Universidad de La Laguna

ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

GRADO EN INGENIERÍA ELECTRÓNICA INDUTRIAL Y AUTOMÁTICA

TRABAJO DE FIN DE GRADO:

ANÁLISIS COMPARATIVO DE TECNOLOGÍAS Y PROTOCOLOS DE COMUNICACIÓN DE SENSORES PARA INTEGRACIÓN EN EDIFICIO DE OFICINAS GESTIONADO POR SOFTWARE DE MONITORIZACIÓN Y CONTROL DE CONFORT, EFICIENCIA ENERGÉTICA Y SALUD

AUTOR: IVÁN RODRÍGUEZ PÉREZ

TUTOR: RICARDO MESA CRUZ

Agradecimientos

En primer lugar, tanto a mi tutor, Ricardo Mesa, como al gerente del Cluster de Construcción Sostenible, Diego Broock, por todo lo que me han ayudado en este proyecto y la confianza que han depositado en mí en todo momento.

A toda mi familia, especialmente mis padres, Pablo y Orlanda, y mi hermana, Marta, que siempre me han apoyado en mis decisiones y me han respaldado en estos cuatro años, sacrificándose para darme todo lo que he necesitado, hacen que cualquier esfuerzo valga la pena.

A mis padrinos, Douglas y Karina, que nunca les ha faltado tiempo para interesarse por mi progresión académica dándome su apoyo incondicional.

A mis compañeros de carrera, especialmente a mi grupo, con los que hemos pasado infinidad de momentos buenos y malos, pero siempre hemos salido adelante.

Finalmente, a mis compañeros de baloncesto durante estos cuatro años, principal forma de desconexión, hacen que todo sea mucho más fácil.

ÍNDICE

ÍNDICE DE ANEXOS

ÍNDICE DE ILUSTRACIONES

Resumen

En el presente proyecto llevaremos a cabo un modelo de Smart Building, seleccionando un conjunto de sensores para analizar diversas variables relacionadas con el confort y la salud, y monitorizar todos los resultados que obtengamos empleando un software denominado CONEFI.

Para desarrollar esta prueba piloto, en primer lugar, llevaremos a cabo un análisis de estrategias para elegir el modelo de instalación más económico para nuestro modelo *low-cost*, seleccionando además el método de recogida de datos más eficiente para cumplir con nuestros requisitos.

Estableceremos un razonamiento comparativo de los diferentes sensores, dispositivos hardware y elementos seleccionados para nuestro proyecto, incluyendo un presupuesto final del proyecto que justifique el gasto total en concepto de material que emplearemos.

Todos los sensores que instalaremos nos indicarán las mediciones en nivel de tensión, por lo que tendremos que programarlos para que indiquen los valores en las unidades correctas y presenten una fiabilidad aceptable para esta prueba. También es preciso configurar adecuadamente los dispositivos de conexión inalámbrica para que funcionen conforme a nuestros requerimientos.

Finalmente, realizaremos las tareas necesarias para establecer la comunicación del dispositivo Hardware con el Software, de forma que podamos visualizar los resultados empleando la herramienta gráfica de CONEFI.

Abstract

In this project, we are going to design a Smart Building model, selecting a set of sensors in order to analyze different variables related to the comfort and healthy, monitoring the results obtained using CONEFI software, which is created by the company.

For developing this pilot test, first, we are going to analyze the strategies for select the most economic installation for our low-cost model, selecting the most efficient method of data collection for meeting the requirements.

We are going to comment the reasoning employed in the selection of all sensors, devices, hardware and other elements selected for our project, including a material budged at the end of this reasoning, just for justifying with numbers, the arguments explained before.

All sensors will give us some information in tension level, so we have to program them to indicate the values given in the correct units, showing us high level of reliability. Moreover, it is important to configure the wireless devices for our requirements, in fact, those elements represent an important part in the monitoring because most of the values come from long distances and the data are given using these devices.

Finally, we are going to work on the communication of the Hardware device with the Software, letting us to visualize the results obtained employing the graphic tool named CONEFI.

CAPÍTULO 1. INTRODUCCIÓN

1. Introducción

1.1. Antecedentes

La estructura energética española se caracteriza principalmente por una escasa explotación de las energías renovables, siendo uno de los países con mayor dependencia energética de Europa.

En 2011 se aprueba el 'Plan de Acción de Ahorro y Eficiencia Energética 2011-2020', con el objetivo de reducir considerablemente la dependencia energética existente en España, haciendo hincapié en las energías renovables. En este sentido, resulta determinante emplear soluciones que sean rentables, aprovechando al máximo las condiciones que presenta cada zona, por ejemplo, en un lugar donde prácticamente no incide la luz solar a lo largo del año no nos interesa invertir en recursos para su explotación, debido a que no proporcionaría beneficio. En Canarias, la mayor demanda energética se produce en el sector predominante, el sector terciario, en el que se invierten pocos recursos energéticos, pese al enorme potencial energético que presentan las islas.

La forma en que es analizada la eficiencia de un edificio es diferente en edificios de nueva planta y en los edificios existentes:

- **Edificios de nueva planta.** Los inconvenientes se reducen, al tratarse de edificios diseñados con vistas al ahorro energético, incluyendo una instalación preparada para la inclusión de cualquier solución energética.
    - **Simulación.** Es la estrategia empleada en edificios de nueva planta. Con la simulación podemos predecir el comportamiento del edificio, analizando los parámetros relacionados con la eficiencia, para adoptar las medidas oportunas para que el edificio alcance un nivel de consumo "casi nulo". Para que la simulación sea lo más fiable posible, la posterior construcción debe seguir estrictamente lo estipulado en la simulación.
- **Edificios existentes.** Este es el caso predominante en las Islas Canarias. Los edificios existentes, presentan numerosos inconvenientes que dificultan la optimización del ahorro energético. No obstante, realizar las operaciones oportunas para que estos edificios se conviertan en eficientes es la solución más sostenible, por encima de la

realización de nuevas obras, que terminan ocasionando un mayor gasto y consumo de recursos.

- **Monitorización.** Al tratarse de edificios existentes, no podemos predecir el comportamiento previo a su construcción, no obstante, lo que sí podemos hacer es estudiar las condiciones que presentan. La monitorización consiste en estudiar ciertas variables, empleando principalmente sensores, para registrar los resultados de forma que podamos compararlos con los valores de referencia, empleando gráficos de comportamiento, para adoptar en función de los resultados obtenidos, las medidas que se consideren oportunas.

Centrándonos en el marco de los edificios existentes, que es el modelo más común en las Islas Canarias, vamos a analizar las variables a monitorizar. Las variables más importantes para tener en cuenta a la hora de estudiar las prestaciones de un edificio existente se pueden observar en la tabla 1:

Tabla 1. Variables de confort, consumo y salud.

Variables		
Confort	Confort Térmico	Estudio de la temperatura, humedad o viento
	Confort Acústico	Estudio del ruido interior y exterior
	Confort Lumínico	Estudio de los parámetros de iluminación. Norma UNE 12464-1 “Iluminación de los lugares de trabajo en interior”.
Consumo	Consumo de energía	Principal parámetro cuando queremos estudiar la eficiencia energética
	Consumo de Agua	Estudio del consumo del agua, fugas o la presión del agua para su reutilización
Salud	Calidad del aire	Principal objeto de estudio, variable directamente relacionada con la salud. Concentración de CO2, benceno o COV
	Calidad del agua	Estudio de agua de abastecimiento y aguas residuales, existiendo una normativa que regula unos mínimos permitidos
	Radiación electromagnética	Estudio de los campos eléctricos y magnéticos en busca de posibles radiaciones dañinas

Con la monitorización o a simulación terminada, es hora de aportar soluciones a los resultados obtenidos, con el objetivo de mejorar la instalación en lo que a sostenibilidad se refiere. Estas estrategias pueden estar relacionadas con el diseño arquitectónico, definiendo el equipamiento necesario para adaptarse al clima cambiante a lo largo del año, o bien con estrategias relacionadas con la ingeniería, es decir, soluciones en materia de ahorro de energía en lo que se refiere a iluminación, temperatura o consumo de agua y electricidad.

1.2. Base del proyecto

Como consecuencia de los avances y nuevas vías de desarrollo en el campo de la ingeniería, en los últimos años ha cobrado fuerza el concepto "Internet of Things" o Internet de las cosas, que consiste en la interacción de cualquier elemento físico (por ejemplo, una vivienda), con internet, empleando sensores, actuadores, software, etc.

En este contexto surgen también los conceptos de "Smart Building" o Construcción Inteligente y "Smart City" o Ciudad Inteligente, que consisten en magnificar el concepto de "Internet of Things" a la totalidad de una infraestructura o incluso una ciudad entera. El objetivo que persiguen las Smart Cities es la sostenibilidad, fomentando el empleo de soluciones tan eficientes como el uso de coches eléctricos, bicicletas, paneles solares o parques eólicos. Otra medida eficiente consiste en controlar las variables relacionadas con el confort y la salud, por ejemplo, la temperatura y la humedad, mediante la instalación de sensores habilitados para ello, con el fin de conseguir incrementar el bienestar de las personas.

Dado el alto impulso que comienza a existir por parte de algunas empresas en este ámbito, es que se ha decidido desarrollar una estrategia propia de construcción inteligente basada en un modelo de bajo coste, surgiendo de este modo el presente proyecto.

1.3. Objetivos

En este apartado vamos a comentar los objetivos del proyecto, que consisten en llevar a cabo la monitorización de una de las cinco plantas de un edificio de oficinas emplazado en la localidad de Santa Cruz de Tenerife, obteniendo datos relacionados principalmente con la salud y el confort, realizando previamente, un análisis de las estrategias a desarrollar. Con esta

prueba piloto trataremos de ejemplificar el buen funcionamiento que puede aportar un sistema de monitorización de bajo coste en tiempo real, desarrollando así el concepto de Smart Building.

Para hacer efectiva la monitorización, se instalarán sensores tanto en el interior como en el exterior del edificio, con el fin de comparar algunos parámetros que sea de interés medir en ambos lugares, como es el caso de la temperatura o la humedad. Los datos recogidos por los sensores son enviados a un software proporcionado por la empresa, el cual debe estar conectado continuamente a la red durante el tiempo de monitorización para la recogida de todos los datos que se vayan obteniendo.

CAPÍTULO 2. CONOCIMIENTOS FUNDAMENTALES

2. Conocimientos fundamentales

2.1. Sensores

Los sensores son dispositivos que se encargan de medir una determinada magnitud, ofreciéndonos como dato un valor de dicha magnitud o una medida que tiene relación directa con la misma y que debemos obtener mediante programación. De esta forma, podemos distinguir entre dos tipos de sensores.

- **Sensor digital.** Puede tomar hasta 2 valores a la salida, estos son 0 y 1, los estados son absolutos, es decir todo o nada. Se suelen emplear para verificar cuando una magnitud sobrepasa un valor determinado que nosotros podemos fijar previamente. En este caso en particular el resultado no guarda relación directa con la magnitud que se mide, es decir, no conocemos el valor de la magnitud, solamente sabemos si sobrepasa el valor fijado.

- **Sensor analógico.** Puede tomar multitud de valores comprendidos en un intervalo definido por 2 valores de referencia, generalmente tierra y V_{CC}. En este caso, los resultados que obtenemos sí son proporcionales a lo que se está midiendo.

Cuando manejamos sensores analógicos, el valor de salida que obtenemos suele ser una medida que guarda algún tipo de relación con la magnitud que deseamos obtener. En algunos casos encontramos sensores que poseen internamente un microprocesador, una memoria y un controlador dentro del mismo, que se encargan de ofrecernos un valor en las magnitudes correctas. Estos sensores unen la función de detectar magnitudes físicas (sensor analógico) y las de procesamiento de la señal (tarea que ejerce comúnmente un microcontrolador), todo ello en el mismo sensor. Este tipo de dispositivos son los sensores inteligentes.

2.1.1. Parámetros que intervienen

Cuando buscamos un sensor, antes que nada, es necesario informarnos acerca de las características que nos interesan, para ello es preciso acudir al datasheet del sensor. En este

sentido, los parámetros más determinantes a la hora de elegir un sensor u otro son los siguientes:

- **Magnitudes.** Margen de medida, resolución, exactitud, estabilidad y tiempo de respuesta.
- **Características de salida.** Sensibilidad, tipo (tensión, corriente), forma de la señal y destino.
- **Características de alimentación.** Tensión, corriente, potencia y frecuencia.
- **Características ambientales.** Margen de temperaturas, humedad, agentes químicos y la posible existencia de atmosferas explosivas.

No obstante, podemos tener en cuenta otros factores que puedan influir en nuestro proyecto por temas relacionados con las condiciones que presente el área de trabajo. Algunas características significativas en este sentido son el peso, las dimensiones y el precio en lo que se refiere al propio sensor, o la longitud del cable, tipo de conectores y costes de verificación, mantenimiento y sustitución en lo que se refiere a las condiciones y material disponible.

2.1.2. Clasificación

Podemos clasificar los sensores de la siguiente manera:

1. **Aporte de energía**

Moduladores o activos. Emplean una fuente de energía auxiliar (termistor).

Generadores o pasivos. La energía de la señal de salida se suministra por la entrada (termopar).

2. **Tipo de salida**

Analógica. Puede adoptar un rango de valores (potenciómetro).

Digital. Solo toma dos valores, encendido o apagado (Codificador de posición).

3. **Tipo de variable a medir**

Los sensores también se pueden clasificar según la variable que deseemos medir. A continuación, vamos a enumerar todas aquellas variables que vamos a automatizar, considerándolas indispensables para la automatización del edificio en concreto, todas ellas relacionadas con el confort y la salud. Emplearemos sensores para medir los siguientes parámetros:

Para el confort térmico:

- Temperatura interior y exterior
- Humedad interior y exterior

Para el confort lumínico:

- Iluminancia

Para la calidad del aire:

- Concentración de CO_2
- Concentración de benceno
- Concentración de tolueno

2.2. Arduino

Arduino es una compañía dedicada a la creación y desarrollo de microcontroladores, así como de los softwares necesarios para su desarrollo. Al tratarse de un hardware libre, todas las referencias, esquemas de conexión y códigos son de acceso público.

El hardware que desarrolla Arduino son los microcontroladores, disponiendo de una amplia gama de dispositivos que ofrecen diferentes características según las necesidades del usuario en cuanto a tamaño o número de pines, entre otros.

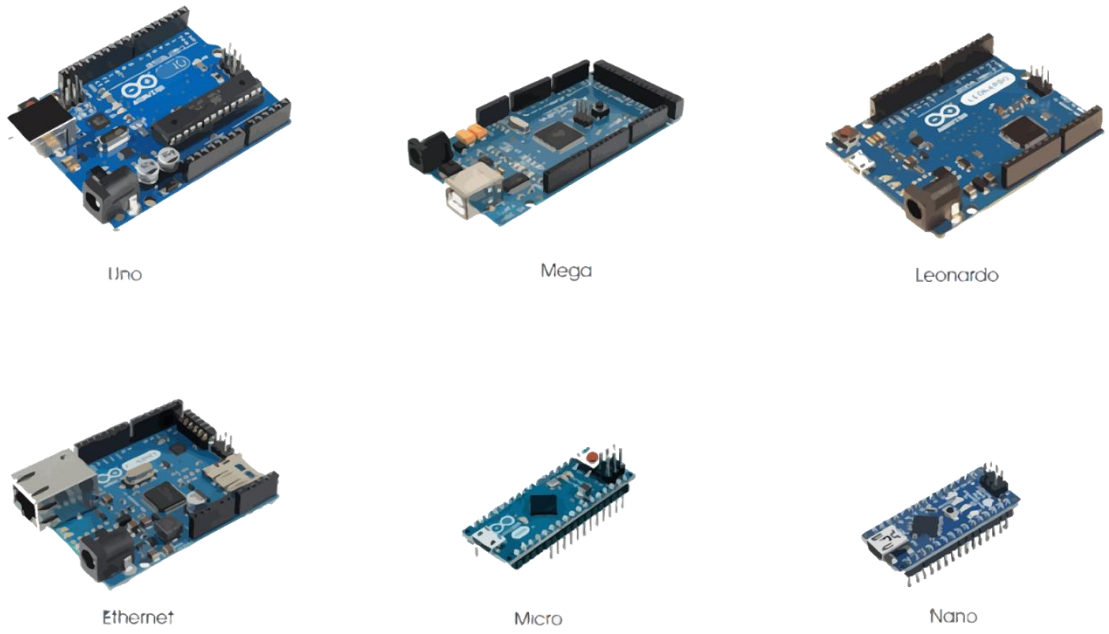


Ilustración 1. Modelos más conocidos de Arduino.

Arduino también trabaja dispositivos o placas que se acoplan a los microcontroladores para mejorar el funcionamiento de los mismos, como es el caso de las Ethernet Shield o los módulos XBee.

2.2.1. Conversión Analógico-Digital

Cuando los datos salen del sensor analógico, se realizan una serie de modificaciones en la señal con el fin de obtener las variables que queremos en las unidades correctas. La conversión de los datos puede ser realizada por el mismo sensor, cuando se trata de sensores inteligentes, pero que requieren igualmente de programación, o empleando un único microcontrolador, que es quien realiza la conversión de datos cuando empleamos varios sensores analógicos conectados al mismo. En este caso vamos a emplear un microcontrolador Arduino para controlar la conversión de los datos de todos los sensores que queremos implementar.

Las señales, normalmente en forma de tensión, se pueden conectar a las entradas analógicas, digitales PWM o las analógicas de bus de un microcontrolador Arduino para poder medir la variable del sensor, y mediante programación obtener el equivalente de esa medida en las unidades que deseamos conseguir. En lo que se refiere a la tensión de referencia V_{CC}, se suelen

emplear 5 o 3,3 voltios, puertos de tensión disponibles en la propia placa de Arduino. En la ilustración 2 podemos observar como existen dos tensiones de referencia y es únicamente la tensión del sensor la que puede variar. V_{OUT} es la señal que llega al microcontrolador.

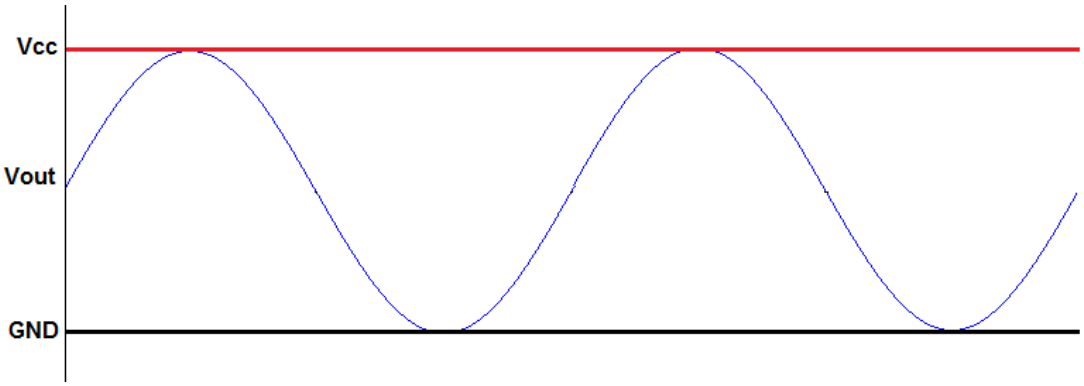


Ilustración 2: Gráfico de tensiones V_{CC}, V_{OUT}, GND.

No obstante, la señal V_{OUT} que acabamos de ver que sale del sensor, cuando es de tipo analógico se encuentra con un problema al llegar al puerto de Arduino, porque los microcontroladores solo entienden señales digitales, por lo que es necesario emplear un Convertidor Analógico-Digital (CAD) o en inglés, Analog-to-Digital Converter (ADC) para convertir una señal analógica en una digital que se encuentre dentro de un rango de valores determinados. En los microcontroladores suelen emplearse convertidores de pequeña señal que suelen estar integrados en los mismos debido a las aplicaciones que suelen tener este tipo de dispositivos, no obstante, en caso de que el CAD resultara insuficiente, se podría implementar un convertidor externo que realice la conversión analógico-digital antes de la llegada de los datos al microcontrolador. En nuestro caso no necesitaremos implementar un convertidor externo, dado que el CAD que posee el microcontrolador es perfecto para nuestros requerimientos.

La conversión analógico-digital se divide en tres etapas: muestreo, cuantificación y codificación. En el caso de los microcontroladores se realizan principalmente dos de ellas, la cuantificación y la codificación, debido a que el proceso de muestreo no se realiza en todos los microcontroladores.

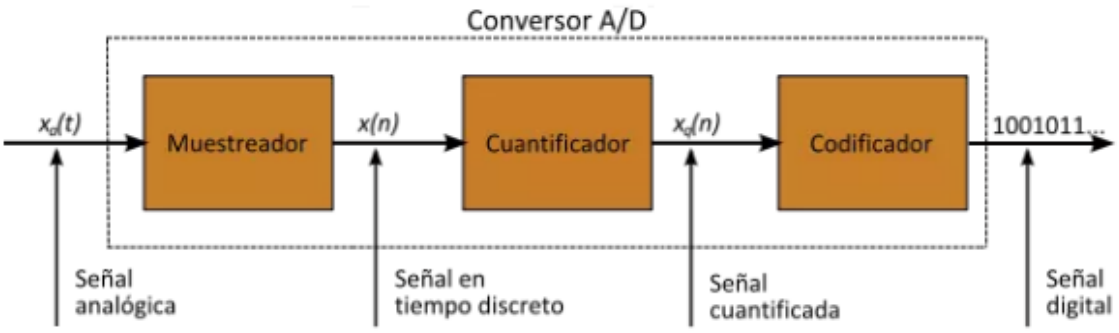


Ilustración 3: Estructura de un Conversor Analógico – Digital (CAD) (https://es.wikipedia.org/wiki/Conversor_de_se%C3%B1al_anal%C3%B3gica_a_digital).

1. Muestreo

Consiste en la obtención de una señal analógica continua en amplitud, pero discreta en tiempo, a partir de una señal analógica continua en tiempo y amplitud. Para que la reconstrucción de la señal sea efectiva, la frecuencia de muestreo debe ser al menos del doble de la frecuencia máxima de la señal de entrada, para asegurar que la reconstrucción de la señal original sea fiable.

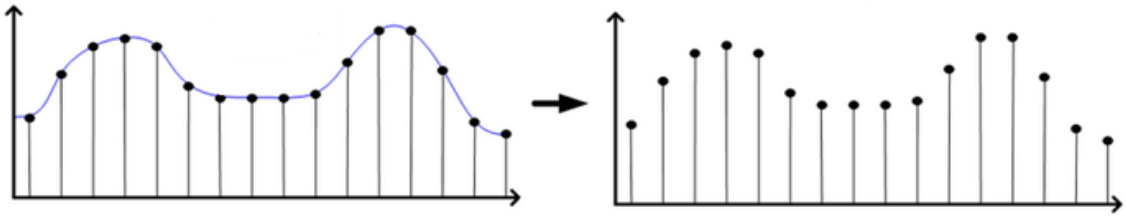


Ilustración 4: Muestreo en un CAD.

2. Cuantificación

Es la operación que consiste en la conversión de una señal de amplitud continua en otra de amplitud discreta, es decir, una señal con un número finito de valores o niveles de amplitud. En función del número de bits podemos obtener los niveles de salida, obteniendo así otros parámetros como el margen de entrada y la resolución del cuantificador. Emplearemos una cuantificación uniforme, donde la diferencia entre cada nivel de salida siempre es la misma, al igual que entre los niveles de decisión.

Para calcular la distancia entre dos niveles de decisión consecutivos se obtiene de la siguiente ecuación:

$$q = \frac{M}{2^n} = \frac{M}{N}$$

Donde:

- M es el margen de entrada (diferencia entre el valor mínimo y máximo en la entrada).
- n es el número de bits que tiene el conversor.
- q es la resolución del conversor (distancia entre dos niveles de decisión consecutivos).
- N es el número de niveles de salida.

Como es de esperar, en el proceso de cuantificación, se emplea el redondeo para poder ofrecer resultados lo más cercanos posible a la realidad. Esto produce la aparición de un error que tiene su valor máximo en los niveles de decisión de $\pm {}^{q}/_{2}$, es decir, la mitad de la resolución del CAD. Sin embargo, en el último intervalo de decisión se genera un error de -q. El error se puede expresar de la siguiente forma:

$$error = valor\ medido - valor\ real$$

También podemos obtener el error máximo que se puede producir de forma teórica y en valor absoluto:

$$|e| \leq \frac{q}{2} = \frac{M}{2^n \cdot n} = \frac{M}{2^{n+1}}$$

En la ilustración 5 se observa el ejemplo de la cuantificación uniforme de un CAD de tres bits. Como ya hemos comentado, cuando el CAD alcanza su nivel de salida más elevado el nivel seguirá ofreciendo el mismo valor a la salida pese a que el nivel analógico continúe subiendo.

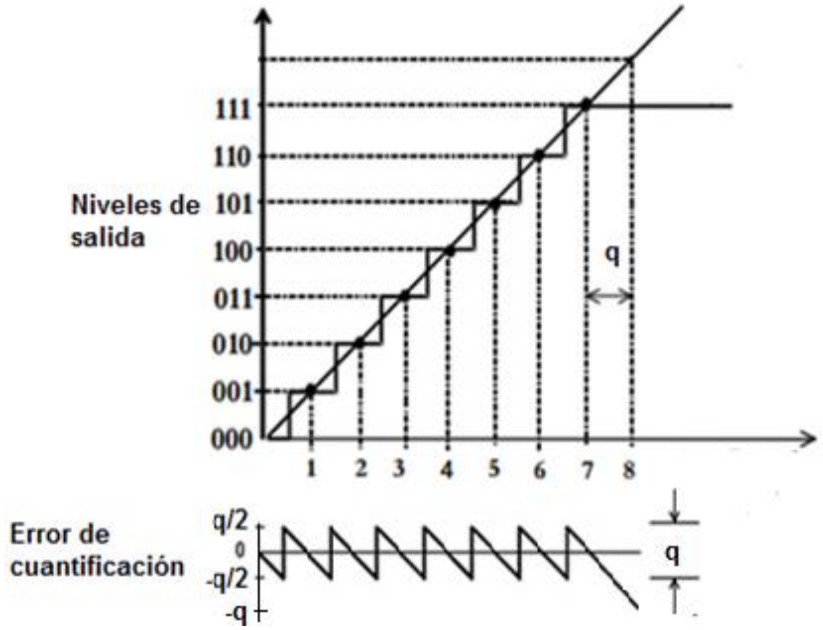


Ilustración 5: Cuantificación y error de un CAD de 3 bits.

En lo que se refiere al microcontrolador Arduino, solo reconoce valores digitales de 10 bits de datos, es decir, reconocen hasta un total de 1024 niveles, teniéndose una conversión analógico digital bastante precisa teniendo en cuenta los valores de tensión de referencia que suelen manejarse.

$$2^{10} = 1024 \text{ niveles (de 0 a 1023)}$$

Normalmente, se emplea una tensión de referencia de 5V, teniéndose así un margen de entrada de 5V. Si dividimos el margen de entrada entre el número de niveles de salida podemos deducir la variación de tensión que debe experimentar el sensor para que se reconozca en el microcontrolador, es decir, obtenemos la resolución del CAD.

$$q = \frac{5}{2^{10}} = \frac{5}{1024} = 5\ mV$$

El error máximo que se podrá cometer, expresado en valor absoluto, para el microcontrolador en cuestión debe ser:

$$|e| \leq \frac{5\ mV}{2} = \frac{5}{2^{10} \cdot 10} = \frac{5}{2^{10+1}} = \frac{5}{2048} = 2{,}5\ mV$$

3. Codificación

Es el proceso mediante el cual se asignan palabras a los valores de tensión analógica que ya han sido cuantificados, para traducirlos al código digital a cada uno de los niveles de amplitud discretos. Normalmente el código que se emplea es el binario, donde las posiciones de los dígitos expresan el nivel al que identifican, teniendo cada una de estas cifras un peso determinado como potencia de dos. Por ejemplo, el número 54 se expresa en binario de la siguiente manera:

$$54_{10} = (2^5 \cdot 1 + 2^4 \cdot 1 + 2^3 \cdot 0 + 2^2 \cdot 1 + 2^1 \cdot 1 + 2^0 \cdot 0)_2 = 110110$$

Como cabe esperar, no se pueden representar todos los valores del margen de entrada en la codificación, por ello es que debemos hablar del valor fondo de escala, que es el valor para el que la salida digital es máxima, de forma que obtendremos la tensión máxima y mínima que trabajará el CAD de forma exacta empleando los puntos críticos de la siguiente ecuación:

$$V_0 = \frac{V_{FE}}{2^n} \sum_{i=0}^{n-1} D_i 2^i = V_{FE} \sum_{i=0}^{n-1} \frac{D_i}{2^{n-i}}$$

En nuestro microcontrolador, el valor fondo de escala será de 5V, y el número de bits será 10, de forma que la tensión mínima a representar serán 0V, y la máxima será la que se expresa a continuación:

$$V_{0,máx.} = V_{FE} \sum_{i=0}^{n-1} \frac{D_i}{2^{n-i}} = V_{FE} \cdot \frac{2^n - 1}{2^n} = 5 \cdot \frac{2^{10} - 1}{2^{10}} = 4{,}995V$$

2.2.2. Bus I²C o IIC

Los circuitos interintegrados (Inter-Integrated Circuits) son un tipo de bus síncrono maestro-esclavo, esto quiere decir que el intercambio de información siempre se realiza cuando el maestro lo permite. Además, no se puede establecer una comunicación directa esclavo-esclavo, sino que tienen que comunicarse siempre con su maestro. Sin embargo, la comunicación maestro-maestro si es posible en un tipo de bus multimaestro.

El bus IIC emplea los pines SDA (Serial Data) y SCL (Serial Clock), de forma que se requieren pocos cables para su conexionado, y donde cada bus tiene una dirección exclusiva de 7 bits de datos. Para el bus IIC, Arduino UNO posee dos pares de pines SDA y SCL, y empleamos la librería Wire.h.

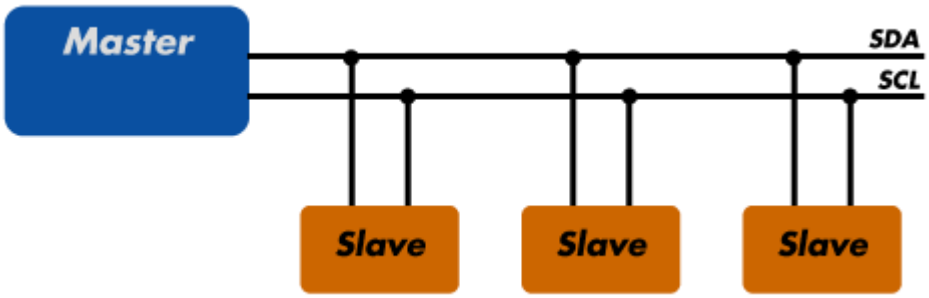


Ilustración 6. Esquema gráfico del tipo de Bus I²C (http://dignal.com/el-bus-i2c/).

2.2.3. Bus SPI

Otro de los buses que emplea Arduino es el Serial Peripheral Interface (SPI) que, al igual que el I²C, se trata de un tipo de bus síncrono maestro-esclavo que se emplea en comunicaciones. Por ejemplo, en nuestro caso realizaremos la comunicación del sensor (esclavo) con el Arduino (maestro). En este caso se emplean los siguientes pines reservados de Arduino: SCK (Serial Clock), MOSI (Master Out, Slave In), MISO (Master In, Slave Out).

El bus SPI posee una alta velocidad de transmisión, sin embargo, está diseñado para distancias muy reducidas.

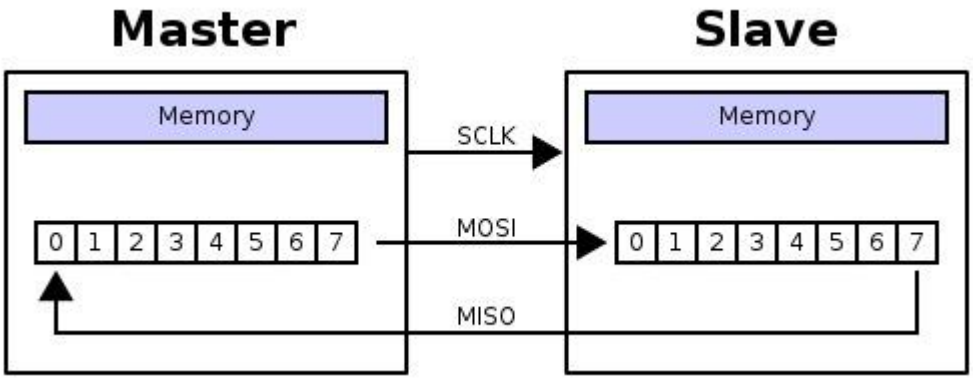


Ilustración 7. Esquema de comunicación de un bus SPI (http://panamahitek.com/como-funciona-el-protocolo-spi/).

2.2.4. Bus One-Wire

El bus OneWire, igual que los anteriores, emplea un tipo de bus maestro-esclavo, que permite el uso de cables más largos sin que se deteriore la señal, de ahí el nombre de este método de comunicación. Este bus fue creado por Dallas Semiconductor, empleándolo como librería OneWire.h junto con otra libraría, que también suele tener el identificador de esta compañía norteamericana, como vemos en la ilustración 10, donde se emplea la librería para un sensor de temperatura de gases y líquidos.

```
#include <OneWire.h>
#include <DallasTemperature.h>
```

Ilustración 8. Librería OneWire.h y DallasTemperature.h

2.2.5. Protocolo ZigBee

ZigBee da nombre a un conjunto de protocolos de comunicación, basados en el estándar IEE 802.15.4 o estándar de área personal (WPAN), que se suelen emplear en instalaciones domóticas, dado su alto nivel de eficiencia, alcance y transmisión de datos, sin embargo, los dispositivos que utilizan este protocolo de comunicación son dispositivos complejos de manejar, dado que debemos configurar los dispositivos individualmente antes de meterlos en la instalación. La topología que implementaremos en este proyecto será punto a punto, dado que solo nos interesa instalar dos dispositivos para establecer la comunicación desde el exterior del edificio hasta el interior.

Los dispositivos que emplean el protocolo ZigBee se denominan XBee, utilizan el programa X-CTU para su configuración, y la librería xbee.h para su programación.

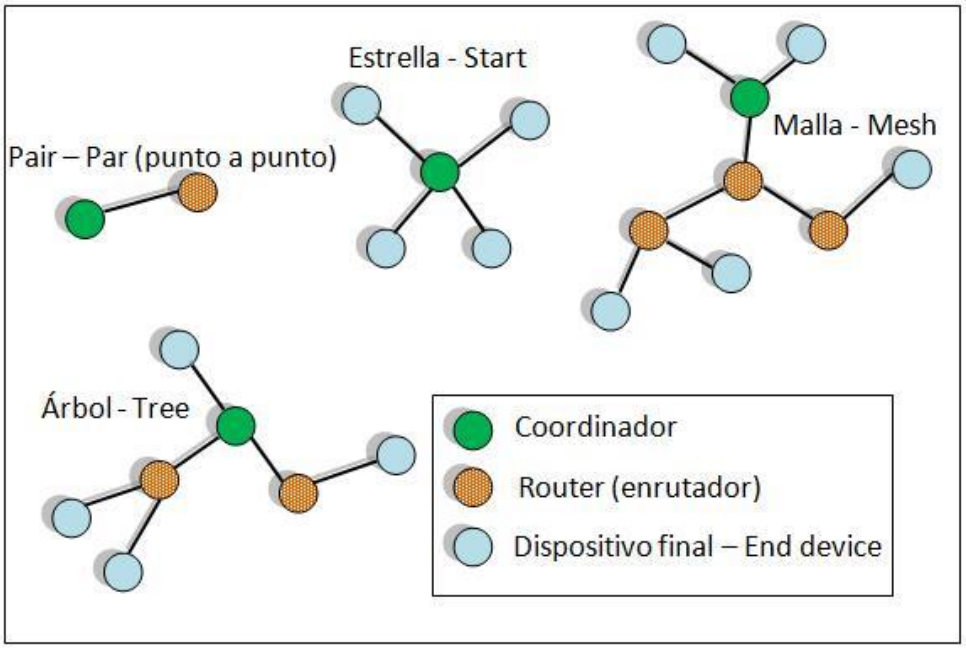


Ilustración 9. Topologías de red (http://plataformaszigbee.blogspot.com.es/2012/05/practica-1-configuracion-y-conceptos.html).

CAPÍTULO 3. ANÁLISIS DE ALTERNATIVAS

3. Análisis de alternativas

3.1. Selección del tipo de instalación

Inicialmente vamos a plantear dos instalaciones perfectamente compatibles con la monitorización, en los que estudiaremos el procedimiento requerido para poder escoger de esta forma el mejor sistema a implementar. En este sentido, ambos métodos tendrán en consideración como último paso del proceso la introducción de datos en un software en el que se podrán recopilar después de ser tomados a intervalos fijos de tiempo predefinidos, para procesarlos y enviar información a los actuadores en función de los datos recibidos.

El software al que se conectan las señales ejerce de cerebro de las operaciones, y habiéndose introducido previamente todas las variables que nos solicita, ejercerán la función de receptor de valores que provienen de los sensores instalados, o en su defecto al nodo central, cada cierto periodo de tiempo predefinido, para estudiar los datos. Una vez finalizado el tiempo de monitorización estudiaremos los resultados obtenidos para obtener unas conclusiones y propondremos una serie de soluciones al respecto.

Los dos casos que consideraremos serán, un sistema empleando cableado de bus, y un sistema centralizado basado en el empleo de un dispositivo hardware que actúa como cerebro de los sensores instalados.

3.1.1. Sistema de bus empleando KNX

Se trata de un sistema descentralizado y centralizado, que hace uso de sensores inteligentes, donde cada sensor tiene su propio procesador para la conversión de los datos que recibe, enviando así valores en las unidades que requiere el software al que son enviados posteriormente (por ello descentralizado). Toda esta información se transmite mediante un cableado de bus KNX, hasta el software disponible (centralización del sistema), donde se recogen los valores para analizarlos y enviar órdenes a los actuadores según proceda.

3.1.2. Sistema centralizado empleando un microcontrolador

Consiste en emplear sensores analógicos que captan señales y las envían en forma de corriente o tensión, porque son incapaces de mostrar señales de forma nítida para poder representarlas, debido a que la amplitud es muy pequeña o porque el dispositivo requiere de

alguna alimentación externa. Es por esto que surge la necesidad de acoplar un circuito electrónico que sea capaz de realizar estas variaciones, empleando en este caso un microcontrolador, para realizar la conversión de todos los datos que se recogen de los diferentes sensores (centralización).

Por último, los datos convertidos se introducen al software para que éste realice la lectura de los mismos y envíe las órdenes pertinentes a los actuadores. En la ilustración 12 se muestra un microcontrolador Arduino, que destaca por su sencillez y manejabilidad, portando varios puertos de entrada/salida al igual que los autómatas programables sin ser tan robustos como estos últimos.

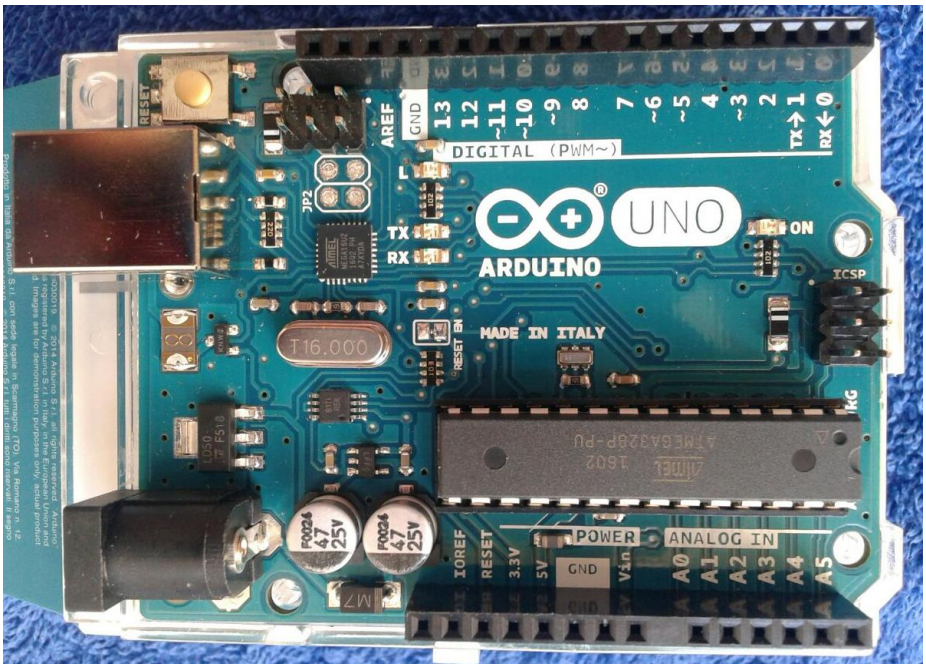


Ilustración 10: Microcontrolador Arduino UNO Rev 3.

3.2. Ventajas y desventajas

El estudio de las ventajas y desventajas son la principal forma de decantarnos por uno u otro sistema teniendo en cuenta las prestaciones que ofrece la instalación, analizando así cuestiones relacionadas con precio, garantía, tolerancia a fallos o facilidad de instalación. De esta forma se obtiene la tabla 2.

Tabla 2. Ventajas y desventajas de de los dos sistemas estudiados.

Sistemas	Ventajas	Desventajas
Sistema de bus	**Tolerancia a fallos.** Si un sensor se avería no afecta a la instalación	**Precio.** Los sensores tienen un precio elevado
	Reducido cableado. No hay que conectar los sensores a un nodo	**Requiere programación.** Todos los sensores deben programarse
	Fiabilidad. Es muy difícil que los sensores fallen	**Mantenimiento costoso.** La avería de un sensor provoca un gasto alto
Sistema centralizado	**Precio.** Se emplean sensores simples y un único controlador.	**Fallos del microcontrolador.** Si el microcontrolador falla, la instalación falla
	Fácil instalación. Se emplean sensores de tipo universal.	**Notable cableado.** Todos los sensores deben estar conectados al microcontrolador
	Fácil manipulación de señales. Las señales que se obtienen son fáciles de manipular	

3.3. Análisis comparativo

El principal objetivo del proyecto es conseguir una instalación que que cumpla los requisitos básicos, sin necesidad de grandes desembolsos económicos, y desarrollando una instalación sencilla, siendo compatible con el software diseñado. Estas metas se plantean con el fin de poder comercializar el trabajo, por lo que debe tener un precio accesible a los usuarios. Por otro lado, nos interesa evitar entornos de programación complicados o el empleo de numerosos dispositivos que programar, para facilitar la modificación del programa en función de las necesidades, así como mejor visualización de errores.

El sistema centralizado, siendo el método más barato y sencillo de manejar, es el que emplearemos para la automatización de una oficina gestionada por software. De esta manera, evitaremos tener que acudir constantemente a los sensores cuando se observen errores y sea necesario revisar la programación, actuando directamente sobre el microcontrolador que se encarga de administrar todos los datos que proporcionan los sensores.

CAPÍTULO 4. ELECCIÓN DE COMPONENTES

4. Elección de componentes

Teniendo en cuenta que nuestro proyecto debe suponer un desembolso económico lo más bajo posible, vamos a elegir los sensores que instalaremos en el edificio, priorizando los elementos que mejor relación calidad – precio presenten, evitando emplear sensores cuya fiabilidad sea escasa o mida alguna variable que no nos interese estudiar. Para optimizar costes, tendremos en cuenta la posibilidad de incluir sensores capaces de medir varias variables en un único dispositivo.

En nuestro presupuesto, no solo hay que tener en cuenta el gasto que conlleva la adquisición de los sensores, también debemos tener en cuenta el cableado, carcasas o el microcontrolador que vayamos a emplear, fijándonos en las prestaciones que nos ofrece, así como los accesorios necesarios, que son los módulos XBee o los respectivos adaptadores USB para su programación.

4.1. Presupuesto

Para justificar correctamente los elementos que vamos a emplear para nuestro proyecto, es necesario llevar a cabo el siguiente razonamiento teórico:

1. **Hardware**

Para nuestra instalación hemos estudiado la posibilidad de implementar tres modelos de Arduino: Mega 2560, Nano y UNO Rev3. El primero nos supondría un gasto excesivo, al no demandar el empleo de tantos pines, que es la principal ventaja que supone utilizar este modelo de hardware. Un Arduino Nano presenta unas características bastante buenas para nuestra instalación, sin embargo, no posee conector de alimentación externa, que es un requisito indispensable. Finalmente, el Arduino UNO sí presenta la posibilidad de conectar a alimentación externa, y es por esto que seleccionaremos este último dispositivo como nuestro hardware.

2. **Módulos inalámbricos**

Para nuestra instalación requeriremos tres dispositivos inalámbricos. En primer lugar, los módulos XBee S2C, los que, pese a su elevado precio y la necesidad de disponer también de un adaptador, han sido seleccionados dentro de la gama de modelos de Digi debido a su alcance para interiores de 60 metros, que permiten establecer la comunicación interior-exterior con garantías.

Para el interior del edificio hemos optado por una alternativa más económica, dado que la distancia no será tan elevada, seleccionado los módulos de radiofrecuencia NRF24L01, que además presentan la ventaja de funcionar perfectamente en distancias inferiores a los 30 metros.

