-USER)	**ON**	B1 push button is connected to PC13.
	OFF	B1 push button is not connected to PC13.
SB12 (NRST)	ON	The NRST signal of the CN4 connector is connected to the NRST pin of the STM32.
	OFF	The NRST signal of the CN4 connector is not connected to the NRST pin of the STM32.
SB15 (SWO)	ON	The SWO signal of the CN4 connector is connected to PB3.
	OFF	The SWO signal is not connected.
SB11 (STM_RST)	**OFF**	No incidence on STM32F103CBT6 (ST-LINK MCU) NRST signal.
	ON	STM32F103CBT6 (ST-LINK MCU) NRST signal is connected to GND.
SB1 (USB-5V)	OFF	USB power management is functional.
	ON	USB power management is disabled.
SB2 (3.3 V)	ON	Output of voltage regulator LD39050PU33R is connected to 3.3V.
	OFF	Output of voltage regulator LD39050PU33R is not connected.
SB21 (LD2-LED)	ON	Green user LED LD2 is connected to D13 of Arduino signal.
	OFF	Green user LED LD2 is not connected.
SB56,SB51 (A4 and A5)	ON	PC1 and PC0 (ADC in) are connected to A4 and A5 (pin 5 and pin 6) on Arduino connector CN8 and ST morpho connector CN7. Thus SB46 and SB52 should be OFF.
	OFF	PC1 and PC0 (ADC in) are disconnected to A4 and A5 (pin 5 and pin 6) on Arduino connector CN8 and ST morpho connector CN7.
SB46,SB52 (I2C on A4 and A5)	OFF	PB9 and PB8 (I2C) are disconnected to A4 and A5 (pin 5 and pin 6) on Arduino connector CN8 and ST morpho connector CN7.
	ON	PB9 and PB8 (I2C) are connected to A4 and A5 (pin 5 and pin 6) on Arduino connector CN8 and ST morpho connector CN7 as I2C signals. Thus SB56 and SB51 should be OFF.

Table 10. Solder bridges (continued)

Bridge	State[1]	Description
SB45 (VBAT/VLCD)	ON	VBAT or VLCD on STM32 is connected to VDD.
	OFF	VBAT or VLCD on STM32 is not connected to VDD.
SB57 (VDDA/VREF+)	ON	VDDA/VREF+ on STM32 is connected to VDD.
	OFF	VDDA/VREF+ on STM32 is not connected to VDD and can be provided from pin 8 of CN5 (Used for external VREF+ provided by Arduino shield)
SB62, SB63 (USART)	**OFF**	PA2 and PA3 on STM32 are disconnected to D1 and D0 (pin 2 and pin 1) on Arduino connector CN9 and ST morpho connector CN10.
	ON	PA2 and PA3 on STM32 are connected to D1 and D0 (pin 2 and pin 1) on Arduino connector CN9 and ST morpho connector CN10 as USART signals. Thus SB13 and SB14 should be OFF.
SB13, SB14 (ST-LINK-USART)	**ON**	PA2 and PA3 on STM32F103CBT6 (ST-LINK MCU) are connected to PA3 and PA2 on STM32 to have USART communication between them. Thus SB61,SB62 and SB63 should be OFF.
	OFF	PA2 and PA3 on STM32F103CBT6 (ST-LINK MCU) are disconnected to PA3 and PA2 on STM32.
SB16,SB50(MCO)[2]	OFF	MCO on STM32F103CBT6 (ST-LINK MCU) are disconnected to PF0/PD0/PH0 on STM32.
	ON	MCO on STM32F103CBT6 (ST-LINK MCU) are connected to PF0/PD0/PH0 on STM32.

1. The default SBx state is shown in bold.
2. Default configuration depends on board version. Refer to *Section 6.7.1: OSC clock supply* for details.
3. Default configuration depends on board version. Refer to *Section 6.7.2: OSC 32 kHz clock supply* for details.

All the other solder bridges present on the STM32 Nucleo board are used to configure several I/Os and power supply pins for compatibility of features and pinout with STM32 supported.

All STM32 Nucleo boards are delivered with the solder-bridges configured according to the target supported STM32.

6.10 Extension connectors

Figure 10 to *Figure 26* show the signals connected by default to Arduino Uno V3 connectors (CN5, CN6, CN8, CN9) and to ST morpho connector (CN7 and CN10), for each STM32 Nucleo board.