Finalmente, para subir los datos a Conefi necesitaremos un dispositivo que nos permita establecer conexión a internet, para ello vamos a considerar dos opciones: la placa de extensión Ethernet Shield, o los módulos ESP8266. La primera establece una conexión segura al emplear cable de conexión de red, no obstante, requieren programación html, complicando el código del programa, que ya es bastante denso y podría producir problemas de estabilidad para el dispositivo hardware. Los módulos ESP8266 presentan la comodidad de trabajar vía wifi, además de ser bastante más económicos, por lo que optaremos por implementar estos módulos en la instalación.

3. Sensores

En lo que se refiere a los sensores, hemos tenido también que decantarnos por la opción que mejor cumpla con los requisitos que solicitamos.

Para medir variables de temperatura y humedad hemos estudiado los mejores modelos para ello: DHT11 y DHT22. El DHT11 es más barato, pero también tiene un error considerable, por lo que lo descartaremos, dedo que un error de hasta 2 grados es superior a lo que nos interesa. El DHT22 mejora bastante las prestaciones de calidad sin exceder mucho el precio, por lo que vamos a elegir este sensor para este apartado. Como queremos también medir la temperatura en el exterior, vamos a seleccionar la sonda DS18B20, dado que es un sensor de temperatura capaz de medir este parámetro en líquidos y gases, de forma que puede soportar condiciones de lluvia sin estropearse.

Para medir los niveles lumínicos vamos a estudiar también dos sensores muy utilizados por su elevado nivel de eficiencia: el TSL2561 y el BH1750. Comparando ambos, no cabe duda de que el sensor BH1750 es mucho más potente dado que el rango de luz que capta es muy grande, sin embargo, seleccionaremos el TSL2561 porque tiene un rango bastante aceptable, siendo además mucho más económico. Otra opción hubiera sido emplear un sensor LDR, pero no obtendríamos resultados tan precisos, por lo que en este caso el que sea más barato aun no nos resulta convincente.

Para gases directamente implicados con la salud como son el dióxido de carbono o el benceno, hemos empleado los sensores MQ135, para el CO_2, y el TGS823, para el benceno, dado que no encontramos otros modelos que detectaran estos gases para poder llevar a cabo una comparación entre ellos.

Finalmente, y siguiendo el desarrollo teórico que hemos comentado, obtenemos la siguiente tabla de presupuesto:

Tabla 3. Presupuesto.

Presupuesto					
Concepto	Modelo	Referencia	Precio unitario (€)	Unidades	Total (€)
Microcontrolador	Arduino UNO R3	K-Electrónica	6	3	18
Módulos ZigBee	XBee S2C	Digikey.es	16,98	2	33,96
Adaptador XBee	-	Digikey.es	23,33	2	46,66
Módulos de radiofrecuencia	NRF24L01	K-Electrónica	2,5	2	5
Módulos de conexión WiFi	ESP8266	K-Electrónica	4,2	2	8,4
Sensor de temp. Y humedad	DHT22	K-Electrónica	5	2	10
Sensor de temperatura	DS18B20	K-Electrónica	2,4	1	2,4
Sensor de iluminación	TSL2561	K-Electrónica	4	2	8
Sensor de CO2 y NOx	MQ135	K-Electrónica	2,9	3	8,7
Sensor de benceno/tolueno	TGS823	Figaro	4,9	2	9,8
Protoboard	400 agujeros	K-Electrónica	2,1	1	2,1
Protoboard	170 agujeros	K-Electrónica	1,2	1	1,2
Cableado	Cobre con cubierta	Electrónica Faro	0,35	40	14

Carcasas Arduino exterior	Metacrilato	Social Makers	13,25	2	26,5
Total	-	-	-	-	194,72

4.2. Ubicación de componentes

La ubicación de los sensores no es tarea fácil en la mayoría de los casos si queremos conseguir resultados fiables, es decir, que no se encuentren influenciados por cualquier condición que pueda darse en donde coloquemos los sensores, como podrían ser conductos de calefacción, ventilación o la incidencia directa de luz solar. Según el sensor que requiramos será necesario estudiar unas variables en concreto, y en función de las dimensiones de la dependencia en el que se toman las mediciones, también será necesario estudiar la posibilidad de instalar varios sensores para que la fiabilidad no disminuya.

Por ejemplo, en un sensor temperatura, a diferencia de las variables de humedad, no son homogéneas en toda una habitación, así que en función de sus dimensiones habrá que estudiar el lugar adecuado para su instalación, y en caso de ser un lugar extenso, disponer de varios sensores para que no disminuya la fiabilidad. También habrá que tener en cuenta que el sensor no esté cerca de conductos de ventilación o luces que puedan modificar los valores de temperatura, así como revisar las posibilidades de conexión inalámbrica en cuestiones de distancia y presupuesto con respecto al coordinador, debido a que un dispositivo inalámbrico siempre es mucho más cómodo y efectivo de instalar, pero en algunas ocasiones podría ser más caro.

En lo que se refiere a los sensores situados en el exterior del edificio los ubicaremos en la fachada este, al ser la zona del edificio donde los valores son más desfavorables (peor caso), sobre todo para los sensores de parámetros de salud, respetando en todo momento las condiciones a las que se debe evitar exponer estos sensores. Por ejemplo, en el caso de un sensor de temperatura habrá que buscar la forma de ubicarlo en una zona en la que no quede influenciado por paredes cercanas que podrían sobrecalentarse, o incluso por chapas metálicas o cualquier otro material. En este caso sí que será indispensable implementar un dispositivo de conexión inalámbrica para enviar datos desde el exterior hasta el interior del edificio.

CAPÍTULO 5. ANÁLISIS DE LOS COMPONENTES

5. Análisis de los componentes

5.1. Microcontrolador Arduino

Para la realización de esta prueba piloto haremos uso de tres microcontroladores Arduino UNO Rev 3. Como podemos ver en la ilustración 13, estos microcontroladores tienen una serie de pines reservados para cumplir funciones determinadas, conocer estos pines es indispensable para que la mayoría de los elementos que emplearemos cumplan su función de forma efectiva.

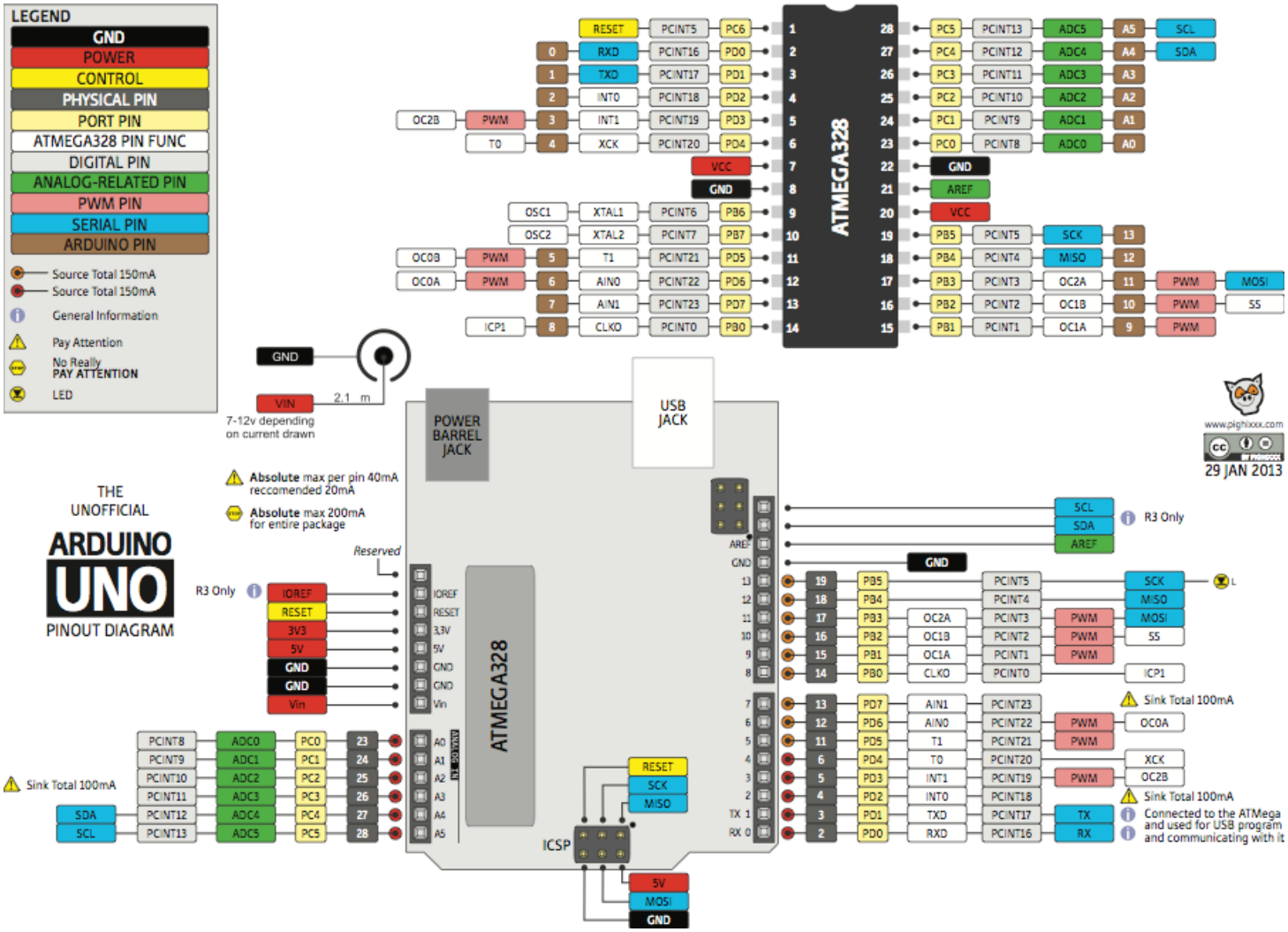


Ilustración 11. Esquema de patillaje de Arduino UNO R3 (https://www.tr3sdland.com/2013/02/arduino-pinout/).

5.2. Sensor de temperatura DHT22

El DHT22, es el único sensor de todos los que vamos a emplear capaz de medir dos variables, temperatura y humedad, haciendo uso del bus I^2C para el envío de ambos datos mediante una trama compuesta por números binarios, por lo que podemos conectar dicho sensor a un pin digital PWM. Para programar el DHT22 emplearemos la librería DHT.h, que se utiliza tanto para este sensor como para el DHT11, que también mide temperatura y humedad. El motivo principal por el que hemos seleccionado este sensor es porque presenta un error inferior al que presentan otros sensores, en las ilustraciones 14 y 15 observamos el comportamiento del error máximo cometido en función de la temperatura/humedad.

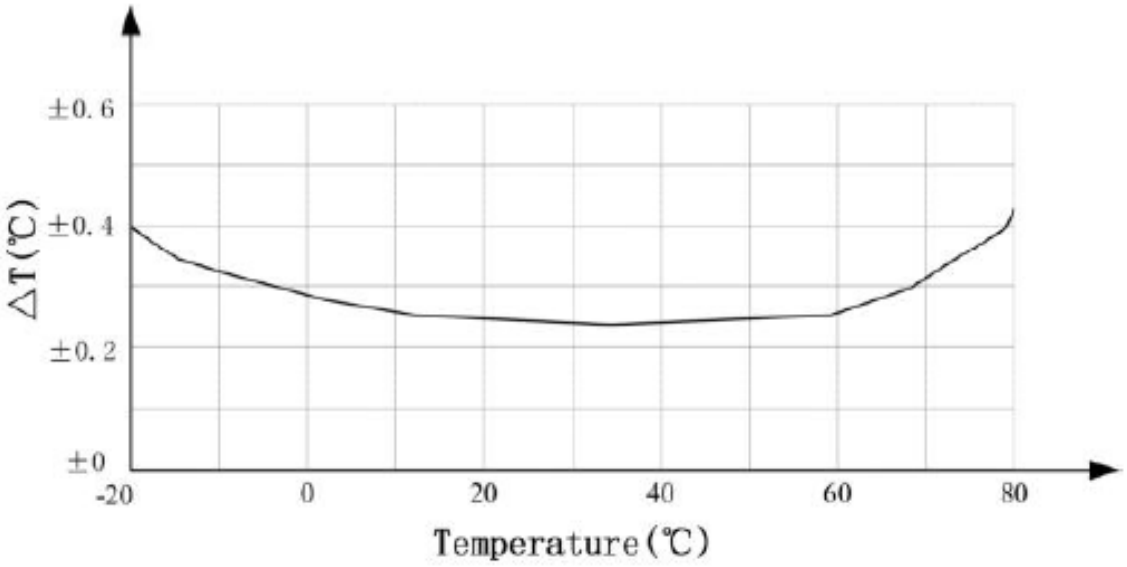


Ilustración 12. Error máximo de temperatura para el sensor DHT22.

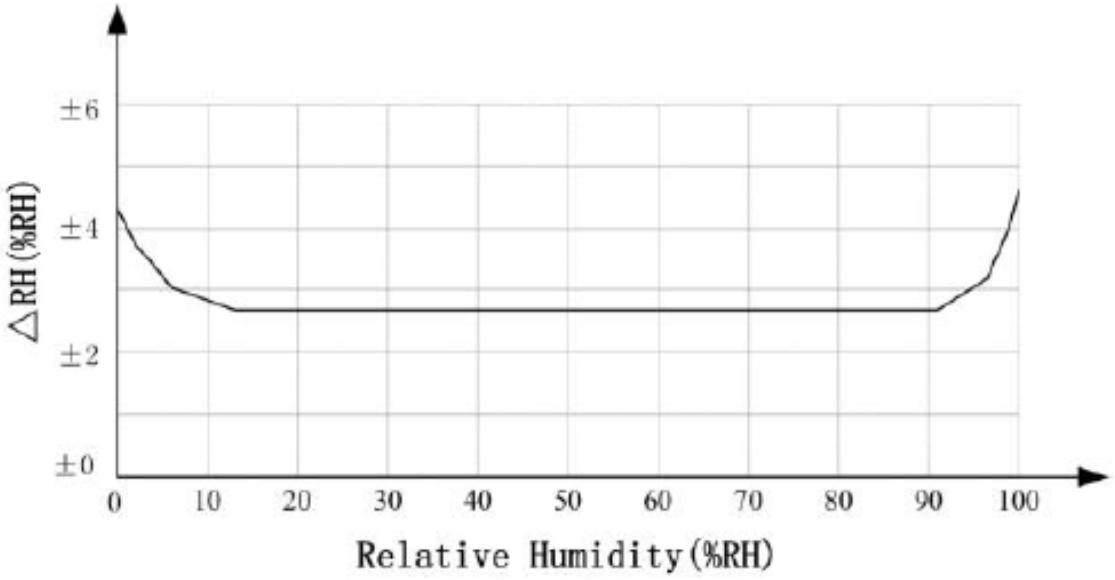


Ilustración 13. Error máximo cometido de humedad para el sensor DHT22.

Además, presenta las siguientes características:

- Emplea un pin SDA (Serial Data) para el envío de datos.
- Su tensión de alimentación típica son 5 voltios.
- Precisión en la medición de temperatura de 0.5ºC.
- Precisión en la medición de humedad de 2 a 5%.
- Frecuencia de muestreo de 0.5 Hz.

Ilustración 14. Sensor DHT22.

5.3. Sensor de temperatura de líquidos y gases DS18B20

Hemos seleccionado la sonda de temperatura DS18B20 con el objetivo de poder tomar las mediciones en lugares al aire libre, sin estar recubiertos, sin temor a que se pueda estropear con la meteorología. Este sensor emplea el bus One-Wire para el envío de una trama de datos de señal digital, haciendo uso de las librerías OneWire.h, y Dallastemperature.h.

Este sensor de temperatura tiene un funcionamiento bastante complejo, por lo que resulta una gran ventaja el hecho de que pueda transmitir los datos a través de un solo cable. Para empezar, como vemos en la ilustración 17, debemos acoplar una resistencia en paralelo de 4,7 kilo ohmios desde la señal de datos hasta la tensión de referencia, cada sensor dispone de una dirección única de 64 bits, que debemos introducir para identificar el sensor.

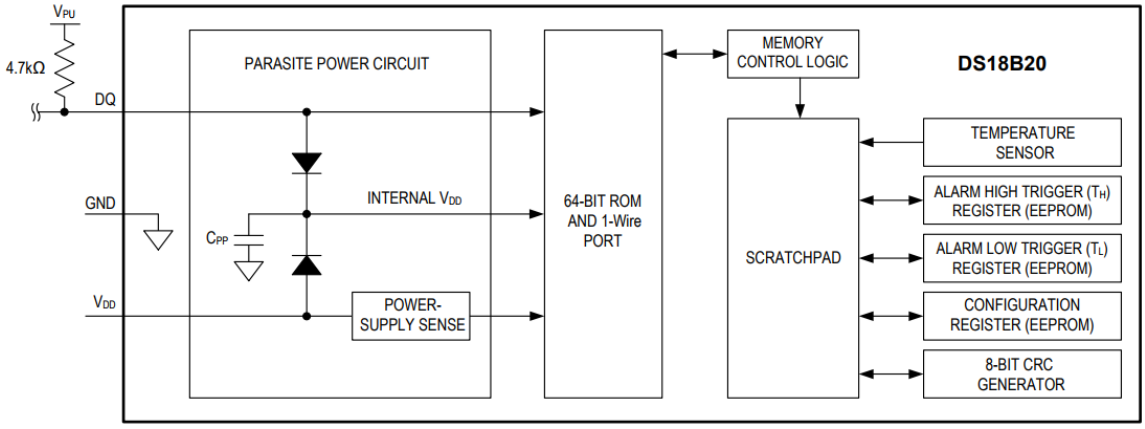


Ilustración 15. Diagrama de bloques del sensor DS18B20.

La resolución de este sensor puede ser de 0.5, 0.25, 0.125 o 0.0625ºC, siendo esta última la resolución por defecto.

Ilustración 16. Sonda de temperatura DS18B20.

5.4. Sensor de gases MQ135

En la hoja de datos de este sensor que mide la calidad del aire, encontramos los siguientes datos de interés:

- R_L: Resistencia a la salida ajustable, hay que calcularla. En el datasheet se recomiendan 20kohm.
- R_O: Resistencia del sensor a 100ppm de NH_3.
- R_S: Resistencia del sensor, es la resistencia interna del MQ135 que varía para la toma de datos. Su valor es el siguiente (30-200 khom):

$$R_S = \left(\frac{V_C - V_{R_L}}{V_{R_L}}\right) \cdot R_L = \left(\frac{V_C}{V_{R_L}} - 1\right) \cdot R_L$$

- El tiempo de calentamiento que necesita un sensor de estas características para que los resultados sean fiables es de 24 horas, en este sensor nos indica que deben ser 48 horas.

También disponemos de un gráfico que describe el comportamiento del sensor capaz de detectar diversos tipos de gases (ilustración 1):

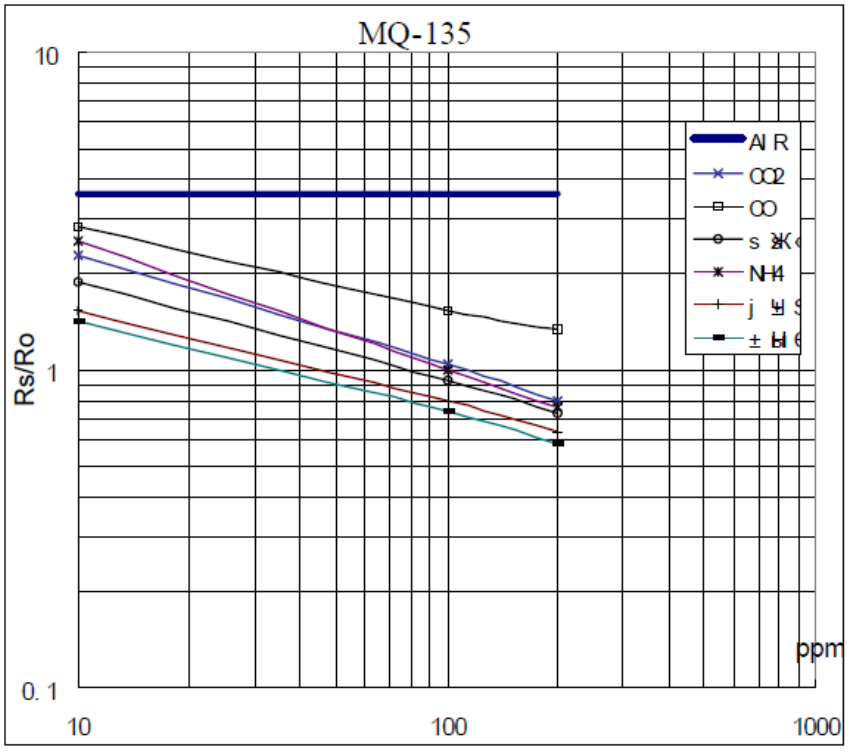


Ilustración 17. Gráfica del sensor MQ135.

Para la calibración de estos sensores de forma precisa necesitamos partir de una muestra conocida para calcular R0, o bien, crear nuestra propia recta empleando varias concentraciones diferentes tomando los valores de los sensores. En nuestro caso, como no disponemos de material ni presupuesto para realizar una calibración empleando muestras conocidas, vamos a apostar por la calibración empleando métodos matemáticos.

En primer lugar, seleccionamos 14 puntos diferentes de la curva característica del CO_2, obteniendo la tabla 4, de forma que podemos deducir tanto la ecuación lineal como la logarítmica del sensor. Ambas ecuaciones las podemos emplear para describir el comportamiento del MQ135 en el código de nuestro programa, no obstante, nos vamos a decantar por la ecuación logarítmica.

Tabla 4. Datos obtenidos de la curva característica del sensor MQ135.

PPM	Rs/Ro		log10(PPM)	log10(Rs/Ro)
10	2,55	→	1	0,4065
15	2,2	→	1,1761	0,3424
20	2	→	1,301	0,301
30	1,8	→	1,4771	0,2553
40	1,58	→	1,6021	0,1987
50	1,4	→	1,699	0,1461
60	1,3	→	1,7782	0,1139
70	1,2	→	1,8451	0,0792
80	1,15	→	1,9031	0,0607
90	1,1	→	1,9542	0,0414
100	1,05	→	2	0,0212
120	1	→	2,0792	0
157,5	0,9	→	2,1973	-0,0458
200	0,8	→	2,301	-0,0969

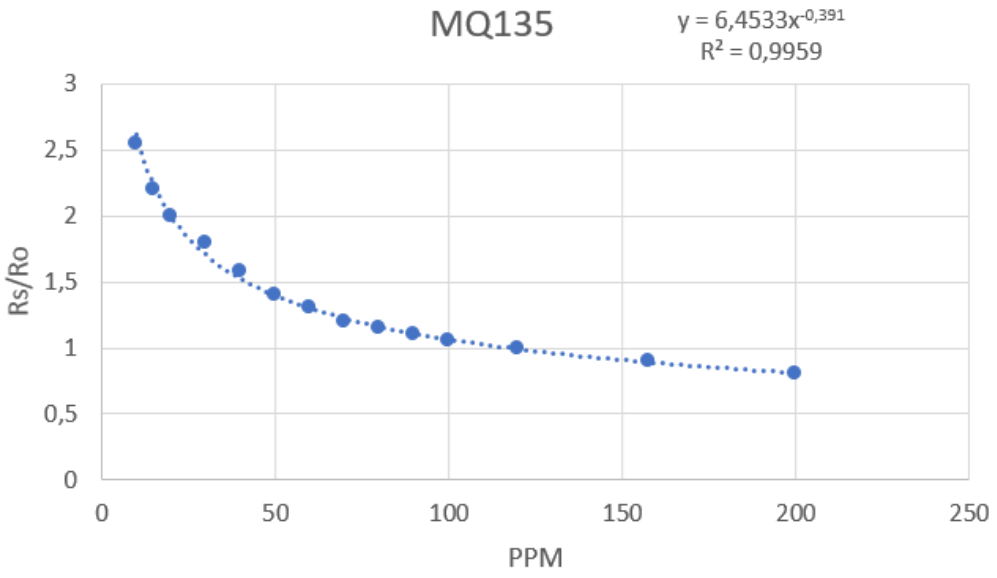


Ilustración 18. Ecuación y recta del sensor MQ135.

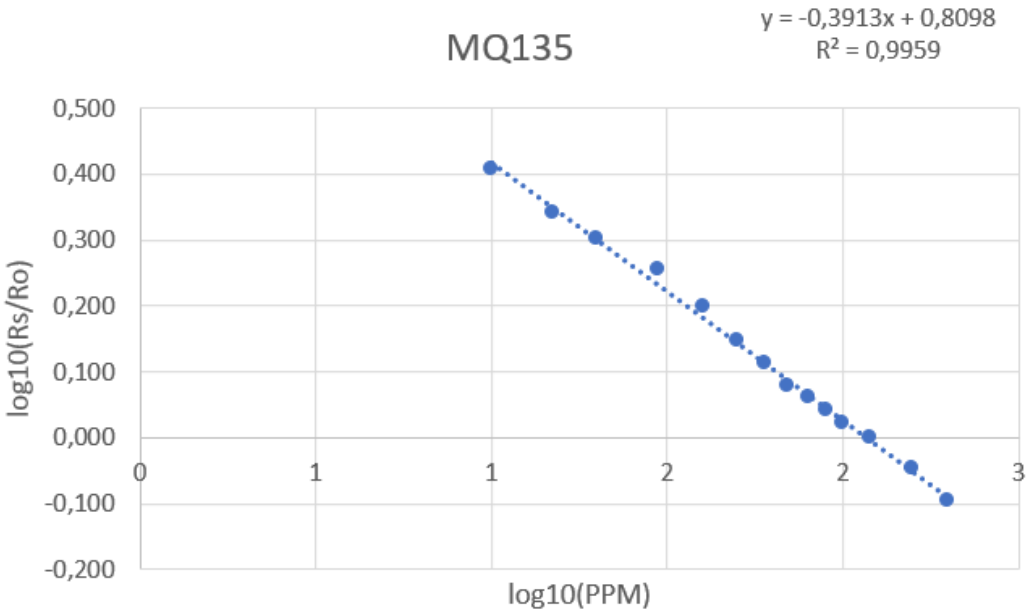


Ilustración 19. Ecuación y recta logarítmica del sensor MQ135.

Del INSHT podemos concluir que el dióxido de carbono "Es un componente del aire exterior en el que se encuentra habitualmente a niveles entre 300 y 400 ppm, pudiendo alcanzar en zonas urbanas valores de hasta 550 ppm" (http://www.insht.es/InshtWeb/Contenidos/Documentacion/FichasTecnicas/NTP/Ficheros/501a600/ntp_549.pdf). De esta forma, vamos a suponer unas condiciones en el interior de la empresa de 390 PPM de CO_2, ligeramente superior a los 387 PPM que es el nivel habitual en ciudad, para obtener de esta forma el valor de R_0. Como se trata de ecuaciones logarítmicas, para obtener R_0 debemos solucionar la siguiente ecuación por aproximación, modificando el valor X hasta que se cumpla la igualdad que estemos buscando:

$$CO_2 = 10^{(A\cdot\log\left(\frac{R_S}{R_0}\right)+B)} = 10^{(-0.3913\cdot\log\left(\frac{R_S}{R_0}\right)+0.8098)};$$

$$390 = 10^{(-0.3913\cdot\log(X)+0.8098)}$$

El valor de X obtenido será equivalente al cociente entre R_s y R_0, despejando se deduce el valor de R_0, cuyo valor será característico de cada sensor. El valor R_0 de un sensor no tiene por qué ser igual para todos los sensores.

$$\frac{R_S}{R_0} = X \rightarrow R_0 = \frac{R_S}{X}$$

Ilustración 20. Sensor MQ135.

5.5. Sensor de gases TGS823

El sensor TGS823 es capaz de detectar diversos gases, con los que podríamos obtener también la calidad del aire, no obstante, de este sensor nos interesa conocer el nivel de benceno. Como podemos ver en la ilustración 27, este sensor lo hemos adquirido sin una PCB que lo soporte, por tanto, debemos realizar el cableado desde las patas del propio sensor como se muestra en la ilustración 23:

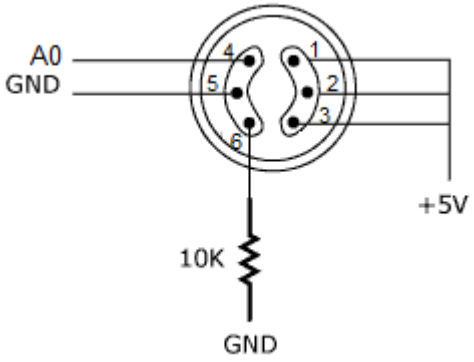


Ilustración 21. Conexionado del sensor TGS823.

Igual que con el sensor MQ135, vamos a realizar una calibración por aproximación matemática para conocer el valor de R0. En la ilustración 24 vemos la gráfica del sensor extraída de su Datasheet.

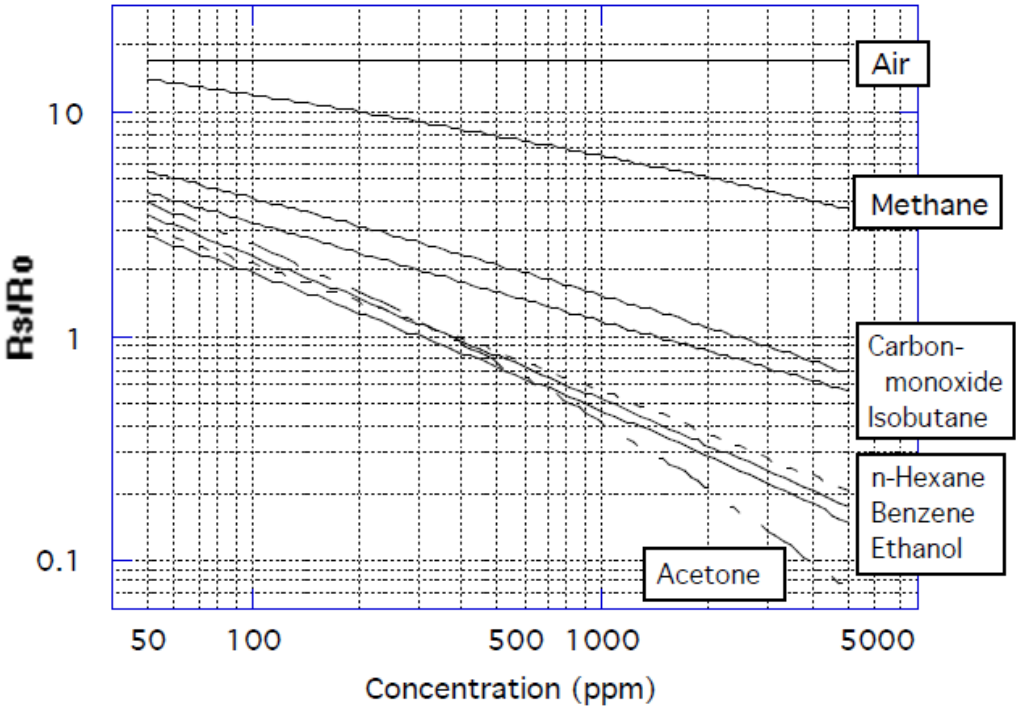


Ilustración 22. Gráfica del sensor TGS823.

Vamos a escoger en este caso un total de 19 puntos diferentes de la curva característica para el benceno, con el objetivo de obtener la ecuación que describe el comportamiento del sensor. De la tabla 5, y empleando la columna de la concentración en PPT, podemos obtener tanto la ecuación lineal como la logarítmica, quedándonos como en el caso anterior, con la ecuación logarítmica.

Tabla 5. Datos obtenidos de la curva característica del sensor TGS823.

PPM	PPT	Rs/Ro		log10(PPM)	log10(PPT)	log10(Rs/Ro)
50	50000000	3,5	→	1,699	7,699	0,544
60	60000000	3,1	→	1,778	7,778	0,491
70	70000000	2,9	→	1,845	7,845	0,462
80	80000000	2,6	→	1,903	7,903	0,415
90	90000000	2,5	→	1,954	7,954	0,398

120	120000000	2	→	2,079	8,079	0,301
200	200000000	1,5	→	2,301	8,301	0,176
300	300000000	1,2	→	2,477	8,477	0,079
400	400000000	0,93	→	2,602	8,602	-0,032
500	500000000	0,81	→	2,699	8,699	-0,092
600	600000000	0,72	→	2,778	8,778	-0,143
700	700000000	0,65	→	2,845	8,845	-0,187
800	800000000	0,59	→	2,903	8,903	-0,229
900	900000000	0,55	→	2,954	8,954	-0,260
1000	1000000000	0,52	→	3,000	9,000	-0,284
2000	2000000000	0,32	→	3,301	9,301	-0,495
3000	3000000000	0,25	→	3,477	9,477	-0,602
4000	4000000000	0,2	→	3,602	9,602	-0,699
5000	5000000000	0,18	→	3,699	9,699	-0,745

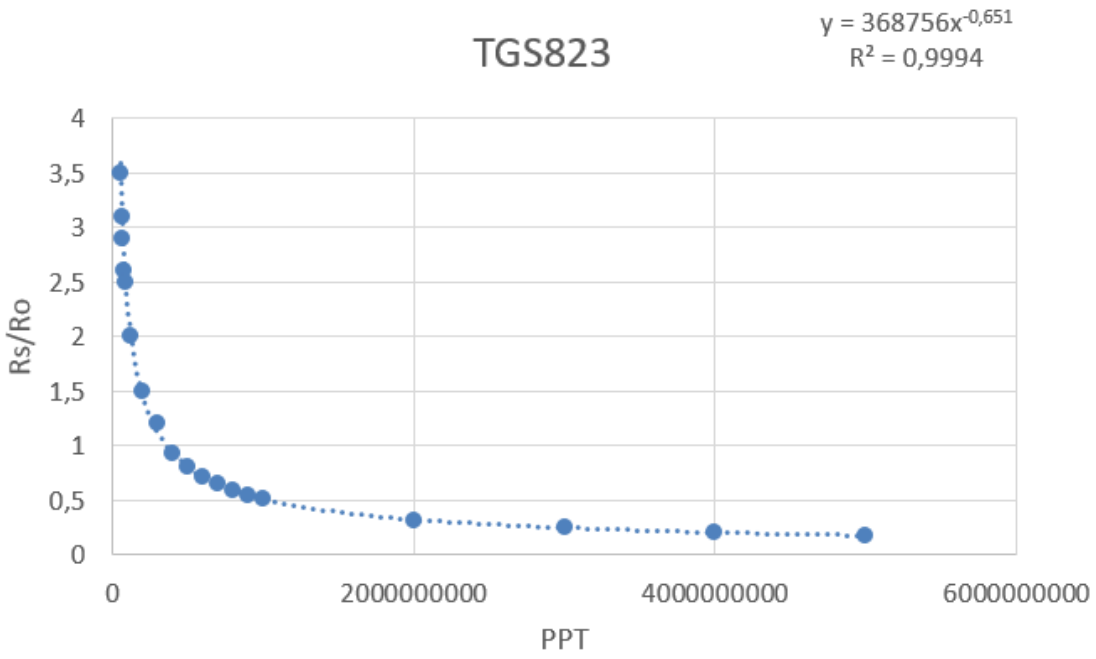


Ilustración 23. Ecuación y recta del sensor TGS823.

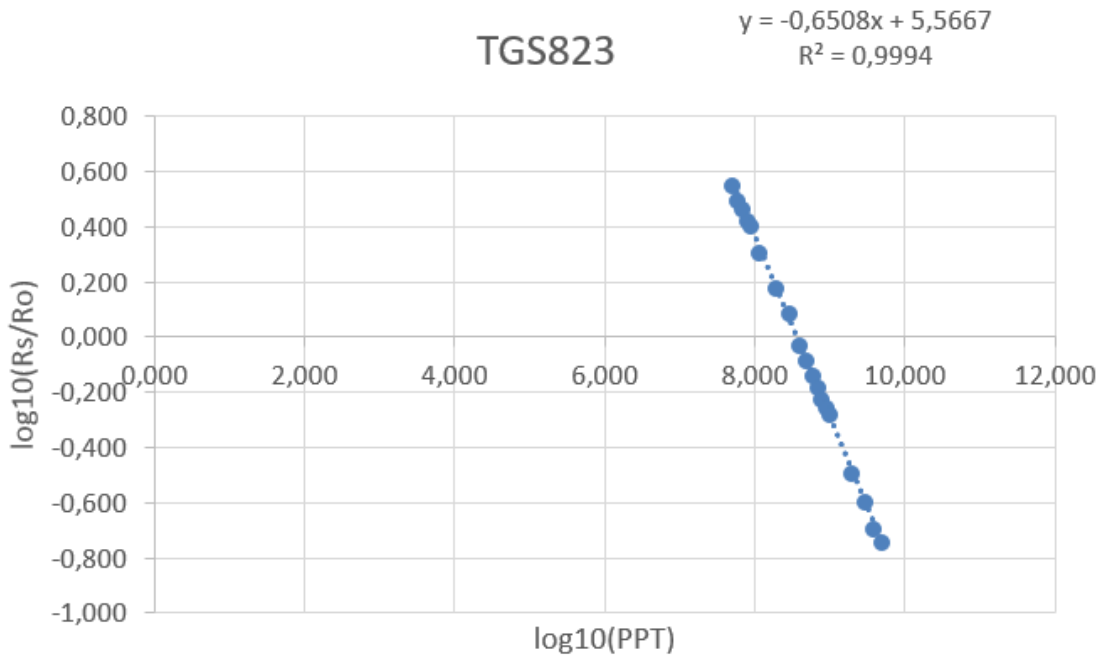


Ilustración 24. Ecuación y recta logarítmica del sensor TGS823.

Como el nivel de benceno es muy bajo, vamos a realizar las mediciones para este gas en partes por trillón (PPT), realizando el mismo procedimiento matemático desarrollado anteriormente. "Los datos disponibles de benceno en aire en el Centro Nacional de Condiciones de Trabajo del INSHT desde 1995 presentan variaciones entre 1 y 50 µg/m^3" (http://www.insht.es/InshtWeb/Contenidos/Documentacion/FichasTecnicas/NTP/Ficheros/401a500/ntp_486.pdf), por lo que vamos a considerar 20 PPT para realizar nuestro cálculo de aproximación.

$$20 = 10^{(-0.6508 \cdot \log(X) + 5.5667)}$$

Una resuelta la ecuación, podemos resolver el cociente de resistencias para obtener el valor de R_0:

$$\frac{R_S}{R_0} = X \rightarrow R_0 = \frac{R_S}{X}$$

Ilustración 25. Sensor TGS823.

5.6. Sensor de iluminación TSL2561

En nuestra instalación dispondremos de 2 sensores TSL2561 que emplearemos para estudiar los niveles lumínicos en el interior del edificio. En estos sensores la tensión de referencia V_{CC} debe estar entre 2,7 y 3,6 voltios (emplear el pin de 3,3V de Arduino), de lo contrario el funcionamiento del sensor no será fiable. En este caso emplearemos la librería TSL2561.h específica para este tipo de sensor. Un diagrama de bloques bastante visual es el que se muestra a continuación, donde dos diodos LED captan la luz solar, pasamos los datos a digital empleando un CAD, y pasan a un registro. Por último, se genera una trama de datos con los valores captados.

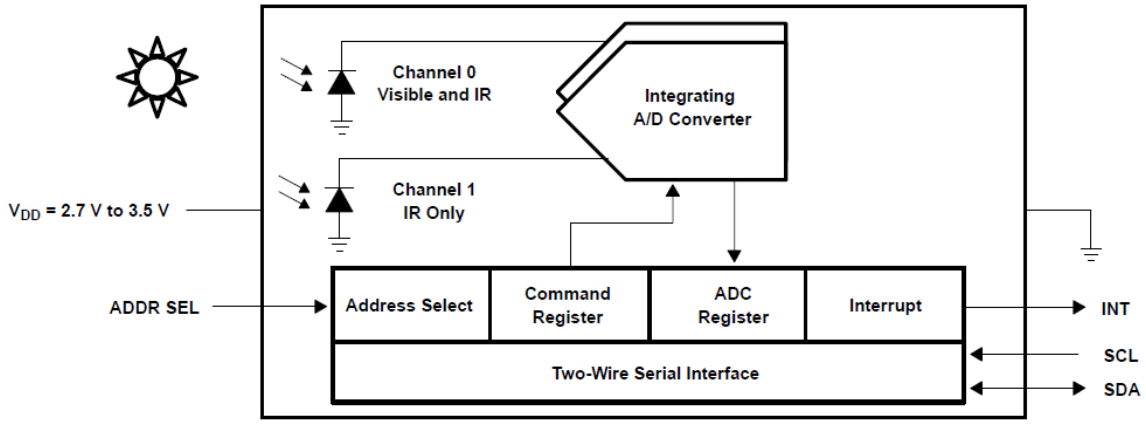


Ilustración 26. Diagrama de bloques del sensor TSL2561.

Los pines que se encargan de enviar datos son el SCL y el SDA, que trabajan empleando el bus I^2C. Los puertos reservados para el conexionado de estos pines son el A4, A5, SCL y SDA.

- **SCL.** Pin de "Serial Clock". Establece una señal de reloj.
- **SDA.** Pin de "Serial Data". Cuando se activa un ciclo de reloj, envía una trama de números binarios, que representan los niveles de iluminación captados.

Ilustración 27. Sensor TSL2561 y sus pines.

5.7. Módulos XBee

Ante la necesidad de realizar mediciones en el exterior del edificio, y la imposibilidad de introducir cables desde el exterior, es indispensable el empleo de dispositivos que transmitan la información de forma inalámbrica. Los módulos que vamos a emplear para nuestro proyecto son los xBee S2C. Los aspectos más importantes que debemos tener en cuenta sobre estos dispositivos son los siguientes:

- Deben ser alimentados a 3,3 Voltios, nunca a 5 Voltios debido a que es bastante probable que se produzca un incorrecto funcionamiento de los dispositivos o incluso llegar a quemarlos.
- Tienen un alcance de 60 metros, en rangos de interiores, lo que los hace bastante eficientes para trabajar en redes domésticas o edificios.
- Poseen una frecuencia de banda de 2,4 GHz.
- Es necesario disponer de un adaptador para su programación. Además, para poder introducirlos en una protoboard también necesitaríamos de estos adaptadores.
- Estos módulos disponen de 4 entradas/salidas analógicas.
- Es importante que en todas las redes exista un único coordinador, todos los demás deben ser routers o dispositivos finales.

Los módulos de la serie XBee, que emplean el protocolo ZigBee, son los que se suelen emplear en los microcontroladores Arduino para el envío y recepción de señales de forma inalámbrica. Estos dispositivos son creados por la compañía Digi International, y deben ser configurados en un programa que pertenece también a esta compañía, su nombre es "X-CTU". Para la conexión con el ordenador mediante cable USB, es necesario acoplar al dispositivo un adaptador, que podemos ver en la ilustración 30, donde también se muestra la función de cada uno de los pines del mismo.

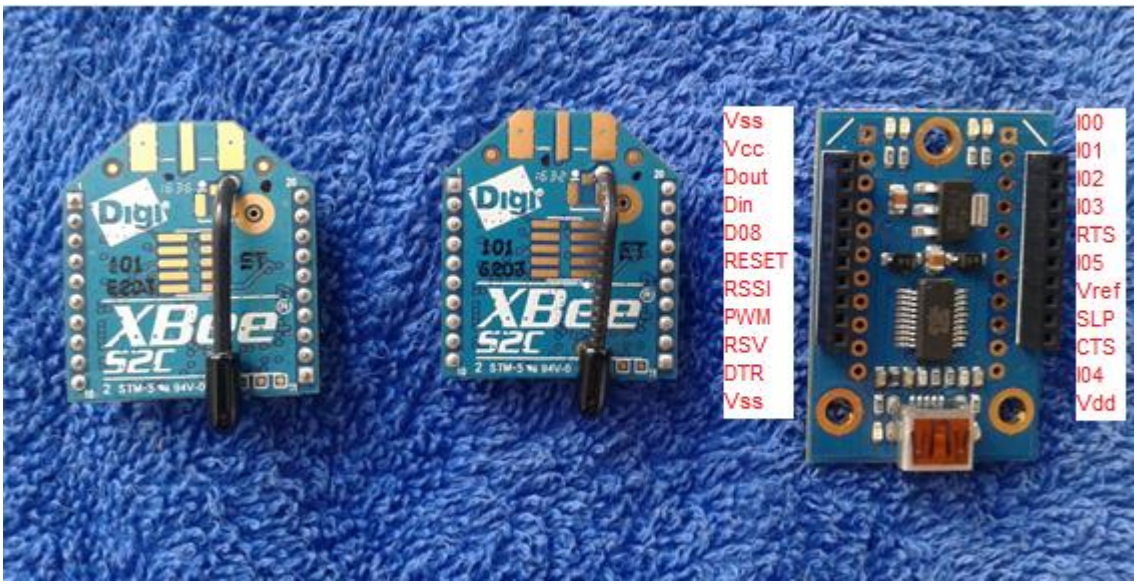


Ilustración 28: XBee S2C y su adaptador.

Los XBee trabajan diferentes modos de funcionamiento, nosotros vamos a centrarnos en dos de ellos, el modo AT (para comprobar que se establece la comunicación de forma correcta entre los dos terminales) empleando el programa X-CTU, y el modo API.

- **Modo Transparente (AT).** El modo AT consiste en simular una conexión serial de forma inalámbrica empleando la comunicación entre los dos dispositivos, conectados al PC mediante sus respectivos adaptadores, y programados para tal fin con el X-CTU. Activando este modo de funcionamiento, podemos observar en la consola como se establece la comunicación correctamente al enviar información de un módulo a otro y viceversa.