Figure 10. NUCLEO-F030R8

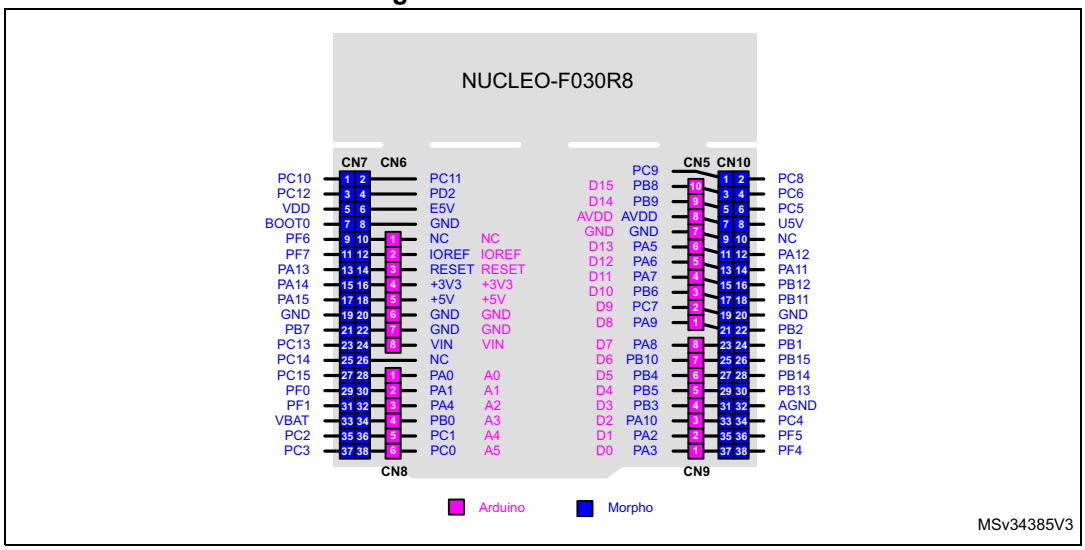


Figure 11. NUCLEO-F070RB

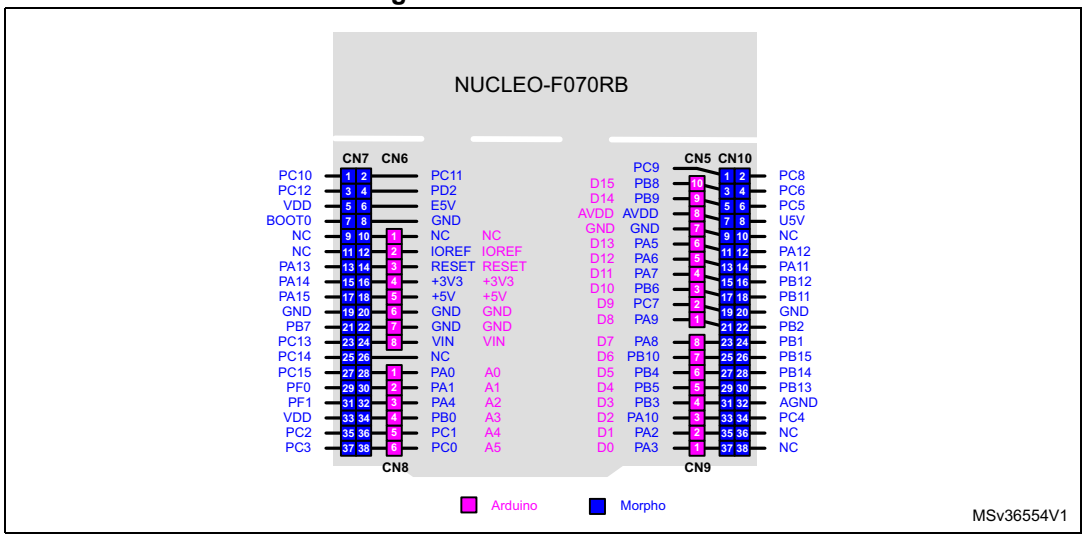


Figure 12. NUCLEO-F072RB

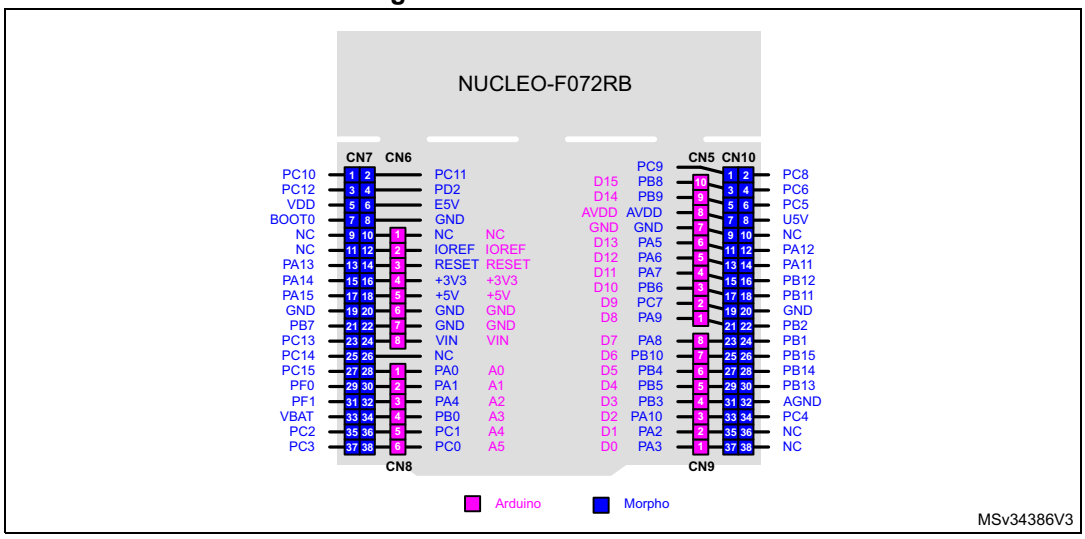


Figure 13. NUCLEO-F091RC

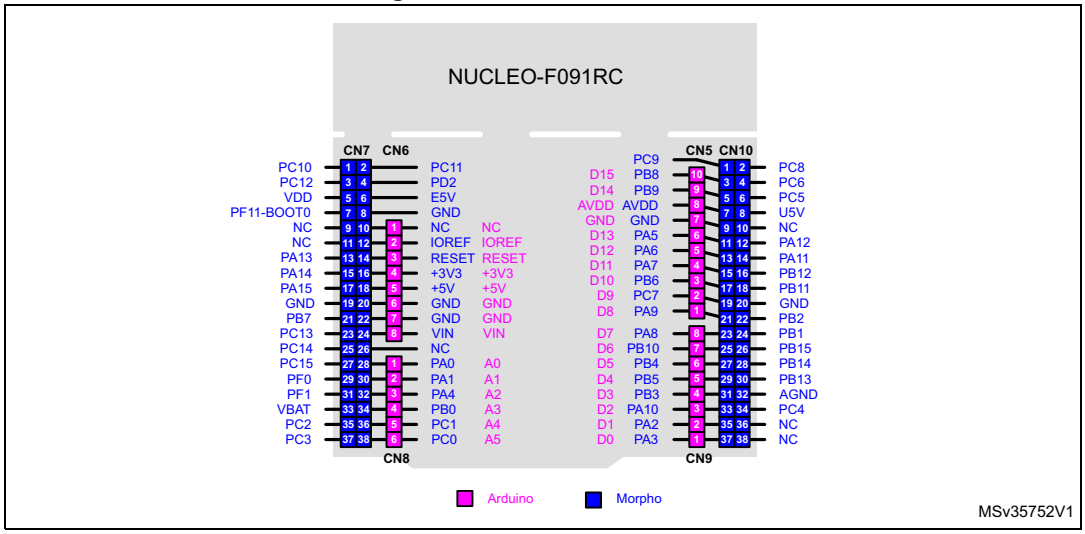


Figure 14. NUCLEO-F103RB

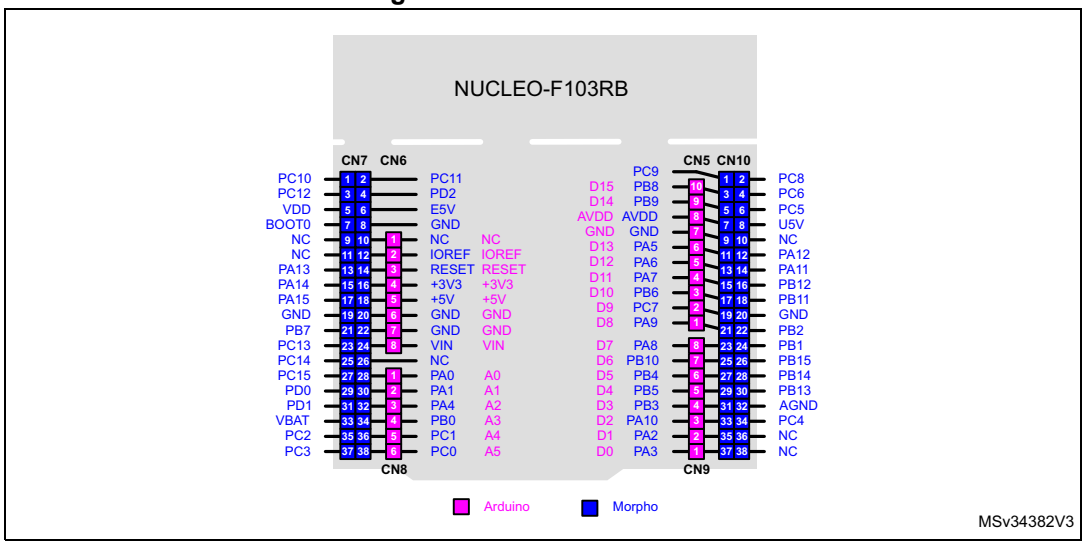


Figure 15. NUCLEO-F302R8

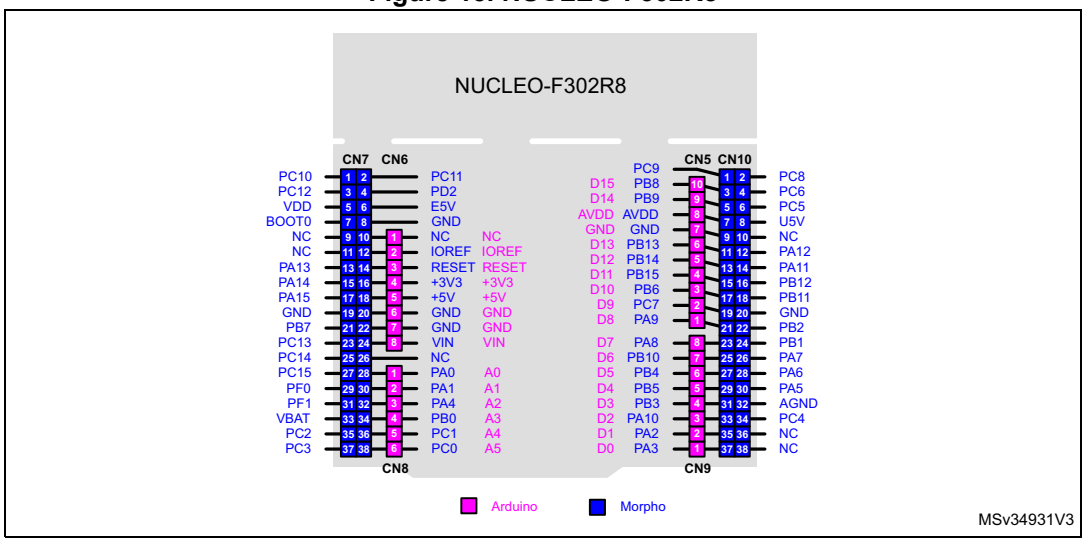


Figure 16. NUCLEO-F303RE

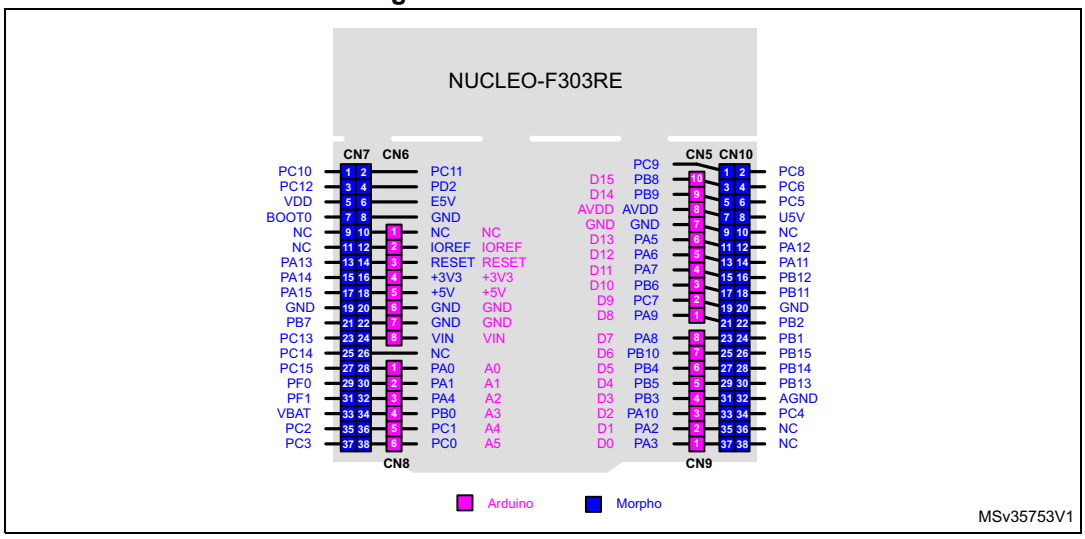


Figure 17. NUCLEO-F334R8

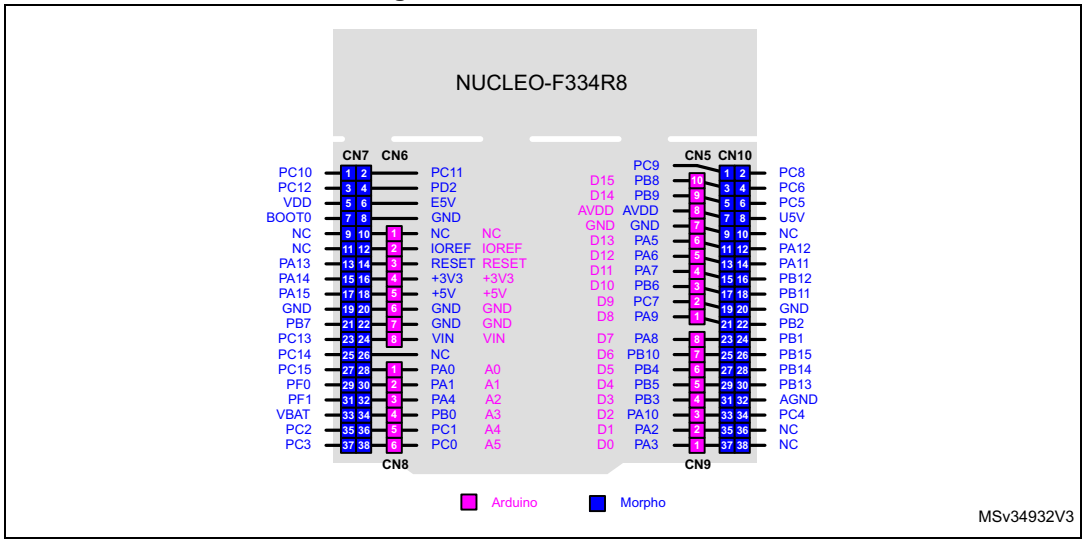


Figure 18. NUCLEO-F401RE

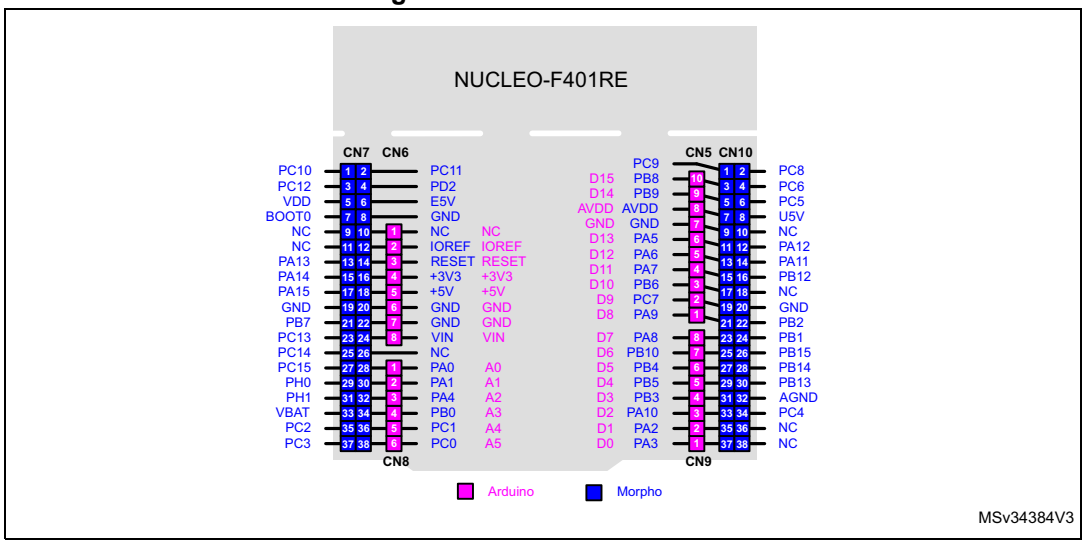


Figure 19. NUCLEO-F411RE

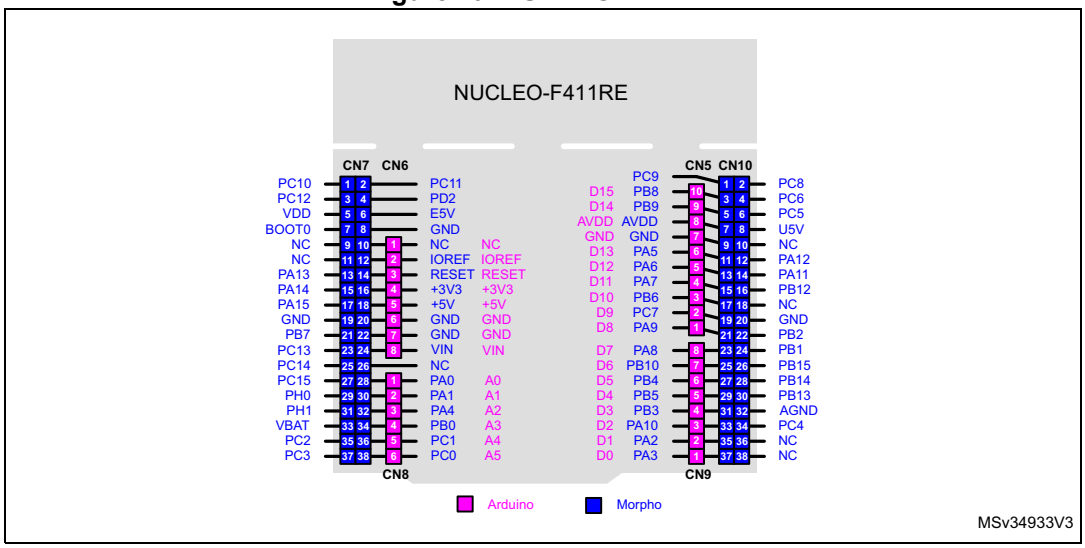


Figure 20. NUCLEO-L053R8

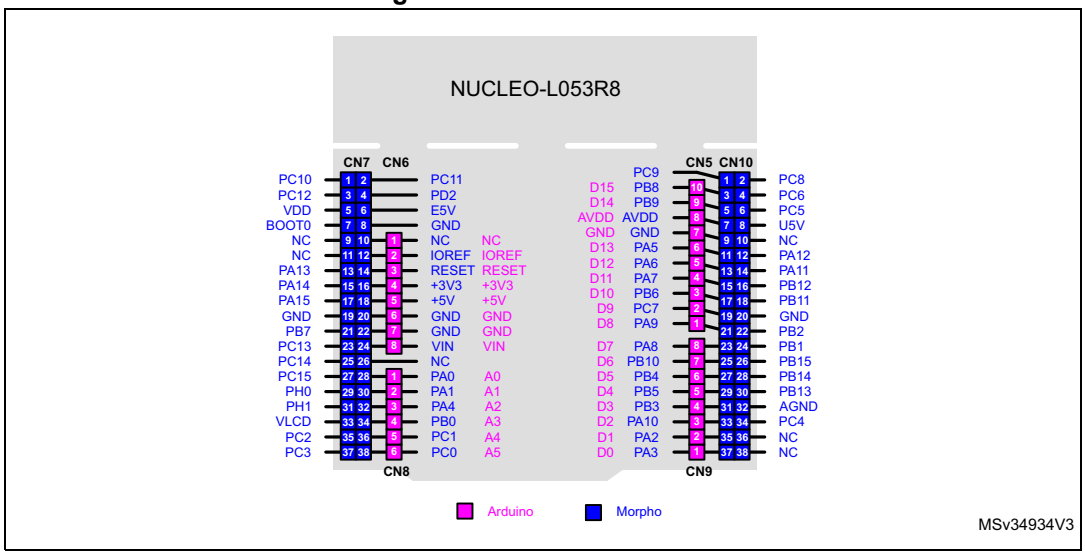


Figure 21. NUCLEO-L073RZ and NUCLEO-L010RB

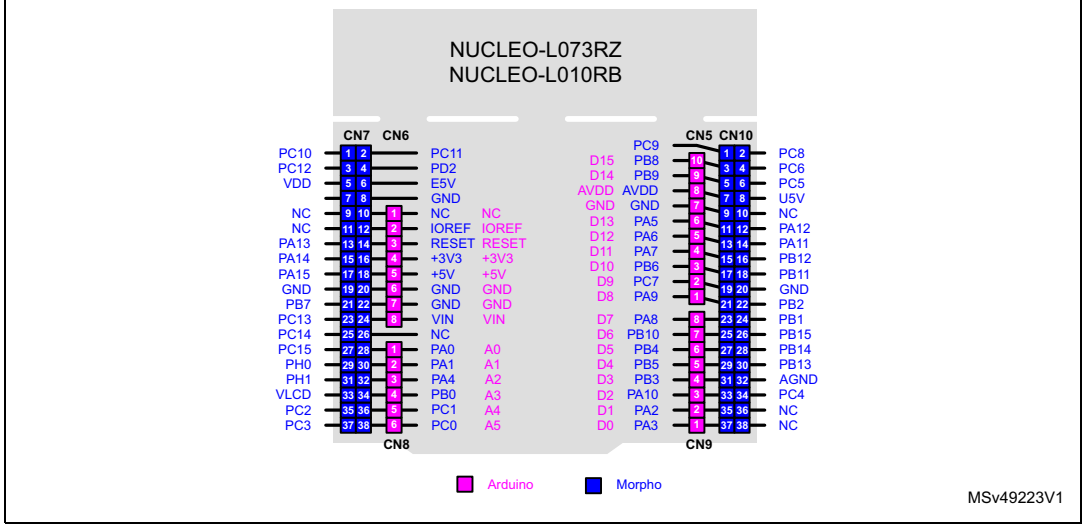


Figure 22. NUCLEO-L152RE

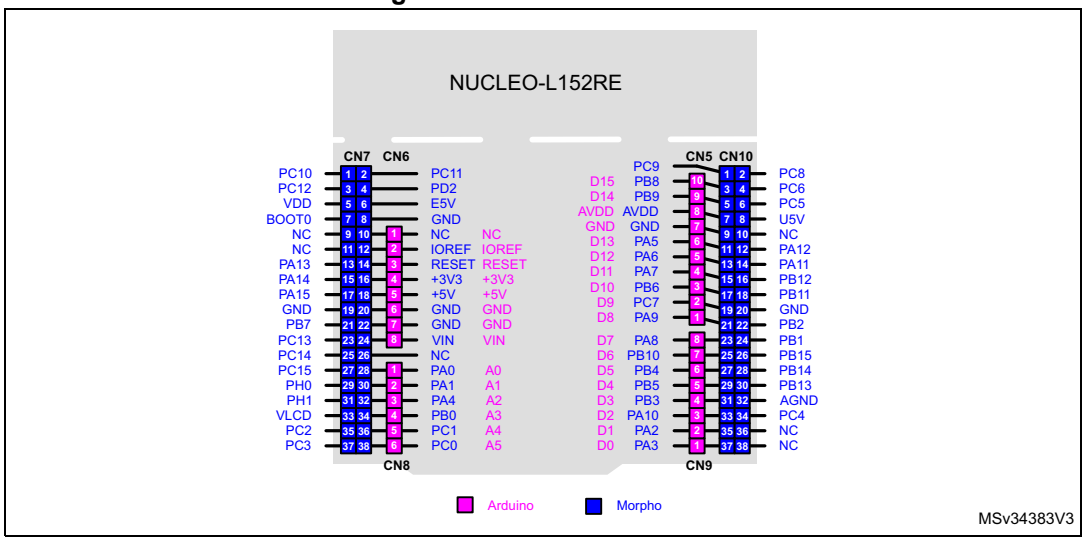


Figure 23. NUCLEO-L452RE

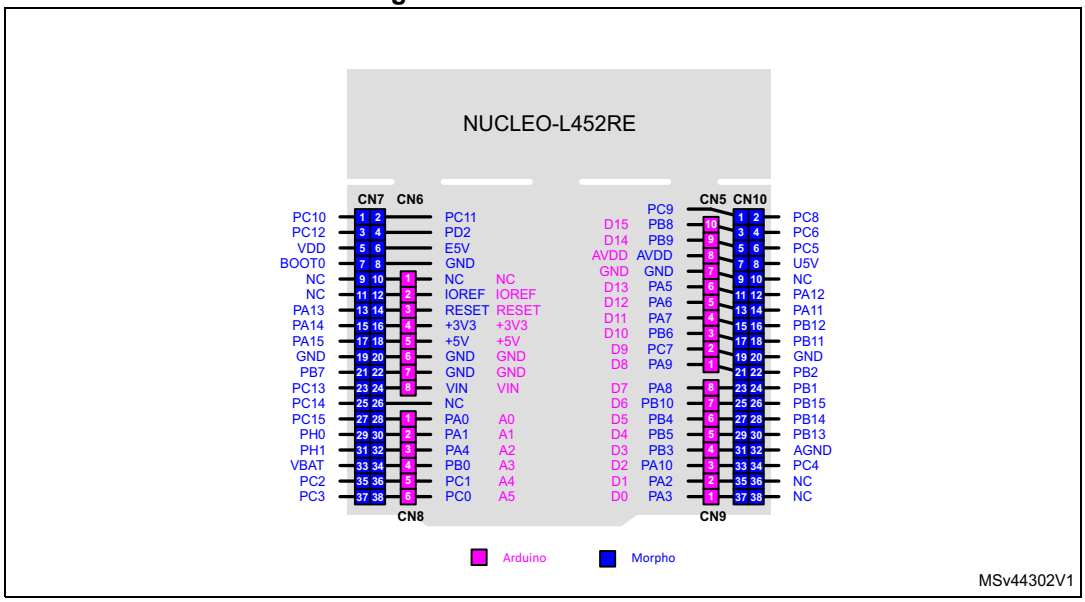


Figure 24. NUCLEO-L476RG

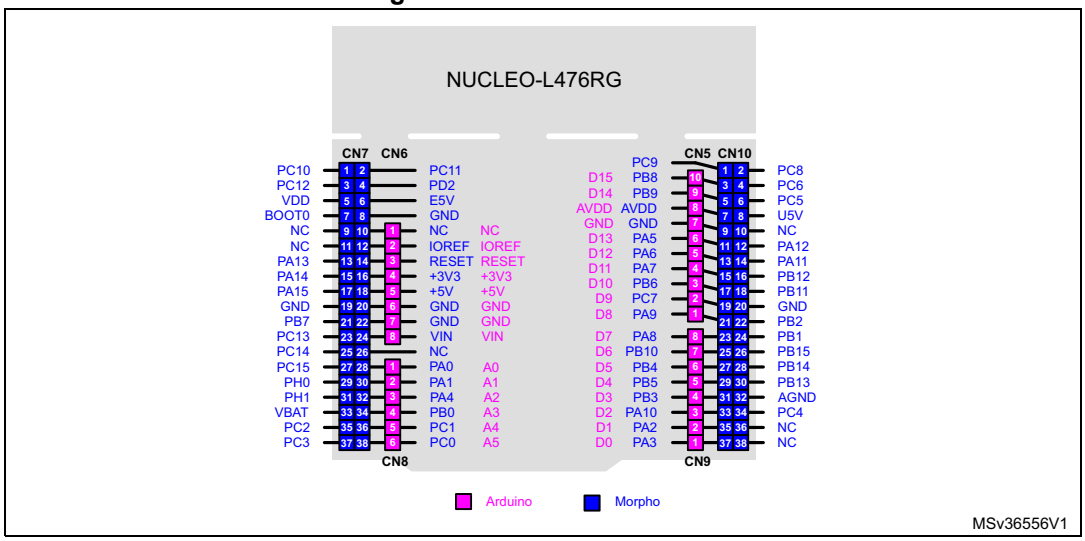


Figure 25. NUCLEO-F446RE

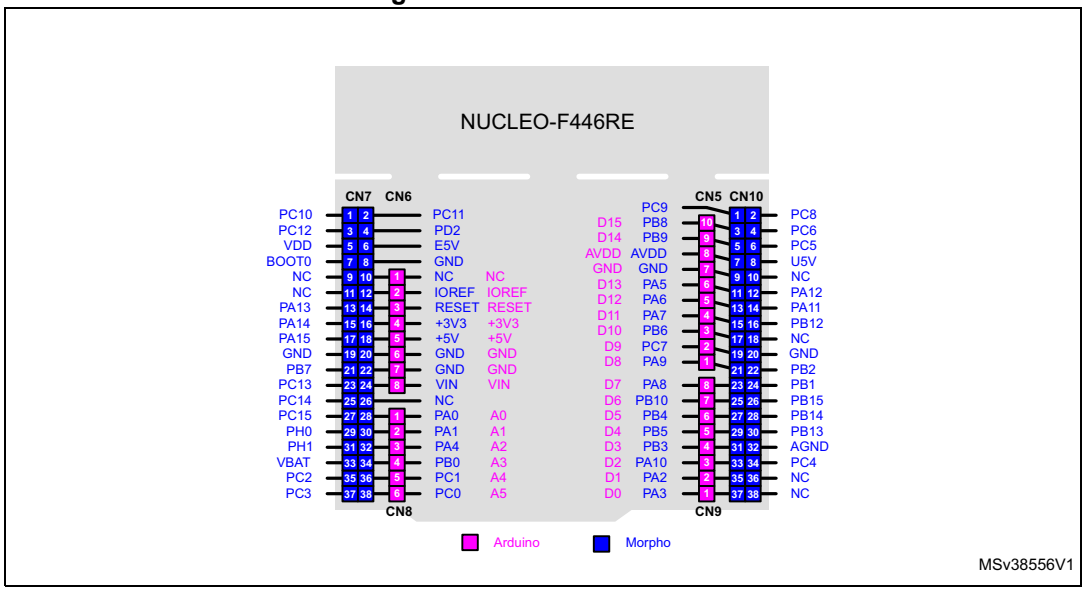
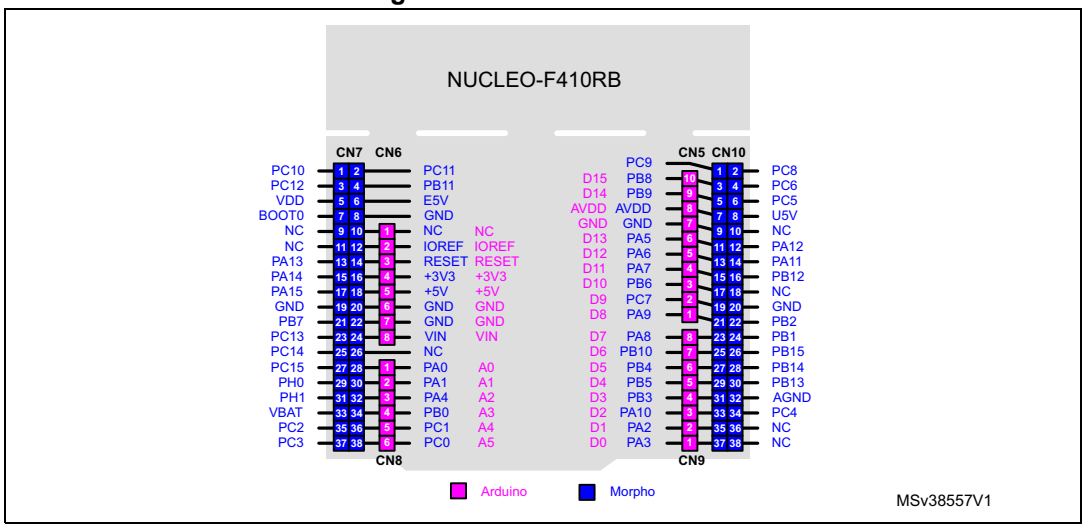


Figure 26. NUCLEO-F410RB



6.11 Arduino connectors

CN5, CN6, CN8 and CN9 are female connectors compatible with Arduino standard. Most shields designed for Arduino can fit to the STM32 Nucleo boards.

The Arduino connectors on STM32 Nucleo board support the Arduino Uno V3.

For compatibility with Arduino Uno V1, apply the following modifications:

- SB46 and SB52 should be ON,
- SB51 and SB56 should be OFF to connect I^2C on A4 (pin 5) and A5 (pin 6 of CN8).

Caution 1: The I/Os of STM32 microcontroller are 3.3 V compatible instead of 5 V for Arduino Uno V3.

Caution 2: SB57 should be removed before implementing Arduino shield with VREF+ power being provided on CN5 pin 8. Refer to *Table 10: Solder bridges* for details on SB57.