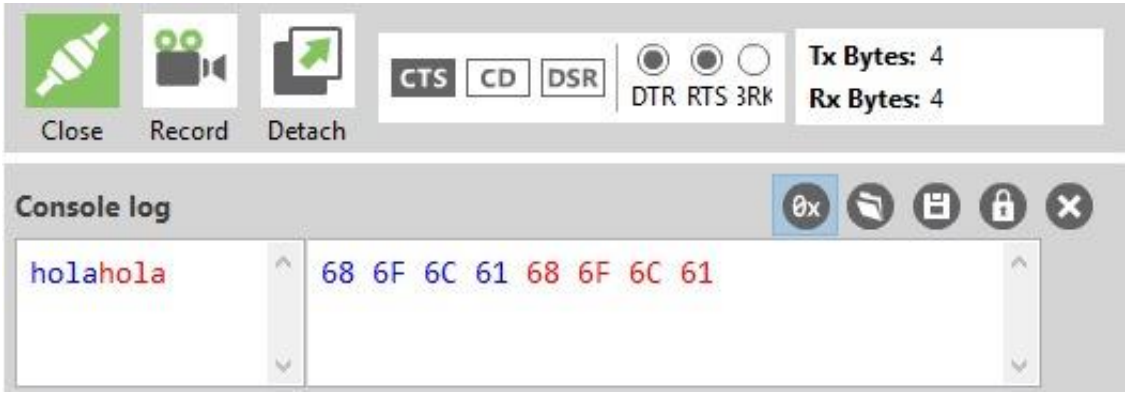


Ilustración 29. Pantallazo de la comunicación del Coordinador con el Router en modo AT.

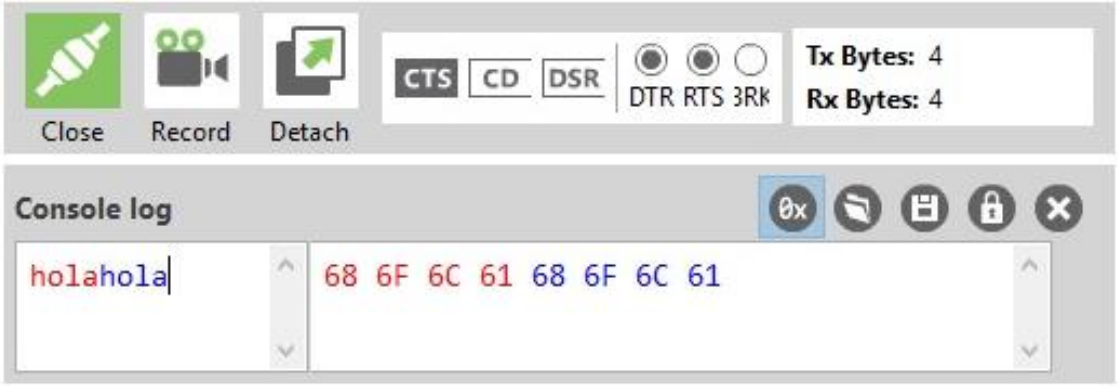


Ilustración 30. Pantallazo de la comunicación del Router con el Coordinador en modo AT.

Como vemos en las dos ilustraciones, en azul se muestran los datos enviados, y en rojo los datos recibidos. En la ventana superior derecha, el número de bytes enviados (Tx) debe coincidir con el número de bytes recibidos (Rx).

- **Modo API (Empleando tramas de datos).** El modo en el que se realizará el envío y recepción de datos será mediante tramas API, que consiste en el envío de una serie de bytes donde cada uno ejerce una función determinada. Una vez configurados debidamente los dos dispositivos, podemos comprobar su funcionamiento de diferentes formas, nosotros vamos a conectar el Router a un Arduino para el envío de dos datos recogidos de dos sensores, por otro lado, conectamos el Coordinador al PC. Abrimos la consola de forma que comenzamos a recibir tramas de datos con un tiempo entre muestra y muestra que ha sido definido por nosotros en la configuración del mismo en el X-CTU.

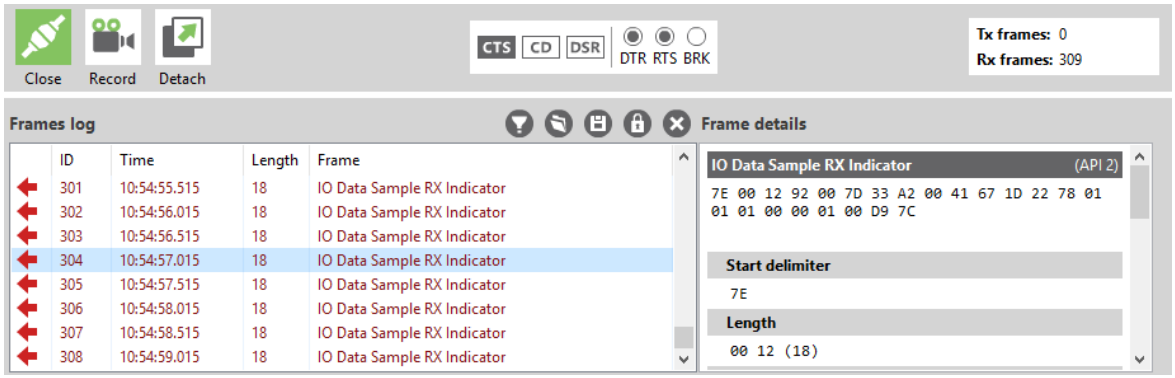


Ilustración 31. Recepción de datos en modo API.

En la ventana de la izquierda observamos todas las tramas que nos van llegando, y a la derecha se muestra la trama completa, así como un desglose de dicha. De esta forma, podemos analizar el contenido de una trama:

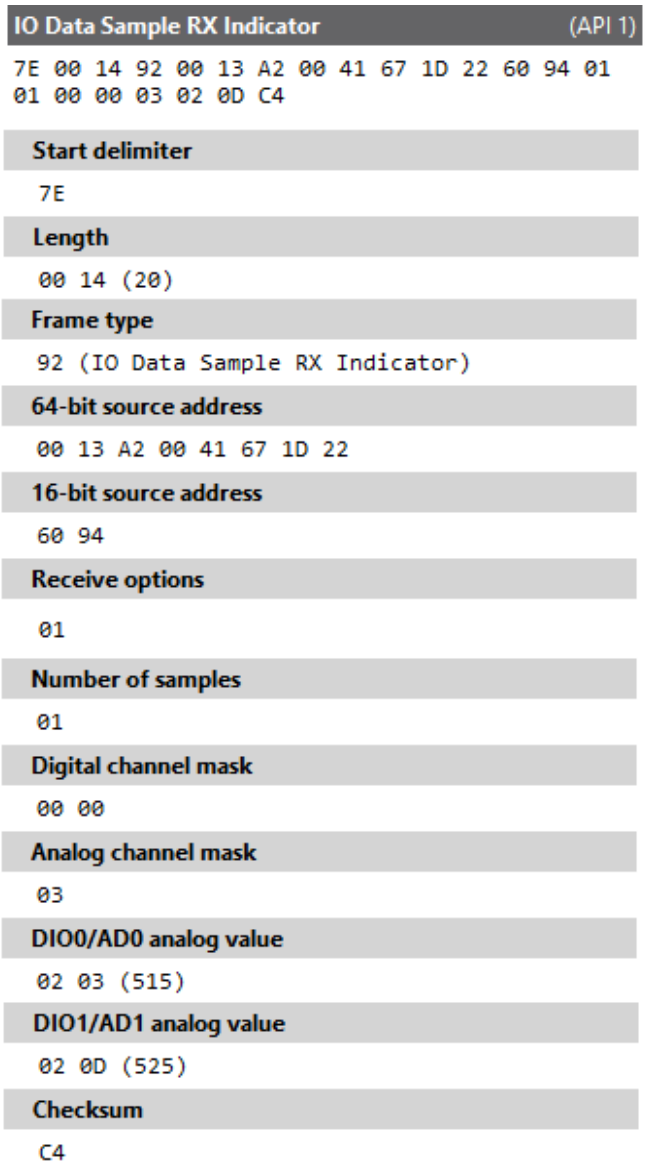


Ilustración 32. Desglose de una trama API.

Tabla 6. Descripción de los bytes que componen una trama API.

BYTE	DESCRIPCION
Start delimiter	Bit de inicio de la trama, siempre es 0x7E
Lenght	Indica el tamaño de la trama
Frame type	Tipo de trama. 0x92 indica muestreo de entradas
64-bit source adress	Dirección del XBee de destino (receptor)
16-bit source adress	Dirección de la red
Receive options	Opciones de recepción. 0x01 indica paquete de confirmacion

Number of samples	Número de muestras, siempre debe ser 0x01
Digital channel mask	Indica el número de pines configurados como digitales (en nuestro caso ninguno)
DIO0/AD0 analog value	Lectura del valor analógico del puerto AD0
DIO1/AD1 analog value	Lectura del valor analógico del puerto AD1
Checksum	Byte de comprobación de que la trama se ha enviado correctamente. [0xFF - (∑ byte 3 hasta el último)]

Con la información que obtenemos de los sensores podemos hacer como en el ejemplo anterior, es decir, conectar la señal de datos al puerto AD0 del XBee para enviar los datos, o bien se pueden llevar a un Arduino, donde se escribe la trama directamente sin necesidad de que el XBee realice la lectura de los datos.

La segunda forma es la más efectiva cuando disponemos de sensores como el DHT22, cuyo valor obtenido es una trama de dígitos binarios que tienen el valor de la temperatura y la humedad, siendo imposible para estos módulos realizar la lectura de todos los datos, al ser una trama más grande de lo que el puerto AD0 puede llegar a leer. Por este motivo, en nuestro montaje vamos a acoplar un Arduino junto al XBee Router y la forma de enviar la información será la desarrollada en segundo lugar.

5.8. Módulos ESP8266

Los módulos ESP8266 son dispositivos que disponen de conexión wifi que emplearemos para la transmisión de datos al entorno de Conefi. Los hemos escogido por su reducido precio y su buena capacidad de funcionamiento, dado que no le vamos a demandar una carga de trabajo excesiva.

5.9. Módulos NRF24L01

Los módulos de radiofrecuencia NRF24L01 los emplearemos para establecer la conexión inalámbrica en el interior del edificio, dado que tomaremos algunas medidas un tanto lejanas al microcontrolador, y resulta más sencillo acoplar estos dispositivos a nuestro montaje que

realizar un cableado para llevar todos estos sensores a Arduino coordinador. La gran ventaja que presentan los NRF24L01 es su precio, siendo bastante baratos, especialmente el modelo 'sin antena', que es el que hemos seleccionado en este caso. Los aspectos más importantes son los siguientes:

- Tensión de alimentación: 3,3 Voltios (su rango de operación es: 1,9 – 3,6 V).
- Modo de transmisión. Bus SPI.
- Velocidad máxima y mínima de datos: 2000 kbps – 2 Mbps.
- Alcance: Según los obstáculos que se interpongan podría alcanzar entre los 20 y los 30 metros.
- Flexibilidad de datos: En este caso podemos enviar gran diversidad de datos, sin necesidad de codificarlos para su envío y recepción.
- Antena: Como ya se ha comentado, esta versión en concreto no dispone de antena de comunicación, sino que dispone de un chip incorporado para estas funciones. De esta forma optimiza el espacio, pero disminuye alcance.

Los módulos NRF24L01 disponen de un total de ocho pines distribuidos como se ve en la ilustración 35, de forma que resulta inviable implementar este dispositivo en una protoboard, debido a que se provocarían cortocircuitos entre estos pines. También podemos ver el chip con el que podemos establecer la comunicación entre módulos. En este caso, los pines de envío y recepción de datos son MOSI y MISO, para los que existen puertos reservados en el Arduino junto al pin SCK.

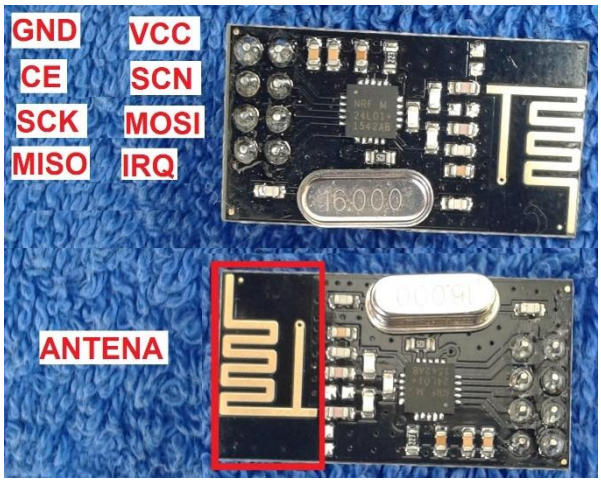


Ilustración 33. Pines y antena correspondientes a los módulos NRF24L01.

El modo de funcionamiento consiste en la creación de un 'tubo', al que se le asigna un identificador que debe ser el mismo en ambos módulos, que funciona como un canal por el que fluye toda la información. Estos dispositivos presentan, entre otras, dos diferencias destacadas con respecto a los módulos XBee: no necesitan de una configuración previa en un programa, y la información no es preciso convertirla a hexadecimal para transmitirla.

5.10. Carcasas protectoras

Para proteger el microcontrolador situado en el exterior del edificio y su respectivo cableado, hemos diseñado un modelo de encapsulado lo suficientemente amplio como para albergar todos los elementos que se requieren, evitando exposiciones a las condiciones meteorológicas. Los sensores no los incluiremos en su interior, debido a que nos interesa obtener valores un tanto alejados de la posición del mismo.

El material que emplearemos para el encapsulado es metacrilato de 3mm de grosor y transparente. En cuanto al programa en el que nos apoyaremos para realizar el diseño es AutoCAD. En la ilustración 36 podemos ver el diseño creado:

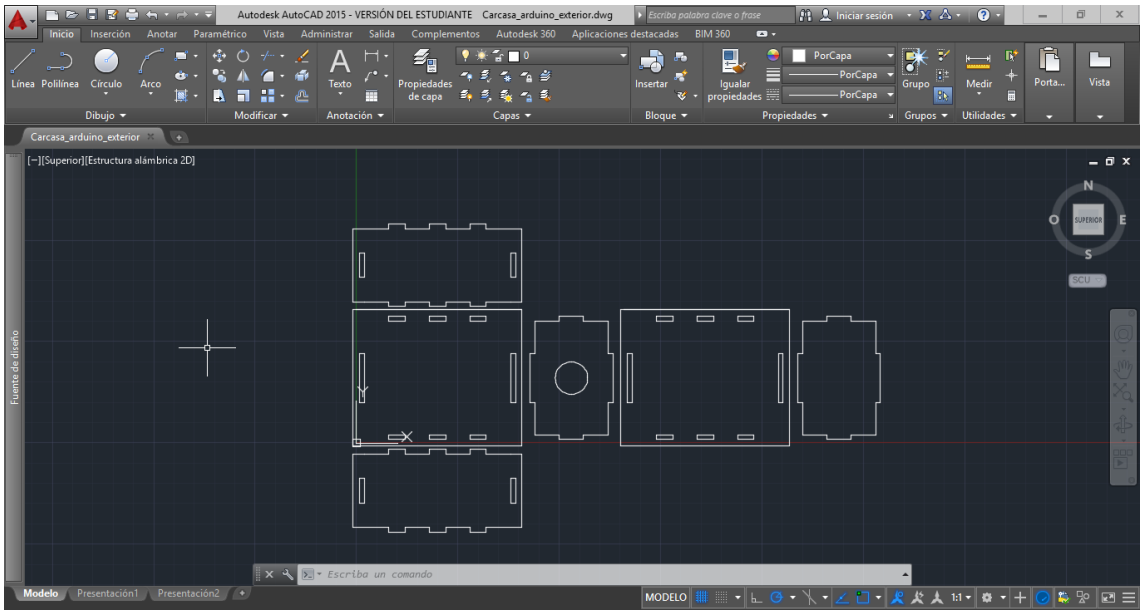


Ilustración 34. Diseño del encapsulado para estación exterior en AutoCAD.

En la ilustración 37 vemos el encapsulado que habíamos diseñado anteriormente. También se ha impreso un encapsulado para el Arduino UNO coordinador, sin embargo, en este caso se ha utilizado una plantilla predeterminada.

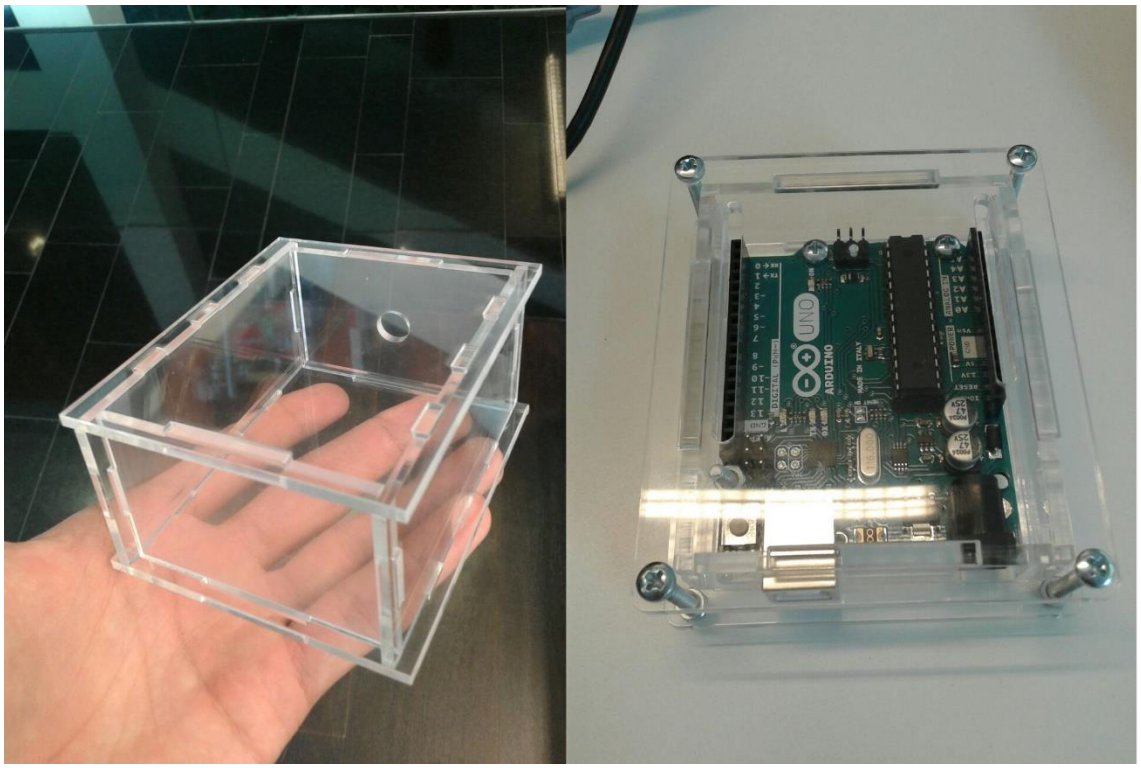


Ilustración 35. Encapsulados para Arduino UNO coordinador y Arduino UNO exterior.

CAPÍTULO 6. IMPLEMENTACIÓN

6. Implementación

6.1. Esquema de montaje

El montaje lo podemos dividir en tres estaciones, cada una controlada por un Arduino. Dos de estas tres estaciones, funcionan a modo de routers, enviando la información mediante comunicación inalámbrica al coordinador, que es quien maneja toda la información, enviándola al sistema de recopilación de datos. El montaje de cada estación es el siguiente:

1. **Estación coordinadora.** Este microcontrolador estará conectado al ordenador administrador, para realizar el volcado de datos de forma efectiva. Tiene conectados los siguientes elementos:
   - MQ135 (2)
   - DHT22
   - TSL2561
   - NRF24L01
   - XBEE S2C
   - ESP8266

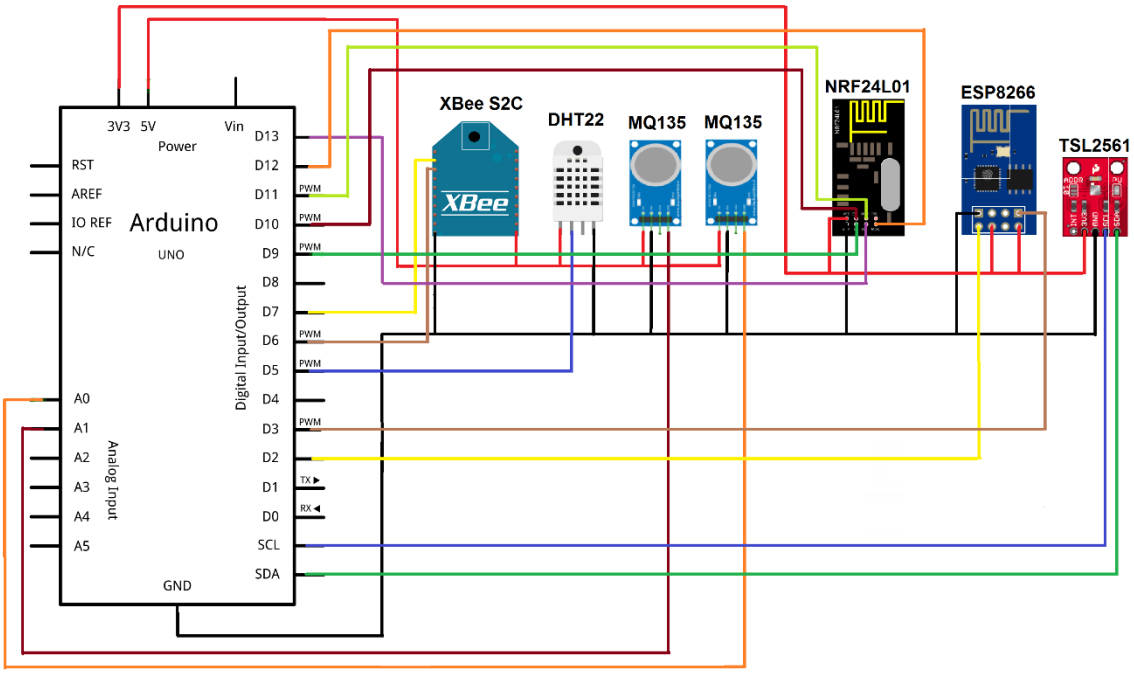


Ilustración 36. Conexionado de la estación que funciona como coordinador.

2. **Estación router interior.** Este microcontrolador se sitúa en el interior del edificio, para estudiar algunas variables en diferentes lugares del edificio. Tiene conectados los siguientes elementos:
   - TGS823
   - TSL2561
   - NRF24L01

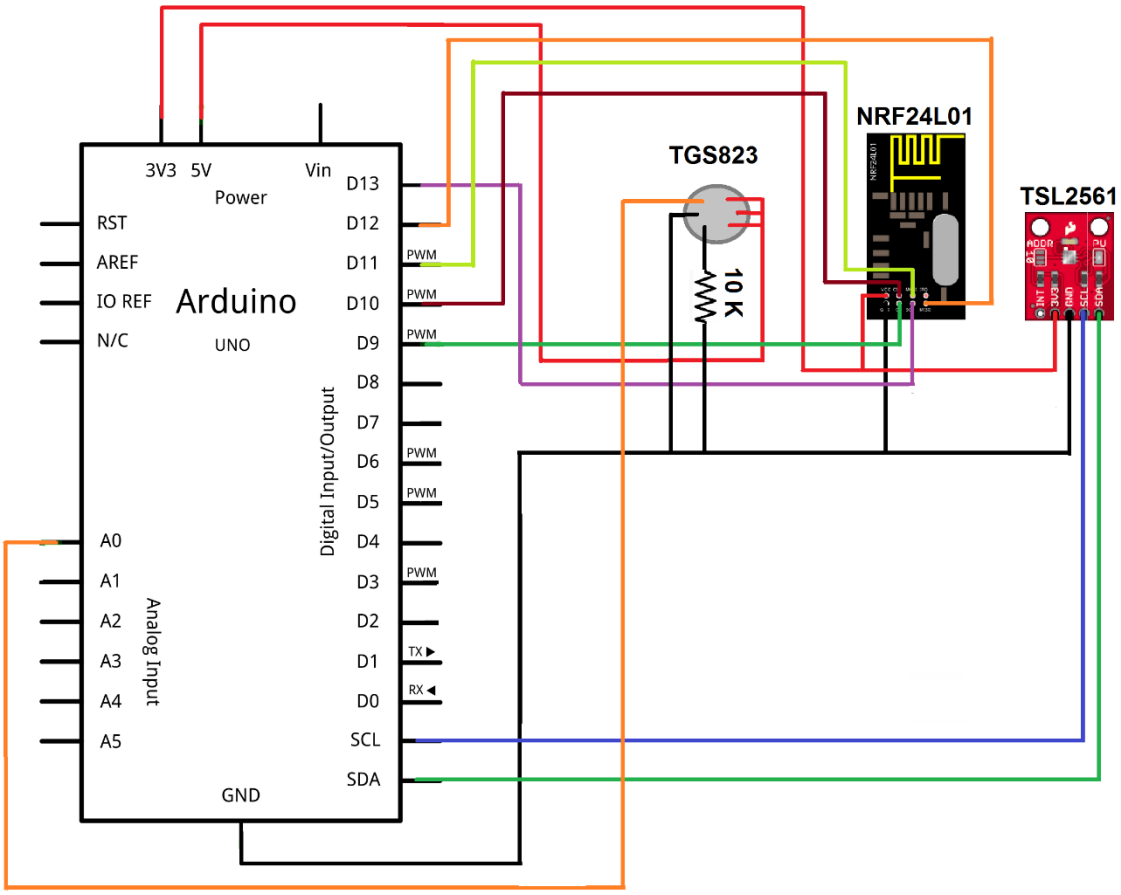


Ilustración 37. Conexionado de la estación que funciona como router interior.

3. **Estación router exterior.** Este microcontrolador se fija en el exterior del edificio, con el objetivo de conocer los parámetros de nuestro interés en dicha zona. Tiene conectados los siguientes elementos:
   - MQ135
   - DHT22
   - DS18B20
   - TGS823

- XBEE S2C

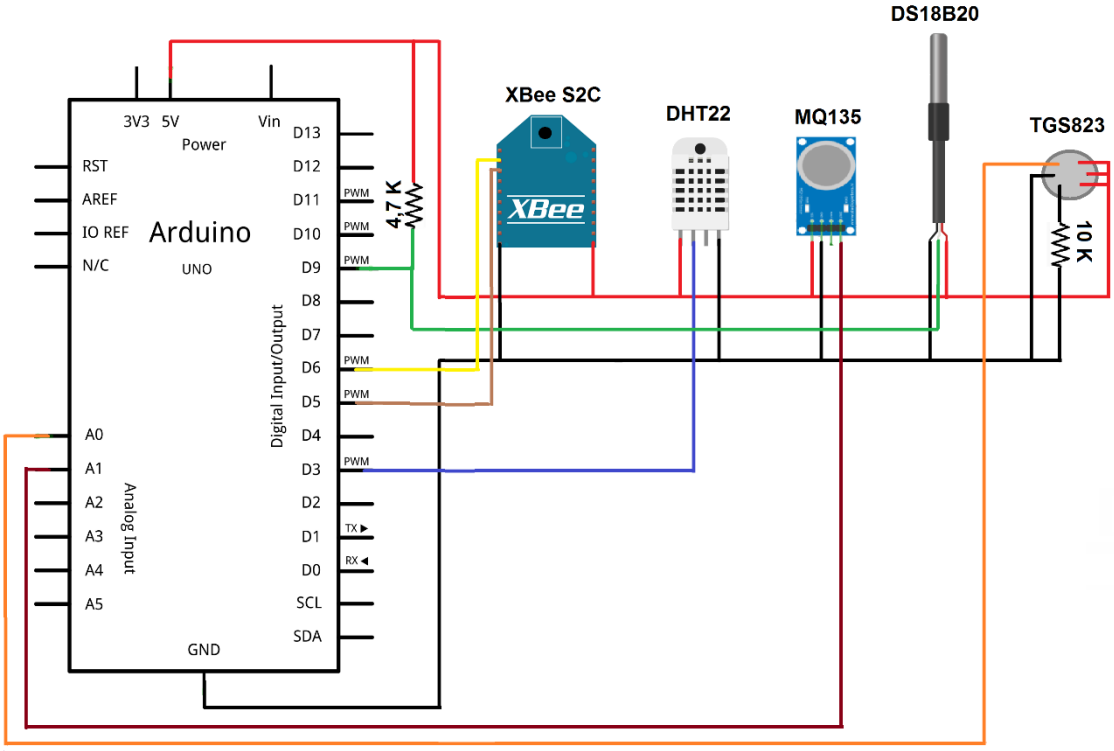


Ilustración 38. Conexionado de la estación que funciona como router exterior.

6.2. Aplicación de monitorización de datos: CONEFI

Conefi, es una aplicación diseñada por parte de la empresa para introducir en ella todos los datos obtenidos de los sensores que han sido instalados, para la monitorización y visualización de los resultados mediante gráficos. En esta aplicación encontramos dos vías de monitorización:

1. **Educa Conefi.** Es el programa web que emplearemos para subir los datos a la red, donde podremos recopilar, en una serie de gráficos, los resultados correspondientes a cada sensor que estemos empleando, para observar los resultados en tiempo real y desde cualquier dispositivo. También tenemos la posibilidad de descargar los resultados que se han obtenido en un periodo de tiempo determinado.
2. **Conefi.** Este programa trabaja en local, es decir, solo podremos visualizar los datos desde el dispositivo que funciona como administrador, sin embargo, posee una ventaja significativa con respecto a Educa Conefi, que consiste en la aportación de lógica e inteligencia y una forma más cómoda de visualización de datos. Por ejemplo, si se

registran temperaturas más elevadas dentro del edificio que fuera, siempre y cuando no estemos niveles de confort, Conefi nos recomendará abrir las ventanas, para evitar el uso de los sistemas de ventilación. Para ello se disponen de líneas orientativas dentro de los gráficos, que delimitan los límites del confort.

En líneas generales, Conefi es una aplicación bastante más potente que Educa Conefi, sobretodo por la aportación de inteligencia a los datos obtenidos, pese a que no esté habilitada para trabajar con ella on-line desde cualquier lugar. Por este motivo es que vamos a volcar todos los datos únicamente en la aplicación de Conefi.

6.2.1. Conefi

En lo que se refiere a la aplicación Conefi, podemos distinguir tanto el área de administración como la de funcionamiento, sin embargo, nosotros nos vamos a centrar en la segunda. El procedimiento que seguimos para configurar la aplicación web conforme a nuestros requerimientos es el siguiente:

1. **Ejecución e identificación en Conefi.** En primer lugar, ejecutamos Conefi y se nos abrirá la página de la aplicación web, como vemos en la ilustración 41, donde se debe completar el formulario de inicio de sesión. Una vez identificados vamos a estudiar las diferentes opciones de la barra de menús.

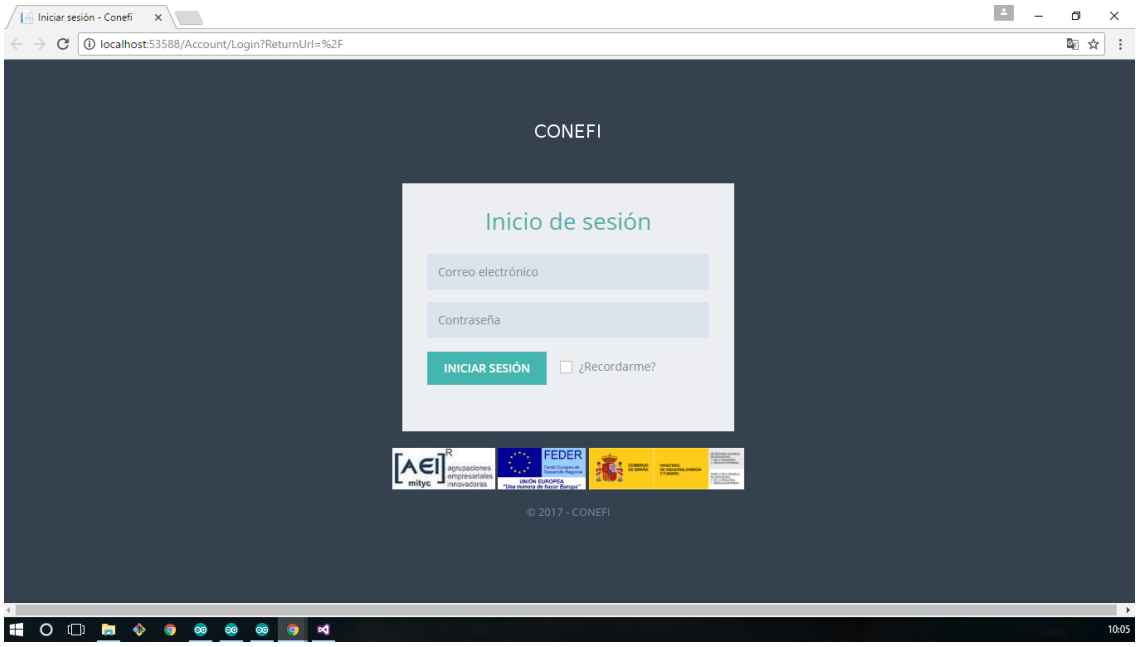


Ilustración 39. Página de inicio de la aplicación Conefi.

2. **Configuración inicial.** En este paso, debemos aportar toda la información relacionada con el edificio, desde lo más general a lo más concreto, de lo contrario realizar la monitorización no sería posible. La información que cumplimentar es la siguiente:

   Datos del edificio. Debemos introducir todos los detalles del edificio en general, relacionados con la ubicación, zona climática o el consumo y potencia. Una vez introducidos estos datos, se genera de forma automática un identificador para este edificio (podremos ver en la pestaña del panel de control) que debemos introducir para la subida de datos a la página.

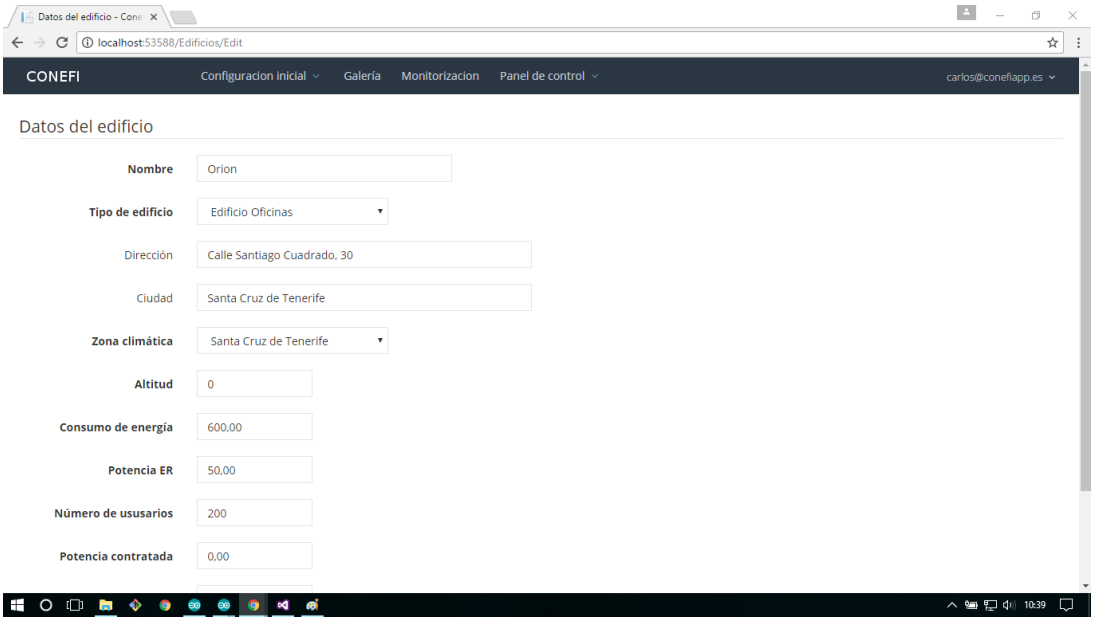


Ilustración 40. Datos del edificio.

Parámetros de la estructura. Comprende las plantas, estancias, huecos, cubiertas, muros y suelos. Cada uno de estos elementos que cabe diferenciar, debemos detallarlos rellenando los formularios que se proponen, para poder ofrecer las soluciones conforme a la disponibilidad de cada estancia en concreto.

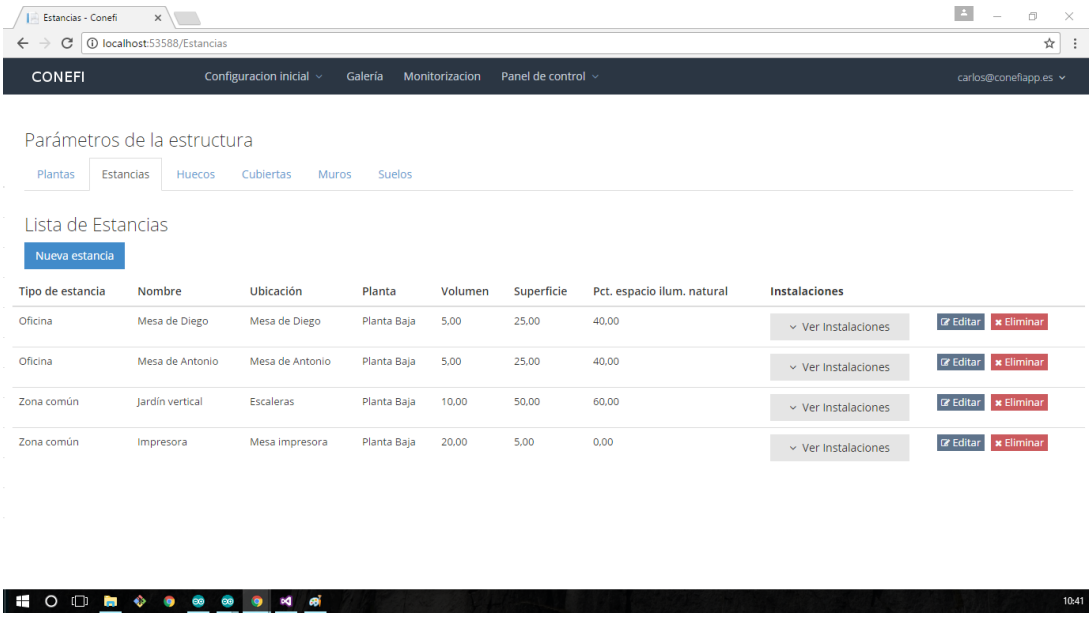


Ilustración 41. Estancias del edificio.

Parámetros de las instalaciones. Aquí debemos indicar todas aquellas prestaciones que ofrece la infraestructura, considerando instalaciones de ACS, refrigeración, calefacción o energías renovables, entre otros.

Parámetros de los sensores. Finalmente, introducimos todos los sensores que hayan sido instalados, atendiendo a tres categorías: sensores de confort, consumo y salud. Igual que con el edificio, debemos crear un identificador para cada sensor, para que los datos se suban conforme al sensor correspondiente. En la instalación piloto solo hemos empleado sensores de confort y salud.

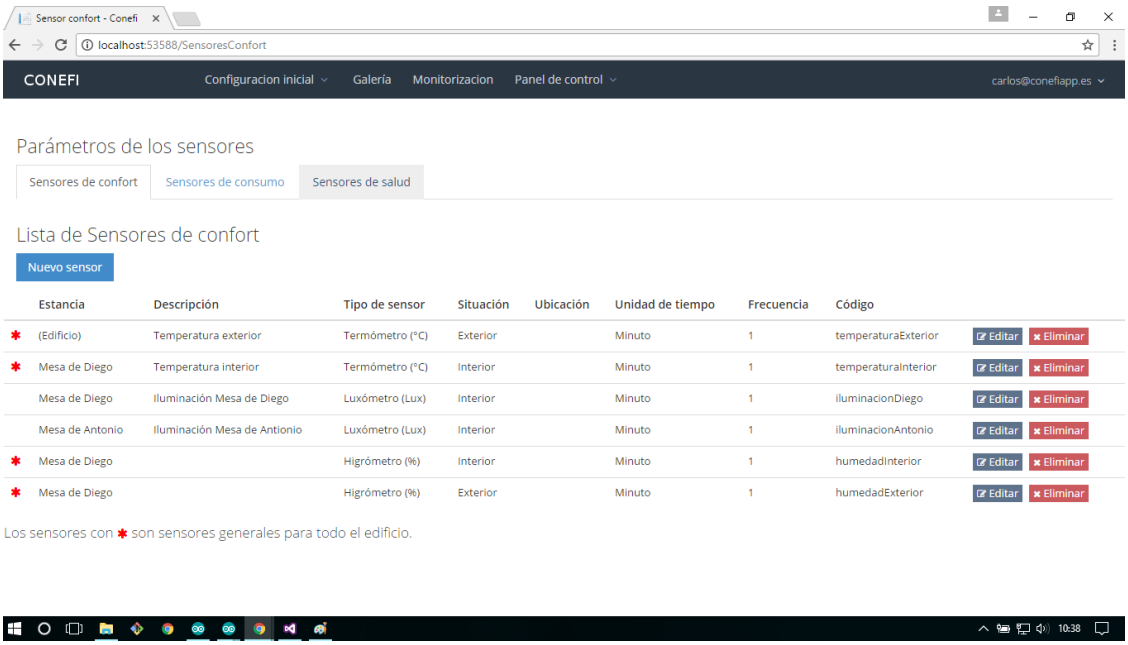


Ilustración 42. Lista de sensores de confort.

Los sensores que se muestran con un asterisco en la ilustración 44 significa que representan valores generales, ya sea para el interior o el exterior.

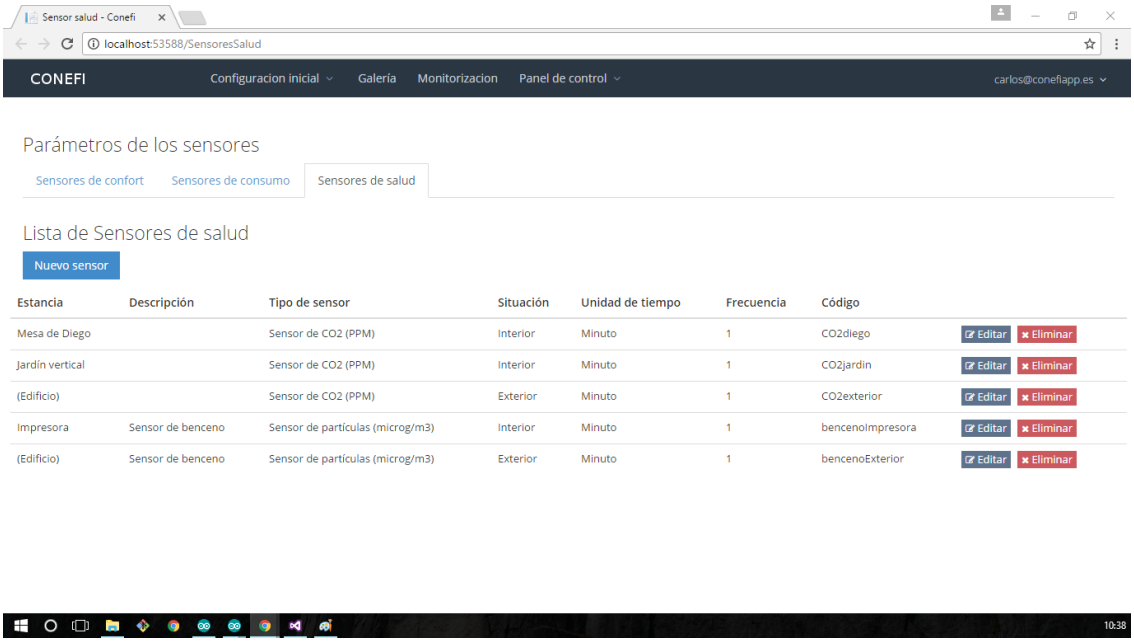


Ilustración 43. Lista de sensores de salud.

3. **Galería.** En este apartado, se ofrece la posibilidad de aportar planos de cada una de las plantas del edificio, permitiéndonos fijarle un número ilimitado de etiquetas asociadas.

4. **Monitorización.** Consiste en la monitorización en tiempo real de las lecturas realizadas por los sensores, que son enviadas a la aplicación. Por pantalla podremos seleccionar los datos que queremos visualizar, teniendo además un sistema de alarmas que nos informa de posibles situaciones en las que la aplicación considera que alguna variable actúa fuera de los niveles de confort/salud, aportando alguna medida correctora al respecto.

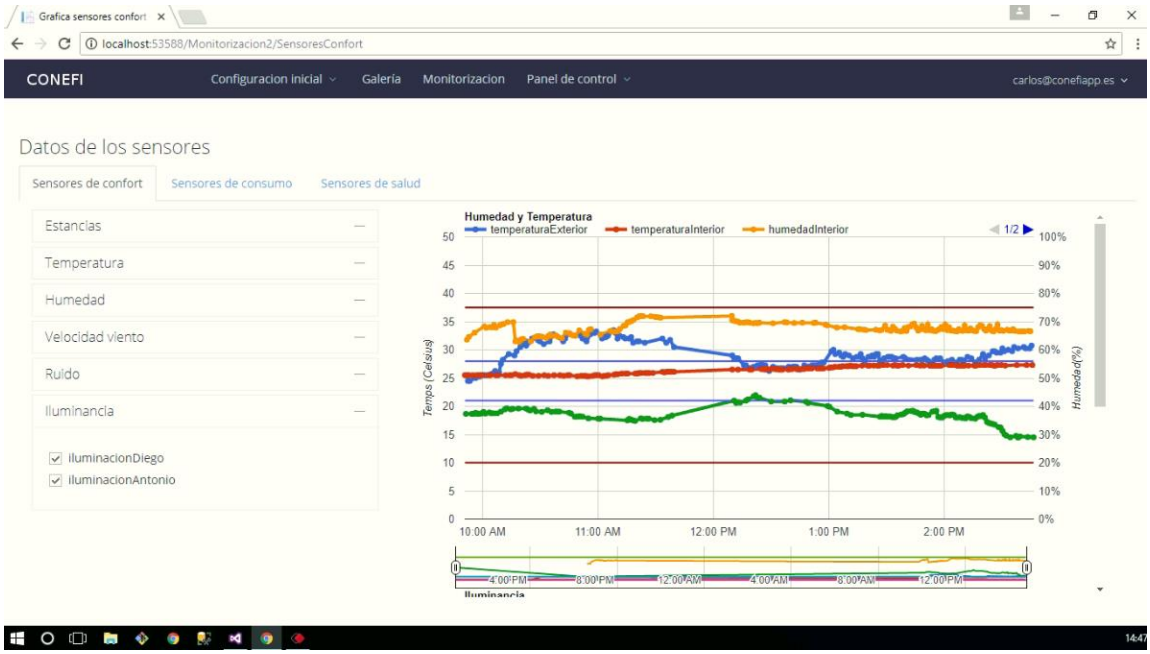


Ilustración 44. Resultados de monitorización empleando Conefi.