Table 11 to *Table 23* show the pin assignment of each main STM32 microcontroller on Arduino connectors.

Table 11. Arduino connectors on NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9(1)	ADC_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	10	D15	PB8	I2C1_SCL
	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground

Table 11. Arduino connectors on NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN5 digital	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM16_CH1N or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3(2)
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2(3)
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.
2. PWM is not supported by D6 on STM32F030 and STM32F070 since the timer is not available on PB10.
3. PWM is not supported by D3 on STM32F030 and STM32F070 since the timer is not available on PB3.

Table 12. Arduino connectors on NUCLEO-F103RB

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF		3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V		5V output
	6	GND		ground
	7	GND		ground
	8	VIN		Power input
CN8 analog	1	A0	PA0	ADC_0
	2	A1	PA1	ADC_1
	3	A2	PA4	ADC_4
	4	A3	PB0	ADC_8
	5	A4	PC1 or PB9(1)	ADC_11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND		ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM3_CH2 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Table 13. Arduino connectors on NUCLEO-F302R8

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN1
	2	A1	PA1	ADC_IN2
	3	A2	PA4	ADC_IN5
	4	A3	PB0	ADC_IN11
	5	A4	PC1 or PB9(1)	ADC_IN7 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN6 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PB13	SPI2_SCK
	5	D12	PB14	SPI2_MISO
	4	D11	PB15	TIM15_CH2 or SPI2_MOSI
	3	D10	PB6	TIM16_CH1N or SPI2_CS
	2	D9	PC7	-
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM16_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Warning: PWM is not supported by D9 on STM32F302 since the timer is not available on PC7.