5. **Panel de control.** En esta opción podemos realizar una de las siguientes operaciones:

   Informes. En este punto se aporta un listado de informes descargables, en formato PDF, de la monitorización realizada para cada uno de los sensores. También se añade el nivel de importancia de cada uno de los informes que se han creado.

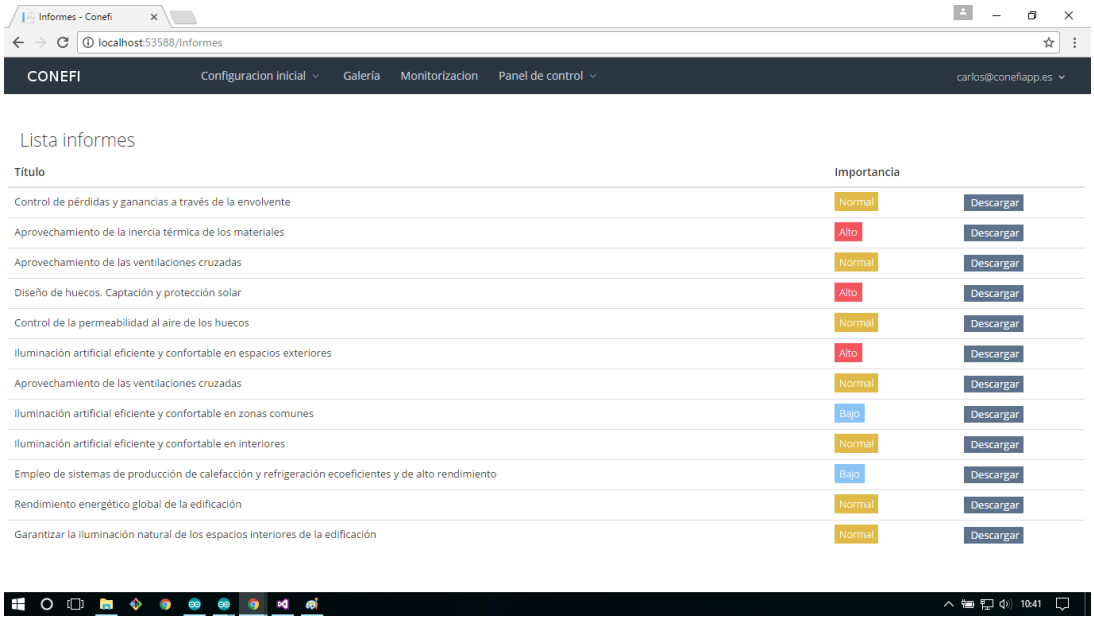


Ilustración 45. Lista de informes descargables de Conefi.

Notificaciones. En esta opción podemos visualizar las alarmas que se hayan producido en tiempo real. Una vez se haya confirmado la visualización de una alarma, esta desaparece de la lista.

Configuración. En este apartado podemos visualizar el identificador del edificio, como habíamos comentado.

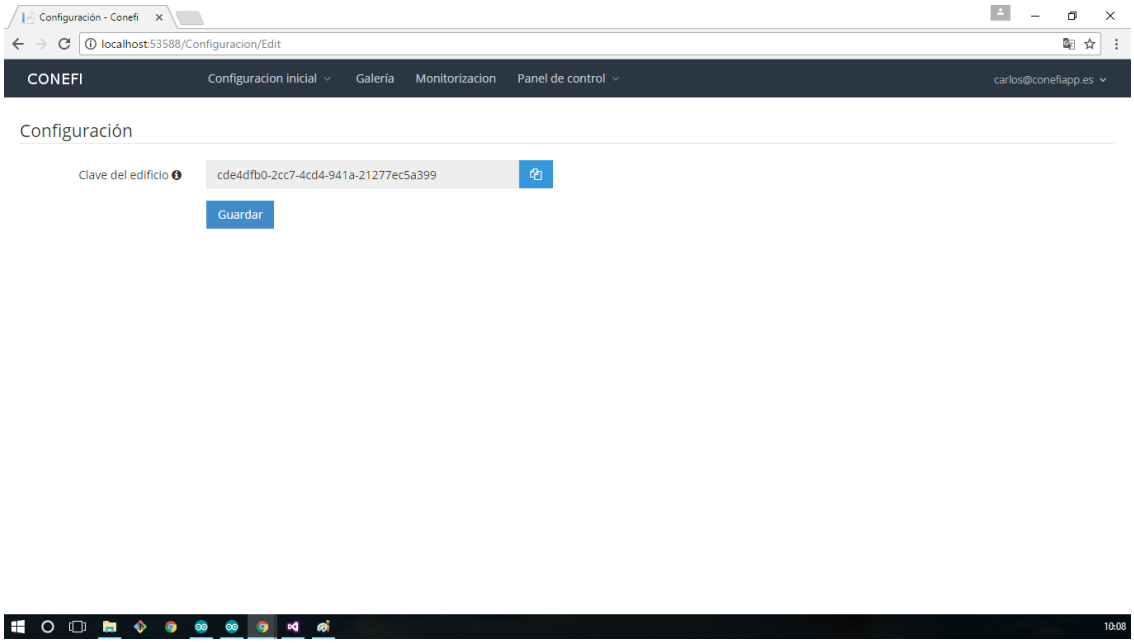


Ilustración 46. El Identificador del edificio la podemos ver en la pestaña de configuración.

Finalmente, cabe comentar que para que la monitorización se produzca de forma lo más cómoda y visual posible para el usuario, la aplicación Conefi hace uso de una llamada AJAX (Asynchronous JavaScript XML) al servidor, de forma que la página se actualiza sin necesidad de realizar esta operación manualmente.

6.2.2. Educa Conefi

Esta es la herramienta web que emplearemos para subir los datos a la red y poder visualizarlos desde cualquier lugar. En este caso, como ocurre con Conefi, también disponemos del área de usuario y el área de administración. Así el funcionamiento de esta aplicación se describe de la siguiente manera:

1. **Área de usuario.** La pantalla principal de Educa Conefi es la que se muestra en la ilustración 49, en la que, según el instituto que seleccionemos, podemos ver los resultados de la monitorización para los parámetros de temperatura, humedad e iluminación, pudiendo hacer una comparación de resultados en diferentes colegios de la isla. También existe la posibilidad de descargarnos un archivo con los datos en formato .CSV (valores separados por comas) donde se distribuyen todos los datos separados generalmente por ';' cuando se trata de números decimales, como sucede en este caso.

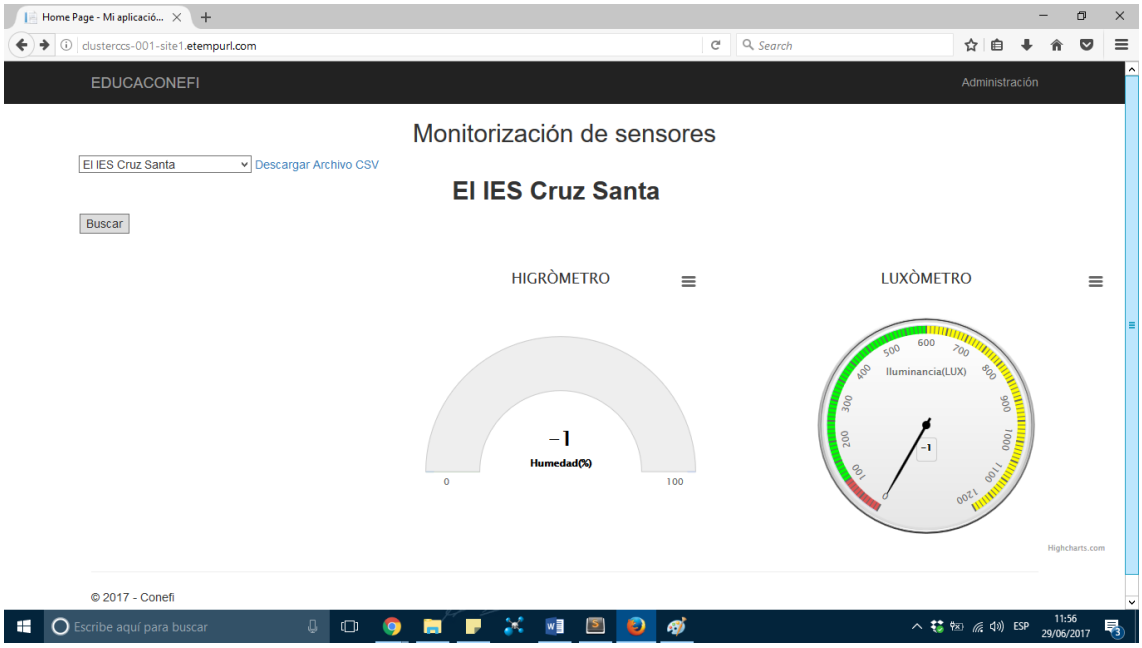


Ilustración 47. Pantalla de inicio de Educa Conefi.

Para cada uno de los parámetros disponemos de un menú en el que podemos seleccionar diferentes vías de descarga del cuadro, para obtener un dato en un momento determinado de forma gráfica. Los métodos disponibles son los siguientes:

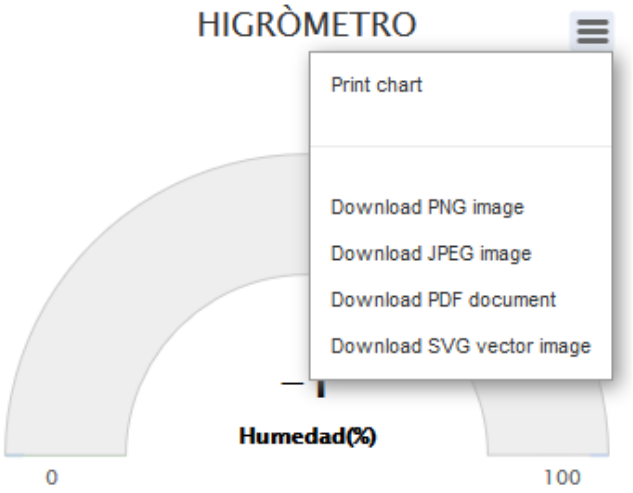


Ilustración 48. Menú de descarga de un cuadro de datos.

Para poder subir los datos al lugar correspondiente, el administrador es quien debe suministrar los identificadores, tanto del instituto como de cada sensor que se desee programar. Todos los sensores pueden ser creados o eliminados por el administrador, esta tarea nunca recalará sobre el usuario que se limita a subir, visualizar y comparar datos.

2. **Área de administración.** En la pantalla de inicio de Educa Conefi, observamos una opción de 'Administración', clicando en esa opción nos aparece el formulario de inicio de sesión para el administrador.

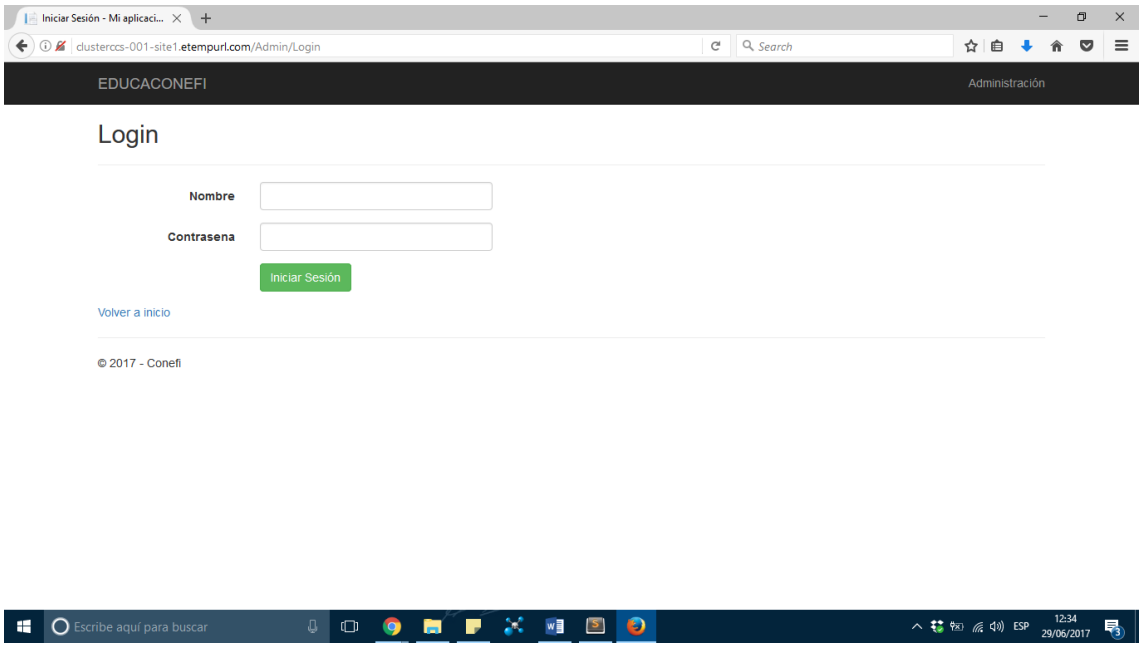


Ilustración 49. Formulario de inicio de sesión como administrador.

Las tareas como administrador residen en la capacidad para crear/eliminar sensores, y ver el número ID de cada uno de los sensores. Para crear un sensor clicamos en la opción 'crear' y rellenamos el formulario de acuerdo con nuestros requerimientos, y finalmente, lo creamos.

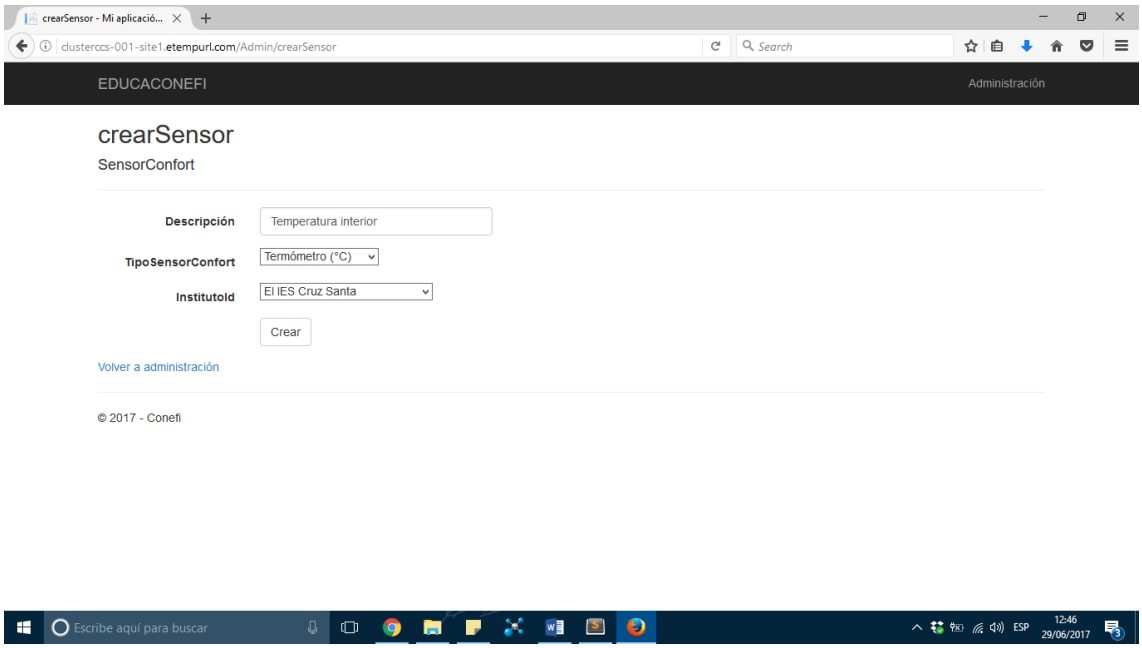


Ilustración 50. Creación de un sensor nuevo en la aplicación Educa Conefi.

6.3. Código empleado

En este apartado vamos a comentar algunas líneas del código que hemos empleado, haciendo hincapié en algunas conversiones de datos o envío de señales que puedan resultar confusas, no obstante, el mismo cuerpo del código está dotado de comentarios suficientes para entender el contenido del mismo.

En primer lugar, destacar el empleo de dos librerías que han sido creadas y diseñadas específicamente para esta aplicación y que están basadas en las funciones que obtuvimos para ambos casos después de analizar sus curvas características.

```
//Incluimos las librerias
#include "XBee.h"
#include "SoftwareSerial.h" //Para desplazar el Tx y Rx a otros pines
#include "DHT.h"
#include "OneWire.h"
#include "DallasTemperature.h" //Libraria para el sensor DS18B20
#include "MQ135.h"
#include "TGS823.h"
```

Ilustración 51. Las librerías MQ135.h y TGS823.h han sido creadas para este proyecto.

En el Arduino que emplea un módulo XBee en modo transmisor necesitamos obtener los datos en hexadecimal, dado que la trama maneja tramas en hexadecimal. Para ello debemos convertir los datos para poder enviarlos. Para ello empleamos las siguientes líneas de código debidamente explicadas con un ejemplo:

```
//Dividimos el resultado en dos paquetes de 8 bits completando con 0.
//*****Ejemplo con el valor 650 para el pin A5: En binario 1010001010*****
datos[0] = CO2 >> 8 & 0xFF; //Desplazamos 8 pos. a la dcha.
//Completamos con 0 -> [00000010]. Y convertimos de binario a hexadecimal:
//[nº binario] * FF -> [00000010] * FF = 02 -> (4 BIN equivalen a 1 HEX)
datos[1] = CO2 & 0xFF; //[10001010] * FF = [1000] * F + [1010] * F = 8A
//Finalmente los concatenamos obteniendo: 02 8A
```

Ilustración 52. Código para pasar de decimal a hexadecimal, envío de datos por los módulos XBee S2C.

Posteriormente, cuando estos datos llegan al Arduino con el XBee S2C receptor, este valor debe volver a pasarse a decimal, para ello basta con realizar la siguiente operación:

$$decimal = \ dato[0]\ x\ 256 + dato[1]$$

Un caso similar al explicado anteriormente ocurre con el sensor TSL2561, donde se obtiene una trama 32 bits, y debemos descomponerlos en dos tramas: los 16 LSB, multiplicando por 0xFFFF, y los 16 MSB, desplazando el número binario en 16 posiciones. Posteriormente podemos calcular los luxes.

```
//obtenemos la luminosidad que detecta el TSL2561
//Los datos aparecen en una trama de datos, empleamos un uint32_t para recogerlos todos
//Un uint32_t equivale a 32 bits.
uint32_t luminosidad = TSL2561.getFullLuminosity();

//Ahora dividimos el numero en 2 partes iguales de 16 bits, es decir,en dos uint16_t:
//1. Multiplicamos el n en binario por FFFF para obtener los 16 LSB
uint16_t luminosidad_LSB = luminosidad & 0xFFFF;
//2. Desplazamos 16 posiciones a la derecha obteniendo los 16 MSB
uint16_t luminosidad_MSB = luminosidad >> 16;

//Calculamos los luxes introduciendo los dos paquetes de datos anteriores
float luxes = TSL2561.calculateLux(luminosidad_LSB, luminosidad_MSB);
```

Ilustración 53. Código para el cálculo de luxes con el TSL2561.

En el código empleado para el Arduino coordinador encontramos algunos aspectos importantes para entender correctamente el razonamiento desarrollado. Para empezar, como se emplea la librería softwareSerial en dos ocasiones para los pines de comunicación del XBee y del ESP8266, debemos tener en cuenta que solamente puede estar una de ellas en funcionamiento de forma simultánea. Por este motivo es que iniciamos y finalizamos los pines de comunicación de forma ordenada, como vemos en la ilustración 56.

También podemos observar que el módulo ESP8266 trabaja de forma predeterminada a 115200 baudios, muy por encima de los 9600 que suelen emplear, aunque también podría trabajar a esta velocidad. En este sentido, no se debe es aumentar la velocidad de 115200 baudios para trabajar con Arduino, dado que podría crear problemas de estabilidad.

```
TxRx.end();
esp8266.begin(115200);
Exterior(temperaturaExterior,humedadExterior,CO2exterior, BENCENOexterior);
esp8266.end();
```

Ilustración 54. Inicio y fin de los pines de comunicación con el esp8266 trabajando a 115200 baudios.

Para subir los datos a la red debemos introducir por método GET la dirección: http://192.168.10.215/Api/SensoresValores?edificioId=x&sensorId=y&valor=z, donde 'x' es la Id del edificio, 'y' es la Id del sensor y 'z' es el dato del sensor. Un aspecto importante está relacionado con la forma de subida de datos, nosotros empleamos el método GET, no obstante, podríamos haber optado por emplear el método POST. El método GET maneja datos de forma visible, mientras que el método POST lo hace coultándolos, es decir, los datos van encriptados de tal forma que no son visibles para el usuario. La diferencia principal entre ambos reside en el nivel de seguridad.

Centrándonos en el formato del link, observamos la presencia de tres caracteres de ordenación:

- **?.** Este carácter se emplea siempre después de la dirección, después del mismo procedemos a introducir toda la información.
- **=.** Este signo se sitúa siempre entre una variable y su resultado.
- **&.** Este carácter se coloca siempre para separar una variable y su resultado del siguiente.

Los datos recogidos tienen que ser subidos desde un equipo/dispositivo que esté conectado al servidor que administra el programa, de lo contrario no se detectará ningún cambio en la página. Para ello hemos empleado el código que se muestra en la ilustración 57, donde poco a poco vamos creando el String del link hasta poder enviarlo al módulo que es quien direcciona los datos a la página para actualizar los datos.

```
String direccion = "AT+CIPSTART=\"TCP\",\"";     // Conexión TCP (http://)
direccion += "192.168.10.215";                   // Identificador del servidor
direccion += "\",80";                            // Puerto 80
ESP8266.println(direccion);                      // Enviar al ESP8266
// Envio de datos en método GET + Identificador del edificio y del sensor
String paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-21277ec5a399";
paquete +="&sensorId=temperaturaInterior&valor=";
paquete += double(temperatura);                  // Dato del sensor
paquete += "\r\n\r\n";
direccion = "AT+CIPSEND=";                       // Comando para enviar la longitud de la trama
direccion += String(paquete.length());
ESP8266.println(direccion);
ESP8266.print(paquete);                          // Enviar el paquete al ESP8266
```

Ilustración 55. Código de direccionamiento de un link empleando Arduino y ESP8266.

CAPÍTULO 7. RESULTADOS DE MONITORIZACIÓN

7. Resultados de monitorización

En este apartado vamos a realizar un pequeño estudio de los resultados obtenidos de la aplicación Conefi para los sensores previamente instalados, dividiendo la monitorización a su vez en 2 grandes bloques: parámetros de confort, y parámetros relacionados con la salud.

7.1. Variables de confort

1. Temperatura y humedad:

En la ilustración 58, podemos observar el comportamiento de la temperatura y humedad interior y exterior, entre las 10 y las 15 horas, dividiendo la misma en tres partes muy diferenciadas:

10:00 – 11:30 AM. Observamos como a medida que pasa el tiempo aumenta la temperatura exterior de forma mucho más rápida que la interior, la humedad exterior comienza a disminuir y la exterior aumenta, debido a la existencia de un jardín vertical que mantiene siempre la humedad elevada en el interior. No obstante, las líneas delgadas azules, para la temperatura, y rojo, para la humedad, indican los límites de confort, de forma que el único parámetro que excede estos límites es la temperatura exterior.

11:30 – 14:30 AM. A partir de esta hora comienza un periodo de lluvia leve, de forma que la temperatura exterior baja considerablemente, de forma inversamente proporcional a la humedad exterior, que sube con motivo de la presencia de ambiente húmedo. En el interior, los parámetros se mantienen constantes.

14:30 – 15:00 AM. En el momento en el que deja de llover y vuelve a salir el sol, vemos como la temperatura exterior comienza a subir nuevamente y la humedad exterior vuelve a reducirse. La temperatura interior alcanza el límite superior de confort, con lo que deberíamos encender el aire acondicionado, por ejemplo.

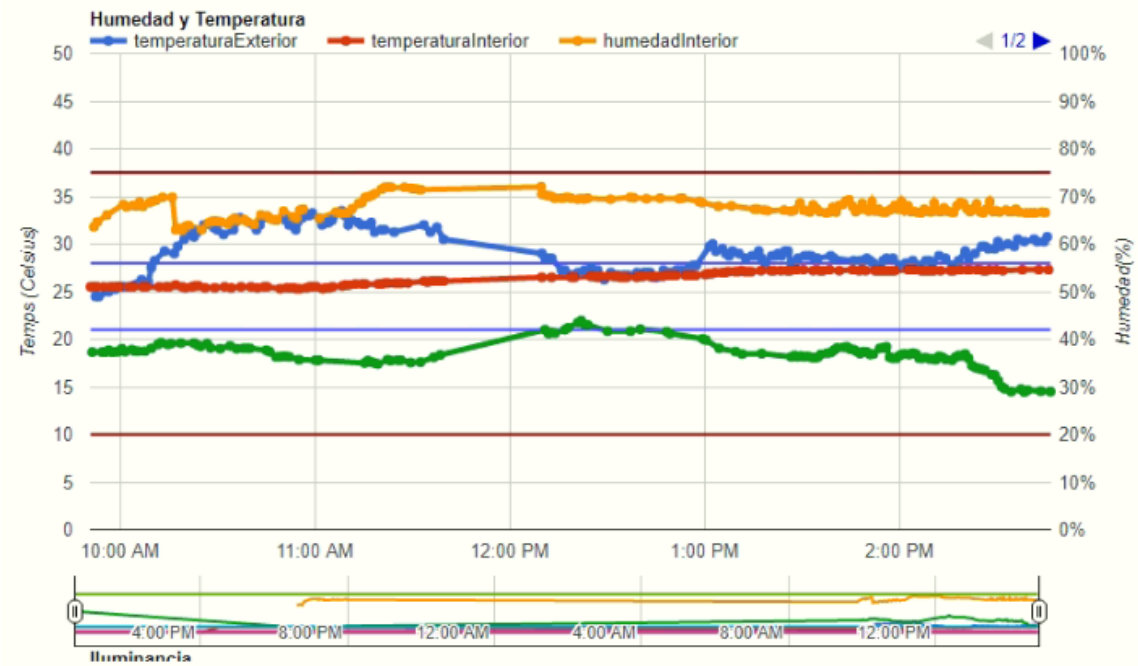


Ilustración 56. Monitorización de temperatura y humedad.

2. Iluminación:

Para estudiar los niveles de iluminación hemos seleccionado el transcurso de una tarde desde las 6 hasta las 8 PM. A lo largo de todo este periodo de tiempo el nivel de iluminación se va reduciendo constantemente hasta la hora de cierre de la oficina, donde también podemos analizar como “iluminacionDiego” tiende a ser superior a “iluminacionAntonio”, debido a que el primero dispone de mejores condiciones de iluminación natural. No obstante, en todo momento se mantienen los niveles por encima del mínimo de 300 lux que exige la norma UNE 12464-1.

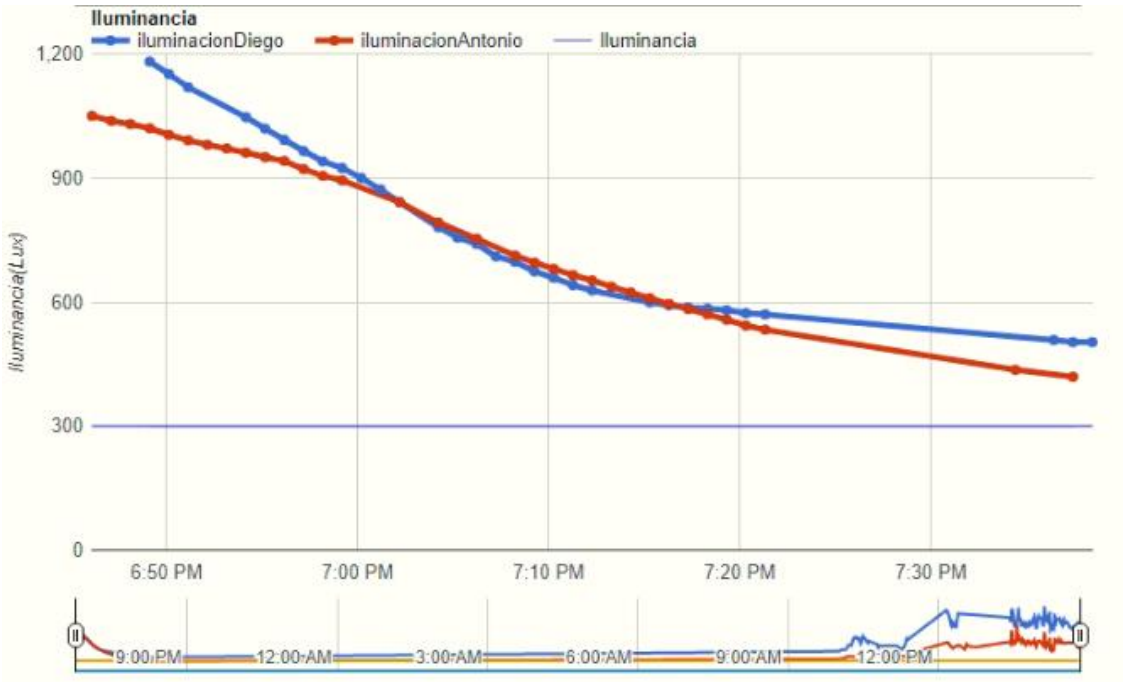


Ilustración 57. Monitorización de iluminación.

7.2. Variables de salud

1. Dióxido de carbono:

Los resultados de dióxido de carbono no son del todo precisos, pero si lo suficientemente orientativos como para poder estudiarlos, de esta forma, podemos observar que el CO_2 exterior se mantiene siempre por encima del nivel interior, bastante comprensible tratándose de un lugar con mucho tránsito de tráfico. En cuanto a las dos medidas de CO_2 interior, vemos que el nivel promedio ronda los 390-400 PPM.

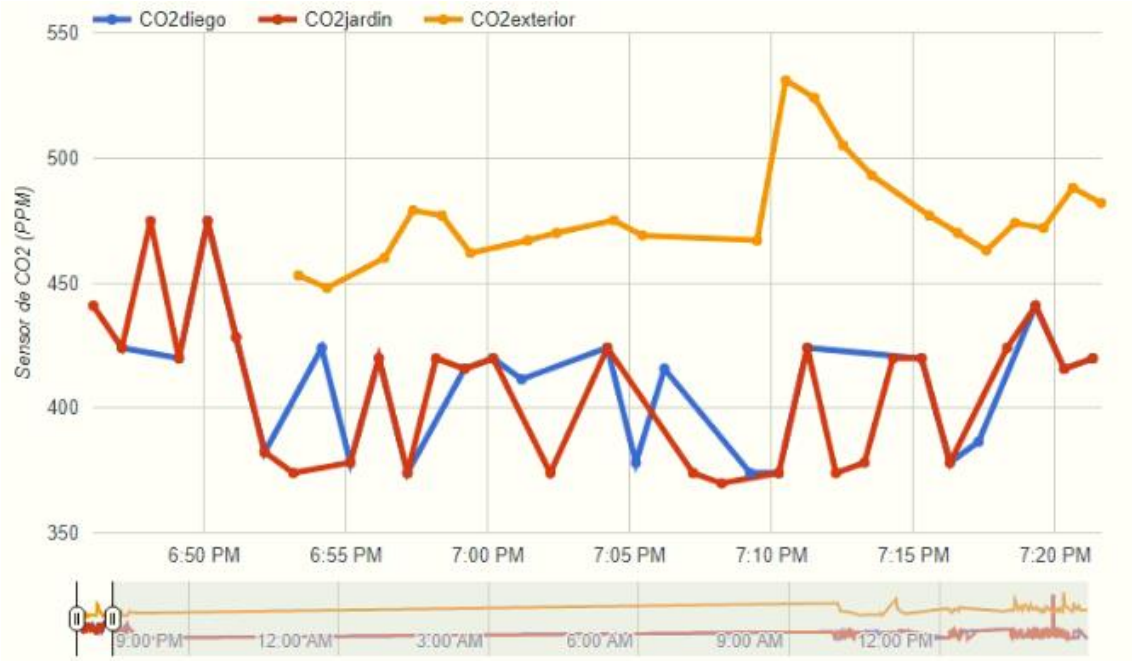


Ilustración 58. Monitorización del dióxido de carbono.

2. Benceno:

Finalmente, el nivel de benceno lo vamos a estudiar considerando un nivel aproximado de 20 PPT o $\mu g/m^3$. Realizando la comparación, vemos como el nivel de benceno interior es inferior al que habíamos fijado, igual que el exterior a excepción de un pequeño intervalo, por lo que podemos considerar estas mediciones como muy aceptables, al no acercarse ni siquiera a las 50 $\mu g/m^3$.

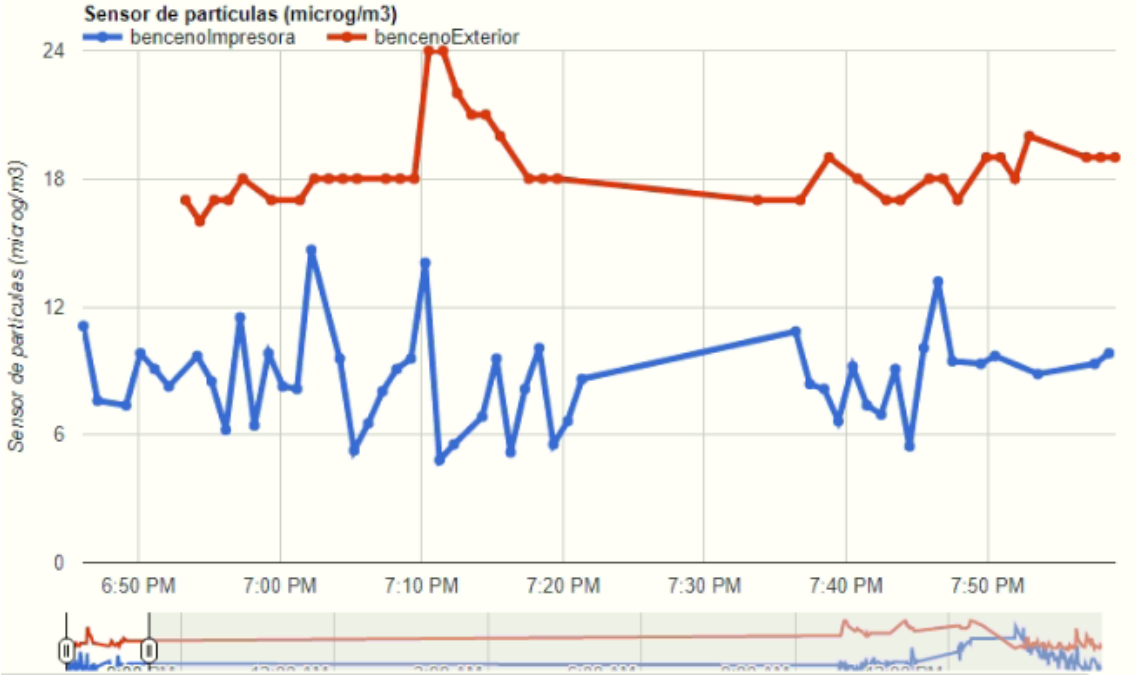


Ilustración 59. Monitorización del nivel de benceno.

CAPÍTULO 8. CONCLUSION

8. Conclusion

The results obtained in this project, based on a construction of an Smart Building as a pilot project, are satisficing after seeing the graphics obtained in the monitoring window, because we are able to orient ourselves with the values obtained from all the sensors in real time.

In this way, the materials budget of this low-cost model has been below the two hundred euros, so we can talk about favorable results for taking real the main objective project. Moreover, we could test some of the wireless connections, like radiofrequency, ZigBee or WiFi, having some information about their quality in parameters which are for example: range or capacity of transmitting information.

Personally, this project has been interesting for me because I had the opportunity to acquire some experience working on a real project with a company, learning a lot of things about the monitoring of the variables extracted from all the sensors that we had installed.

In conclusion, monitoring the comfort and healthy variables using different sensors, hardware and software systems, is a good way for introducing us in Smart Buildings. Without any doubt, now we could start thinking about to create a home automation system based on this way of monitoring employing all these sensors.

CAPÍTULO 9. BIBLIOGRAFÍA

9. Bibliografía

[1]https://github.com/

[2]https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwiiqoj_4OjUAhWmF8AKHahMC70QFggnMAA&url=http%3A%2F%2Fwww.editex.es%2FRecuperarFichero.aspx%3FId%3D19543&usg=AFQjCNE7kgAAE_FbLEMIX_AmkeQWsOh-8g

[3]https://www.knx.org/media/docs/Flyers/KNX-Solutions/KNX-Solutions_es.pdf

[4]http://diymakers.es/sensores-en-entradas-analogicas-de-arduino/

[5]https://www.theben.es/Espana/Inicio/Informacion-actual/Sabia-que/Automatizacion-de-viviendas-y-de-edificios-KNX/KNX-tecnologia

[6]http://srcsl.com/que-es-un-sensor-pt100/

[7]http://www.arian.cl/downloads/nt-004.pdf

[8]http://informesdelaconstruccion.revistas.csic.es/index.php/informesdelaconstruccion/article/viewArticle/3259/3665

[9]http://eie.ucr.ac.cr/uploads/file/proybach/pb_08_II/pb0811t.pdf

[10]http://slideplayer.es/slide/3178592/

[11]https://es.wikipedia.org/wiki/ZigBee#Ejemplos_de_dispositivos_Zigbee

[12]http://www.heremeter.com/general-introduction-about-zigbee.html

[13]https://aprendiendoarduino.wordpress.com/tag/xbee/

[14]http://xbee.cl/que-es-xbee/

[15]https://www.luisllamas.es/

[16]https://es.wikipedia.org/wiki/Wikipedia:Portada

[17] https://forum.arduino.cc/

CAPÍTULO 10. ANEXOS

10. Anexos

I. Código Arduino "Router Exterior"

```
/* Este es el código que implemtaremos en el Arduino NANO que ejercerá las funciones
 * de transmisor de datos desde el exterior del edificio piloto empleando conexión
 * inalámbrica (ZigBee) para enviar al Arduino coordinador todos los datos obtenidos
 * de las mediciones realizadas para la comparación de datos dentro-fuera. Los elementos
 * atribuídos a este microcontrolador son:
 *     -DHT22
 *     -MQ135
 *     -TGS823. En PPT
 *     -XBee S2C. (router)
 * Cabe destacar que para los sensores de gas MQ135 y TGS823 se han diseñado librerias
 * propias basadas en los resultados obtenidos del estudio de las curvas características.
 */
//Incluímos las librerias
#include "XBee.h"
#include "SoftwareSerial.h" //Para desplazar el Tx y Rx a otros pines
#include "DHT.h"
#include "OneWire.h"
#include "DallasTemperature.h" //Libraría para el sensor DS18B20
#include "MQ135.h"
#include "TGS823.h"
//Definimos todos los pines de los componentes
const int Din = 6; //Din se conecta al Dout del XBee
```

```
const int Dout = 5; //Dout se conecta al Din del XBee
const int pinMQ135 = A0;
const int pinTGS823 = A1;
const int pinDHT22 = 3;
const int pinDS18B20 = 9;
//Definimos un paquete de 8 posiciones para enviar los datos de los 4 sensores por XBee
//Cada número decimal empleará 8 posiciones de digitos hexadecimales, de ahí el uint8_t
uint8_t datos[8];
//Definimos la direccion del bus del sensor DS18B20
DeviceAddress ID = { 0x28, 0xFF, 0x7B, 0x8E, 0x65, 0x15, 0x02, 0xA6 };
//Creamos los objetos de las librerías
XBee xBeeS2C = XBee();
SoftwareSerial TxRx(Din, Dout);
DHT dht22(pinDHT22, DHT22);
OneWire busDatos(pinDS18B20);
DallasTemperature DS18B20(&busDatos);
MQ135 MQ135(pinMQ135);
TGS823 TGS823(pinTGS823);
//Definimos la direccion del XBee receptor y los parametros a enviarle
XBeeAddress64 destino = XBeeAddress64(0x0013A200, 0x4154DF0F);
ZBTxRequest trama = ZBTxRequest(destino, datos, sizeof(datos));
void setup() {
  //Iniciamos todos los elementos necesarios
```

```
  Serial.begin(9600);
  xBeeS2C.begin(TxRx);
  TxRx.begin(9600);
  DS18B20.begin();
  dht22.begin();
  //Definimos los pines declarados anteriormente como input o output
  pinMode (pinDS18B20, INPUT);
  pinMode (pinMQ135, INPUT);
  pinMode (pinDHT22, INPUT);
  pinMode (pinDS18B20, INPUT);
  pinMode (Din, INPUT);
  pinMode (Dout, OUTPUT);
}
void loop() {
  //Obtenemos los valores de los 4 sensores instalados
  int CO2 = MQ135.getPPMexterior(analogRead(pinMQ135));
  int BENCENO = TGS823.getPPT(analogRead(pinTGS823)*100);
  int humedad = dht22.readHumidity()*100;
  DS18B20.requestTemperatures(); //Solicitamos una medición para DS18B20
  int temperatura = DS18B20.getTempC(ID)*100;
  //Ahora vamos a convrtir todos los resultados a hexadecimal, para ello:
  //Dividimos el resultado en dos paquetes de 8 bits completando con 0.
  //*****Ejemplo con el valor 650 para el pin A5: En binario 1010001010*****
```

```
datos[0] = CO2 >> 8 & 0xFF; //Desplazamos 8 pos. a la dcha.
//Completamos con 0 -> [00000010]. Y convertimos de binario a hexadecimal:
//[nº binario] * FF -> [00000010] * FF = 02 -> (4 BIN equivalen a 1 HEX)
datos[1] = CO2 & 0xFF; //[10001010] * FF = [1000] * F + [1010] * F = 8A
//Finalmente los concatenamos obteniendo: 02 8A

//Realizamos el mismo proceso con los otros 3 resultados
datos[2] = humedad >> 8 & 0xFF;
datos[3] = humedad & 0xFF;
datos[4] = temperatura >> 8 & 0xFF;
datos[5] = temperatura & 0xFF;
datos[6] = BENCENO >> 8 & 0xFF;
datos[7] = BENCENO & 0xFF;
//Escribimos en el XBee el paquete de datos para que se envie al módulo XBee receptor
xBeeS2C.send(trama);
//Mostramos los resultados en decimal y hexadecimal por el monitor serial
//(En caso de estar conectado el monitor serial)
Serial.print("Benceno: ");
Serial.println(BENCENO);
Serial.print("Temperatura= ");
Serial.print(temperatura); //Temperatura multiplicada por 100
Serial.print(" C -> ");
Serial.print(datos[4], HEX);
```

```
  Serial.print(" ");
  Serial.println(datos[5], HEX);
  Serial.println(" ");
  Serial.print("Humedad= ");
  Serial.print(humedad); //Humedad multiplicada por 100
  Serial.print(" % -> ");
  Serial.print(datos[2], HEX);
  Serial.print(" ");
  Serial.println(datos[3], HEX);
  Serial.println(" ");
  Serial.print("CO2= ");
  Serial.print(CO2);
  Serial.print(" PPM -> ");
  Serial.print(datos[0], HEX);
  Serial.print(" ");
  Serial.println(datos[1], HEX);
  Serial.println(" ");
  //Finalmente, esperamos un minuto entre dos mediciones consecutivas
  delay(60000);
}
```