Table 14. Arduino connectors on NUCLEO-F303RE

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_IN1
	2	A1	PA1	ADC1_IN2
	3	A2	PA4	ADC2_IN1
	4	A3	PB0	ADC3_IN12
	5	A4	PC1 or PB9(1)	ADC12_IN7 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC12_IN6 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-

Table 14. Arduino connectors on NUCLEO-F303RE (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX

1. Refer to *Table 10: Solder bridges* for details.

Table 15. Arduino connectors on NUCLEO-F334R8

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_IN1
	2	A1	PA1	ADC1_IN2
	3	A2	PA4	ADC2_IN1
	4	A3	PB0	ADC1_IN11
	5	A4	PC1 or PB9(1)	ADC_IN7 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN6 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM16_CH1N or SPI1_CS

Table 15. Arduino connectors on NUCLEO-F334R8 (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN5 digital	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Table 16. Arduino connectors on NUCLEO-F401RE and NUCLEO-F411RE

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_0
	2	A1	PA1	ADC1_1
	3	A2	PA4	ADC1_4
	4	A3	PB0	ADC1_8
	5	A4	PC1 or PB9(1)	ADC1_11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC1_10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground

Table 16. Arduino connectors on NUCLEO-F401RE and NUCLEO-F411RE (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN5 digital	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM1_CH1N or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Table 17. Arduino connectors on NUCLEO-L053R8

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9(1)	ADC_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM12_CH2 or SPI1_MOSI
	3	D10	PB6	SPI1_CS
	2	D9	PC7	TIM12_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM12_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Warning: PWM is not supported by D10 on STM32L053 since the timer is not available on PB6.

Table 18. Arduino connectors on NUCLEO-L010RB and NUCLEO-L073RZ

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9(1)	ADC_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM22_CH2 or SPI1_MOSI
	3	D10	PB6	SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-

Table 18. Arduino connectors on NUCLEO-L010RB and NUCLEO-L073RZ (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Warning: PWM is not supported by D10 on STM32L073 since the timer is not available on PB6.

Table 19. Arduino connectors on NUCLEO-F446RE

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC123_IN0
	2	A1	PA1	ADC123_IN1
	3	A2	PA4	ADC12_IN4
	4	A3	PB0	ADC12_IN8
	5	A4	PC1 or PB9(1)	ADC123_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC123_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	10	D15	PB8	I2C1_SCL

Table 19. Arduino connectors on NUCLEO-F446RE (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN5 digital	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM14_CH1 || SPI1_MOSI
	3	D10	PB6	TIM4_CH1 || SPI1_CS
	2	D9	PC7	TIM8_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Table 20. Arduino connectors on NUCLEO-F410RB

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_0
	2	A1	PA1	ADC1_1
	3	A2	PA4	ADC1_4
	4	A3	PB0	ADC1_8
	5	A4	PC1 or PB9(1)	ADC1_11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC1_10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	PB8	D15	I2C1_SCL
	9	PB9	D14	I2C1_SDA
	8	-	AREF	AVDD
	7	-	GND	ground
	6	PA5	D13	SPI1_SCK
	5	PA6	D12	SPI1_MISO
	4	PA7	D11	TIM1_CH1N || SPI1_MOSI
	3	PB6	D10	SPI1_CS
	2	PC7	D9	-
	1	PA9	D8	-
CN9 digital	8	PA8	D7	-
	7	PB10	D6	-
	6	PB4	D5	-
	5	PB5	D4	-
	4	PB3	D3	-
	3	PA10	D2	-
	2	PA2	D1	USART2_TX
	1	PA3	D0	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Warning: PWM is not supported by D3, D5, D6, D9 and D10 on STM32F410RB since timer is not available on PB6, PC7, PB10, PB4, PB3.