II. Código Arduino "Router Interior"

```
/* Este es el código que implemtaremos en el Arduino UNO que ejercerá las funciones
 * de transmisor de datos en el interior del edificio piloto empleando conexión
 * inalámbrica (radiofrecuencia), para el envío de las señales al Arduino Coordinador,
 * habiendo analizado estas señales previamente. Los elementos que emplearemos son:
 *     -TSL2561
 *     -TGS823. En este sensor vamos a medir la concentración en PPT, dado que las
 *          concentraciones de benceno son bastante reducidas en ambiente
 *     -NRF24L01. (emisor)
 * Cabe destacar que para el sensor de gas TGS823 se han diseñado una libreria
 * propia basada en los resultados obtenidos del estudio de su curva característica.
 */
//Incluímos las librerías
#include "SPI.h"
#include "RF24.h" //Libreria para el módulo de radiofrecuencia NRF24L01
#include "TSL2561.h"
#include "TGS823.h"
//Definimos el tiempo de integración del TSL2561 en 402ms
#define TSL2561_INTEGRATIONTIME_402MS
//Definimos todos los pines de los componentes
const int pinTGS823 = A0;
const int pinCE = 9;
const int pinCSN = 10;
//Definimos un paquete de datos de dos posiciones donde depositar los valores para enviar
```

```
//En este caso son de tipo float porque no es necesario realizar la conversion a hex.
float datos[2];
//Definimos la dirección de la comunicación de los NRF24L01 en hexadecimal
//Para depositar todos estos números en hexadecimal definimos un uint64_t
//Esta direccion debe ser la misma en los dos módulos
//Lo definimos como constante porque el canal no debe cambiar su ID
const uint64_t ID = 0xCAFE20172018;
//Definimos los objetos de las librerías
RF24 NRF24L01(pinCE, pinCSN); //Especificamos los pines CE y CSN
TSL2561 TSL2561(TSL2561_ADDR_FLOAT); //ADDR no conectado (no aparece en nuestro sensor)
TGS823 TGS823(pinTGS823);

void setup(){
  //Iniciamos todos los elementos necesarios
  NRF24L01.begin();
  Serial.begin(9600);
  TSL2561.begin();
  //Activamos el "Modo Escritura" del módulo NRF24L01 especificando el ID
  NRF24L01.openWritingPipe(ID);
  //Con el objetivo de que los módulos funcionen a su alcance máximo:
  NRF24L01.setPALevel( RF24_PA_MAX ); //Elevamos la potencia de transmision al maximo
  NRF24L01.setDataRate( RF24_250KBPS ); //Reducimos la velocidad de transmision al minimo
```

```
  //Definimos los pines declarados como input o output
  pinMode (pinTGS823, INPUT);
  pinMode (SCL, INPUT);
  pinMode (SDA, INPUT);
  pinMode (pinCE, INPUT);
  pinMode (pinCSN, INPUT);
  //Incluímos una espera de un segundo para que exista un decalaje entre los datos
  //Llegarán los datos provinientes del exterior, y posteriormente los del interior
  delay(1000);
}
void loop(){
 //Calculamos la concentración de benceno
 float BENCENO = TGS823.getPPT(analogRead(pinTGS823));

 //obtenemos la luminosidad que detecta el TSL2561
 //Los datos aparecen en una trama de datos, empleamos un uint32_t para recogerlos todos
 //Un uint32_t equivale a 32 bits.
 uint32_t luminosidad = TSL2561.getFullLuminosity();
 //Ahora dividimos el numero en 2 partes iguales de 16 bits, es decir,en dos uint16_t:
 //1. Multiplicamos el n en binario por FFFF para obtener los 16 LSB
 uint16_t luminosidad_LSB = luminosidad & 0xFFFF;
 //2. Desplazamos 16 posiciones a la derecha obteniendo los 16 MSB
 uint16_t luminosidad_MSB = luminosidad >> 16;
```

```
//Calculamos los luxes introduciendo los dos paquetes de datos anteriores
float luxes = TSL2561.calculateLux(luminosidad_LSB, luminosidad_MSB);
//Escribimos en el NRF24L01 el paquete de datos para que se envíe al módulo receptor
NRF24L01.write(datos, sizeof(datos));
//Mostramos los resultados por el monitor serial (en caso de estar activado)
Serial.print("Lux: ");
Serial.println(luxes);
Serial.print(" luxes ");
Serial.print("BENCENO: ");
Serial.print(BENCENO);
Serial.println(" PPT");
datos[0]= luxes;
datos[1]= BENCENO;
//Establecemos el tiempo de espera entre dos mediciones consecutivas en un minuto.
delay(60000);
}
```

III. Código Arduino "Coordinador"

```
/* Este es el código que implemtaremos en el Arduino UNO que ejercerá las funciones
 * de coordinador, en el que se reciben todos los datos recogidos en toda la
 * instalación para introducirlas en CONEFI. Los elementos que acoplaremos a este
 * microcontrolador son:
 *     -TSL2561
 *     -TGS823
 *     -MQ135. (2 sensores)
 *     -DHT22
 *     -XBee S2C. (coordinador)
 *     -NRF24L01. (receptor)
 *     -ESP8266. Para establecer la conexion WiFi con la red
 * Cabe destacar que para el sensor de gas TGS823 se han diseñado una libreria
 * propia basada en los resultados obtenidos del estudio de su curva característica.
 */
//Incluímos las librerías
#include "MQ135.h"
#include "XBee.h"
#include "SoftwareSerial.h" //Para desplazar el Tx y Rx a otros pines
#include "SPI.h"
#include "RF24.h"
#include "TSL2561.h"
#include "DHT.h"
#include "MQ135.h"
```

```
//Definimos todos los pines de loc componentes
const int pinDHT22 = 5;
const int pinMQ135 = A0; //Mesa de diego
const int pinMQ135_2 = A1; //Jardin
const int pinCE = 9;
const int pinSCN = 10;
//Definimos el nombre y contraseña de la red WiFi que vayamos a emplear para el ESP8266
String ssid="MOVISTAR_5100";            //Aquí escribimos el nombre de la red WiFi
String password ="********************";  //Aquí escribimos la contraseña del WiFi
boolean DEBUG=true;
//Escribimos el ID que debe coincidir en el emisor y el receptor
const uint64_t ID = 0xCAFE20172018;
//Creamos los paquetes del mismo tamaño de los otros 2 que habiamos creado
//Para depositar aqui los valores recibidos desde los módulos inalámbricos
float trama[2];
uint8_t resultados[6];
//Creamos los objetos de las lubrerías
SoftwareSerial TxRx(6, 7);
SoftwareSerial esp8266(2,3);
XBee xBeeS2C = XBee(); //Creamos el objeto del xBee
XBeeResponse response = XBeeResponse();
ZBRxResponse datos = ZBRxResponse(); //Datos para obtener el paquete desde el Router
RF24 NRF24L01(pinCE, pinSCN); //Definiendo los pines CE y CSN
```

```
TSL2561 TSL2561(TSL2561_ADDR_FLOAT); //ADDR no conectado (no aparece en nuestro sensor)
MQ135 MQ135(pinMQ135);
DHT dht22(pinDHT22, DHT22);
void setup() {
  //Inicializamos los elementos que sean necesarios
  Serial.begin(9600);
  xBeeS2C.begin(TxRx);
  TxRx.begin(9600);
  esp8266.begin(115200);
  NRF24L01.begin();
  //Activamos el "Modo Lectura" del NRF24L01
  NRF24L01.openReadingPipe(1, ID);
  NRF24L01.startListening();
  //Como en el otro módulo, establecemos los parámetros para que el alcance sea el maximo
  NRF24L01.setPALevel( RF24_PA_MAX ) ;
  NRF24L01.setDataRate( RF24_250KBPS );
  //Iniciamos el ESP8266 en MODO CLIENTE, introduciendo el nombre y contraseña del WiFi
  esp8266.println("AT+CWMODE=1");
  esp8266.println("AT+CWJAP=\""+ssid+"\",\""+password+"\"");
}

void loop() {
    TxRx.begin(9600);
```

```
  //Leemos la trama del XBee a la espera de que detecte la llegada de un bit de inicio
  //El bit de inicio siempre es 0x7E
  xBeeS2C.readPacket();
  //Serial.println("Lee el paquete");
  //Cuando se detecta la llegada de esta nueva trama, entra en el bucle if
  if (xBeeS2C.getResponse().isAvailable()) {
   //Serial.println("Entra en el primer if");
   if ( xBeeS2C.getResponse().getApiId() == ZB_RX_RESPONSE) {
    //Obtenemos los datos de la trama
    xBeeS2C.getResponse().getZBRxResponse(datos);
    //Recorremos todos los datos obtenidos
    for (size_t i = 0; i < datos.getDataLength(); i++) {
     //Guardamos todos los resultados en un vector de "resultados"
     resultados[i] = datos.getData(i);
    }
    //Ahora llevamos todos los resultados a variables individuales:
    //Pasándolos de hexadecimal a decimal, dividuendolos posteriormente entre 100
    //para obtener los valores reales
    float BENCENOexterior = (((resultados[6] * 256) + resultados[7])/100);
    float temp = (((resultados[4] * 256) + resultados[5]));
    float hum = (((resultados[2] * 256) + resultados[3]));
    float temperaturaExterior = temp / 100;
    float humedadExterior = hum / 100;
```

```
      float CO2exterior = (((resultados[0] * 256) + resultados[1]));
      //Metemos todos los datos en la función Exterior, donde subiremos las variables
      //de una en una, dado que el programa solo reconoce las subidas de 1 en 1.
      //No obstante, permite la posibilidad de subirlas con una distancia de 1 ms
      //entre ellas, de forma que no es necesario esperar mucho para ver los resultados
      TxRx.end();
      esp8266.begin(115200);
      Exterior(temperaturaExterior,humedadExterior,CO2exterior, BENCENOexterior);
      esp8266.end();
    }
  }
  //En cuando a los módulos NRF24L01, cuando se detecta la llegada
  //de una nueva trama de datos entramos en este bucle if
  if (NRF24L01.available()){
    //Leemos la trama de valores obtenida posición a poición
    NRF24L01.read(trama,sizeof(trama));
    //Ahora llevamos los valores a variables independientes
    float lux = trama[0];
    float BENCENO = trama[1];
    //Vamos a analizar también los sensores conectados al módulo Arduino coordinador
    //dentro de este bucle para dividir así la llegada de resultados:
    //Por un lado los valores desde el exterior, y por el otro desde el interior
    //Primero calculamos los valores de los sensores
```

```
   float CO2 = MQ135.getPPMinterior(analogRead(pinMQ135));
   float CO2_2 = MQ135.getPPMinterior(analogRead(pinMQ135));
   float humedad = dht22.readHumidity();
   float temperatura = dht22.readTemperature();
   uint32_t luminosidad = TSL2561.getFullLuminosity();
   //Ahora dividimos el numero anterior en 2 partes iguales, es decir, en dos uint16_t:
   //1. Multiplicamos el n en binario por FFFF para obtener los 16 LSB
   uint16_t luminosidad_LSB = luminosidad & 0xFFFF;
   //2. Desplazamos 16 posiciones a la derecha obteniendo los 16 MSB
   uint16_t luminosidad_MSB = luminosidad >> 16;
   //Calculamos los luxes empleando la funcion calculateLux e introduciendo ambos
parametros
   float luxes = TSL2561.calculateLux(luminosidad_LSB, luminosidad_MSB);
   //Todos los resultados obtenidos en el interior los llevamos a la función Interior
   //Al igual que en el caso anterior los datos se subirán 1 a 1 con una distancia entre
   //cada uno de 1 ms.
   esp8266.begin(115200);
   delay(10);
   Interior(temperatura,humedad,CO2, CO2_2, luxes, lux, BENCENO);
  }
}
//FUNCIONES
boolean Interior(float temperatura, float humedad, float CO2diego, float CO2jardin, float
luxDiego, float luxAntonio, float BENCENO){
```

```
  //TEMPERATURA
  String ATComando = "AT+CIPSTART=\"TCP\",\"";
  ATComando += "192.168.10.215";
  ATComando += "\",80";
  esp8266.println(ATComando);
  Serial.println(ATComando);
  String paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
  paquete +="&sensorId=temperaturaInterior&valor=";
  paquete += String(temperatura);
  paquete += "\r\n\r\n";
  ATComando = "AT+CIPSEND=";
  ATComando += String(paquete.length());
  esp8266.println(ATComando);
  Serial.println(ATComando);
  delay(1);
  esp8266.print(paquete);
  Serial.print(paquete);
  //HUMEDAD
  ATComando = "AT+CIPSTART=\"TCP\",\"";
  ATComando += "192.168.10.215";
  ATComando += "\",80";
  esp8266.println(ATComando);
  Serial.println(ATComando);
```

```
 paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
 paquete +="&sensorId=humedadInterior&valor=";
 paquete += String(humedad);
 paquete += "\r\n\r\n";
 ATComando = "AT+CIPSEND=";
 ATComando += String(paquete.length());
 esp8266.println(ATComando);
 Serial.println(ATComando);
 delay(1);
 esp8266.print(paquete);
 Serial.print(paquete);
 //CO2diego
 ATComando = "AT+CIPSTART=\"TCP\",\"";
 ATComando += "192.168.10.215";
 ATComando += "\",80";
 esp8266.println(ATComando);
 Serial.println(ATComando);
 paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
 paquete +="&sensorId=CO2diego&valor=";
 paquete += String(CO2diego);
 paquete += "\r\n\r\n";
 ATComando = "AT+CIPSEND=";
```

```
 ATComando += String(paquete.length());
 esp8266.println(ATComando);
 Serial.println(ATComando);
 delay(1);
 esp8266.print(paquete);
 Serial.print(paquete);
 //CO2jardin
 ATComando = "AT+CIPSTART=\"TCP\",\"";
 ATComando += "192.168.10.215";
 ATComando += "\",80";
 esp8266.println(ATComando);
 Serial.println(ATComando);
 paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
 paquete +="&sensorId=CO2jardin&valor=";
 paquete += String(CO2jardin);
 paquete += "\r\n\r\n";
 ATComando = "AT+CIPSEND=";
 ATComando += String(paquete.length());
 esp8266.println(ATComando);
 Serial.println(ATComando);
 delay(1);
 esp8266.print(paquete);
 Serial.print(paquete);
```

```
 //LUXDIEGO
 ATComando = "AT+CIPSTART=\"TCP\",\"";
 ATComando += "192.168.10.215";
 ATComando += "\",80";
 esp8266.println(ATComando);
 Serial.println(ATComando);
 paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
 paquete +="&sensorId=iluminacionDiego&valor=";
 paquete += String(luxDiego);
 paquete += "\r\n\r\n";
 ATComando = "AT+CIPSEND=";
 ATComando += String(paquete.length());
 esp8266.println(ATComando);
 Serial.println(ATComando);
 delay(1);
 esp8266.print(paquete);
 Serial.print(paquete);
 //LUX ANTONIO
 ATComando = "AT+CIPSTART=\"TCP\",\"";
 ATComando += "192.168.10.215";
 ATComando += "\",80";
 esp8266.println(ATComando);
 Serial.println(ATComando);
```

```
  paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
  paquete +="&sensorId=iluminacionAntonio&valor=";
  paquete += String(luxAntonio);
  paquete += "\r\n\r\n";
  ATComando = "AT+CIPSEND=";
  ATComando += String(paquete.length());
  esp8266.println(ATComando);
  Serial.println(ATComando);
  delay(1);
  esp8266.print(paquete);
  Serial.print(paquete);
  //BENCENO IMPRESORA
  ATComando = "AT+CIPSTART=\"TCP\",\"";
  ATComando += "192.168.10.215";
  ATComando += "\",80";
  esp8266.println(ATComando);
  Serial.println(ATComando);
  paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
  paquete +="&sensorId=bencenoImpresora&valor=";
  paquete += String(BENCENO);
  paquete += "\r\n\r\n";
  ATComando = "AT+CIPSEND=";
```

```
 ATComando += String(paquete.length());
 esp8266.println(ATComando);
 Serial.println(ATComando);
 delay(1);
 esp8266.print(paquete);
 Serial.print(paquete);
 return true;
}
boolean Exterior(float temperatura, float humedad,float CO2, float BENCENO){
 //TEMPERATURA
 String ATComando = "AT+CIPSTART=\"TCP\",\"";
 ATComando += "192.168.10.215";
 ATComando += "\",80";
 esp8266.println(ATComando);
 Serial.println(ATComando);
 String paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
 paquete +="&sensorId=temperaturaExterior&valor=";
 paquete += String(temperatura);
 paquete += "\r\n\r\n";
 ATComando = "AT+CIPSEND=";
 ATComando += String(paquete.length());
 esp8266.println(ATComando);
 Serial.println(ATComando);
```

```
 delay(1);
 esp8266.print(paquete);
 Serial.print(paquete);
 //HUMEDAD
 ATComando = "AT+CIPSTART=\"TCP\",\"";
 ATComando += "192.168.10.215";
 ATComando += "\",80";
 esp8266.println(ATComando);
 Serial.println(ATComando);
 paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
 paquete +="&sensorId=humedadExterior&valor=";
 paquete += String(humedad);
 paquete += "\r\n\r\n";
 ATComando = "AT+CIPSEND=";
 ATComando += String(paquete.length());
 esp8266.println(ATComando);
 Serial.println(ATComando);
 delay(1);
 esp8266.print(paquete);
 Serial.print(paquete);
 //CO2EXTERIOR
 ATComando = "AT+CIPSTART=\"TCP\",\"";
 ATComando += "192.168.10.215";
```

```
 ATComando += "\",80";
 esp8266.println(ATComando);
 Serial.println(ATComando);
 paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
 paquete +="&sensorId=CO2exterior&valor=";
 paquete += String(CO2);
 paquete += "\r\n\r\n";
 ATComando = "AT+CIPSEND=";
 ATComando += String(paquete.length());
 esp8266.println(ATComando);
 Serial.println(ATComando);
 delay(1);
 esp8266.print(paquete);
 Serial.print(paquete);
 //BENCENOEXTERIOR
 ATComando = "AT+CIPSTART=\"TCP\",\"";
 ATComando += "192.168.10.215";
 ATComando += "\",80";
 esp8266.println(ATComando);
 Serial.println(ATComando);
 paquete = "GET /Api/SensoresValores?edificioId=cde4dfb0-2cc7-4cd4-941a-
21277ec5a399";
 paquete +="&sensorId=bencenoExterior&valor=";
```

```
  paquete += String(BENCENO);
  paquete += "\r\n\r\n";
  ATComando = "AT+CIPSEND=";
  ATComando += String(paquete.length());
  esp8266.println(ATComando);
  Serial.println(ATComando);
  delay(1);
  esp8266.print(paquete);
  Serial.print(paquete);
  return true;
}
```

IV. Librería MQ135

Librería MQ135 (.h)

```
// Autor: Iván Rodríguez Pérez

// MQ135.h - Librería específica para medir benceno con el sensor MQ135

#ifndef MQ135_h

#define MQ135_h

 #include "Arduino.h"

//Definimos la constante Rload

#define Rload 20.0

//Definimos la constante R0 calculada matematicamente

#define R0interior 19000

#define R0exterior 18500

//Definimos los valores RA y RB, obtenidos de la curva del sensor

#define RA 0.8098

#define RB 0.3913

class MQ135 {

 private:

  uint8_t _pin;

 public:

  MQ135(uint8_t pin);

  float getPPMinterior(int digitalValue);

  float getPPMexterior(int digitalValue);

};

#endif
```

Librería MQ135 (.cpp)

```
// Autor: Iván Rodríguez Pérez

// MQ135.cpp - Descripción de la funcion correspondiente al sensor MQ135

#include "MQ135.h"

//Esta funcion es la que nos permite crear el objeto de la librería

MQ135::MQ135(uint8_t pin) {

  _pin = pin;

}

//Con esta funcion podemos obtener la concentración de CO2 interior

float MQ135::getPPMinterior(int digitalValue){

 float Rs = ( Rload * ( 1023 - digitalValue ) / digitalValue );

 float cociente = ( Rs / R0interior );

 float PPMinterior = pow ( 10 , (        RA - RB * log ( cociente ) ) );

 return PPMinterior;

}

//Con esta funcion podemos obtener la concentración de CO2 exterior

float MQ135::getPPMexterior(int digitalValue){

 float Rs = ( Rload * ( 1023 - digitalValue ) / digitalValue );

 float cociente = ( Rs / R0exterior );

 float PPMexterior = pow ( 10 , RA - RB * log ( cociente ) );

 return PPMexterior;

}
```

V. Librería TGS823

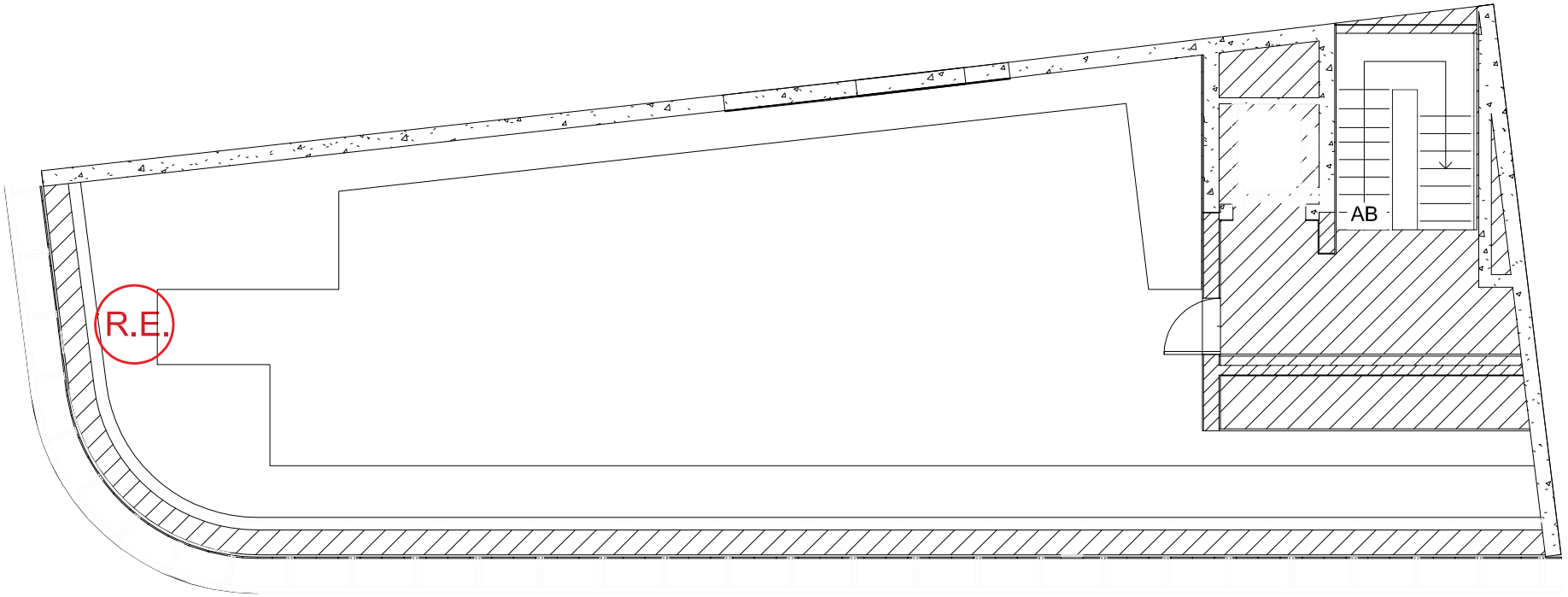
Librería TGS823 (.h)

```
// Autor: Iván Rodríguez Pérez

// TGS823.h - Librería específica para medir benceno con el sensor TGS823

#ifndef TGS823_h

#define TGS823_h

 #include "Arduino.h"

//Definimos la constante Rload

#define Rl 10.0

//Definimos la constante R0 calculada matematicamente, para los 2 sensores es la misma

#define R0 1.25

//Definimos los valores A y B, obtenidos de la curva del sensor

#define A 3.6143

#define B 0.6508

class TGS823 {

 private:

  uint8_t _pin;

 public:

  //Definimos las funciones de esta librería

  //Posteriormente las definimo en el .cpp

  TGS823(uint8_t pin);

  float getPPT(int digitalValue);

};

#endif
```

Librería TGS823 (.cpp)

```
// Autor: Iván Rodríguez Pérez

// TGS823.cpp - Descripción de la funcion correspondiente al sensor TGS823

//Instanciamos la librería TGS823.h

#include "TGS823.h"

//Funcion para crear el objeto

TGS823::TGS823(uint8_t pin) {

  _pin = pin;

}

//Obtenemos la concentracion de benceno en PPT

float TGS823::getPPT(int digitalValue){

 float tension = ( digitalValue * 5 / 1023 );

 float Rs = ( Rl * ( 5 - tension ) / tension );

 float cociente = ( Rs / R0 );

 float PPB = pow ( 10 , A - B * log ( cociente ) );

 return PPB;

}
```

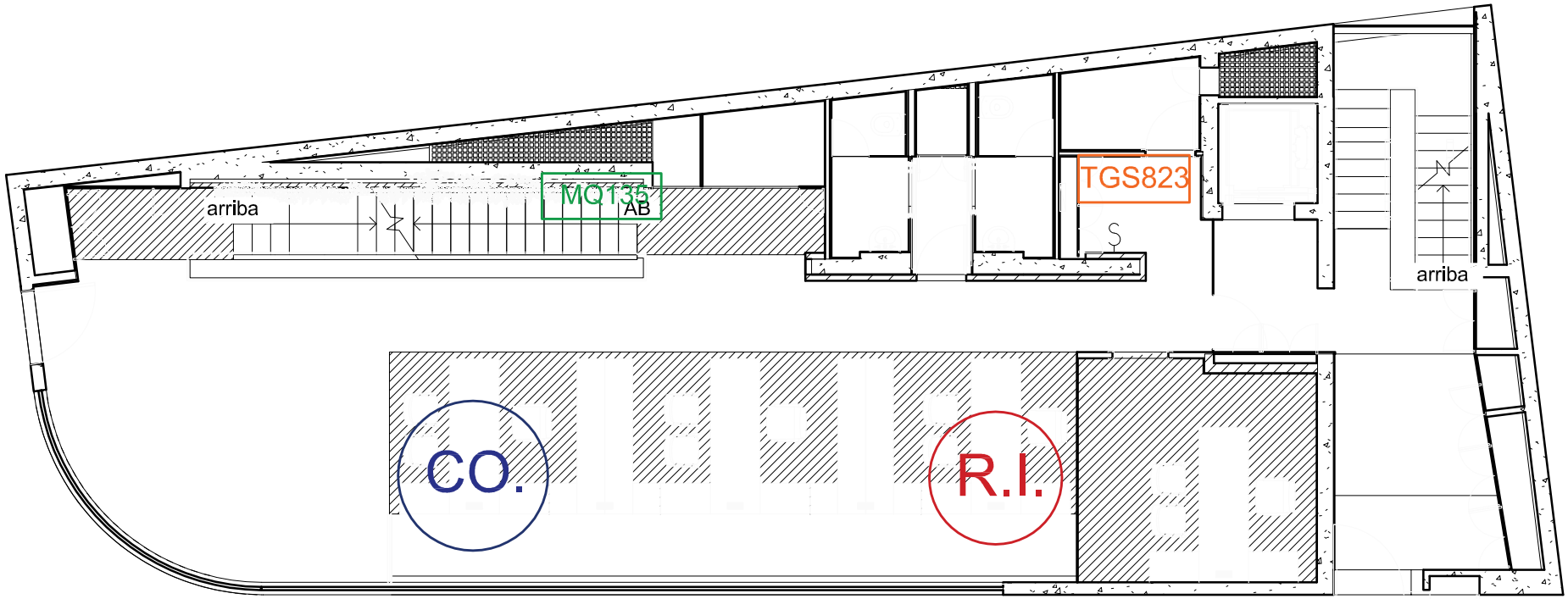
VI. Planos



LEYENDA	
R.E.	Estación "Router Exterior" con todos los sensores y dispositivos incorporados

Análisis comparativo de tecnologías y protocolos de comunicación de sensores para integración en edificio de oficinas gestionado por software de monitorización y control de confort, eficiencia energética y salud

	Fecha:	Autor:		
Dibujado	03/07/2017	Iván	ULL Universidad de La Laguna	Escuela Técnica Superior de Ingeniería y Tecnología
Comprobado	03/07/2017	Rodríguez Pérez		Grado en Ingeniería Electrónica, Industrial y Automática
Id. s. normas	UNE-EN-DIN			Universidad de La Laguna
ESCALA: 1:100	UBICACIÓN DE LA ESTACIÓN "ROUTER EXTERIOR" EN PLANTA CUBIERTA			Nº de Plano: 1



LEYENDA

CO.	Estación "Coordinador" con todos los sensores y dispositivos incorporados excepto uno de los sensores MQ135
R.I.	Estación "Router Interior" con todos los sensores y dispositivos incorporados excepto el sensor TGS823
MQ135	Sensor MQ135 correspondiente a la Estación "Coordinador"
TGS823	Sensor TGS823 correspondiente a la Estación "Router Interior"

Análisis comparativo de tecnologías y protocolos de comunicación de sensores para integración en edificio de oficinas gestionado por software de monitorización y control de confort, eficiencia energética y salud

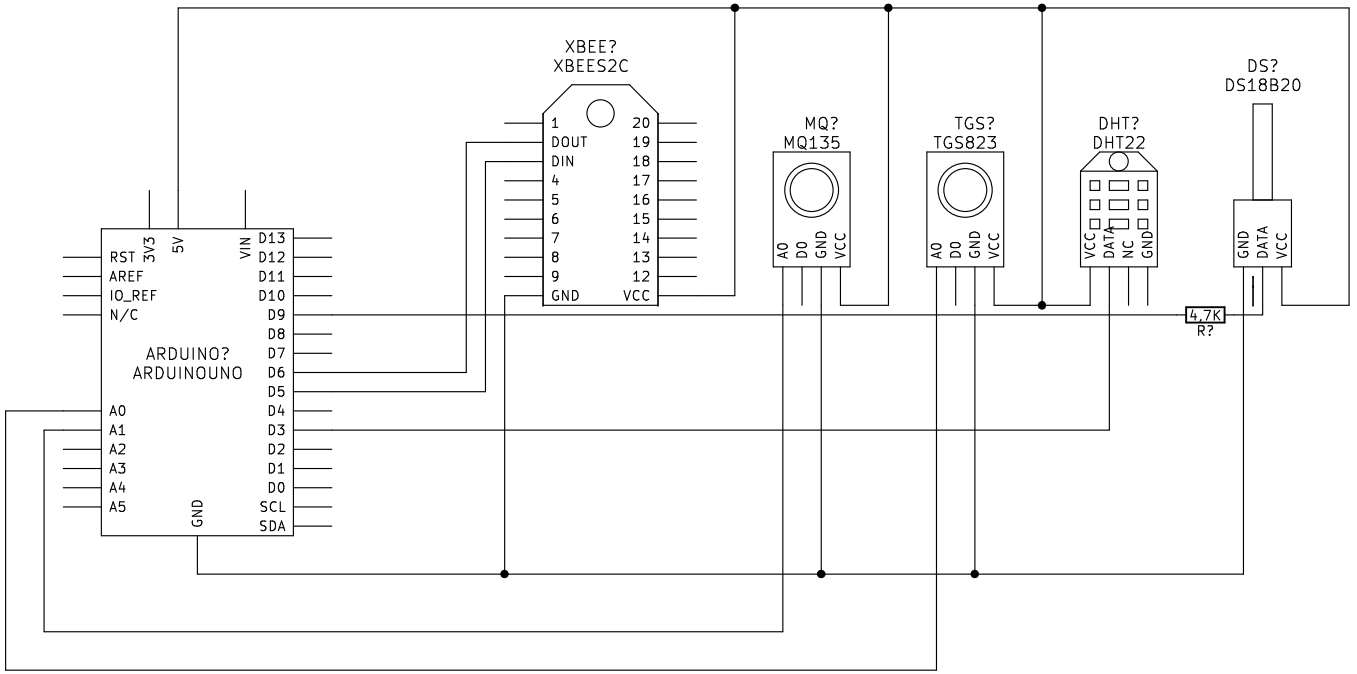
	Fecha:	Autor:
Dibujado	03/07/2017	Iván
Comprobado	03/07/2017	Rodríguez Pérez
Id. s. normas	UNE-EN-DIN	
ESCALA: 1:100	UBICACIÓN DE LAS ESTACIONES "COORDINADOR" Y "ROUTER INTERIOR" EN PLANTA BAJA	Nº de Plano: 2

Universidad de La Laguna

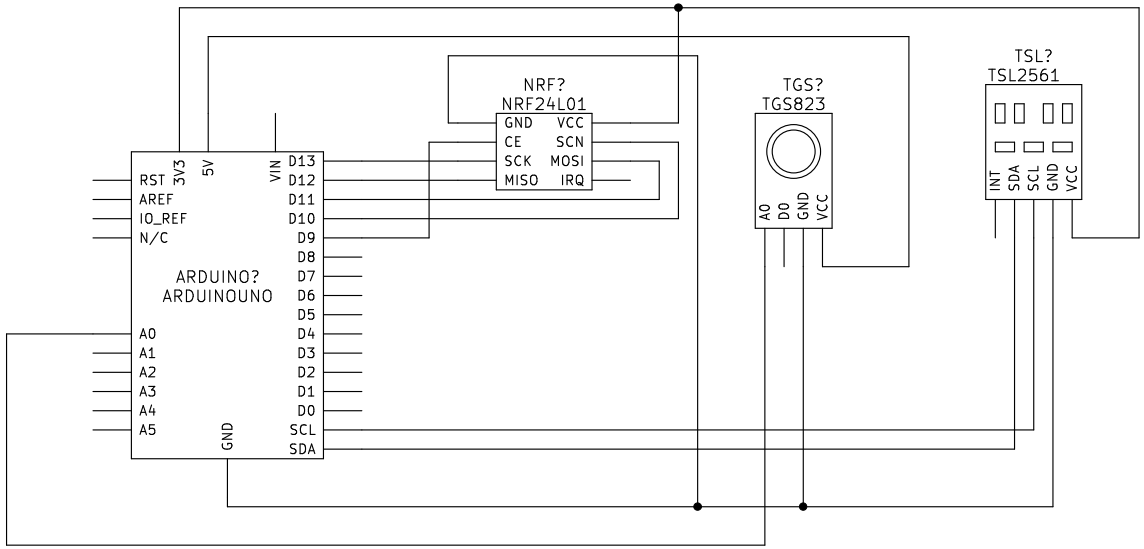
Escuela Técnica Superior de Ingeniería y Tecnología

Grado en Ingeniería Electrónica, Industrial y Automática

Universidad de La Laguna



Análisis comparativo de tecnologías y protocolos de comunicación de sensores para integración en edificio de oficinas gestionado por software de monitorización y control de confort, eficiencia energética y salud				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica, Industrial y Automática
Dibujado	03/07/2017	Iván		
Comprobado	03/07/2017	Rodríguez Pérez		Universidad de La Laguna
Id. s. normas	UNE-EN-DIN			
Escala:	ESTACIÓN "ROUTER EXTERIOR"			Nº Plano: 3



Análisis comparativo de tecnologías y protocolos de comunicación de sensores para integración en edificio de oficinas gestionado por software de monitorización y control de confort, eficiencia energética y salud

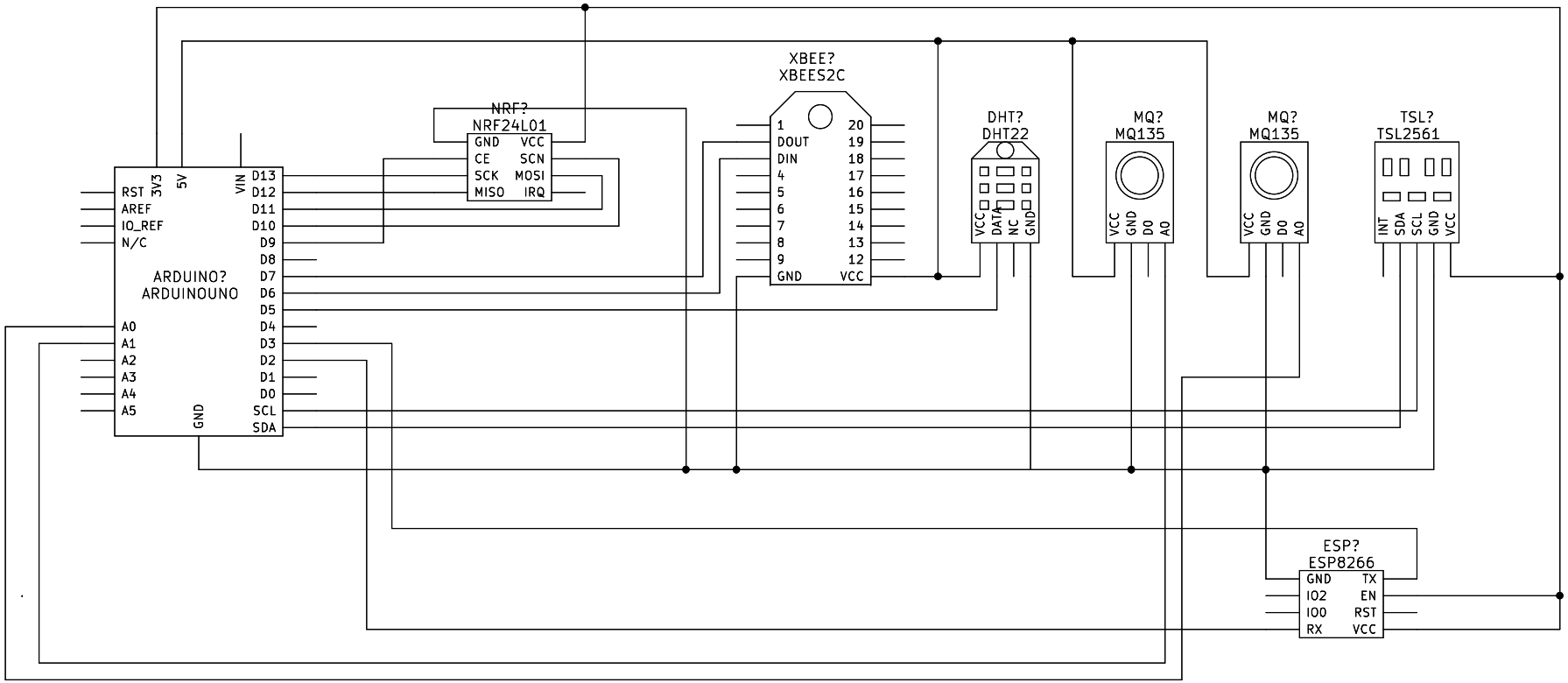
	Fecha	Autor
Dibujado	03/07/2017	Iván
Comprobado	03/07/2017	Rodríguez Pérez
Id. s. normas	UNE-EN-DIN	
Escala:	ESTACIÓN "ROUTER INTERIOR"	

Universidad de La Laguna

Grado en Ingeniería Electrónica, Industrial y Automática

Universidad de La Laguna

Nº Plano: 4



Análisis comparativo de tecnologías y protocolos de comunicación de sensores para integración en edificio de oficinas gestionado por software de monitorización y control de confort, eficiencia energética y salud

	Fecha	Autor
Dibujado	03/07/2017	Iván
Comprobado	03/07/2017	Rodríguez Pérez
Id. s. normas	UNE-EN-DIN	
Escala:	ESTACIÓN "COORDINADOR"	

Universidad de La Laguna

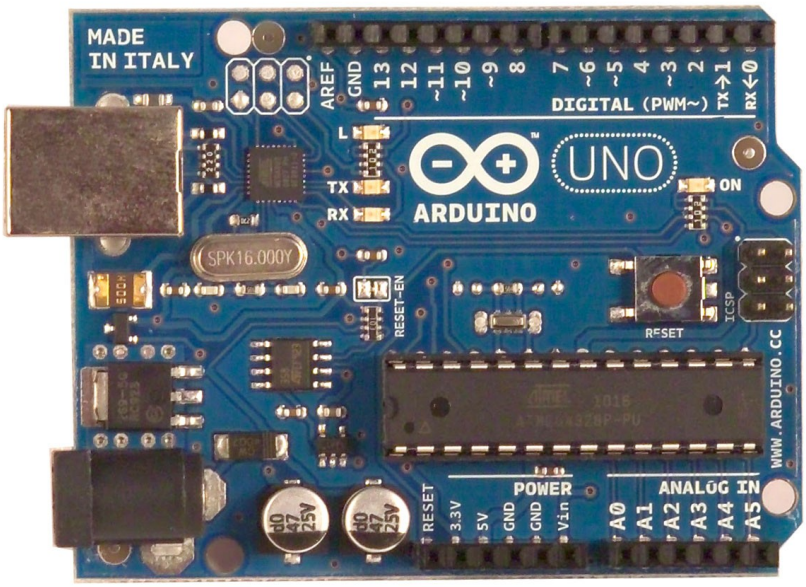
Grado en Ingeniería Electrónica, Industrial y Automática

Universidad de La Laguna

Nº Plano: 5

VII. Datasheets

Arduino UNO



Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Index

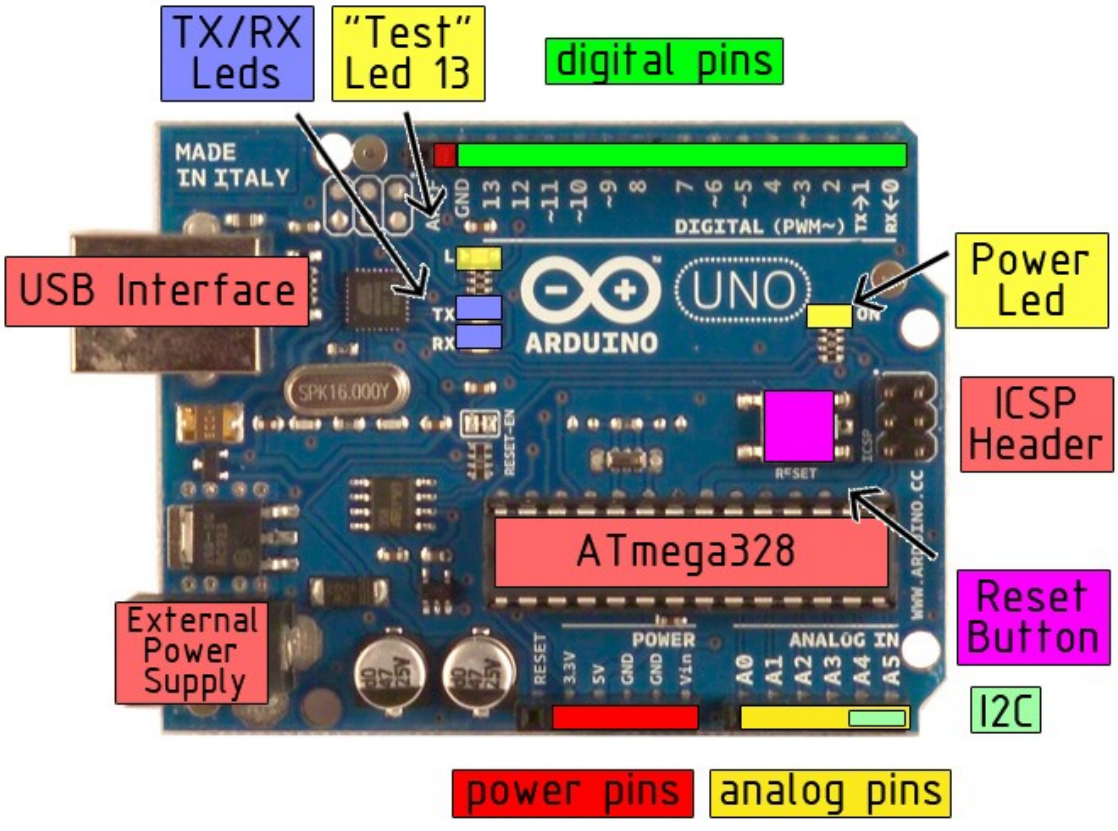
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

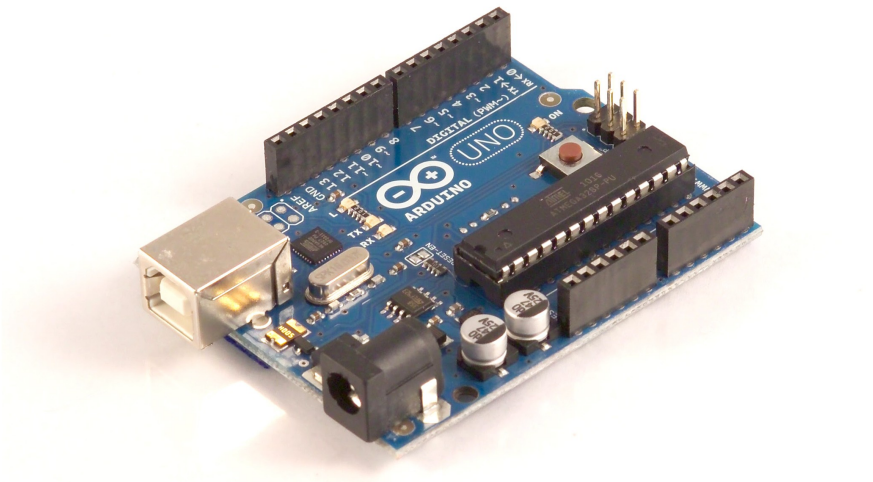
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.