Table 21. Arduino connectors on NUCLEO-L152RE

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC_IN0
	2	A1	PA1	ADC_IN1
	3	A2	PA4	ADC_IN4
	4	A3	PB0	ADC_IN8
	5	A4	PC1 or PB9(1)	ADC_IN11 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC_IN10 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM11_CH1 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-

Table 21. Arduino connectors on NUCLEO-L152RE (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN9 digital	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Table 22. Arduino connectors on NUCLEO-L452RE

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC1_IN5
	2	A1	PA1	ADC1_IN6
	3	A2	PA4	ADC1_IN9
	4	A3	PB0	ADC1_IN15
	5	A4	PC1 or PB9(1)	ADC1_IN2 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC1_IN1 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM1_CH1N or SPI1_MOSI
	3	D10	PB6	TIM16_CH1N or SPI1_CS
	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-

Table 22. Arduino connectors on NUCLEO-L452RE (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN9 digital	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

Table 23. Arduino connectors on NUCLEO-L476RG

Connector	Pin	Pin name	STM32 pin	Function
Left connectors				
CN6 power	1	NC	-	-
	2	IOREF	-	3.3V Ref
	3	RESET	NRST	RESET
	4	+3.3V	-	3.3V input/output
	5	+5V	-	5V output
	6	GND	-	ground
	7	GND	-	ground
	8	VIN	-	Power input
CN8 analog	1	A0	PA0	ADC12_IN5
	2	A1	PA1	ADC12_IN6
	3	A2	PA4	ADC12_IN9
	4	A3	PB0	ADC12_IN15
	5	A4	PC1 or PB9(1)	ADC123_IN2 (PC1) or I2C1_SDA (PB9)
	6	A5	PC0 or PB8(1)	ADC123_IN1 (PC0) or I2C1_SCL (PB8)
Right connectors				
CN5 digital	10	D15	PB8	I2C1_SCL
	9	D14	PB9	I2C1_SDA
	8	AREF	-	AVDD
	7	GND	-	ground
	6	D13	PA5	SPI1_SCK
	5	D12	PA6	SPI1_MISO
	4	D11	PA7	TIM17_CH1 or SPI1_MOSI
	3	D10	PB6	TIM4_CH1 or SPI1_CS

Table 23. Arduino connectors on NUCLEO-L476RG (continued)

Connector	Pin	Pin name	STM32 pin	Function
CN5 digital	2	D9	PC7	TIM3_CH2
	1	D8	PA9	-
CN9 digital	8	D7	PA8	-
	7	D6	PB10	TIM2_CH3
	6	D5	PB4	TIM3_CH1
	5	D4	PB5	-
	4	D3	PB3	TIM2_CH2
	3	D2	PA10	-
	2	D1	PA2	USART2_TX
	1	D0	PA3	USART2_RX

1. Refer to *Table 10: Solder bridges* for details.

6.12 ST morpho connector

The ST morpho connector consists in male pin headers (CN7 and CN10) accessible on both sides of the board. They can be used to connect the STM32 Nucleo board to an extension board or a prototype/wrapping board placed on top or on bottom side of the STM32 Nucleo board. All signals and power pins of the STM32 are available on ST morpho connector. This connector can also be probed by an oscilloscope, logical analyzer or voltmeter.

Table 24 to *Table 33* show the pin assignments of each STM32 on ST morpho connector.

Table 24. ST morpho connector on NUCLEO-F030R8

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	PF6	-	10	9	GND	-	10
11	PF7	IOREF	12	11	PA5	PA12	12
13	PA13	RESET	14	13	PA6	PA11	14
15	PA14	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13(3)	VIN	24	23	PA8	PB1	24
25	PC14(3)	-	26	25	PB10	PB15	26

Table 24. ST morpho connector on NUCLEO-F030R8 (continued)

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
27	PC15	PA0	28	27	PB4	PB14	28
29	PF0	PA1	30	29	PB5	PB13	30
31	PF1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	PF5	36
37	PC3	PC0 or PB8(4)	38	37	PA3	PF4	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Table 25. ST morpho connector on NUCLEO-F070RB

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PF0	PA1	30	29	PB5	PB13	30
31	PF1	PA4	32	31	PB3	AGND	32
33	VDD	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Table 26. ST morpho connector on NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F303RE, NUCLEO-F334R8

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)(2)	GND	8	7	AVDD	U5V(3)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(4)	RESET	14	13	PA6	PA11	14
15	PA14(4)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PF0	PA1	30	29	PB5	PB13	30
31	PF1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(5)	36	35	PA2	-	36
37	PC3	PC0 or PB8(5)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. CN7 pin 7 (BOOT0) can be configured by engineering byte as PF11 on NUCLEO-F091RC.
3. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
4. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommended to use them as IO pins if ST-LINK part is not cut.
5. Refer to *Table 10: Solder bridges* for details.

Table 27. ST morpho connector on NUCLEO-F103RB

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6

Table 27. ST morpho connector on NUCLEO-F103RB (continued)

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	**Name**	**Name**	**Pin**	**Pin**	**Name**	**Name**	**Pin**
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PD0	PA1	30	29	PB5	PB13	30
31	PD1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. The default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5 V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommended to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Table 28. ST morpho connector on NUCLEO-F302R8

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PB13	PA12	12
13	PA13(3)	RESET	14	13	PB14	PA11	14
15	PA14(3)	+3.3V	16	15	PB15	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PA7	26
27	PC15	PA0	28	27	PB4	PA6	28
29	PF0	PA1	30	29	PB5	PA5	30
31	PF1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Table 29. ST morpho connector on NUCLEO-F401RE, NUCLEO-F411RE, NUCLEO-F446RE

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	-	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Table 30. ST morpho connector on NUCLEO-L010RB, NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VLCD	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.Two unused jumpers are available on CN11 and CN12 (bottom side of the board).
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Table 31. ST morpho connector on NUCLEO-L452RE

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	**Name**	**Name**	**Pin**	**Pin**	**Name**	**Name**	**Pin**
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	PH3 / BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pins 5-7 of CN7.
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Table 32. ST morpho connector on NUCLEO-L476RG

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	**Name**	**Name**	**Pin**	**Pin**	**Name**	**Name**	**Pin**
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	PB11	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.
2. U5V is 5 V power from ST-LINK/V2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Table 33. ST morpho connector on NUCLEO-F410RB

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin	Name	Name	Pin	Pin	Name	Name	Pin
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PB11	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0(1)	GND	8	7	AVDD	U5V(2)	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13(3)	RESET	14	13	PA6	PA11	14
15	PA14(3)	+3.3V	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	-	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9(4)	36	35	PA2	-	36
37	PC3	PC0 or PB8(4)	38	37	PA3	-	38

1. Default state of BOOT0 is 0. It can be set to 1 when a jumper is on pin5-7 of CN7.
2. U5V is 5V power from ST-LINKV2-1 USB connector and it rises before +5V.
3. PA13 and PA14 share with SWD signals connected to ST-LINK/V2-1, it is not recommend to use them as IO pins if ST-LINK part is not cut.
4. Refer to *Table 10: Solder bridges* for details.

Appendix A Electrical schematics

Figure 27 to *Figure 30* show the electrical schematics of the STM32 Nucleo-64 board.

Figure 27. Top and Power

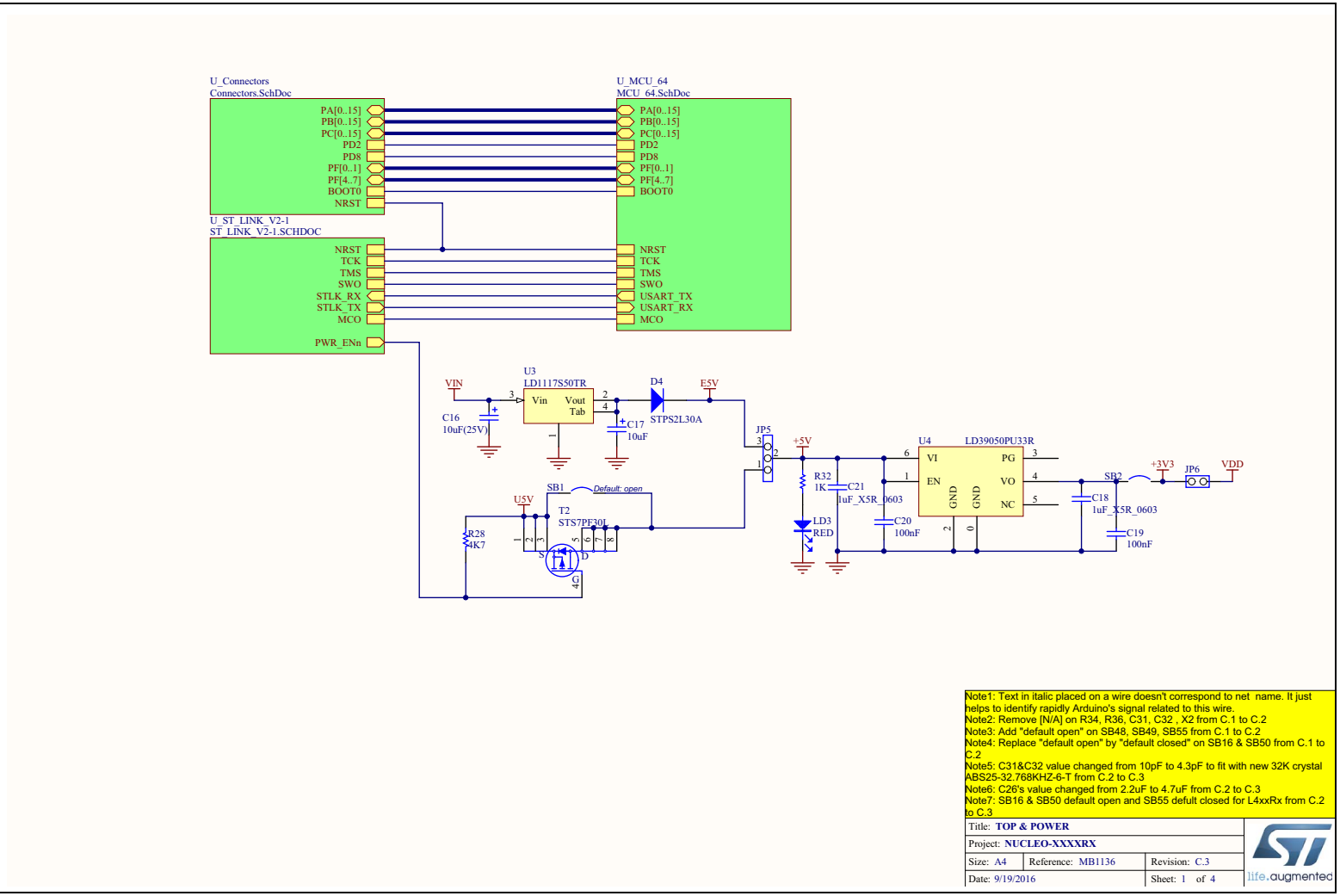


Figure 28. STM32 MCU

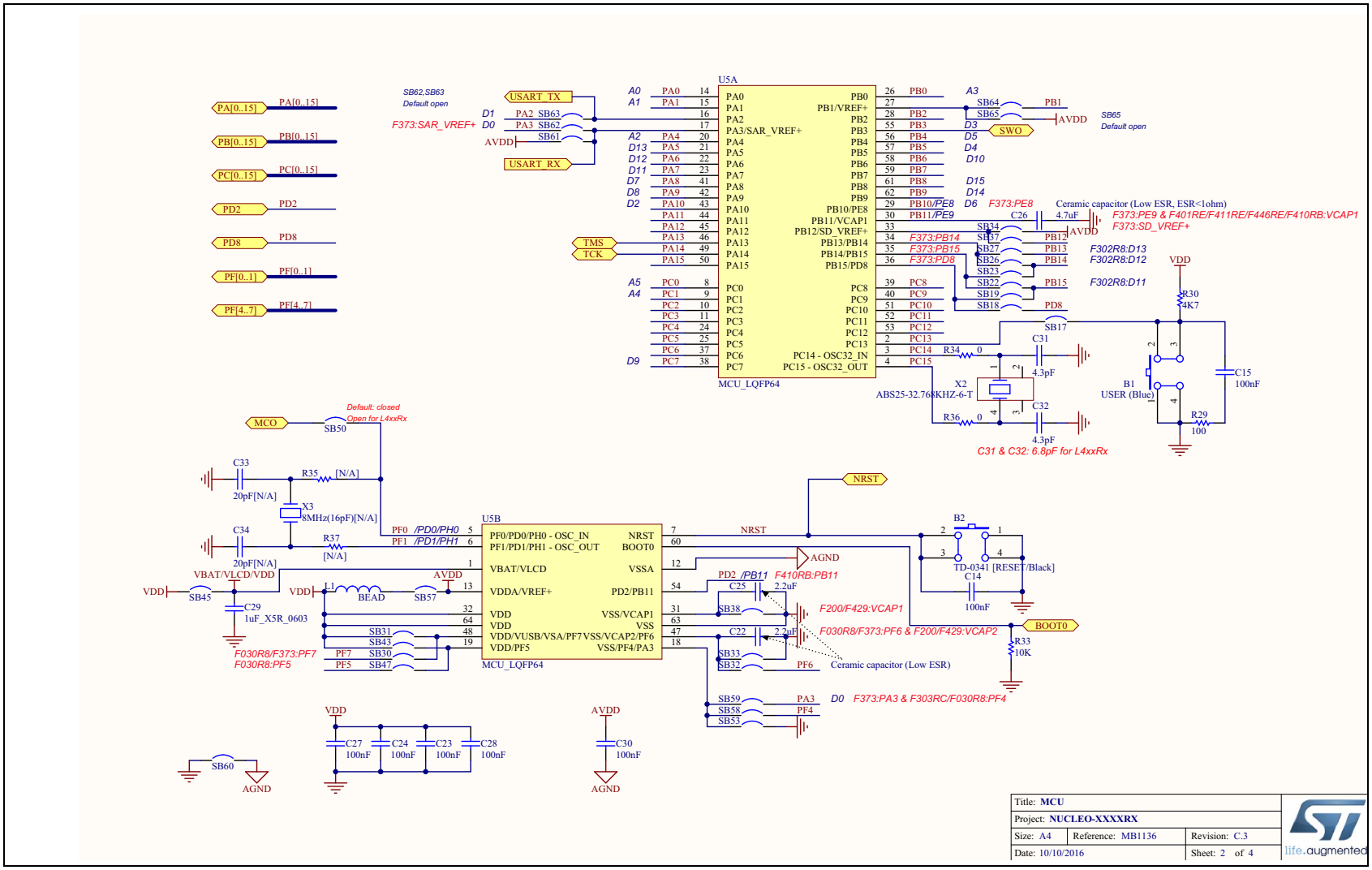


Figure 29. ST-LINK/V2-1

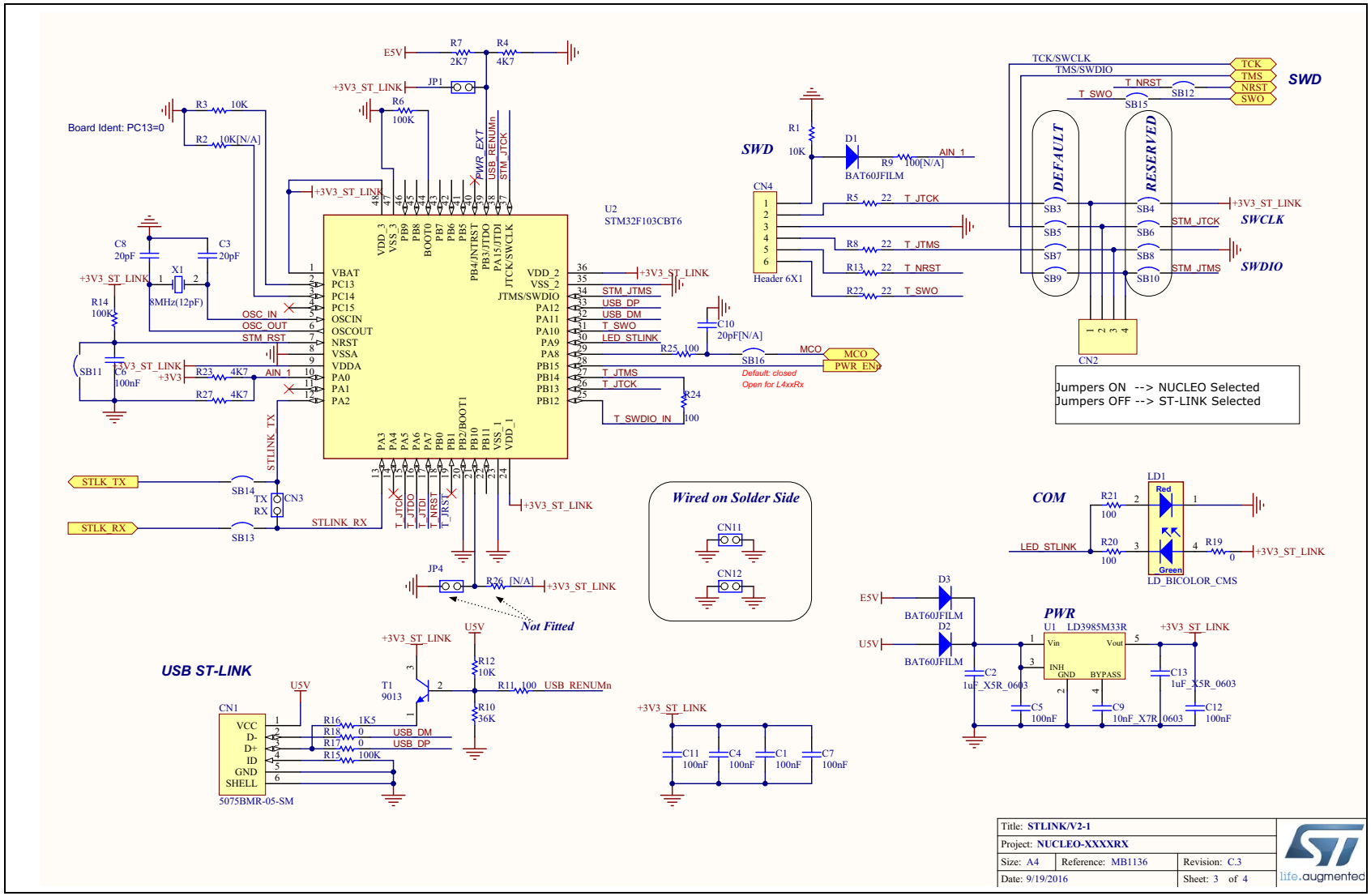
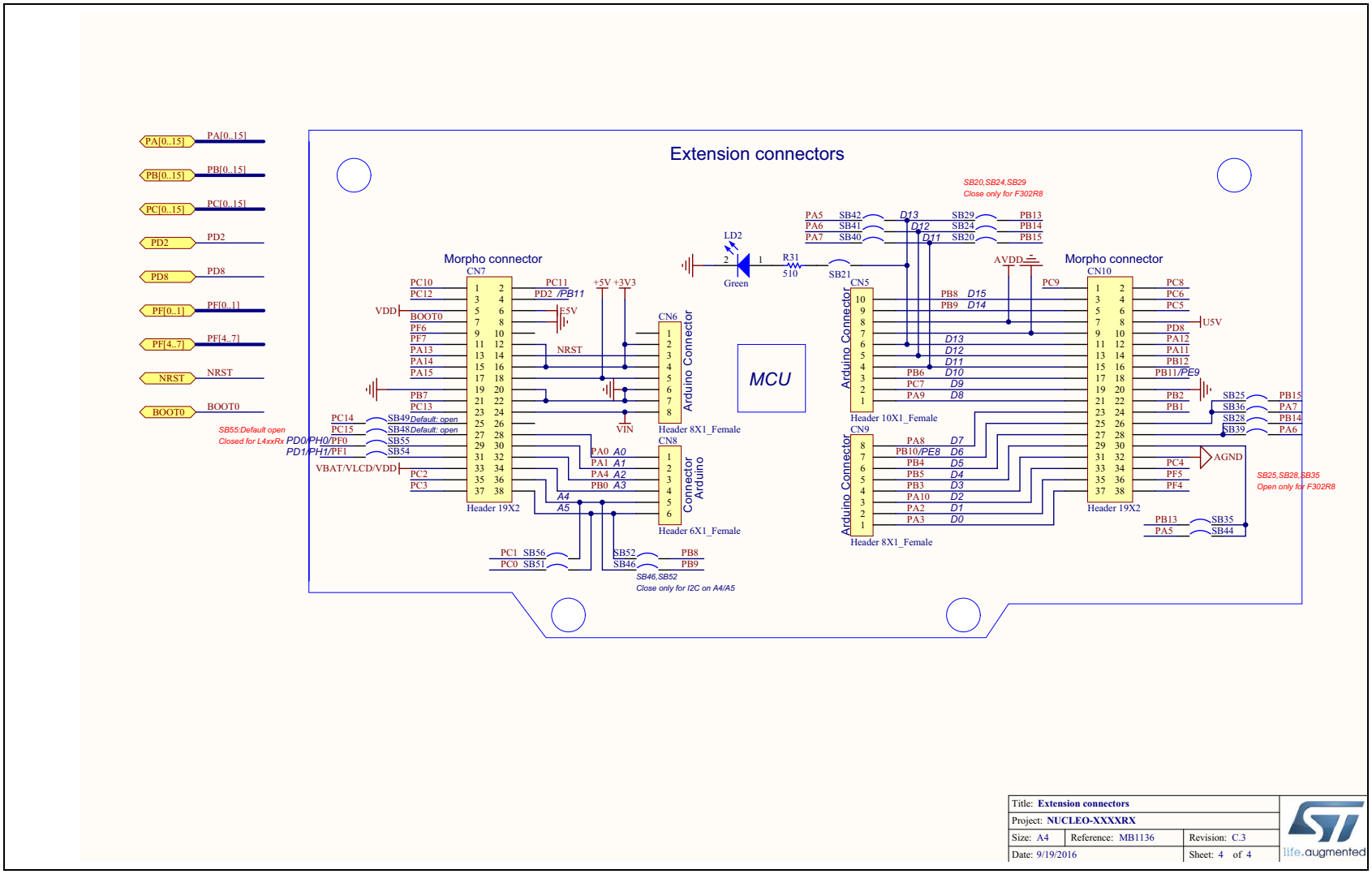


Figure 30. Extension connectors



Revision history

Table 34. Document revision history

Date	Revision	Changes
10-Feb-2014	1	Initial release.
13-Feb-2014	2	Updated *Figure 1*, *Chapter 5.5* and *Table 10*.
11-Apr-2014	3	Extended the applicability to NUCLEO-F302R8. Updated *Table 1: Ordering information*, *Section 6.11: Arduino connectors* and *Section 6.12: ST morpho connector*. Updated *Figure 1*
10-June-2014	4	Updated the board figure: *Figure 1*. Updated HSE and LSE configuration description: *Section 6.7.1*, *Section 5.5* and *Section 6.7.2*. Extended the applicability to NUCLEO-F334R8, NUCLEO-F411RE and NUCLEO-L053R8.
20-June-2014	5	Updated the electrical schematics figures: *Figure 27*, *Figure 28*, *Figure 29* and *Figure 30*. Refer to the AN2867 for oscillator design guide for STM32 microcontrollers in *Section 6.7.1: OSC clock supply* and *Section 6.7.2: OSC 32 KHz clock supply*.
30-Sept-2014	6	Extended the applicability to NUCLEO-F091RC and NUCLEO-F303RE; Updated *Table 1: Ordering information*; Updated *Table 11: Arduino connectors on NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC*; Updated *Table 26: ST morpho connector on NUCLEO-F072RB, NUCLEO-F091RC, NUCLEO-F303RE, NUCLEO-F334R8*; Updated *Figure 6: Typical configuration*; Added *Figure 13: NUCLEO-F091RC*; Added *Figure 16: NUCLEO-F303RE*; Updated *Section 6.7.2: OSC 32 KHz clock supply*; Updated *Figure 27: Top and Power(1/4)* ,*Figure 28: STM32 MCU*;

Table 34. Document revision history (continued)

Date	Revision	Changes