How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install Windows Install Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

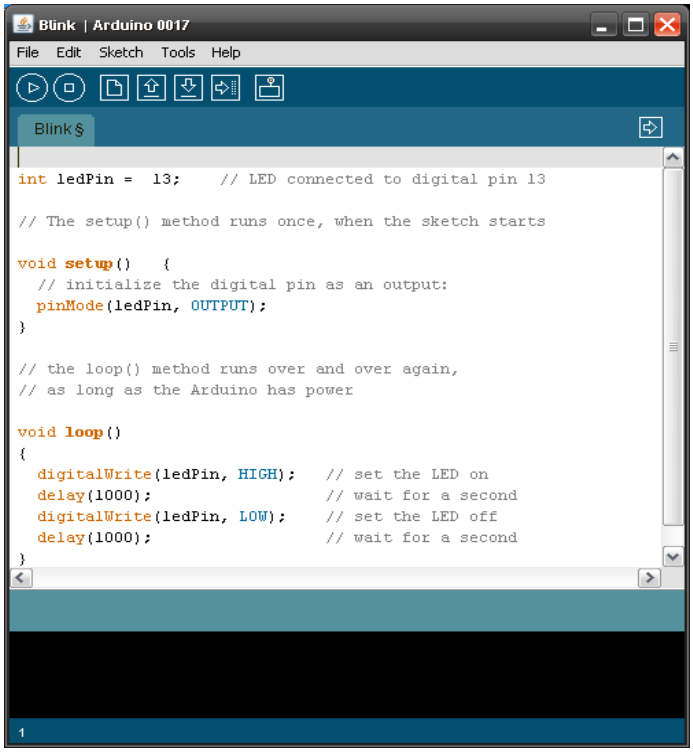
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

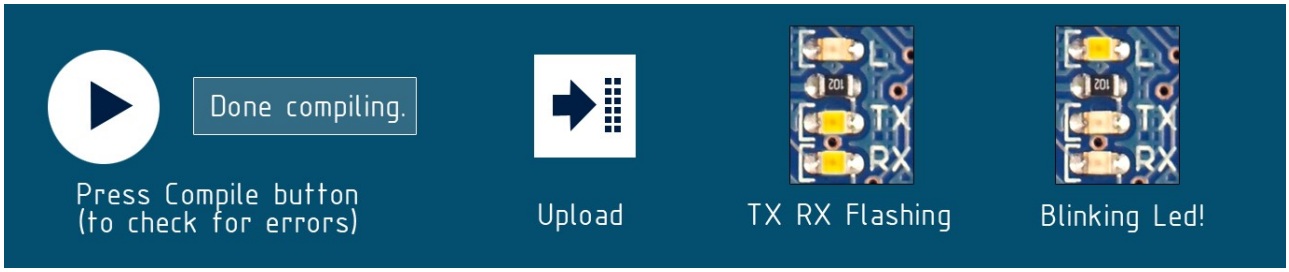
File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your skecth you'll see something very close to the screenshot on the right.

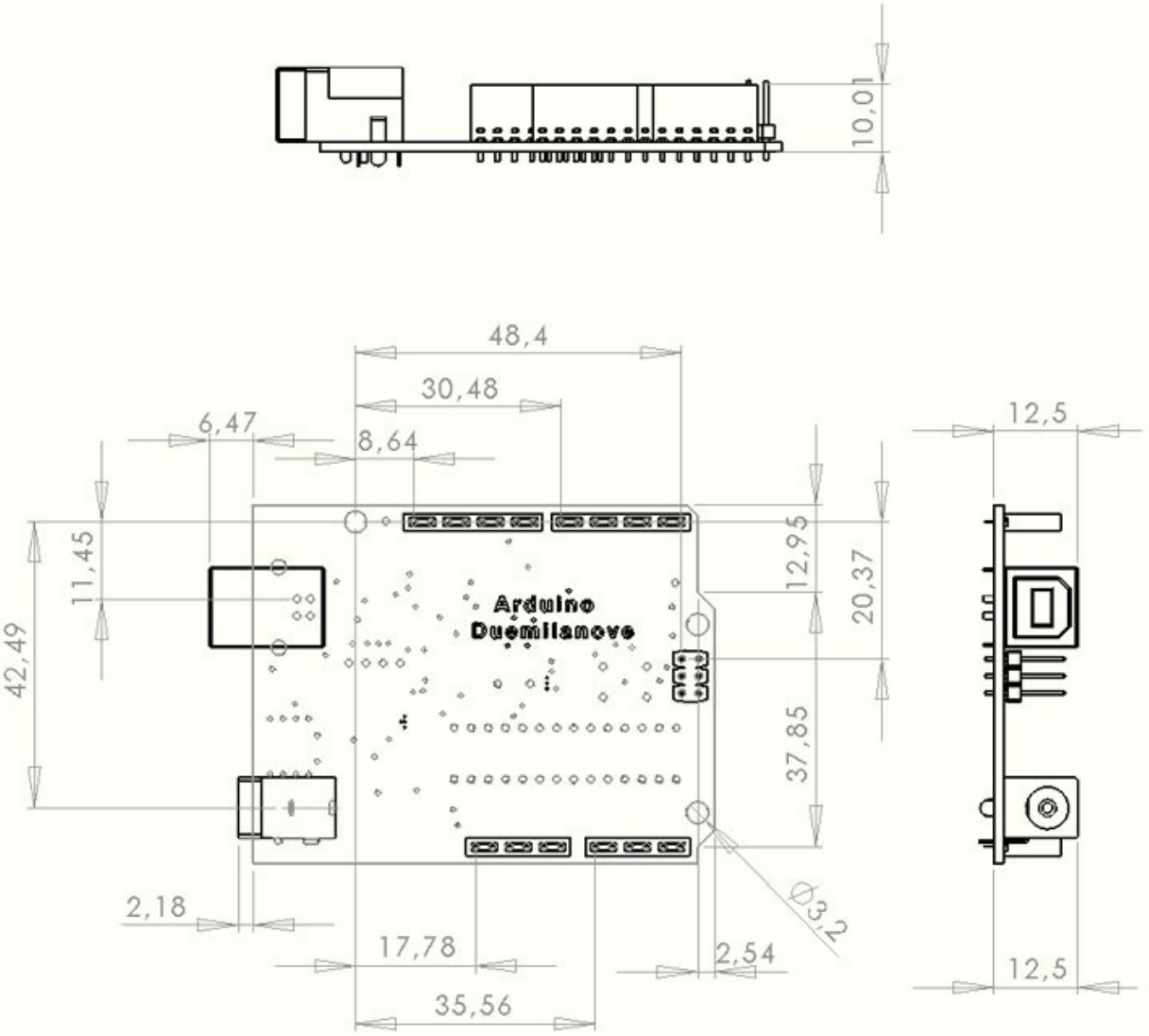
In **Tools>Board** select

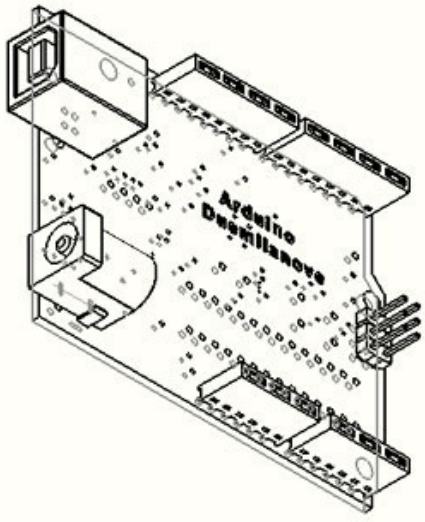
Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.





Dimensioned Drawing





Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

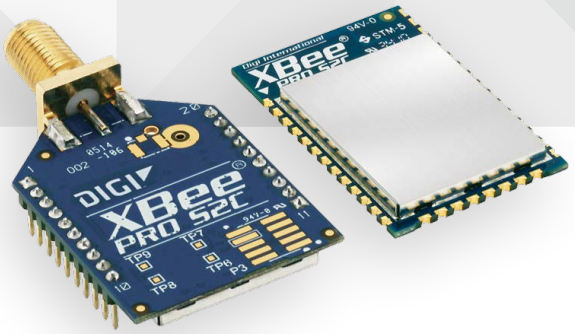
4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

Enviromental Policies

The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

EMBEDDED RF
MODULES FOR OEMS



XBEE® S2C 802.15.4 RF MODULES

Low-cost, easy-to-deploy modules provide critical end-point connectivity to devices and sensors

XBee RF modules provide OEMs with a common footprint shared by multiple platforms, including multipoint and ZigBee/Mesh topologies, and both 2.4 GHz and 900 MHz solutions. OEMs deploying the XBee can substitute one XBee for another, depending upon dynamic application needs, with minimal development, reduced risk and shorter time-to-market.

XBee 802.15.4 RF modules are ideal for applications requiring low latency and predictable communication timing. Providing quick, robust communication in point-to-point, peer-to-peer, and multipoint/star configurations, XBee 802.15.4 products enable robust end-point connectivity with ease. Whether deployed as a pure cable replacement for simple serial communication, or as part of a more complex hub-and-spoke network of sensors, XBee 802.15.4 RF modules maximize performance and ease of development.

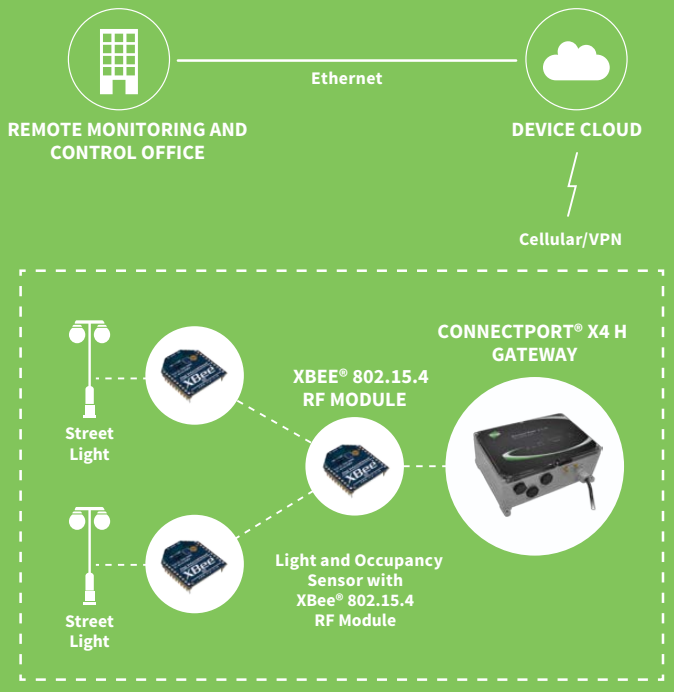
XBee 802.15.4 modules seamlessly interface with compatible gateways, device adapters and range extenders, providing developers with true beyond-the-horizon connectivity.

The updated XBee S2C 802.15.4 module is built with the SiliconLabs EM357 SoC and offers improved power consumption, support for over-the-air firmware updates, and provides an upgrade path to DigiMesh® or ZigBee® mesh protocols if desired.

BENEFITS

- Simple, out-of-the-box RF communications, no configuration needed
- Point-to-multipoint network topology
- 2.4 GHz for worldwide deployment
- Common XBee footprint for a variety of RF modules
- Industry leading sleep current of sub 1uA
- Firmware upgrades via UART, SPI or over the air
- Migratable to DigiMesh and ZigBee PRO protocols and vice-versa

APPLICATION EXAMPLE

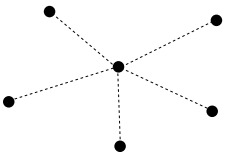


RELATED PRODUCTS

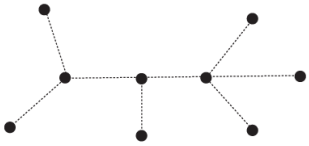
SPECIFICATIONS	XBee® S2C 802.15.4	XBee-PRO® S2C 802.15.4
PERFORMANCE		
TRANSCEIVER CHIPSET	Silicon Labs EM357 SoC	
DATA RATE	RF 250 Kbps, Serial up to 1 Mbps	
INDOOR/URBAN RANGE	200 ft (60 m)	300 ft (90 m)
OUTDOOR/RF LINE-OF-SIGHT RANGE	4000 ft (1200 m)	2 miles (3200 m)
TRANSMIT POWER	3.1 mW (+5 dBm) / 6.3 mW (+8 dBm) boost mode	63 mW (+18 dBm)
RECEIVER SENSITIVITY (1% PER)	-100 dBm / -102 dBm boost mode	-101 dBm
FEATURES		
SERIAL DATA INTERFACE	UART, SPI	
CONFIGURATION METHOD	API or AT commands, local or over-the-air (OTA)	
FREQUENCY BAND	ISM 2.4 GHz	
FORM FACTOR	Through-Hole, Surface Mount	
HARDWARE	S2C	
ADC INPUTS	(4) 10-bit ADC inputs	
DIGITAL I/O	15	
ANTENNA OPTIONS	Through-Hole: PCB Antenna, U.FL Connector, RPSMA Connector, or Integrated Wire SMT: RF Pad, PCB Antenna, or U.FL Connector	
OPERATING TEMPERATURE	-40° C to +85° C	
DIMENSIONS (L X W X H) AND WEIGHT	Through-Hole: 0.960 x 1.087 in (2.438 x 2.761 cm) SMT: 0.866 x 1.33 x 0.120 in (2.199 x 3.4 x 0.305 cm)	Through-Hole: 0.960 x 1.297 in (2.438 x 3.294 cm) SMT: 0.866 x 1.33 x 0.120 in (2.199 x 3.4 x 0.305 cm)
NETWORKING AND SECURITY		
PROTOCOL	XBee 802.15.4 (Proprietary 802.15.4)	
UPDATABLE TO DIGIMESH PROTOCOL	Yes	
UPDATABLE TO ZIGBEE PROTOCOL	Yes	
INTERFERENCE IMMUNITY	DSSS (Direct Sequence Spread Spectrum)	
ENCRYPTION	128-bit AES	
RELIABLE PACKET DELIVERY	Retries/Acknowledgements	
IDS	PAN ID and addresses, cluster IDs and endpoints (optional)	
CHANNELS	16 channels	15 channels
POWER REQUIREMENTS		
SUPPLY VOLTAGE	2.1 to 3.6V	2.7 to 3.6V
TRANSMIT CURRENT	33 mA @ 3.3 VDC / 45 mA boost mode	120 mA @ 3.3 VDC
RECEIVE CURRENT	28 mA @ 3.3 VDC / 31 mA boost mode	31 mA @ 3.3 VDC
POWER-DOWN CURRENT	<1 µA @ 25° C	<1 µA @ 25° C
REGULATORY APPROVALS		
FCC, IC (NORTH AMERICA)	Yes	Yes
ETSI (EUROPE)	Yes	No
RCM (AUSTRALIA AND NEW ZEALAND)	No (Coming soon)	No (Coming soon)
TELEC (JAPAN)	No (Coming soon)	No (Coming soon)

PART NUMBERS	DESCRIPTION
KIT	
XKB2-A2T-WWC	Wireless Connectivity Kit with XBee 802.15.4 (S2C)
MODULES	
XB24CAWIT-001	XBee 802.15.4 through-hole module w/ wire antenna
XB24CAPIT-001	XBee 802.15.4 through-hole module w/ PCB antenna
XB24CAUIT-001	XBee 802.15.4 through-hole module w/ U.fl connector
XB24CASIT-001	XBee 802.15.4 through-hole module w/ RPSMA connector
XB24CAPIS-001	XBee 802.15.4 SMT module w/ PCB antenna
XB24CAUIS-001	XBee 802.15.4 SMT module w/ U.fl connector
XB24CARIS-001	XBee 802.15.4 SMT module w/ RF Pad connector
XBP24CAWIT-001	XBee-PRO 802.15.4 through-hole module w/ wire antenna
XBP24CAPIT-001	XBee-PRO 802.15.4 through-hole module w/ PCB antenna
XBP24CAUIT-001	XBee-PRO 802.15.4 through-hole module w/ U.fl connector
XBP24CASIT-001	XBee-PRO 802.15.4 through-hole module w/ RPSMA connector
XBP24CAPIS-001	XBee-PRO 802.15.4 SMT module w/ PCB antenna
XBP24CAUIS-001	XBee-PRO 802.15.4 SMT module w/ U.fl connector
XBP24CARIS-001	XBee-PRO 802.15.4 SMT module w/ RF Pad connector

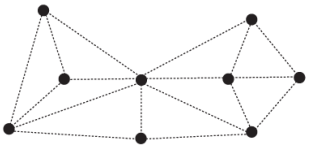
802.15.4 – Star



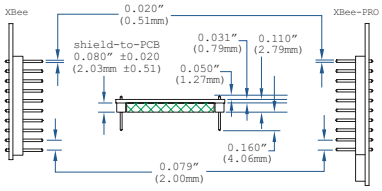
ZigBee - Migratable



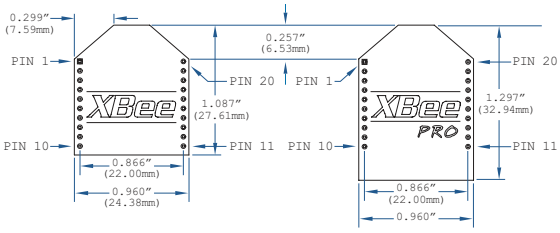
DigiMesh - Migratable



(Side Views)



(Top View)



(Top View)

DIGI SERVICE AND SUPPORT / You can purchase with confidence knowing that Digi is always available to serve you with expert technical support and our industry leading warranty. For detailed information visit www.digi.com/support.

91003287
A3/616

DIGI INTERNATIONAL WORLDWIDE HQ
877-912-3444 / 952-912-3444 / www.digi.com

DIGI INTERNATIONAL FRANCE
+33-1-55-61-98-98 / www.digi.fr

DIGI INTERNATIONAL JAPAN
+81-3-5428-0261 / www.digi-intl.co.jp

DIGI INTERNATIONAL SINGAPORE
+65-6213-5380

DIGI INTERNATIONAL CHINA
+86-21-50492199 / www.digi.com.cn

ESP8266EX

Datasheet

Version 5.4

About This Guide

This document introduces the specifications of ESP8266EX, including the following topics.

Chapter	Title	Subject
Chapter 1	Overview	Provides an overview of ESP8266, including its features, protocols, technical parameters and applications.
Chapter 2	Pin Definitions	Provides the pin layout and the relevant description.
Chapter 3	Functional Description	Describes major functional modules integrated on ESP8266EX including CPU, flash and memory, clock, radio, Wi-Fi, and low-power management.
Chapter 4	Peripheral Interface	Provides descriptions of peripheral interfaces integrated on ESP8266EX.
Chapter 5	Electrical Specifications	Lists the electrical data of ESP8266EX.
Chapter 6	Package Information	Illustrates the package details for ESP8266EX.
Appendix I	Pin List	Provides detailed pin information, including digital die pin list, buffer sheet, register list, and strapping pin list.
Appendix II	Learning Resources	Provides a list of ESP8266-related must-read documents and must-have resources.

Release Notes

Date	Version	Release Notes
2015.12	V4.6	Updated Chapter 3.
2016.02	V4.7	Updated Section 3.6 and Section 4.1.
2016.04	V4.8	Updated Chapter 1.
2016.08	V4.9	Updated Chapter 1.
2016.11	V5.0	Added Appendix II “Learning Resources”.
2016.11	V5.1	Changed the power consumption during Deep-sleep from 10 μA to 20 μA in Table 5-2.
2016.11	V5.2	Changed the crystal frequency range from “26 MHz to 52 MHz” to “24 MHz to 52 MHz” in Section 3.3.
2016.12	V5.3	Changed the minimum working voltage from 3.0V to 2.5V.
2017.04	V5.4	Changed chip input and output impedance from 50Ω to 39+j6 Ω.

Table of Contents

1. Overview

Espressif's ESP8266EX delivers highly integrated Wi-Fi SoC solution to meet users' continuous demands for efficient power usage, compact design and reliable performance in the Internet of Things industry.

With the complete and self-contained Wi-Fi networking capabilities, ESP8266EX can perform either as a standalone application or as the slave to a host MCU. When ESP8266EX hosts the application, it promptly boots up from the flash. The integrated high-speed cache helps to increase the system performance and optimize the system memory. Also, ESP8266EX can be applied to any micro-controller design as a Wi-Fi adaptor through SPI / SDIO or I2C / UART interfaces.

ESP8266EX integrates antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules. The compact design minimizes the PCB size and requires minimal external circuitries.

Besides the Wi-Fi functionalities, ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor and on-chip SRAM. It can be interfaced with external sensors and other devices through the GPIOs. Software Development Kit (SDK) provides sample codes for various applications.

Espressif Systems' Smart Connectivity Platform (ESCP) enables sophisticated features including fast switch between sleep and wakeup mode for energy-efficient purpose, adaptive radio biasing for low-power operation, advance signal processing, spur cancellation and radio co-existence mechanisms for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation.

1.1. Wi-Fi Protocols

- 802.11 b/g/n/e/i support.
- Wi-Fi Direct (P2P) support.
- P2P Discovery, P2P GO (Group Owner) mode, GC(Group Client) mode and P2P Power Management.
- Infrastructure BSS Station mode / P2P mode / SoftAP mode support.
- Hardware accelerators for CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WAPI (SMS4), WEP (RC4), CRC.
- WPA/WPA2 PSK, and WPS driver.
- Additional 802.11i security features such as pre-authentication, and TSN.
- Open Interface for various upper layer authentication schemes over EAP such as TLS, PEAP, LEAP, SIM, AKA, or customer specific.
- 802.11n support (2.4 GHz).
- Supports MIMO 1×1 and 2×1, STBC, A-MPDU and A-MSDU frame aggregation and 0.4μs guard interval.

- WMM power low U-APSD.
- Multiple queue management to fully utilize traffic prioritization defined by 802.11e standard.
- UMA compliant and certified.
- 802.1h/RFC1042 frame encapsulation.
- Scattered DMA for optimal CPU off load on Zero Copy data transfer operations.
- Antenna diversity and selection (software managed hardware).
- Clock/power gating combined with 802.11-compliant power management dynamically adapted to current connection condition providing minimal power consumption.
- Adaptive rate fallback algorithm sets the optimum transmission rate and Tx power based on actual SNR and packet loss information.
- Automatic retransmission and response on MAC to avoid packet discarding on slow host environment.
- Seamless roaming support.
- Configurable packet traffic arbitration (PTA) with dedicated slave processor based design provides flexible and exact timing Bluetooth co-existence support for a wide range of Bluetooth Chip vendors.
- Dual and single antenna Bluetooth co-existence support with optional simultaneous receive (Wi-Fi/Bluetooth) capability.

1.2. Main Technical Specifications

Table 1-1. Main Technical Specifications

Categories	Items	Parameters
Wi-Fi	Standards	FCC/CE/TELEC/SRRC
	Protocols	802.11 b/g/n/e/i
	Frequency Range	2.4G ~ 2.5G (2400M ~ 2483.5M)
	Tx Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit micro controller
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5V ~ 3.6V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	-40°C ~ 125°C
	Storage Temperature Range	-40°C ~ 125°C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

1.3. Applications

- Home Appliances
- Home Automation
- Smart Plugs and Lights
- Mesh Network
- Industrial Wireless Control
- Baby Monitors
- IP Cameras
- Sensor Networks
- Wearable Electronics
- Wi-Fi Location-aware Devices
- Security ID Tags
- Wi-Fi Position System Beacons

2. Pin Definitions

Figure 2-1 shows the pin layout for 32-pin QFN package.

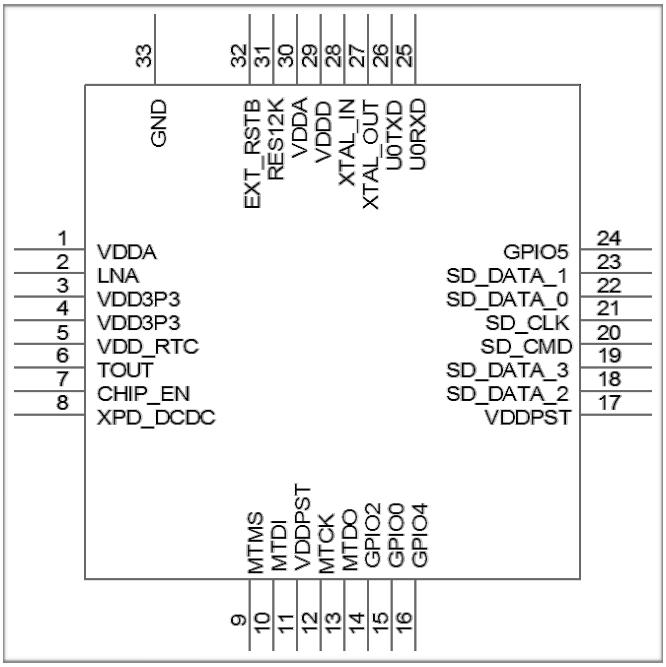


Figure 2-1. Pin Layout

Table 2-1 lists the definitions and functions of each pin.

Table 2-1. ESP8266EX Pin Definitions

Pin	Name	Type	Function
1	VDDA	P	Analog Power 2.5V ~ 3.6V
2	LNA	I/O	RF antenna interface Chip output impedance=39+j6 Ω. It is suggested to retain the π-type matching network to match the antenna.
3	VDD3P3	P	Amplifier Power 2.5V ~ 3.6V
4	VDD3P3	P	Amplifier Power 2.5V ~ 3.6V
5	VDD_RTC	P	NC (1.1V)
6	TOUT	I	ADC pin. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously.
7	CHIP_PU	I	Chip Enable High: On, chip works properly Low: Off, small current consumed
8	XPD_DCDC	I/O	Deep-sleep wakeup (need to be connected to EXT_RSTB); GPIO16

Pin	Name	Type	Function
9	MTMS	I/O	GPIO 14; HSPI_CLK
10	MTDI	I/O	GPIO 12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8V ~ 3.3V)
12	MTCK	I/O	GPIO 13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO 15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART Tx during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO 4
17	VDDPST	P	Digital/IO Power Supply (1.8V ~ 3.3V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 200Ω); SPIHD; HSPIHD; GPIO 9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200Ω); SPIWP; HSPIWP; GPIO 10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200Ω); SPI_CS0; GPIO 11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200Ω); SPI_CLK; GPIO 6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200Ω); SPI_MSIO; GPIO 7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200Ω); SPI_MOSI; GPIO 8
24	GPIO5	I/O	GPIO 5
25	U0RXD	I/O	UART Rx during flash programming; GPIO 3
26	U0TXD	I/O	UART Tx during flash progamming; GPIO 1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 2.5V ~ 3.6V
30	VDDA	P	Analog Power 2.5V ~ 3.6V
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: Active)

> 📖 ***Note:***
>
> *GPIO2, GPIO0, and MTDO are configurable on PCB as the 3-bit strapping register that determines the booting mode and the SDIO timing mode.*

3. Functional Description

The functional diagram of ESP8266EX is shown as in Figure 3-1.

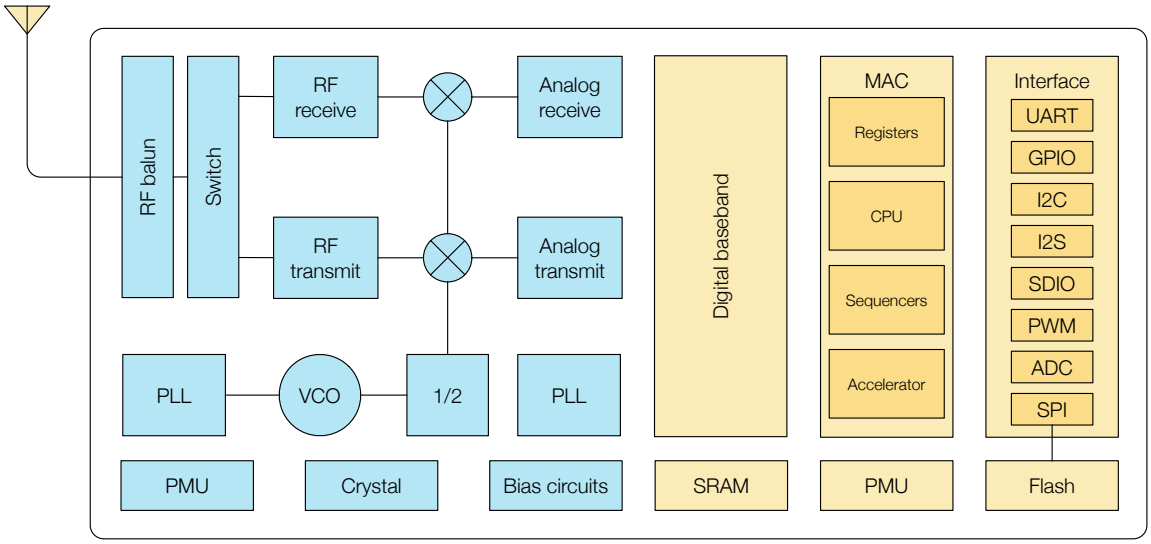


Figure 3-1. Functional Block Diagram

3.1. CPU, Memory, and Flash

3.1.1. CPU

ESP8266EX integrates Tensilica L106 32-bit micro controller (MCU) and ultra-low-power 16-bit RSIC. The CPU clock speed is 80 MHz. It can also reach a maximum value of 160 MHz. Real Time Operation System (RTOS) is enabled. Currently, only 20% of MIPS has been occupied by the Wi-Fi stack, the rest can all be used for user application programming and development. The CPU includes the interfaces as below.

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

3.1.2. Memory

ESP8266EX Wi-Fi SoC integrates memory controller and memory units including SRAM and ROM. MCU can access the memory units through iBus, dBus, and AHB interfaces. All memory units can be accessed upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

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

- RAM size < 50 kB, that is, when ESP8266EX is working under the Station mode and connects to the router, programmable space accessible in heap + data section is around 50 kB.
- There is no programmable ROM in the SoC, therefore, user program must be stored in an external SPI flash.

3.1.3. External Flash

ESP8266EX uses external SPI flash to store user programs, and supports up to 16 MB memory capacity theoretically.

The minimum flash memory of ESP8266EX is shown in Table 3-1.

Table 3-1. Minimum Flash Memory

OTA	Minimum Flash Memory
Disabled	512 kB
Enabled	1 MB

3.2. AHB and AHB Blocks

The AHB block performs as an arbiter. It controls the AHB interfaces through the MAC, SDIO (host) and CPU. Depending on the address, the AHB data requests can go into one of the two slaves.

- APB block
- Flash controller (usually for standalone applications)

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

The APB block acts as a decoder that only accesses the programmable registers within the main blocks of ESP8266EX. Depending on the address, the APB request can go to radio, SI/SPI, SDIO (host), GPIO, UART, real-time clock (RTC), MAC or digital baseband.

3.3. Clock

3.3.1. High Frequency Clock

The high frequency clock on ESP8266EX is used to drive both transmit and receive mixers. This clock is generated from internal crystal oscillator and external crystal. The crystal frequency ranges from 24 MHz to 52 MHz.

The internal calibration inside the crystal oscillator ensures that a wide range of crystals can be used, nevertheless the quality of the crystal is still a factor to consider to have reasonable phase noise and good Wi-Fi sensitivity. Refer to Table 3-2 to measure the frequency offset.

Table 3-2. High Frequency Clock Specifications

Parameter	Symbol	Min	Max	Unit
Frequency	FXO	24	52	MHz
Loading capacitance	CL	–	32	pF
Motional capacitance	CM	2	5	pF
Series resistance	RS	0	65	Ω
Frequency tolerance	ΔFXO	-15	15	ppm
Frequency vs temperature (-25°C ~ 75°C)	ΔFXO,Temp	-15	15	ppm

3.3.2. External Clock Requirements

An externally generated clock is available with the frequency ranging from 24 MHz to 52 MHz. The following characteristics are expected to achieve good performance of radio.

Table 3-3. External Clock Reference

Parameter	Symbol	Min	Max	Unit
Clock amplitude	VXO	0.2	1	Vpp
External clock accuracy	ΔFXO,EXT	-15	15	ppm
Phase noise @1kHz offset, 40 MHz clock	–	–	-120	dBc/Hz
Phase noise @10kHz offset, 40 MHz clock	–	–	-130	dBc/Hz
Phase noise @100kHz offset, 40 MHz clock	–	–	-138	dBc/Hz

3.4. Radio

ESP8266EX radio consists of the following blocks.

- 2.4 GHz receiver
- 2.4 GHz transmitter
- High speed clock generators and crystal oscillator
- Real time clock
- Bias and regulators
- Power management

3.4.1. Channel Frequencies

The RF transceiver supports the following channels according to IEEE802.11b/g/n standards.

Table 3-4. Frequency Channel

Channel No.	Frequency (MHz)	Channel No.	Frequency (MHz)
1	2412	8	2447
2	2417	9	2452
3	2422	10	2457
4	2427	11	2462
5	2432	12	2467
6	2437	13	2472
7	2442	14	2484

3.4.2. 2.4 GHz Receiver

The 2.4 GHz receiver down-converts the RF signals to quadrature baseband signals and converts them to the digital domain with 2 high resolution high speed ADCs. To adapt to varying signal channel conditions, RF filters, automatic gain control (AGC), DC offset cancelation circuits and baseband filters are integrated within ESP8266EX.

3.4.3. 2.4 GHz Transmitter

The 2.4 GHz transmitter up-converts the quadrature baseband signals to 2.4 GHz, and drives the antenna with a high-power CMOS power amplifier. The function of digital calibration further improves the linearity of the power amplifier, enabling a state of art performance of delivering +19.5 dBm average power for 802.11b transmission and +16 dBm for 802.11n transmission.

Additional calibrations are integrated to offset any imperfections of the radio, such as:

- Carrier leakage
- I/Q phase matching
- Baseband nonlinearities

These built-in calibration functions reduce the product test time and make the test equipment unnecessary.

3.4.4. Clock Generator

The clock generator generates quadrature 2.4 GHz clock signals for the receiver and transmitter. All components of the clock generator are integrated on the chip, including all inductors, varactors, filters, regulators and dividers.

The clock generator has built-in calibration and self test circuits. Quadrature clock phases and phase noise are optimized on-chip with patented calibration algorithms to ensure the best performance of the receiver and transmitter.

3.5. Wi-Fi

ESP8266EX implements TCP/IP, the full 802.11 b/g/n/e/i WLAN MAC protocol and Wi-Fi Direct specification. It supports not only basic service set (BSS) operations under the distributed control function (DCF) but also P2P group operation compliant with the latest Wi-Fi P2P protocol. Low level protocol functions are handled automatically by ESP8266EX.

- RTS/CTS
- acknowledgement
- fragmentation and defragmentation
- aggregation
- frame encapsulation (802.11h/RFC 1042)
- automatic beacon monitoring / scanning, and
- P2P Wi-Fi direct

Like P2P discovery procedure, passive or active scanning is performed autonomously once initiated by the appropriate command. Power management is handled with minimum interaction with host to minimize active duty period.

3.6. Power Management

ESP8266EX is designed with advanced power management technologies and intended for mobile devices, wearable electronics and the Internet of Things applications.

The low-power architecture operates in 3 modes: active mode, sleep mode and Deep-sleep mode. ESP8266EX consumes about 20 μA of power in Deep-sleep mode (with RTC clock still running) and less than 1.0 mA (DTIM=3) or less than 0.6 mA (DTIM=10) to stay connected to the access point.

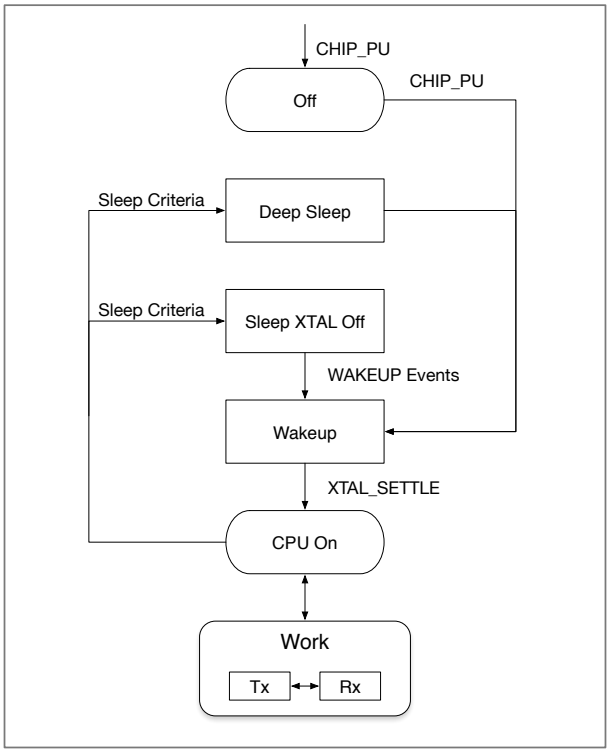


Figure 3-2. Power Management

- **Off**: CHIP_PU pin is low. The RTC is disabled. All registers are cleared.
- **Deep-sleep**: Only RTC is powered on – the rest of the chip is powered off. Recovery memory of RTC can save basic Wi-Fi connection information.
- **Sleep**: Only the RTC is operating. The crystal oscillator is disabled. Any wake-up events (MAC, host, RTC timer, external interrupts) will put the chip into the wakeup mode.
- **Wakeup**: In this state, the system switches from the sleep states to the PWR mode. The crystal oscillator and PLLs are enabled.
- **On**: The high speed clock is able to operate and sent to each block enabled by the clock control register. Lower level clock gating is implemented at the block level, including the CPU, which can be gated off using the WAITI instruction while the system is on.

4. Peripheral Interface

4.1. General Purpose Input/Output Interface (GPIO)

ESP8266EX has 17 GPIO pins which can be assigned to various functions by programming the appropriate registers.

Each GPIO can be configured with internal pull-up or pull-down, or set to high impedance, and when configured as an input, the data are stored in software registers; the input can also be set to edge-trigger or level trigger CPU interrupts. In short, the IO pads are bi-directional, non-inverting and tristate, which includes input and output buffer with tristate control inputs.

These pins can be multiplexed with other functions such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button etc.

For low power operations, the GPIOs can also be set to hold their state. For instance, when the chip is powered down, all output enable signals can be set to hold low.

Optional hold functionality can be built into the IO if requested. When the IO is not driven by the internal or external circuitry, the hold functionality can be used to hold the state to the last used state. The hold functionality introduces some positive feedback into the pad. Hence, the external driver that drives the pad must be stronger than the positive feedback. The required drive strength is small — in the range of 5 µA to pull apart the latch.

4.2. Secure Digital Input/Output Interface (SDIO)

ESP8266EX has one Slave SDIO, the definitions of which are described as Table 4-1.

Table 4-1. Pin Definitions of SDIOs

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

📖 ***Note:***

4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

4.3. Serial Peripheral Interface (SPI/HSPI)

ESP8266EX has 3 SPIs.

- One general Slave/Master SPI
- One Slave SDIO/SPI
- One general Slave/Master HSPI

Functions of all these pins can be implemented via hardware. The pin definitions are described as below.

4.3.1. General SPI (Master/Slave)

Table 4-2. Pin Definitions of SPIs

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

📖 ***Note:***

SPI mode can be implemented via software programming. The clock frequency is 80 MHz at maximum.

4.3.2. HSPI (Slave)

Table 4-3. Pin Definitions of HSPI (Slave)

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

4.4. I2C Interface

ESP8266EX has one I2C used to connect with micro-controller and other peripheral equipments such as sensors. The pin definition of I2C is as below.

Table 4-4. Pin Definitions of I2C

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

Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized via software programming, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

4.5. I2S Interface

ESP8266EX has one I2S data input interface and one I2S data output interface. I2S interfaces are mainly used in applications such as data collection, processing, and transmission of audio data, as well as the input and output of serial data. For example, LED lights (WS2812 series) are supported. The pin definition of I2S is shown in Table 4-5. I2S functionality can be enabled via software programming by using multiplexed GPIOs, and linked list DMA is supported.

Table 4-5. Pin Definitions of I2S

I2S Data Input			
Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	I2SI_DATA
MTCK	12	IO13	I2SI_BCK
MTMS	9	IO14	I2SI_WS
MTDO	13	IO15	I2SO_BCK
U0RXD	25	IO3	I2SO_DATA
GPIO2	14	IO2	I2SO_WS

4.6. Universal Asynchronous Receiver Transmitter (UART)

ESP8266EX has two UART interfaces UART0 and UART, the definitions are shown in Table 4-6.

Table 4-6. Pin Definitions of UART

Pin Type	Pin Name	Pin Num	IO	Function Name
UART0	U0RXD	25	IO3	U0RXD
	U0TXD	26	IO1	U0TXD
	MTDO	13	IO15	U0RTS

Pin Type	Pin Name	Pin Num	IO	Function Name
	MTCK	12	IO13	U0CTS
UART1	GPIO2	14	IO2	U1TXD
	SD_D1	23	IO8	U1RXD

Data transfers to/from UART interfaces can be implemented via hardware. The data transmission speed via UART interfaces reaches 115200 x 40 (4.5 Mbps).

UART0 can be used for communication. It supports fluid control. Since UART1 features only data transmit signal (Tx), it is usually used for printing log.

> 📖 ***Note:***
>
> *By default, UART0 outputs some printed information when the device is powered on and booting up. The baud rate of the printed information is relevant to the frequency of the external crystal oscillator. If the frequency of the crystal oscillator is 40 MHz, then the baud rate for printing is 115200; if the frequency of the crystal oscillator is 26 MHz, then the baud rate for printing is 74880. If the printed information exerts any influence on the functionality of the device, it is suggested to block the printing during the power-on period by changing (U0TXD, U0RXD) to (MTDO, MTCK).*

4.7. Pulse-Width Modulation (PWM)

ESP8266EX has four PWM output interfaces. They can be extended by users themselves. The pin definitions of the PWM interfaces are defined as below.

Table 4-7. Pin Definitions of PWM

Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	PWM0
MTDO	13	IO15	PWM1
MTMS	9	IO14	PWM2
GPIO4	16	IO4	PWM3

The functionality of PWM interfaces can be implemented via software programming. For example, in the LED smart light demo, the function of PWM is realized by interruption of the timer, the minimum resolution reaches as high as 44 ns. PWM frequency range is adjustable from 1000 μs to 10000 μs, i.e., between 100 Hz and 1 kHz. When the PWM frequency is 1 kHz, the duty ratio will be 1/22727, and a resolution of over 14 bits will be achieved at 1 kHz refresh rate.

4.8. IR Remote Control

One Infrared remote control interface is defined as below.

Table 4-8. Pin Definitions of IR Remote Control

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	IR Tx
GPIO5	24	IO 5	IR Rx

The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are used by this interface. The frequency of modulated carrier signal is 38 kHz, while the duty ratio of the square wave is 1/3. The transmission range is around 1m which is determined by two factors: one is the maximum value of rated current, the other is internal current-limiting resistance value in the infrared receiver. The larger the resistance value, the lower the current, so is the power, and vice versa. The transmission angle is between 15° and 30° which is determined by the radiation direction of the infrared receiver.

4.9. ADC (Analog-to-Digital Converter)

ESP8266EX is embedded with a 10-bit precision SARADC. TOUT (Pin6) is defined as below.

Table 4-9. Pin Definition of ADC

Pin Name	Pin Num	Function Name
TOUT	6	ADC Interface

The following two functions can be implemented using ADC (Pin 6). However, they cannot be implemented at the same time.

- Test the power supply voltage of VDD3P3 (Pin 3 and Pin 4).

Hardware Design	TOUT must be floating.
RF Initialization Parameter	The 107th byte of ***esp_init_data_default.bin*** (0 ~ 127 bytes), `vdd33_const` must be set to `0xFF`.
RF Calibration Process	Optimize the RF circuit conditions based on the testing results of VDD3P3 (Pin 3 and Pin 4).
User Programming	Use `system_get_vdd33` instead of `system_adc_read`.

- Test the input voltage of TOUT (Pin 6).

Hardware Design	The input voltage range is 0 to 1.0V when TOUT is connected to external circuit.
RF Initialization Parameter	The value of the 107th byte of ***esp_init_data_default.bin*** (0 ~ 127 bytes), `vdd33_const` must be set to the real power supply voltage of Pin 3 and Pin 4. The working power voltage range of ESP8266EX is between 1.8V and 3.6V, while the unit of `vdd33_const` is 0.1V, therefore, the effective value range of `vdd33_const` is 18 to 36.

RF Calibration Process	Optimize the RF circuit conditions based on the value of `vdd33_const`. The permissible error is ±0.2V.
User Programming	Use `system_adc_read` instead of `system_get_vdd33`.

📖 ***Notes:***

esp_init_data_default.bin *is provided in SDK package which contains RF initialization parameters (0 ~ 127 bytes).*

You can define the 107th byte in ***esp_init_data_default.bin*** *to* `vdd33_const` *as below.*

- *If vdd33_const = 0xff, the power voltage of Pin 3 and Pin 4 will be tested by the internal self-calibration process of ESP8266EX itself. RF circuit conditions should be optimized according to the testing results.*
- *If 18 =< vdd33_const =< 36, ESP8266EX RF Calibration and optimization process is implemented via (vdd33_const/10).*
- *If vdd33_const < 18 or 36 < vdd33_const < 255, ESP8266EX RF Calibration and optimization process is implemented via the default value 2.5V.*

4.10. LED Light and Button

ESP8266EX features 17 GPIOs, all of which can be assigned to support various functions of LED lights and buttons. Definitions of some GPIOs that are assigned with certain functions in demo application design are shown as below.

Table 4-10. Pin Definition of LED and Button

Pin Name	Pin Num	IO	Function Name
MTCK	12	IO 13	Button (Reset)
GPIO0	15	IO 0	Wi-Fi Light
MTDI	10	IO 12	Link Light

Altogether three interfaces have been defined, one is for the button, while the other two are for LED light. Generally, MTCK is used for controlling the reset button; GPIO0 is used as an signal to indicate the Wi-Fi working state; MTDI is used as a signal light to indicate communication status between the device and the server.

📖 ***Note:***

Most interfaces described in this chapter can be multiplexed. Pin definitions that can be defined is not limited to the ones herein mentioned; you can customize the functions of the pins according to your specific application scenarios via software programming and hardware design.