19-Jan-2015	7	Extended the applicability to NUCLEO-F070RB, NUCLEO-L073RZ and NUCLEO-L476RG; Updated *Table 1: Ordering information*; Updated *Section 6.2: Embedded ST-LINK/V2-1*; Updated *Section 6.7.1: OSC clock supply*; Added *Figure 11: NUCLEO-F070RB*; Added *Figure 21: NUCLEO-L073RZ*; Added *Figure 24: NUCLEO-L476RG* Updated *Table 11: Arduino connectors on NUCLEO-F030R8, NUCLEO-F070RB, NUCLEO-F072RB, NUCLEO-F091RC* Added *Table 18: Arduino connectors on NUCLEO-L073RZ* Added *Table 23: Arduino connectors on NUCLEO-L476RG* Added *Table 25: ST morpho connector on NUCLEO-F070RB* Updated *Table 30: ST morpho connector on NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE* Added *Table 32: ST morpho connector on NUCLEO-L476RG* Updated schematics from *Figure 27: Top and Power(1/4)* to *Figure 30: Extension connectors*
08-Jul-2015	8	Extended the applicability to Updated *Table 1: Ordering information*; Added *Figure 25: NUCLEO-F446RE* and *Figure 26: NUCLEO-F410RB* Updated *Section 6.11: Arduino connectors on page 37* and *Section 6.12: ST morpho connector on page 53*
04-Aug-2015	9	Added *Section 5.4: NUCLEO-L476RG bootloader limitations*.
17-Nov-2015	10	Updated *Section 6.9: Solder bridges* and *Section 6.7.1: OSC clock supply*.
29-Nov-2015	11	Updated *Introduction*, *Section 3: Ordering information*, *Section 6.10: Extension connectors*, *Section 6.11: Arduino connectors*, *Section 6.12: ST morpho connector* to add NUCLEO-L452RE.
15-Dec-2017	12	Updated document title and cover page. Updated *Chapter 2: Product marking* and *Section 5.3: Development toolchains*. Expanded document scope to NUCLEO-L010RB: – Updated *Table 1: Ordering information* – Updated *Table 18: Arduino connectors on NUCLEO-L010RB and NUCLEO-L073RZ* – Updated *Table 21: NUCLEO-L073RZ and NUCLEO-L010RB* – Updated *Table 30: ST morpho connector on NUCLEO-L010RB, NUCLEO-L053R8, NUCLEO-L073RZ, NUCLEO-L152RE*
3-Apr-2019	13	Updated document title, *Introduction*, *Chapter 2: Ordering information*, *Section 2.1: Product marking*, *Section 2.2: Codification*, and *Section 5.1: Getting started*. Added *Chapter 3: Development environment* and *Section 3.3: Demonstration software*.