5. Electrical Specifications

5.1. Electrical Characteristics

Table 5-1. Electrical Characteristics

Parameters		Conditions	Min	Typical	Max	Unit
Storage Temperature Range		-	-40	Normal	125	°C
Maximum Soldering Temperature		IPC/JEDEC J-STD-020	-	-	260	°C
Working Voltage Value		—	2.5	3.3	3.6	V
I/O	V_{IL}/V_{IH}	-	-0.3/0.75V_{IO}	-	0.25V_{IO}/3.6	V
	V_{OL}/V_{OH}	-	N/0.8V_{IO}	-	0.1V_{IO}/N	
	I_{MAX}	-	-	-	12	mA
Electrostatic Discharge (HBM)		TAMB=25°C	-	-	2	KV
Electrostatic Discharge (CDM)		TAMB=25°C	-	-	0.5	KV

5.2. Power Consumption

Table 5-2. Power Consumption

Parameters	Min	Typical	Max	Unit
Tx802.11b, CCK 11Mbps, P OUT=+17dBm	-	170	-	mA
Tx 802.11g, OFDM 54Mbps, P OUT =+15dBm	-	140	-	mA
Tx 802.11n, MCS7, P OUT =+13dBm	-	120	-	mA
Rx 802.11b, 1024 bytes packet length , -80dBm	-	50	-	mA
Rx 802.11g, 1024 bytes packet length, -70dBm	-	56	-	mA
Rx 802.11n, 1024 bytes packet length, -65dBm	-	56	-	mA
Modem-sleep①	-	15	-	mA
Light-sleep②	-	0.9	-	mA
Deep-sleep③	-	20	-	µA
Power Off	-	0.5	-	µA

> 📖 ***Notes:***
>
> ① *Modem-sleep mode is used in the applications that require the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it shuts down the Wi-Fi Modem circuit while maintaining a Wi-Fi connection with no data transmission to optimize power consumption. E.g. in DTIM3, maintaining a sleep of 300 ms with a wakeup of 3 ms cycle to receive AP's Beacon packages at interval requires about 15 mA current.*
>
> ② *During **Light-sleep** mode, the CPU may be suspended in applications like Wi-Fi switch. Without data transmission, the Wi-Fi Modem circuit can be turned off and CPU suspended to save power consumption according to the 802.11 standards (U-APSD). E.g. in DTIM3, maintaining a sleep of 300 ms with a wakeup of 3ms to receive AP's Beacon packages at interval requires about 0.9 mA current.*
>
> ③ *During **Deep-sleep** mode, Wi-Fi is turned off. For applications with long time lags between data transmission, e.g. a temperature sensor that detects the temperature every 100s, sleeps for 300s and wakes up to connect to the AP (taking about 0.3 ~ 1s), the overall average current is less than 1mA. The current of 20 µA is acquired at the voltage of 2.5V.*

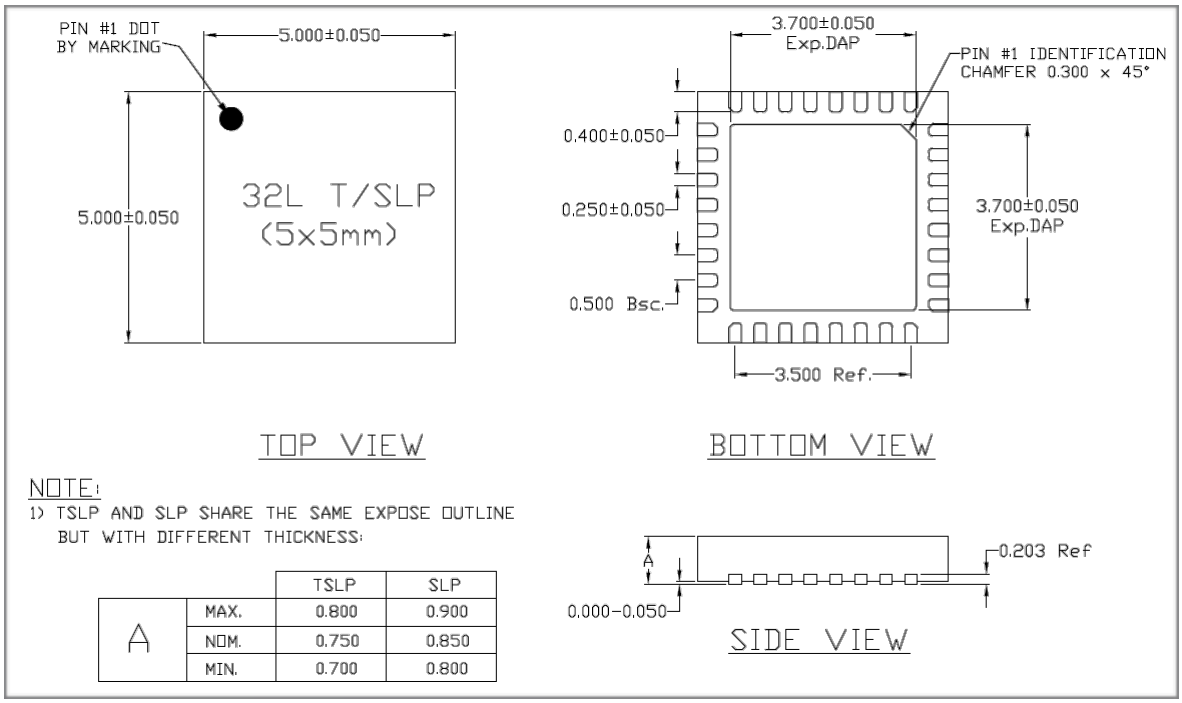
5.3. Wi-Fi Radio Characteristics

The following data are from tests conducted at room temperature with 3.3V and 1.1V power supplies.

Table 5-3. Wi-Fi Radio Characteristics

Parameters	Min	Typical	Max	Unit
Input frequency	2412	-	2484	MHz
Output impedance	-	39+j6	-	Ω
Input reflection	-	-	-10	dB
Output power of PA for 72.2 Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1 Mbps	-	-98	-	dBm
CCK, 11 Mbps	-	-91	-	dBm
6 Mbps (1/2 BPSK)	-	-93	-	dBm
54 Mbps (3/4 64-QAM)	-	-75	-	dBm
HT20, MCS7 (65 Mbps, 72.2 Mbps)	-	-72	-	dBm
Adjacent Channel Rejection				
OFDM, 6 Mbps	-	37	-	dB
OFDM, 54 Mbps	-	21	-	dB
HT20, MCS0	-	37	-	dB
HT20, MCS7	-	20	-	dB

6. Package Information



		TSLP	SLP
A	MAX.	0.800	0.900
	NOM.	0.750	0.850
	MIN.	0.700	0.800

Figure 6-1. ESP8266EX Package

I. Appendix - Pin List

For detailed pin information, please see *ESP8266 Pin List*.

- Digital Die Pin List
- Buffer Sheet
- Register List
- Strapping List

> 📖 ***Notes:***
>
> - *INST_NAME refers to the IO_MUX REGISTER defined in* ***eagle_soc.h****, for example MTDI_U refers to PERIPHS_IO_MUX_MTDI_U.*
> - *Net Name refers to the pin name in schematic.*
> - *Function refers to the multifunction of each pin pad.*
> - *Function number 1 ~ 5 correspond to FUNCTION 0 ~ 4 in SDK. For example, set MTDI to GPIO12 as follows.*
>
> ```
> - #define FUNC_GPIO12  3 //defined in eagle_soc.h
> - PIN_FUNC_SELECT(PERIPHS_IO_MUX_MTDI_U,FUNC_GPIO12)
> ```

II. Appendix - Learning Resources

II.1. Must-Read Documents

- *ESP8266 Quick Start Guide*

  Description: This document is a quick user guide to getting started with ESP8266. It includes an introduction to the ESP-LAUNCHER, instructions on how to download firmware to the board and run it, how to compile the AT application, as well as the structure and debugging method of RTOS SDK. Basic documentation and other related resources for the ESP8266 are also provided.
- *ESP8266 SDK Getting Started Guide*

  Description: This document takes ESP-LAUNCHER and ESP-WROOM-02 as examples of how to use the ESP8266 SDK. The contents include preparations before compilation, SDK compilation and firmware download.
- *ESP8266 Pin List*

  Description: This link directs you to a list containing the type and function of every ESP8266 pin.
- *ESP8266 System Description*

  Description: This document provides a technical description of the ESP8266 series of products, including ESP8266EX, ESP-LAUNCHER and ESP-WROOM.
- *ESP8266 Hardware Matching Guide*

  Description: This document introduces the frequency offset tuning and antenna impedance matching for ESP8266 in order to achieve optimal RF performance.
- *ESP8266 Technical Reference*

  Description: This document provides an introduction to the interfaces integrated on ESP8266. Functional overview, parameter configuration, function description, application demos and other pieces of information are included.
- *ESP8266 Hardware Resources*

  Description: This zip package includes manufacturing specifications of the ESP8266 board and the modules, manufacturing BOM and schematics.
- *FAQ*

II.2. Must-Have Resources

- *ESP8266 SDKs*

Description: This webpage provides links both to the latest version of the ESP8266 SDK and the older ones.

- *ESP8266 Tools*

  Description: This webpage provides links to both the ESP8266 flash download tools and the ESP8266 performance evaluation tools.
- *ESP8266 APK*
- *ESP8266 Certification and Test Guide*
- *ESP8266 BBS*
- *ESP8266 Resources*

Espressif IOT Team

www.espressif.com

Single chip 2.4 GHz Transceiver nRF24L01

FEATURES

- True single chip GFSK transceiver
- Complete OSI Link Layer in hardware
- Enhanced ShockBurst™
- Auto ACK & retransmit
- Address and CRC computation
- On the air data rate 1 or 2Mbps
- Digital interface (SPI) speed 0-8 Mbps
- 125 RF channel operation
- Short switching time enable frequency hopping
- Fully RF compatible with nRF24XX
- 5V tolerant signal input pads
- 20-pin package (QFN20 4x4mm)
- Uses ultra low cost +/- 60 ppm crystal
- Uses low cost chip inductors and 2-layer PCB
- Power supply range: 1.9 to 3.6 V

APPLICATIONS

- Wireless mouse, keyboard, joystick
- Keyless entry
- Wireless data communication
- Alarm and security systems
- Home automation
- Surveillance
- Automotive
- Telemetry
- Intelligent sports equipment
- Industrial sensors
- Toys

GENERAL DESCRIPTION

nRF24L01 is a single chip radio transceiver for the world wide 2.4 - 2.5 GHz ISM band. The transceiver consists of a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator, a demodulator, modulator and Enhanced ShockBurst™ protocol engine. Output power, frequency channels, and protocol setup are easily programmable through a SPI interface. Current consumption is very low, only 9.0mA at an output power of -6dBm and 12.3mA in RX mode. Built-in Power Down and Standby modes makes power saving easily realizable.

QUICK REFERENCE DATA

Parameter	Value	Unit
Minimum supply voltage	1.9	V
Maximum output power	0	dBm
Maximum data rate	2000	kbps
Supply current in TX mode @ 0dBm output power	11.3	mA
Supply current in RX mode @ 2000 kbps	12.3	mA
Temperature range	-40 to +85	°C
Sensitivity @ 1000 kbps	-85	dBm
Supply current in Power Down mode	900	nA

Table 1 nRF24L01 quick reference data

Type Number	Description	Version
nRF24L01	20 pin QFN 4x4, RoHS & SS-00259 compliant	D
nRF24L01 IC	Bare Dice	D
nRF24L01-EVKIT	Evaluation kit (2 test PCB, 2 configuration PCB, SW)	1.0

Table 2 nRF24L01 ordering information

BLOCK DIAGRAM

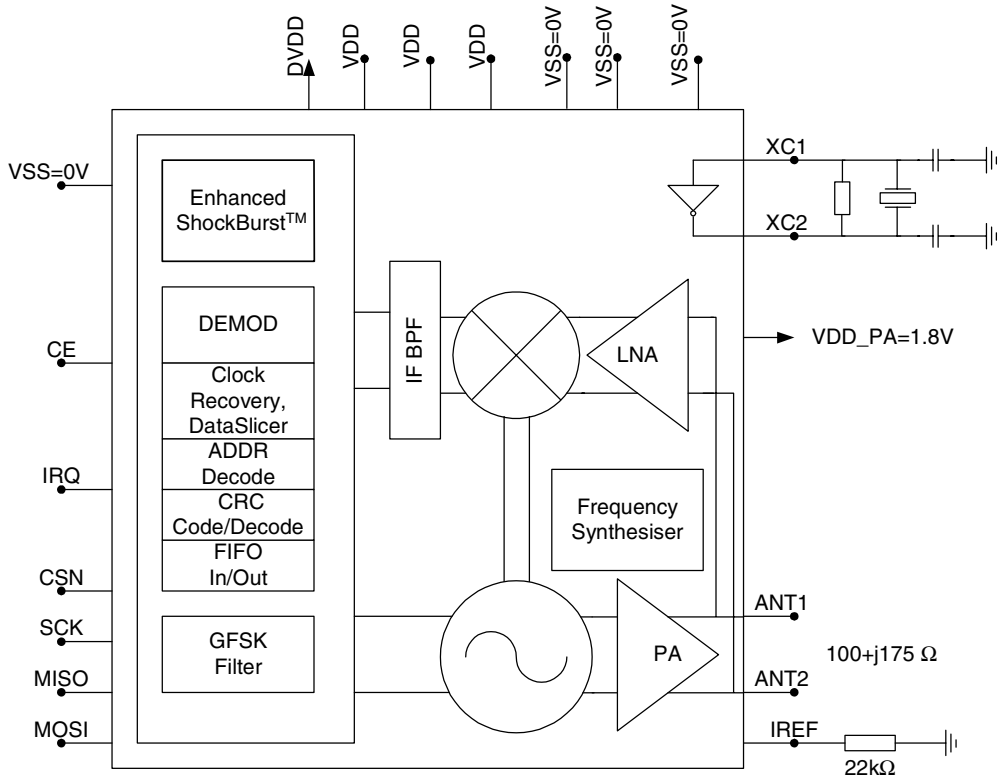


Figure 1 nRF24L01 with external components.

PIN FUNCTIONS

Pin	Name	Pin function	Description
1	CE	Digital Input	Chip Enable Activates RX or TX mode
2	CSN	Digital Input	SPI Chip Select
3	SCK	Digital Input	SPI Clock
4	MOSI	Digital Input	SPI Slave Data Input
5	MISO	Digital Output	SPI Slave Data Output, with tri-state option
6	IRQ	Digital Output	Maskable interrupt pin
7	VDD	Power	Power Supply (+3V DC)
8	VSS	Power	Ground (0V)
9	XC2	Analog Output	Crystal Pin 2
10	XC1	Analog Input	Crystal Pin 1
11	VDD_PA	Power Output	Power Supply (+1.8V) to Power Amplifier
12	ANT1	RF	Antenna interface 1
13	ANT2	RF	Antenna interface 2
14	VSS	Power	Ground (0V)
15	VDD	Power	Power Supply (+3V DC)
16	IREF	Analog Input	Reference current
17	VSS	Power	Ground (0V)
18	VDD	Power	Power Supply (+3V DC)
19	DVDD	Power Output	Positive Digital Supply output for de-coupling purposes
20	VSS	Power	Ground (0V)

Table 3 nRF24L01 pin function

PIN ASSIGNMENT

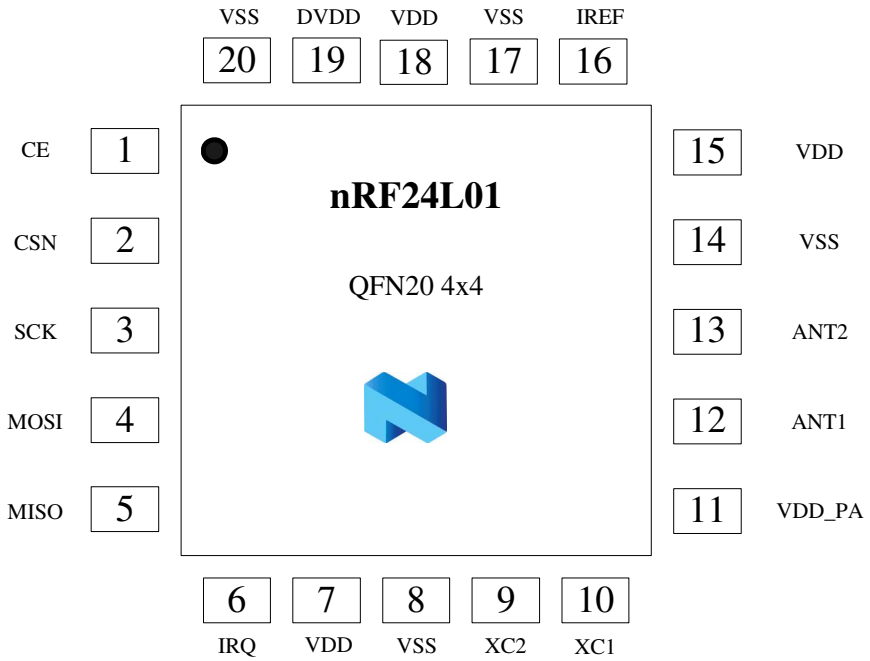


Figure 2 nRF24L01 pin assignment (top view) for a QFN20 4x4 package.

ELECTRICAL SPECIFICATIONS

Conditions: VDD = +3V, VSS = 0V, T_A = - 40ºC to + 85ºC

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
	Operating conditions					
VDD	Supply voltage		1.9	3.0	3.6	V
TEMP	Operating Temperature		-40	+27	+85	ºC
	Digital input pin					
V_{IH}	HIGH level input voltage	[1]	0.7VDD		5.25	V
V_{IL}	LOW level input voltage		VSS		0.3VDD	V
	Digital output pin					
V_{OH}	HIGH level output voltage (I_{OH}=-0.25mA)		VDD- 0.3		VDD	V
V_{OL}	LOW level output voltage (I_{OL}=0.25mA)		VSS		0.3	V
	General RF conditions					
f_{OP}	Operating frequency	[2]	2400		2525	MHz
f_{XTAL}	Crystal frequency			16		MHz
Δf_{1M}	Frequency deviation @ 1000kbps			±160		kHz
Δf_{2M}	Frequency deviation @ 2000kbps			±320		kHz
R_{GFSK}	Data rate ShockBurst™		>0		2000	kbps
$F_{CHANNEL}$	Channel spacing @ 1000kbps			1		MHz
$F_{CHANNEL}$	Channel spacing @ 2000kbps			2		MHz
	Transmitter operation					
P_{RF}	Maximum Output Power	[3]		0	+4	dBm
P_{RFC}	RF Power Control Range		16	18	20	dB
P_{RFCR}	RF Power Accuracy				±4	dB
P_{BW}	20dB Bandwidth for Modulated Carrier (2000kbps)			1800	2000	kHz
P_{RF1}	1st Adjacent Channel Transmit Power 2MHz				-20	dBm
P_{RF2}	2nd Adjacent Channel Transmit Power 4MHz				-50	dBm
I_{VDD}	Supply current @ 0dBm output power	[4]		11.3		mA
I_{VDD}	Supply current @ -18dBm output power			7.0		mA
I_{VDD}	Average Supply current @ -6dBm output power, Enhanced ShockBurst™	[5]		0.05		mA
I_{VDD}	Supply current in Standby-I mode	[6]		32		µA
I_{VDD}	Supply current in power down			900		nA

[1] All digital inputs handle up to 5.25V signal inputs. Keep in mind that the VDD of the nRF24L01 must match the V_{IH} of the driving device for output pins.
[2] Usable band is determined by local regulations
[3] Antenna load impedance = 15Ω+j88Ω
[4] Antenna load impedance = 15Ω+j88Ω. Effective data rate 1000kbps or 2000 kbps
[5] Antenna load impedance = 15Ω+j88Ω. Effective data rate 10kbps and full packets
[6] Given for a 12pF crystal. Current when using external clock is dependent on signal swing.

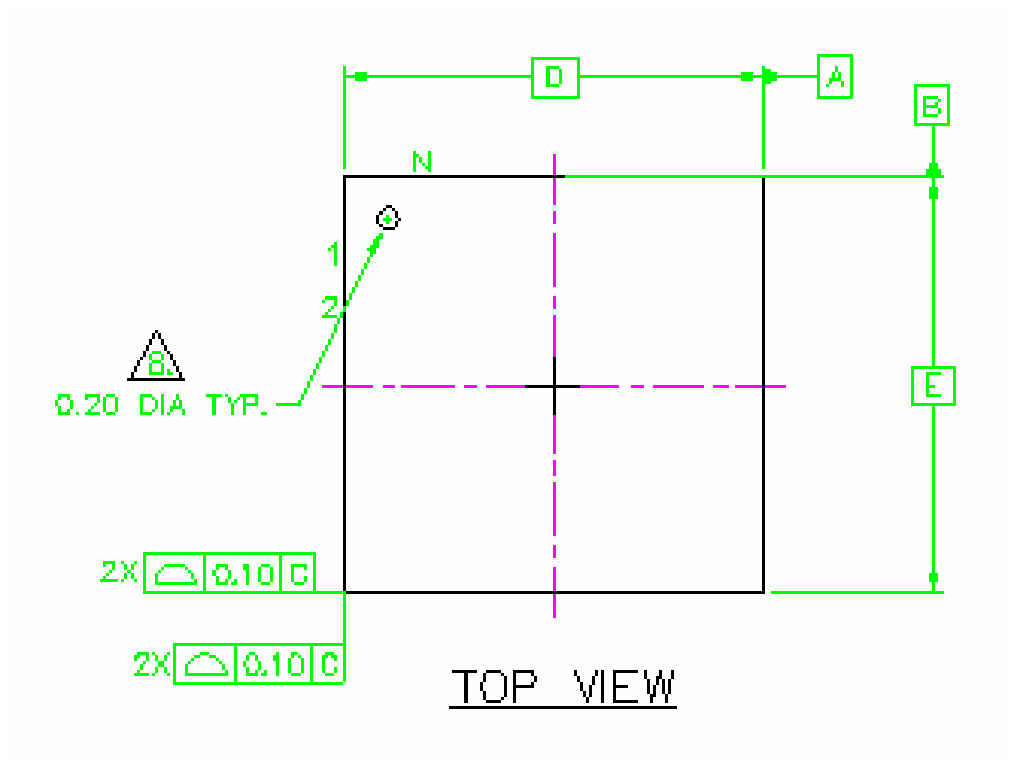
	Receiver operation					
I_{VDD}	Supply current one channel 2000kbps			12.3		mA
I_{VDD}	Supply current one channel 1000kbps			11.8		mA
RX_{SENS}	Sensitivity at 0.1%BER (@2000kbps)			-82		dBm
RX_{SENS}	Sensitivity at 0.1%BER (@1000kbps)			-85		dBm
C/I_{CO}	C/I Co-channel (@2000kbps)	[7]		7[8]/11[9]		dB
C/I_{1ST}	1st Adjacent Channel Selectivity C/I 2MHz			1/4		dB
C/I_{2ND}	2nd Adjacent Channel Selectivity C/I 4MHz			-21/-20		dB
C/I_{3RD}	3rd Adjacent Channel Selectivity C/I 6MHz			-27/-27		dB
C/I_{CO}	C/I Co-channel (@1000kbps)	[10]		9[11]/12[12]		dB
C/I_{1ST}	1st Adjacent Channel Selectivity C/I 1MHz			8/8		dB
C/I_{2ND}	2nd Adjacent Channel Selectivity C/I 2MHz			-22/-21		dB
C/I_{3RD}	3rd Adjacent Channel Selectivity C/I 3MHz			-30/-30		dB

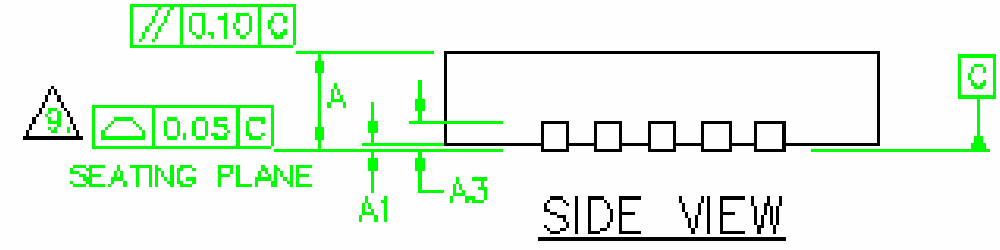
Table 4 nRF24L01 RF specifications

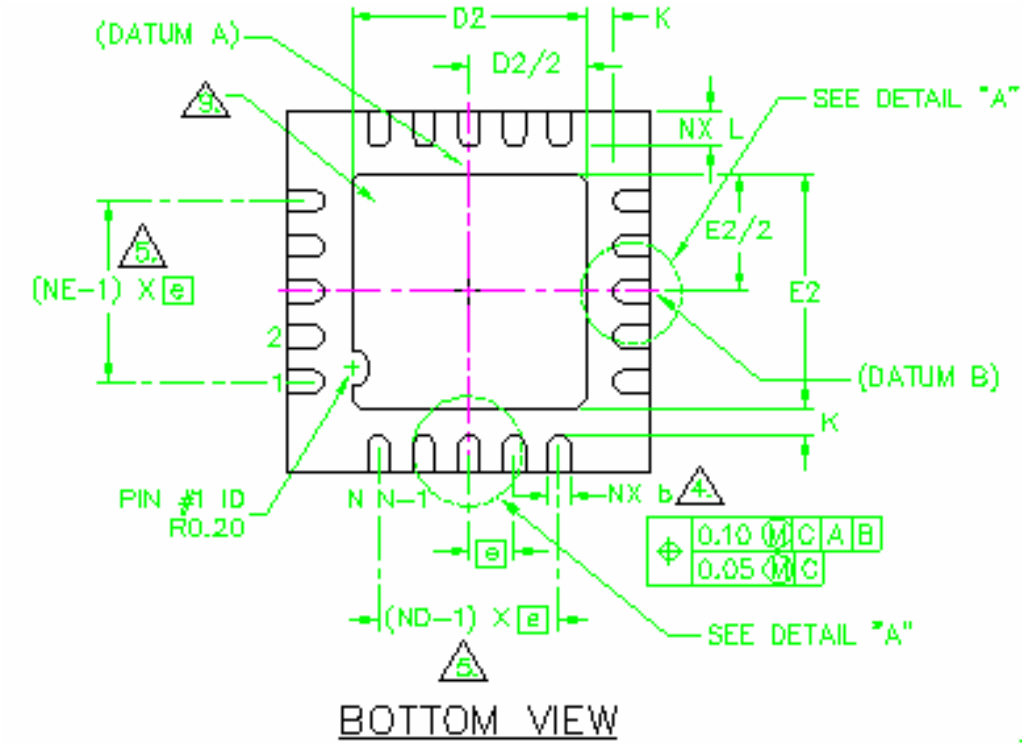
[7] Data rate is 2000kbps for the following C/I measurements
[8] According to ETSI EN 300 440-1 V1.3.1 (2001-09) page 27
[9] nRF24L01 equal modulation on interfering signal
[10] Data rate is 1000kbps for the following C/I measurements
[11] According to ETSI EN 300 440-1 V1.3.1 (2001-09) page 27
[12] nRF24L01 equal modulation on interfering signal

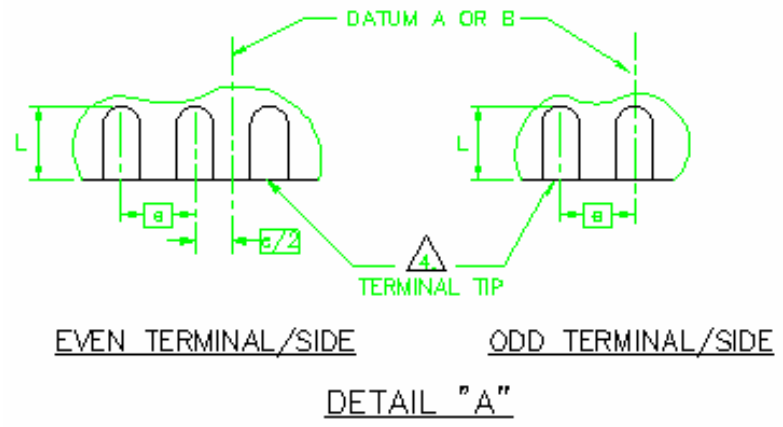
PACKAGE OUTLINE

nRF24L01 uses the QFN20 4x4 package, with matt tin plating.









Package Type		A	A1	A3	K	D/E	e	D2/E2	L	L1	b
Saw QFN20	**Min**	0.80	0.00					2.50	0.35		0.18
(4x4 mm)	**Typ.**	0.85	0.02	0.20	0.20	4.0 BSC[13]	0.5 BSC	2.60	0.40	0.15	0.25
	Max	0.95	0.05	REF.	min			2.70	0.45	max	0.30

Figure 3 nRF24L01 Package Outline.

[13] BSC: Basic Spacing between Centers, ref. JEDEC standard 95, page 4 17-11/A

Package marking:

n	R	F		*B*	*X*
2	4	L	0	1	
Y	*Y*	*W*	*W*	*L*	*L*

Abbreviations:

B – Build Code, i.e. unique code for production sites, package type and test platform
X – "X" grade, i.e. Engineering Samples (optional)
YY – 2 digit Year number
WW – 2 digit Week number
LL – 2 letter wafer lot number code

Absolute Maximum Ratings

Supply voltages
VDD ..... - 0.3V to + 3.6V
VSS 0V
Input voltage
V_I ..... - 0.3V to 5.25V
Output voltage
V_O ..... VSS to VDD

Total Power Dissipation
P_D (T_A=85°C) 60mW

Temperatures
Operating Temperature…. - 40°C to + 85°C
Storage Temperature……. - 40°C to + 125°C

Note: Stress exceeding one or more of the limiting values may cause permanent damage to the device.

ATTENTION!

Electrostatic Sensitive Device
Observe Precaution for handling.

Glossary of Terms

Term	Description
ACK	Acknowledgement
ART	Auto Re-Transmit
CE	Chip Enable
CLK	Clock
CRC	Cyclic Redundancy Check
CSN	Chip Select NOT
ESB	Enhanced ShockBurst™
GFSK	Gaussian Frequency Shift Keying
IRQ	Interrupt Request
ISM	Industrial-Scientific-Medical
LNA	Low Noise Amplifier
LSB	Least Significant Bit
LSByte	Least Significant Byte
Mbps	Megabit per second
MCU	Micro Controller Unit
MISO	Master In Slave Out
MOSI	Master Out Slave In
MSB	Most Significant Bit
MSByte	Most Significant Byte
PCB	Printed Circuit Board
PER	Packet Error Rate
PID	Packet Identity Bits
PLD	Payload
PRX	Primary RX
PTX	Primary TX
PWR_DWN	Power Down
PWR_UP	Power Up
RoHS	Restriction of use of Certain Hazardous Substances
RX	Receive
RX_DR	Receive Data Ready
SPI	Serial Peripheral Interface
TX	Transmit
TX_DS	Transmit Data Sent

Table 5 Glossary

FUNCTIONAL DESCRIPTION

Modes of operation

The nRF24L01 can be set in the following main modes depending on the level of the following primary I/Os and configuration registers:

Mode	PWR_UP register	PRIM_RX register	CE	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFO
TX mode	1	0	1➔0	Stays in TX mode until packet transmission is finished
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

Table 6 nRF24L01 main modes

An overview of the nRF24L01 I/O pins in different modes is given in Table 7.

Pin functions in the different modes of nRF24L01

Pin Name	Direction	TX Mode	RX Mode	Standby Modes	Power Down
CE	Input	High Pulse >10µs	High	Low	-
CSN	Input	SPI Chip Select, active low			
SCK	Input	SPI Clock			
MOSI	Input	SPI Serial Input			
MISO	Tri-state Output	SPI Serial Output			
IRQ	Output	Interrupt, active low			

Table 7 Pin functions of the nRF24L01

Standby Modes

Standby-I mode is used to minimize average current consumption while maintaining short start up times. In this mode, part of the crystal oscillator is active. In Standby-II mode some extra clock buffers are active compared to Standby-I mode. Standby-II occurs when CE is held high on a PTX device with empty TX FIFO. The configuration word content is maintained during Standby modes. SPI interface may be activated. For start up time see Table 13.

Power Down Mode

In power down nRF24L01 is disabled with minimal current consumption. When entering this mode the device is not active, but all registers values available from the SPI interface are maintained during power down and the SPI interface may be activated (CSN=0). For start up time see Table 13. The power down is controlled by the PWR_UP bit in the CONFIG register.

Packet Handling Methods

nRF24L01 has the following Packet Handling Methods:

- ShockBurst™ (compatible with nRF2401, nRF24E1, nRF2402 and nRF24E2 with 1Mbps data rate, see page 26)
- Enhanced ShockBurst™

ShockBurst™

ShockBurst™ makes it possible to use the high data rate offered by nRF24L01 without the need of a costly, high-speed microcontroller (MCU) for data processing/clock recovery. By placing all high speed signal processing related to RF protocol on-chip, nRF24L01 offers the application microcontroller a simple SPI compatible interface, the data rate is decided by the interface-speed the micro controller itself sets up. By allowing the digital part of the application to run at low speed, while maximizing the data rate on the RF link, ShockBurst™ reduces the average current consumption in applications.
In ShockBurst™ RX, IRQ notifies the MCU when a valid address and payload is received respectively. The MCU can then clock out the received payload from an nRF24L01 RX FIFO.
In ShockBurst™ TX, nRF24L01 automatically generates preamble and CRC, see Table 12. IRQ notifies the MCU that the transmission is completed. All together, this means reduced memory demand in the MCU resulting in a low cost MCU, as well as reduced software development time. nRF24L01 has a three level deep RX FIFO (shared between 6 pipes) and a three level deep TX FIFO. The MCU can access the FIFOs at any time, in power down mode, in standby modes, and during RF packet transmission. This allows the slowest possible SPI interface compared to the average data-rate, and may enable usage of an MCU without hardware SPI.

Enhanced ShockBurst™

Enhanced ShockBurst™ is a packet handling method with functionality that makes bi-directional link protocol implementation easier and more efficient. In a typical bi-directional link, one will let the terminating part acknowledge received packets from the originating part in order to make it possible to detect data loss. Data loss can then be recovered by retransmission. The idea with Enhanced ShockBurst™ is to let nRF24L01 handle both acknowledgement of received packets and retransmissions of lost packets, without involvement from the microcontroller.

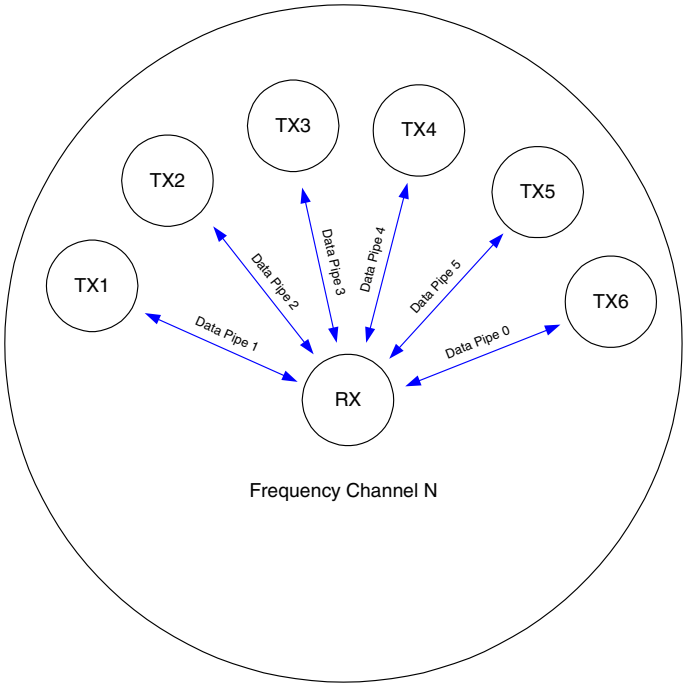


Figure 4: nRF24L01 in a star network configuration

An nRF24L01 configured as primary RX (PRX) will be able to receive data trough 6 different data pipes, see Figure 4. A data pipe will have a unique address but share the same frequency channel. This means that up to 6 different nRF24L01 configured as primary TX (PTX) can communicate with one nRF24L01 configured as PRX, and the nRF24L01 configured as PRX will be able to distinguish between them. Data pipe 0 has a unique 40 bit configurable address. Each of data pipe 1-5 has an 8 bit unique address and shares the 32 most significant address bits. All data pipes can perform full Enhanced ShockBurst™ functionality.
nRF24L01 will use the data pipe address when acknowledging a received packet. This means that nRF24L01 will transmit ACK with the same address as it receives payload at. In the PTX device data pipe 0 is used to received the acknowledgement, and therefore the receive address for data pipe 0 has to be equal to the transmit address to be able to receive the acknowledgement. See Figure 5 for addressing example.

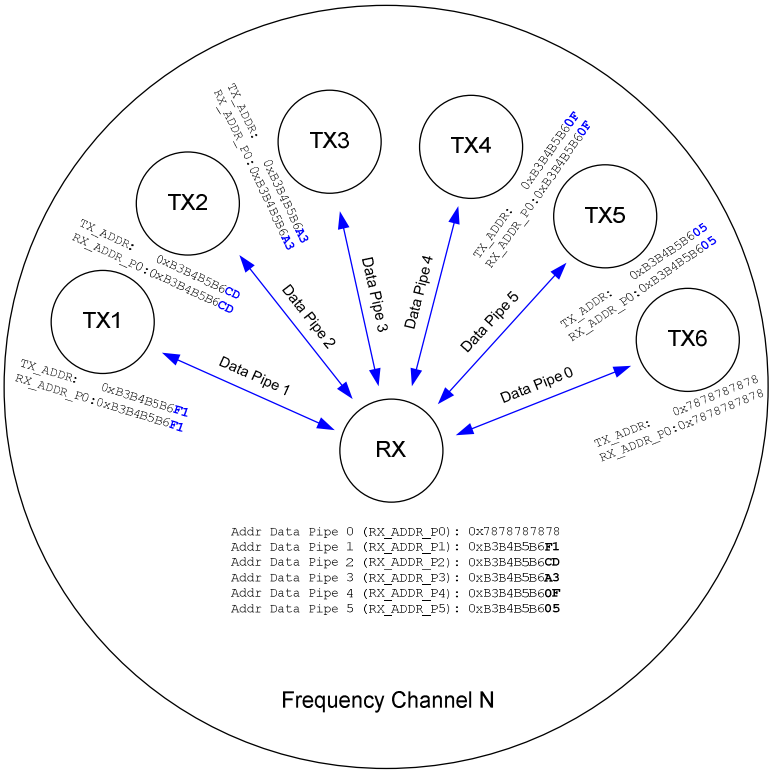


Figure 5: Example on how the acknowledgement addressing is done

An nRF24L01 configured as PTX with Enhanced ShockBurst™ enabled, will use the ShockBurst™ feature to send a packet whenever the microcontroller wants to. After the packet has been transmitted, nRF24L01 will switch on its receiver and expect an acknowledgement to arrive from the terminating part. If this acknowledgement fails to arrive, nRF24L01 will retransmit the same packet until it receives an acknowledgement or the number of retries exceeds the number of allowed retries given in the SETUP_RETR_ARC register. If the number of retries exceeds the number of allowed retries, this will be showed by the STATUS register bit MAX_RT which gives an interrupt.
Whenever an acknowledgement is received by an nRF24L01 it will consider the last transmitted packet as delivered. It will then be cleared from the TX FIFO, and the TX_DS IRQ source will be set high.

With Enhanced ShockBurst™ nRF24L01 offers the following benefits:

- Highly reduced current consumption due to short time on air and sharp timing when operating with acknowledgement traffic
- Lower system cost. Since the nRF24L01 handles all the high-speed link layer operations, like re-transmission of lost packet and generating acknowledgement to received packets, it is no need for hardware SPI on the system microcontroller to interface the nRF24L01. The interface can be done by using general purpose IO pins on a low cost microcontroller where the SPI is emulated in firmware. With the nRF24L01 this will be sufficient speed even when running a bi-directional link.
- Greatly reduced risk of “on-air” collisions due to short time on air
- Easier firmware development since the link layer is integrated on chip

Enhanced ShockBurst™ Transmitting Payload:

1. The configuration bit PRIM_RX has to be low.
2. When the application MCU has data to send, the address for receiving node (TX_ADDR) and payload data (TX_PLD) has to be clocked into nRF24L01 via the SPI interface. The width of TX-payload is counted from number of bytes written into the TX FIFO from the MCU. TX_PLD must be written continuously while holding CSN low. TX_ADDR does not have to be rewritten if it is unchanged from last transmit. If the PTX device shall receive acknowledge, data pipe 0 has to be configured to receive the acknowledgement. The receive address for data pipe 0 (RX_ADDR_P0) has to be equal to the transmit address (TX_ADDR) in the PTX device. For the example in Figure 5 the following address settings have to be performed for the TX5 device and the RX device:
   TX5 device: TX_ADDR = 0xB3B4B5B605
   TX5 device: RX_ADDR_P0 = 0xB3B4B5B605
   RX device: RX_ADDR_P5 = 0xB3B4B5B605
3. A high pulse on CE starts the transmission. The minimum pulse width on CE is 10 µs.
4. nRF24L01 ShockBurst™:
   - Radio is powered up
   - 16 MHz internal clock is started.
   - RF packet is completed (see the packet description)
   - Data is transmitted at high speed (1 Mbps or 2 Mbps configured by MCU).
5. If auto acknowledgement is activated (ENAA_P0=1) the radio goes into RX mode immediately. If a valid packet has been received in the valid acknowledgement time window, the transmission is considered a success. The TX_DS bit in the status register is set high and the payload is removed from TX FIFO. If a valid acknowledgement is not received in the specified time window, the payload is resent (if auto retransmit is enabled). If the auto retransmit counter (ARC_CNT) exceeds the programmed maximum limit (ARC), the MAX_RT bit in the status register is set high. The payload in TX FIFO is NOT removed. The IRQ pin will be active when MAX_RT or TX_DS is high. To turn off the IRQ pin, the interrupt source must be reset by writing to the status register (see Interrupt chapter). If no acknowledgement is received for a packet after the maximum number of retries, no further packets can be sent before the MAX_RX interrupt is cleared. The packet loss counter (PLOS_CNT) is incremented at each MAX_RT interrupt. I.e. ARC_CNT counts the number of retries that was required to get a single packet through. PLOS_CNT counts the number of packets that did not get through after maximum number of retries.
6. The device goes into Standby-I mode if CE is low. Otherwise next payload in TX FIFO will be sent. If TX FIFO is empty and CE is still high, the device will enter Standby-II mode.
7. If the device is in Standby-II mode, it will go to Standby-I mode immediately if CE is set low.

Enhanced ShockBurst™ Receive Payload:

1. RX is selected by setting the PRIM_RX bit in the configuration register to high. All data pipes that shall receive data must be enabled (EN_RXADDR register),

auto acknowledgement for all pipes running Enhanced ShockBurst™ has to be enabled (EN_AA register), and the correct payload widths must be set (RX_PW_Px registers). Addresses have to be set up as described in item 2 in the Enhanced ShockBurst™ transmit payload chapter above.

2. Active RX mode is started by setting CE high.
3. After 130µs nRF24L01 is monitoring the air for incoming communication.
4. When a valid packet has been received (matching address and correct CRC), the payload is stored in the RX-FIFO, and the RX_DR bit in status register is set high. The IRQ pin will be active when RX_DR is high. RX_P_NO in status register will indicate what data pipe the payload has been received in.
5. If auto acknowledgement is enabled, an acknowledgement is sent back.
6. MCU sets the CE pin low to enter Standby-I mode (low current mode).
7. MCU can clock out the payload data at a suitable rate via the SPI interface.
8. The device is now ready for entering TX or RX mode or power down mode.

Two way communication with payload in both directions

If payload shall be sent in both directions, the PRIM_RX register must be toggled by redefining the device from PRX to PTX or vice versa. The controlling processors must handle the synchronicity between a PTX and a PRX. Data buffering in both RX FIFO and TX FIFO simultaneously is possible, but restricted to data pipes 1 to 5. The third level in TX FIFO shall only be written in RX, TX or Standby-II mode if data is stored in RX FIFO

Auto Acknowledgement (RX)

The auto acknowledgement function reduces the load of the external microcontroller, and may remove the need for dedicated SPI hardware in a mouse/keyboard or comparable systems, and hence reduce cost and average current consumption. Auto acknowledgement can be configured individually for each data pipe via the SPI interface.

If auto acknowledgement is enabled and a valid packet (correct data pipe address and CRC) is received, the device will enter TX mode and send an acknowledgement packet. After the device has sent the acknowledgement packet, normal operation resumes, and the mode is determined by the PRIM_RX register and CE pin.

Auto Re-Transmission (ART) (TX)

An auto retransmission function is available. It will be used at the TX side in an auto acknowledgement system. In the SETUP_RETR register it will be possible to state how many times the data in the data register will be resent if data is not acknowledged. After each sending, the device will enter RX mode and wait a specified time period for acknowledgement. When the acknowledgement packet is received, the device will return to normal transmit function. If there is no more unsent data in the TX FIFO and the CE pin is low, the device will go into Standby-I mode. If the acknowledgement is not received, the device will go back to TX mode and resend the data. This will continue until acknowledgment is received, or a time out occurs

(i.e. the maximum number of sending is reached). The only way to reset this is to set the PWR_UP bit low or let the auto retransmission finish. A packet loss counter will be incremented each time a packet does not succeed to reach the destination before time out. (Time out is indicated by the MAX_RT interrupt.) The packet loss counter is reset when writing to the RF channel register.

Packet Identity (PID) and CRC used by Enhanced ShockBurst™

Each packet contains a two bit wide PID field to detect if the received packet is new or resent. The PID will prevent that the PRX device presents the same payload more than once to the microcontroller. This PID field is incremented at the TX side for each new packet received via the SPI interface. The PID and CRC field is used by the PRX device to determine whether a packet is resent or new. When several data is lost on the link, the PID fields may in some cases become equal to last received PID. If a packet has the same PID as the previous packet, nRF24L01 will compare the CRC sums from both packets. If they also are equal, the last received packet is considered as a copy of the previous and is discarded.

1: PRX device:
The PRX device compares the received PID with the last PID. If the PID fields are different, the packet is considered to be new. If the PID is equal to last received PID, the received packet might be the same as last time. The receiver must check if the CRC is equal to the previous CRC. If the CRC is equal to the previous one, the packet is probably the same, and will be discarded.

2: PTX device:
The transmitter increments the PID field each time it sends a new packet.

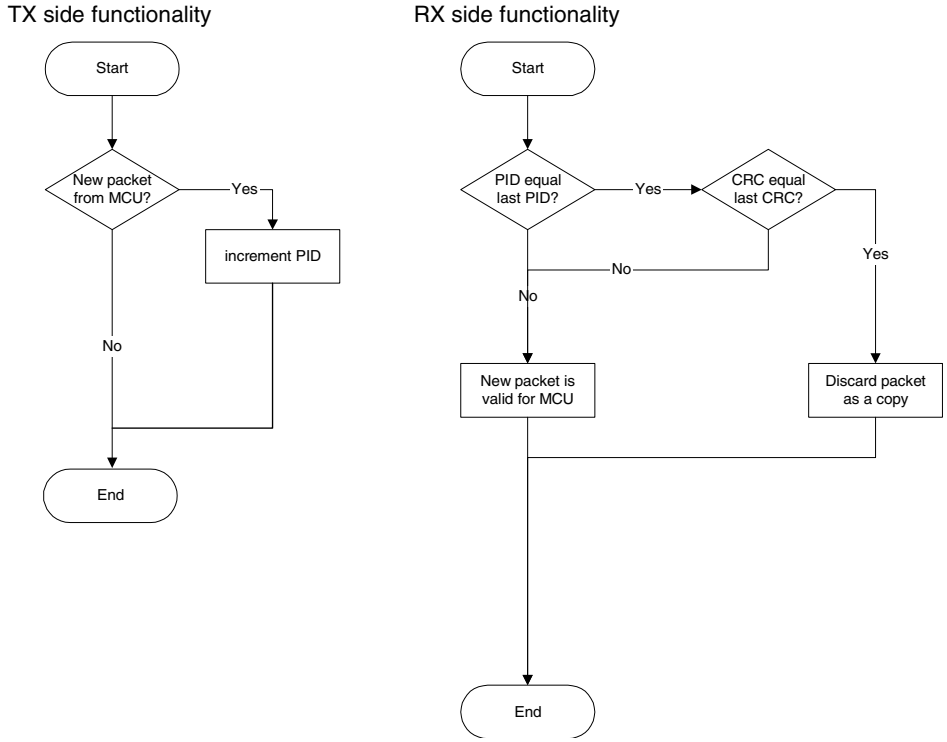


Figure 6 PID generation/detection

The length of the CRC is configurable through the SPI interface. It is important to notice that the CRC is calculated over the whole packet including address, PID and payload. No packet is accepted as correct if the CRC fails. This is an extra requirement for packet acceptance that is not illustrated in the figure above.

Stationary Disturbance Detection – CD

Carrier Detect (CD) is set high when an in-band RF signal is detected in RX mode, otherwise CD is low. The internal CD signal is filtered before presented to CD register. The internal CD signal must be high for at least 128µs.
In Enhanced ShockBurst™ it is recommended to use the Carrier Detect functionality only when the PTX device does not succeed to get packets through, as indicated by the MAX_RT interrupt for single packets and by the packet loss counter (PLOS_CNT) if several packets are lost. If the PLOS_CNT in the PTX device indicates to high rate of packet losses, the device can be configured to a PRX device for a short time (T_{stbt2a} + CD-filter delay = 130µs+128µs = 258µs) to check CD. If CD was high (jam situation), the frequency channel should be changed. If CD was low (out of range), it may continue on the same frequency channel, but perform other adjustments. (A dummy write to the RF_CH will clear the PLOS_CNT.)

Data Pipes

nRF24L01 configured as PRX can receive data addressed to 6 different data pipes in one physical frequency channel. Each data pipe has its own unique address and can be configured to have individual behavior.

The data pipes are enabled with the bits in the EN_RXADDR register. By default only data pipe 0 and 1 are enabled.
The address for each data pipe is configured in the RX_ADDR_Px registers. Always ensure that none of the data pipes have the exact same address.

Data pipe 0 has a unique 40 bit configurable address. Data pipes 1-5 share the 32 most significant address bits and have only the LSByte unique for each data pipe. Figure 7 shows an example of how data pipes 0-5 are addressed. All pipes can have up to 40 bit address, but for pipe 1-5 only the LSByte is different, and the LSByte must be unique for all pipes.

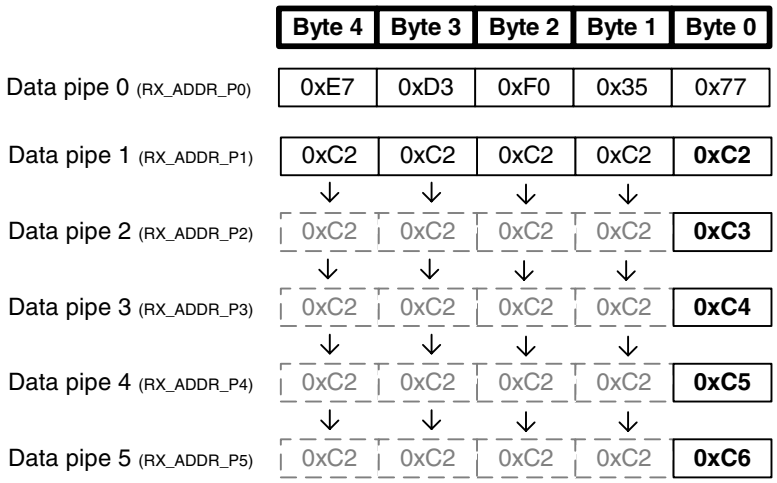


Figure 7: Addressing data pipes 0-5

When a packet has been received at one of the data pipes and the data pipe is setup to generate acknowledgement, nRF24L01 will generate an acknowledgement with an address that equals the data pipe address where the packet was received.

Some configuration settings are common to all data pipes and some are individual. The following settings are common to all data pipes:

- CRC enabled/disabled (CRC always enabled when ESB is enabled)
- CRC encoding scheme
- RX address width
- Frequency channel
- RF data rate
- LNA gain
- RF output power

DEVICE CONFIGURATION

All configuration of nRF24L01 is defined by values in some configuration registers. All these registers are writable via the SPI interface.

SPI Interface

The SPI interface is a standard SPI interface with a maximum data rate of 10Mbps. Most registers are readable.

SPI Instruction Set

The available commands to be used on the SPI interface are shown below. Whenever CSN is set low the interface expects an instruction. Every new instruction must be started by a high to low transition on CSN.

In parallel to the SPI instruction word applied on the MOSI pin, the STATUS register is shifted serially out on the MISO pin.

The serial shifting SPI commands is on the format:

<**Instruction word**: MSBit to LSBit (one byte)>

<**Data bytes**: LSByte to MSByte, MSBit in each byte first>

See Figure 8 and Figure 9.

Instruction Name	Instruction Format [binary]	# Data Bytes	Operation
R_REGISTER	000A AAAA	1 to 5 LSByte first	Read registers. AAAAA = 5 bit Memory Map Address
W_REGISTER	001A AAAA	1 to 5 LSByte first	Write registers. AAAAA = 5 bit Memory Map Address *Executable in power down or standby modes only.*
R_RX_PAYLOAD	0110 0001	1 to 32 LSByte first	Read RX-payload: 1 – 32 bytes. A read operation will always start at byte 0. Payload will be deleted from FIFO after it is read. Used in RX mode.
W_TX_PAYLOAD	1010 0000	1 to 32 LSByte first	Used in TX mode. Write TX-payload: 1 – 32 bytes. A write operation will always start at byte 0.
FLUSH_TX	1110 0001	0	Flush TX FIFO, used in TX mode
FLUSH_RX	1110 0010	0	Flush RX FIFO, used in RX mode Should not be executed during transmission of acknowledge, i.e. acknowledge package will not be completed.
REUSE_TX_PL	1110 0011	0	Used for a PTX device Reuse last sent payload. Packets will be repeatedly resent as long as CE is high. TX payload reuse is active until W_TX_PAYLOAD or FLUSH TX is executed. TX payload reuse must not be activated or deactivated during package transmission
NOP	1111 1111	0	No Operation. Might be used to read the STATUS register

Table 8 Instruction set for the nRF24L01 SPI interface.

The W_REGISTER and R_REGISTER may operate on single or multi-byte registers. When accessing multi-byte registers one will read or write MSBit of LSByte first. The

writing can be terminated before all bytes in a multi-byte register has been written. In this case the unwritten MSByte(s) will remain unchanged. E.g. the LSByte of RX_ADDR_P0 can be modified by writing only one byte to the RX_ADDR_P0 register. The content of the status register will always be read to MISO after a high to low transition on CSN.

Interrupt

The nRF24L01 has an active low interrupt pin (IRQ). The interrupt pin is activated when TX_DS, RX_DR or MAX_RT is set high in status register. When MCU writes '1' to the interrupt source, the IRQ pin will go inactive. The interrupt mask part of the CONFIG register is used to mask out the interrupt sources that are allowed to set the IRQ pin low. By setting one of the MASK bits high, the corresponding interrupt source will be disabled. By default all interrupt sources are enabled.

SPI Timing

The interface supports SPI. SPI operation and timing is given in Figure 8 to Figure 10 and in Table 9 and Table 10. The device must be in one of the standby modes or power down mode before writing to the configuration registers. In Figure 8 to Figure 10 the following notations are used:
Cn – SPI Instruction Bit
Sn – Status Register Bit
Dn – Data Bit (note: LSByte to MSByte, MSBit in each byte first)

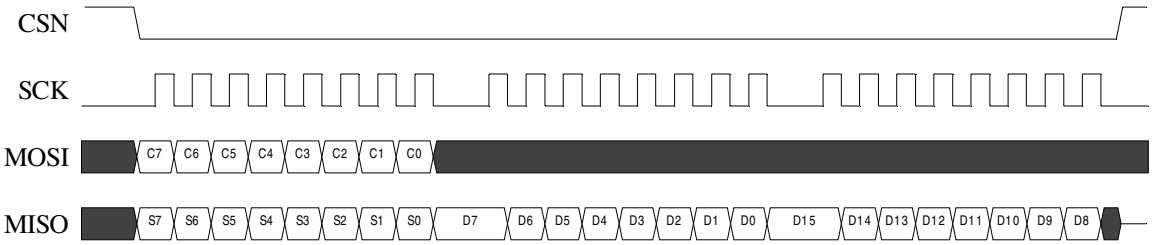


Figure 8 SPI read operation.

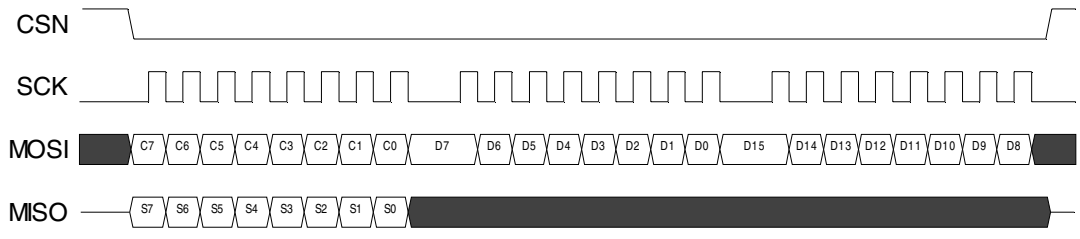


Figure 9 SPI write operation.

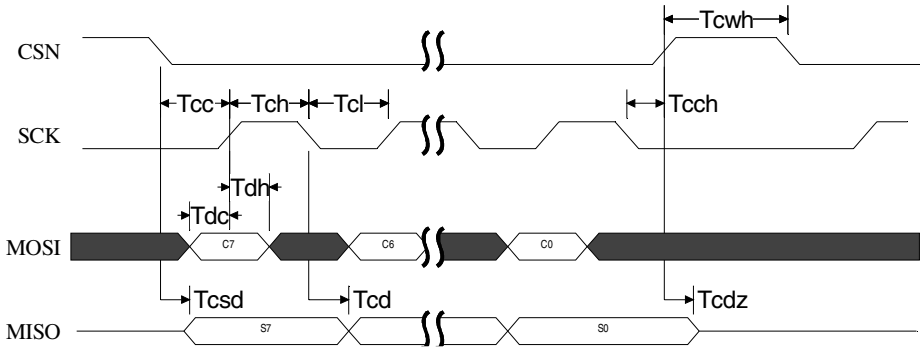


Figure 10 SPI NOP timing diagram.

PARAMETER	SYMBOL	MIN	MAX	UNITS
Data to SCK Setup	Tdc	2		ns
SCK to Data Hold	Tdh	2		ns
CSN to Data Valid	Tcsd		38	ns
SCK to Data Valid	Tcd		55	ns
SCK Low Time	Tcl	40		ns
SCK High Time	Tch	40		ns
SCK Frequency	Fsck	0	8	MHz
SCK Rise and Fall	Tr,Tf		100	ns
CSN to SCK Setup	Tcc	2		ns
SCK to CSN Hold	Tcch	2		ns
CSN Inactive time	Tcwh	50		ns
CSN to Output High Z	Tcdz		38	ns

Table 9 SPI timing parameters (C_{Load} = 5pF).

PARAMETER	SYMBOL	MIN	MAX	UNITS
Data to SCK Setup	Tdc	2		ns
SCK to Data Hold	Tdh	2		ns
CSN to Data Valid	Tcsd		42	ns
SCK to Data Valid	Tcd		58	ns
SCK Low Time	Tcl	40		ns
SCK High Time	Tch	40		ns
SCK Frequency	Fsck	0	8	MHz
SCK Rise and Fall	Tr,Tf		100	ns
CSN to SCK Setup	Tcc	2		ns
SCK to CSN Hold	Tcch	2		ns
CSN Inactive time	Tcwh	50		ns
CSN to Output High Z	Tcdz		42	ns

Table 10 SPI timing parameters (C_{Load} = 10pF).

MEMORY MAP

All undefined bits in the table below are redundant. They will be read out as '0'.

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
00	CONFIG				Configuration Register
	Reserved	7	0	R/W	Only '0' allowed
	MASK_RX_DR	6	0	R/W	Mask interrupt caused by RX_DR 1: Interrupt not reflected on the IRQ pin 0: Reflect RX_DR as active low interrupt on the IRQ pin
	MASK_TX_DS	5	0	R/W	Mask interrupt caused by TX_DS 1: Interrupt not reflected on the IRQ pin 0: Reflect TX_DS as active low interrupt on the IRQ pin
	MASK_MAX_RT	4	0	R/W	Mask interrupt caused by MAX_RT 1: Interrupt not reflected on the IRQ pin 0: Reflect MAX_RT as active low interrupt on the IRQ pin
	EN_CRC	3	1	R/W	Enable CRC. Forced high if one of the bits in the EN_AA is high
	CRCO	2	0	R/W	CRC encoding scheme '0' - 1 byte '1' – 2 bytes
	PWR_UP	1	0	R/W	1: POWER UP, 0:POWER DOWN
	PRIM_RX	0	0	R/W	1: PRX, 0: PTX
01	EN_AA Enhanced ShockBurst™				Enable 'Auto Acknowledgment' Function Disable this functionality to be compatible with nRF2401, see page 26
	Reserved	7:6	00	R/W	Only '00' allowed
	ENAA_P5	5	1	R/W	Enable auto ack. data pipe 5
	ENAA_P4	4	1	R/W	Enable auto ack. data pipe 4
	ENAA_P3	3	1	R/W	Enable auto ack. data pipe 3
	ENAA_P2	2	1	R/W	Enable auto ack. data pipe 2
	ENAA_P1	1	1	R/W	Enable auto ack. data pipe 1
	ENAA_P0	0	1	R/W	Enable auto ack. data pipe 0
02	EN_RXADDR				Enabled RX Addresses
	Reserved	7:6	00	R/W	Only '00' allowed
	ERX _P5	5	0	R/W	Enable data pipe 5.
	ERX _P4	4	0	R/W	Enable data pipe 4.
	ERX _P3	3	0	R/W	Enable data pipe 3.
	ERX _P2	2	0	R/W	Enable data pipe 2.
	ERX _P1	1	1	R/W	Enable data pipe 1.
	ERX _P0	0	1	R/W	Enable data pipe 0.
03	SETUP_AW				Setup of Address Widths (common for all data pipes)
	Reserved	7:2	000000	R/W	Only '000000' allowed
	AW	1:0	11	R/W	RX/TX Address field width '00' - Illegal '01' - 3 bytes '10' - 4 bytes '11' – 5 bytes LSByte will be used if address width below 5 bytes
04	SETUP_RETR				Setup of Automatic Retransmission
	ARD	7:4	0000	R/W	Auto Re-transmit Delay '0000' – Wait 250+86uS

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
					‘0001’ – Wait 500+86uS ‘0010’ – Wait 750+86uS …….. ‘1111’ – Wait 4000+86uS (Delay defined from end of transmission to start of next transmission)[14]
	ARC	3:0	0011	R/W	Auto Retransmit Count ‘0000’ –Re-Transmit disabled ‘0001’ – Up to 1 Re-Transmit on fail of AA …… ‘1111’ – Up to 15 Re-Transmit on fail of AA
05	RF_CH				RF Channel
	Reserved	7	0	R/W	Only '0' allowed
	RF_CH	6:0	0000010	R/W	Sets the frequency channel nRF24L01 operates on
06	RF_SETUP				RF Setup Register
	Reserved	7:5	000	R/W	Only '000' allowed
	PLL_LOCK	4	0	R/W	Force PLL lock signal. Only used in test
	RF_DR	3	1	R/W	Data Rate ‘0’ – 1 Mbps ‘1’ – 2 Mbps
	RF_PWR	2:1	11	R/W	Set RF output power in TX mode '00' – -18 dBm '01' – -12 dBm '10' – -6 dBm '11' – 0 dBm
	LNA_HCURR	0	1	R/W	Setup LNA gain
07	STATUS				Status Register (In parallel to the SPI instruction word applied on the MOSI pin, the STATUS register is shifted serially out on the MISO pin)
	Reserved	7	0	R/W	Only '0' allowed
	RX_DR	6	0	R/W	Data Ready RX FIFO interrupt. Set high when new data arrives RX FIFO[15]. Write 1 to clear bit.
	TX_DS	5	0	R/W	Data Sent TX FIFO interrupt. Set high when packet sent on TX. If AUTO_ACK is activated, this bit will be set high only when ACK is received. Write 1 to clear bit.
	MAX_RT	4	0	R/W	Maximum number of TX retries interrupt Write 1 to clear bit. If MAX_RT is set it must be cleared to enable further communication.
	RX_P_NO	3:1	111	R	Data pipe number for the payload

[14] Accurate formula for delay from start of transmission, to start of re-transmission:
TRD (us) = 250us * (ARD+1) + 4us *(AW + PW + CRCW) +138,5us.
TRD= total retransmit delay, AW=Address Width (#bytes), PW=Payload Width(#bytes)
, CRCW= CRC Width (#bytes)

[15] The Data Ready interrupt is set by a new packet arrival event. The procedure for handling this interrupt should be: 1) read payload via SPI, 2) clear RX_DR interrupt, 3) read FIFO_STATUS to check if there are more payloads available in RX FIFO, 4) if there are more data in RX FIFO, repeat from 1).

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
					available for reading from RX_FIFO 000-101: Data Pipe Number 110: Not Used 111: RX FIFO Empty
	TX_FULL	0	0	R	TX FIFO full flag. 1: TX FIFO full. 0: Available locations in TX FIFO.
08	OBSERVE_TX				Transmit observe register
	PLOS_CNT	7:4	0	R	Count lost packets. The counter is overflow protected to 15, and discontinue at max until reset. The counter is reset by writing to RF_CH. See page 14 and 17.
	ARC_CNT	3:0	0	R	Count resent packets. The counter is reset when transmission of a new packet starts. See page 14.
09	CD				
	Reserved	7:1	000000	R	
	CD	0	0	R	Carrier Detect. See page 17.
0A	RX_ADDR_P0	39:0	0xE7E7E7E7E7	R/W	Receive address data pipe 0. 5 Bytes maximum length. (LSByte is written first. Write the number of bytes defined by SETUP_AW)
0B	RX_ADDR_P1	39:0	0xC2C2C2C2C2	R/W	Receive address data pipe 1. 5 Bytes maximum length. (LSByte is written first. Write the number of bytes defined by SETUP_AW)
0C	RX_ADDR_P2	7:0	0xC3	R/W	Receive address data pipe 2. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]
0D	RX_ADDR_P3	7:0	0xC4	R/W	Receive address data pipe 3. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]
0E	RX_ADDR_P4	7:0	0xC5	R/W	Receive address data pipe 4. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]
0F	RX_ADDR_P5	7:0	0xC6	R/W	Receive address data pipe 5. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]
10	TX_ADDR	39:0	0xE7E7E7E7E7	R/W	Transmit address. Used for a PTX device only. (LSByte is written first) Set RX_ADDR_P0 equal to this address to handle automatic acknowledge if this is a PTX device with Enhanced ShockBurst™ enabled. See page 14.
11	RX_PW_P0				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P0	5:0	0	R/W	Number of bytes in RX payload in data pipe 0 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
12	RX_PW_P1				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P1	5:0	0	R/W	Number of bytes in RX payload in data pipe 1 (1 to 32 bytes). 0 Pipe not used

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
					1 = 1 byte … 32 = 32 bytes
13	RX_PW_P2				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P2	5:0	0	R/W	Number of bytes in RX payload in data pipe 2 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
14	RX_PW_P3				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P3	5:0	0	R/W	Number of bytes in RX payload in data pipe 3 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
15	RX_PW_P4				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P4	5:0	0	R/W	Number of bytes in RX payload in data pipe 4 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
16	RX_PW_P5				
	Reserved	7:6	00	R/W	Only '00' allowed
	RX_PW_P5	5:0	0	R/W	Number of bytes in RX payload in data pipe 5 (1 to 32 bytes). 0 Pipe not used 1 = 1 byte … 32 = 32 bytes
17	FIFO_STATUS				FIFO Status Register
	Reserved	7	0	R/W	Only '0' allowed
	TX_REUSE	6	0	R	Reuse last sent data packet if set high. The packet will be repeatedly resent as long as CE is high. TX_REUSE is set by the SPI instruction REUSE_TX_PL, and is reset by the SPI instructions W_TX_PAYLOAD or FLUSH TX
	TX_FULL	5	0	R	TX FIFO full flag. 1: TX FIFO full. 0: Available locations in TX FIFO.
	TX_EMPTY	4	1	R	TX FIFO empty flag. 1: TX FIFO empty. 0: Data in TX FIFO.
	Reserved	3:2	00	R/W	Only '00' allowed
	RX_FULL	1	0	R	RX FIFO full flag. 1: RX FIFO full. 0: Available locations in RX FIFO.
	RX_EMPTY	0	1	R	RX FIFO empty flag. 1: RX FIFO empty. 0: Data in RX FIFO.
N/A	TX_PLD	255:0	X	W	Written by separate SPI command TX data payload register 1 - 32 bytes. This register is implemented as a FIFO with 3 levels. Used in TX mode only
N/A	RX_PLD	255:0	X	R	Written by separate SPI command RX data payload register. 1 - 32 bytes. This register is implemented as a FIFO

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
					with 3 levels. All receive channels share the same FIFO

Table 11 Memory map of nRF24L01

Configuration for compatibility with nRF24XX

How to setup nRF24L01 to receive from an nRF2401/nRF2402/nRF24E1/nRF24E2:

- Use same CRC configuration as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
- Set the PRIM_RX bit to 1
- Disable auto acknowledgement on the data pipe that will be addressed
- Use the same address width as the PTX device
- Use the same frequency channel as the PTX device
- Select data rate 1Mbit/s on both nRF24L01 and nRF2401/nRF2402/nRF24E1/nRF24E2
- Set correct payload width on the data pipe that will be addressed
- Set PWR_UP and CE high

How to setup nRF24L01 to transmit to an nRF2401/nRF24E1:

- Use same CRC configuration as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
- Set the PRIM_RX bit to 0
- Set the Auto Retransmit Count to 0 to disable the auto retransmit functionality
- Use the same address width as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
- Use the same frequency channel as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
- Select data rate 1Mbit/s on both nRF24L01 and nRF2401/nRF2402/nRF24E1/nRF24E2
- Set PWR_UP high
- Clock in a payload that has the same length as the nRF2401/nRF2402/nRF24E1/nRF24E2 is configured to receive
- Pulse CE to send the packet

PACKET DESCRIPTION

An Enhanced ShockBurst™ packet with payload (1-32 bytes).

A ShockBurst™ packet compatible to nRF2401/nRF2402/nRF24E1/nRF24E2 devices.

Preamble	• Preamble is used to detect 0 and 1 levels. It is stripped off (RX) and added (TX) by nRF24L01.
Address	• The address field contains the receiver address. • The address can be 3, 4 or 5 bytes wide • The address fields can be individually configured for all RX channels and the TX channel • Address is automatically removed from received packets.[16]
Flags	• PID: Packet Identification. 2 bits that is incremented for each new payload • 7 bits reserved for packet compatibility with future products • Not used when compatible to nRF2401/nRF24E1
Payload	• 1 - 32 bytes wide.
CRC	• The CRC is optional. • 0-2 bytes wide CRC • The polynomial for 8 bits CRC check is $X^8 + X^2 + X + 1$ • The polynomial for 16 bits CRC check is $X^{16} + X^{12} + X^5 + 1$.

Table 12 Data packet description

[16] Suggested use of addresses. In general more bits in the address gives less false detection, which in the end may give lower data Packet-Error-Rate (PER).

A. The address made by (5, 4, or 3) equal bytes are not recommended because it in general will make the packet-error-rate increase.
B. Addresses where the level shift only one time (i.e. 000FFFFFFF) could often be detected in noise that may give a false detection, which again may give raised packet-error-rate.
C. Addresses as a continuation of the preamble (hi-low toggling) will raise the Packet-Error-Rate (PER).

IMPORTANT TIMING DATA

The following timing applies for operation of nRF24L01.

nRF24L01 Timing Information

nRF24L01 timing	Max.	Min.	Name
Power Down ➔ Standby mode	1.5ms		Tpd2stby
Standby modes ➔ TX/RX mode	130µs		Tstby2a
Minimum CE high		10µs	Thce
Delay from CE pos. edge to CSN low		4µs	Tpece2csn

Table 13 Operational timing of nRF24L01

When the nRF24L01 is in power down it must always settle in Standby for 1.5ms before it can enter one of the TX or RX modes. Note that the configuration word will be lost if VDD is turned off and that the device then must be configured before going to one of the TX or RX mode.

Enhanced ShockBurst™ timing

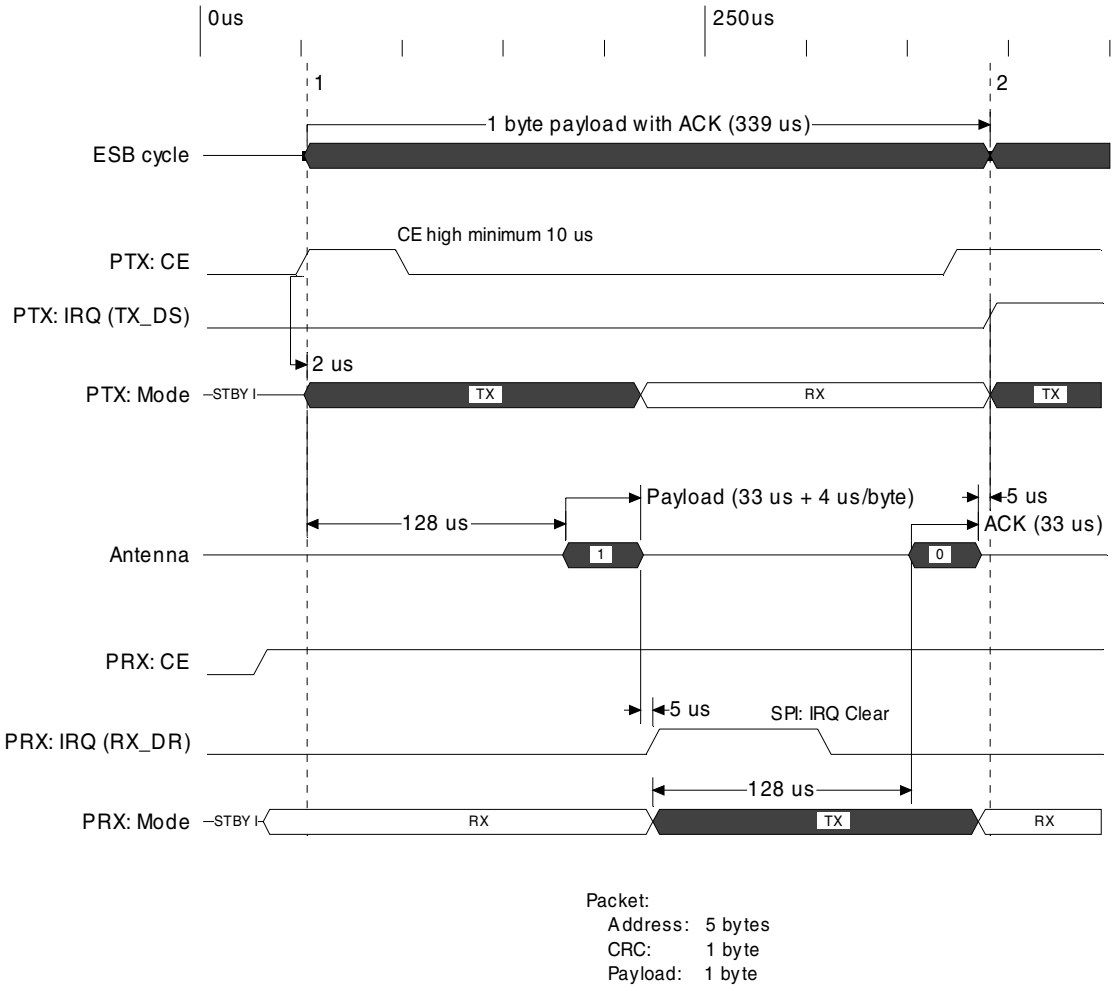


Figure 11 Timing of Enhanced ShockBurst™ for one packet upload (2Mbps).

In Figure 11 the sending of one packet and the acknowledgement of this packet is shown. The loading of payload to the PTX device is not shown in the figure. The PRX device is turned into RX mode (CE=1), and the PTX device is set into TX mode (CE=1 for minimum 10 µs). After 130 µs the transmission starts and is finished after another 37 µs (1 byte payload). The transmission ends, and the PTX device is automatically turned around to RX mode to wait for the acknowledgement from the PRX device. After the PTX device has received the acknowledgement it gives an interrupt to the MCU (IRQ (TX_DS) =>TX-data sent). After the PRX device has received the packet it gives an interrupt to the MCU (IRQ (RX_DR) =>RX-data ready).

PERIPHERAL RF INFORMATION

Antenna output

The ANT1 & ANT2 output pins provide a balanced RF output to the antenna. The pins must have a DC path to VDD, either via a RF choke or via the center point in a dipole antenna. A load of 15Ω+j88Ω (simulated values) is recommended for maximum output power (0dBm). Lower load impedance (for instance 50 Ω) can be obtained by fitting a simple matching network between the load and ANT1 and ANT2.

Output Power adjustment

SPI RF-SETUP (RF_PWR)	RF output power	DC current consumption
11	0 dBm	11.3 mA
10	-6 dBm	9.0 mA
01	-12 dBm	7.5 mA
00	-18 dBm	7.0 mA

Conditions: VDD = 3.0V, VSS = 0V, T_A = 27ºC, Load impedance = 15Ω+j88Ω.

Table 14 RF output power setting for the nRF24L01.

Crystal Specification

Frequency accuracy includes initial accuracy (tolerance) and stability over temperature and aging.

Frequency	C_L	ESR max	C_{0max}	Frequency accuracy
16MHz	8 – 16 pF	100 Ω	7.0pF	±60ppm

Table 15 Crystal specification of the nRF24L01

To achieve a crystal oscillator solution with low power consumption and fast start-up time, it is recommended to specify the crystal with a low value of crystal load capacitance. Specifying a lower value of crystal parallel equivalent capacitance, C_0 will also work, but this can increase the price of the crystal itself. Typically C_0=1.5pF at a crystal specified for C_{0max}=7.0pF.

The crystal load capacitance, C_L, is given by:

$C_L = \frac{C_1' \cdot C_2'}{C_1' + C_2'}$, where $C_1' = C_1 + C_{PCB1} + C_{I1}$ and $C_2' = C_2 + C_{PCB2} + C_{I2}$

C_1 and C_2 are SMD capacitors as shown in the application schematics. C_{PCB1} and C_{PCB2} are the layout parasitic on the circuit board. C_{I1} and C_{I2} are the capacitance seen into the XC1 and XC2 pin respectively; the value is typical 1pF.

nRF24L01 sharing crystal with a micro controller.

When using a micro controller to drive the crystal reference input XC1 of the nRF24L01 transceiver some rules must be followed.

Crystal parameters:

When the micro controller drives the nRF24L01 clock input, the requirement of load capacitance C_L is set by the micro controller only. The frequency accuracy of ±60 ppm is still required to get a functional radio link. The nRF24L01 will load the crystal by 0.5pF at XC1 in addition to the PBC routing.

Input crystal amplitude & Current consumption

The input signal should not have amplitudes exceeding any rail voltage, but any DC-voltage within this is OK. Exceeding rail voltage will excite the ESD structure and the radio performance is degraded below specification. If testing the nRF24L01 with a RF source with no DC offset as the reference source, the input signal will go below the ground level, which is not acceptable.

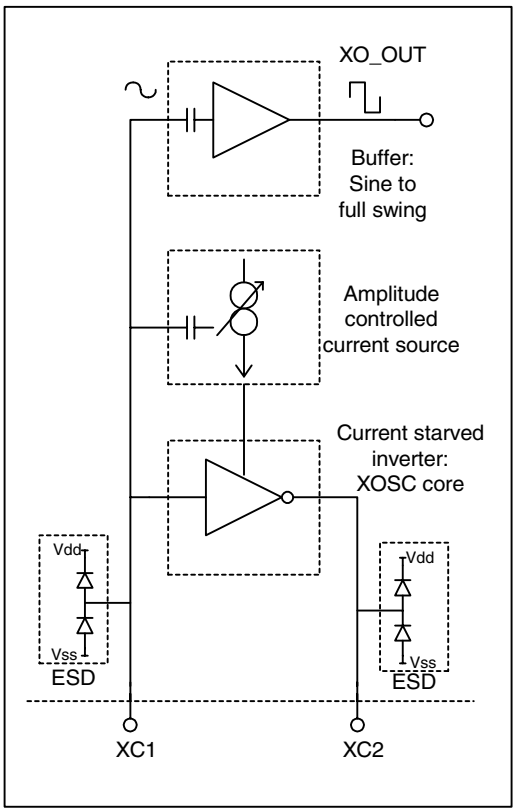


Figure 12 Principle of crystal oscillator

The nRF24L01 crystal oscillator is amplitude regulated. To achieve low current consumption and also good signal-to-noise ratio when using an external clock, it is recommended to use an input signal larger than 0.4 V-peak. When clocked externally, XC2 is not used and can be left as an open pin.

PCB layout and de-coupling guidelines

A well-designed PCB is necessary to achieve good RF performance. Keep in mind that a poor layout may lead to loss of performance, or even functionality, if due care is not taken. A fully qualified RF-layout for the nRF24L01 and its surrounding components, including matching networks, can be downloaded from **www.nordicsemi.no**.

A PCB with a minimum of two layers including a ground plane is recommended for optimum performance. The nRF24L01 DC supply voltage should be de-coupled as close as possible to the VDD pins with high performance RF capacitors, see Table 16. It is preferable to mount a large surface mount capacitor (e.g. 4.7µF tantalum) in parallel with the smaller value capacitors. The nRF24L01 supply voltage should be filtered and routed separately from the supply voltages of any digital circuitry.

Long power supply lines on the PCB should be avoided. All device grounds, VDD connections and VDD bypass capacitors must be connected as close as possible to the nRF24L01 IC. For a PCB with a topside RF ground plane, the VSS pins should be connected directly to the ground plane. For a PCB with a bottom ground plane, the best technique is to have via holes as close as possible to the VSS pads. At least one via hole should be used for each VSS pin.

Full swing digital data or control signals should not be routed close to the crystal or the power supply lines.

APPLICATION EXAMPLE

nRF24L01 with single ended matching network crystal, bias resistor, and decoupling capacitors.

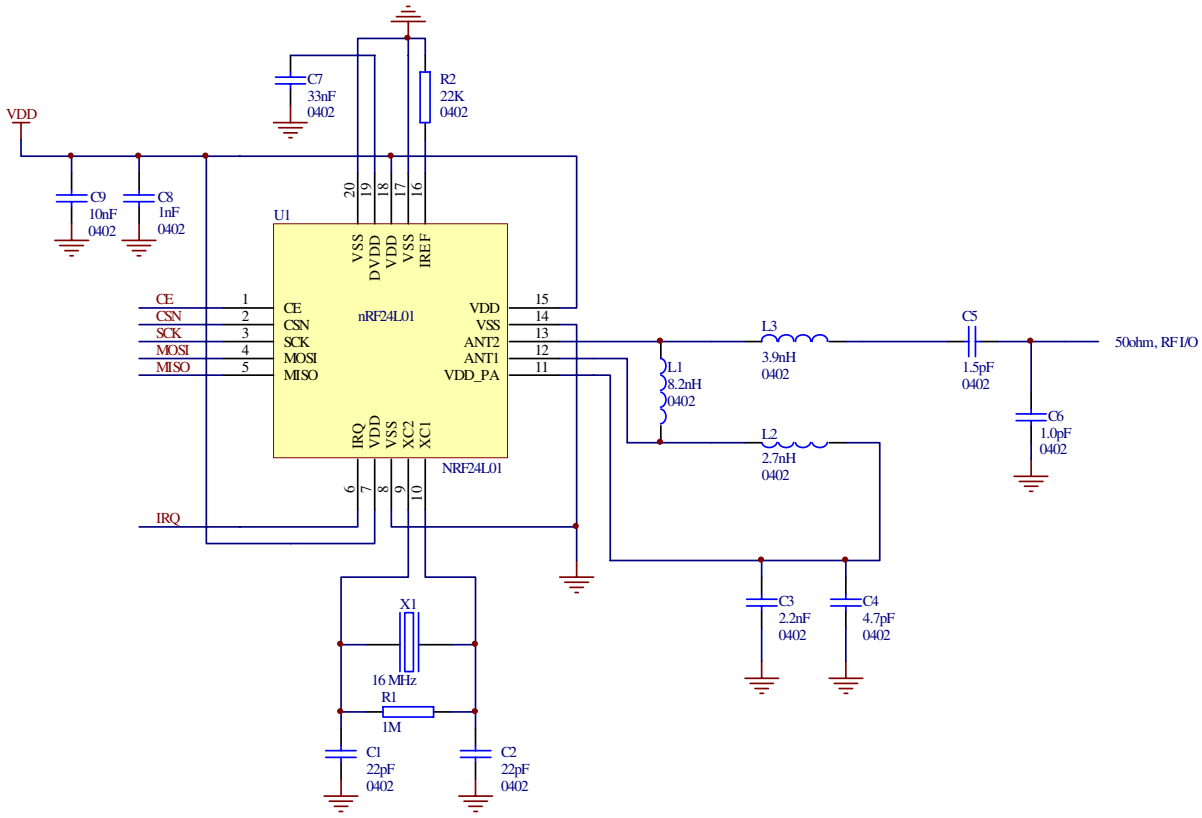


Figure 13 nRF24L01 schematic for RF layouts with single ended 50Ω RF output.

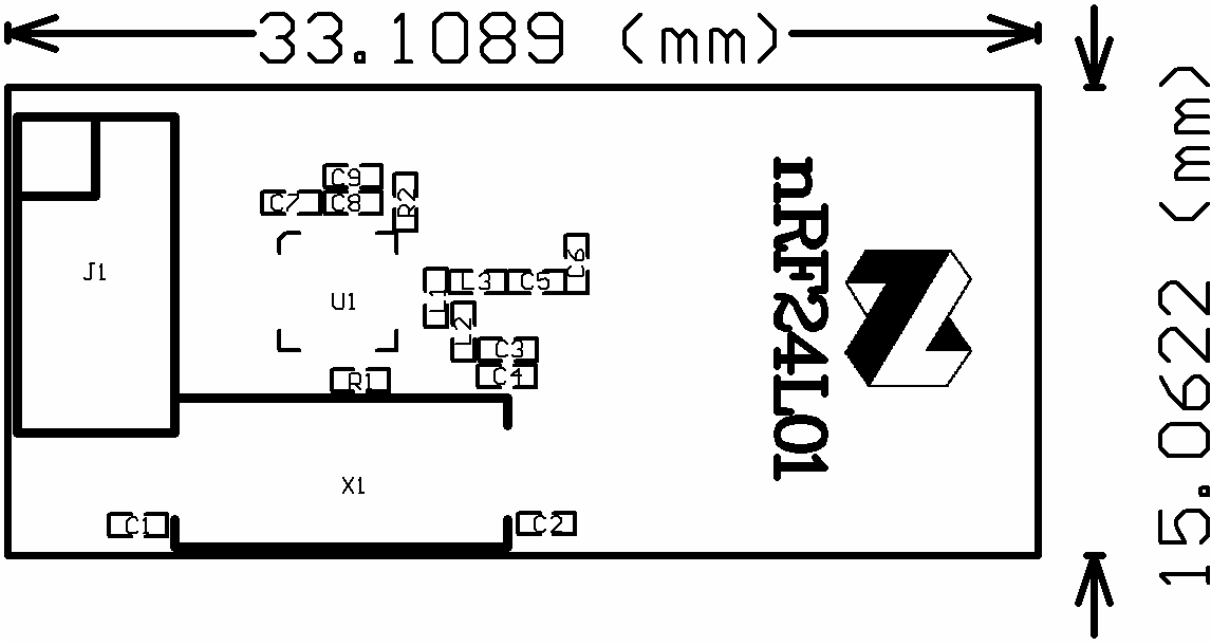
Part	Designator	Footprint	Description
22pF[17]	C1	0402	NPO, +/- 2%, 50V
22pF[17]	C2	0402	NPO, +/- 2%, 50V
2.2nF	C3	0402	X7R, +/- 10%, 50V
4.7pF	C4	0402	NPO, +/- 0.25 pF, 50V
1.5pF	C5	0402	NPO, +/- 0.1 pF, 50V
1,0pF	C6	0402	NPO, +/- 0.1 pF, 50V
33nF	C7	0402	X7R, +/- 10%, 50V
1nF	C8	0402	X7R, +/- 10%, 50V
10nF	C9	0402	X7R, +/- 10%, 50V
8,2nH	L1	0402	chip inductor +/- 5%
2.7nH	L2	0402	chip inductor +/- 5%
3,9nH	L3	0402	chip inductor +/- 5%
1M	R1	0402	+/-10%
22K	R2	0402	+/- 1 %
nRF24L01	U1	QFN20 4x4	
16MHz	X1		+/-60ppm, C_L=12pF[17]

Table 16 Recommended components (BOM) in nRF24L01 with antenna matching network

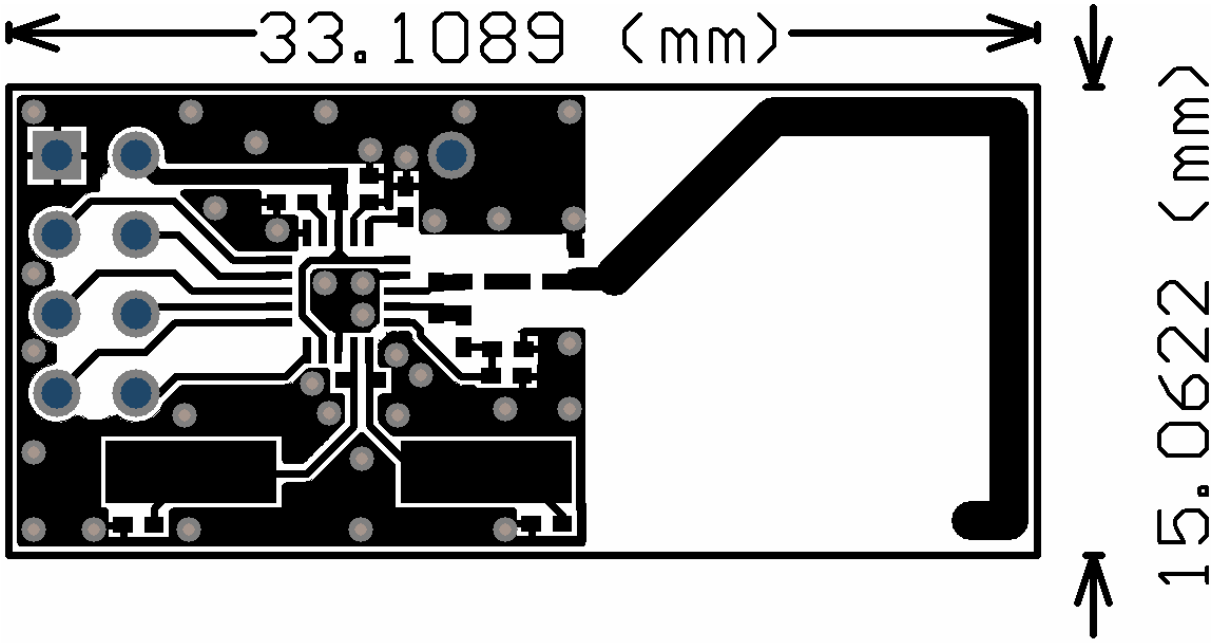
[17] C1 and C2 must have values that match the crystals load capacitance, C_L.

PCB layout examples

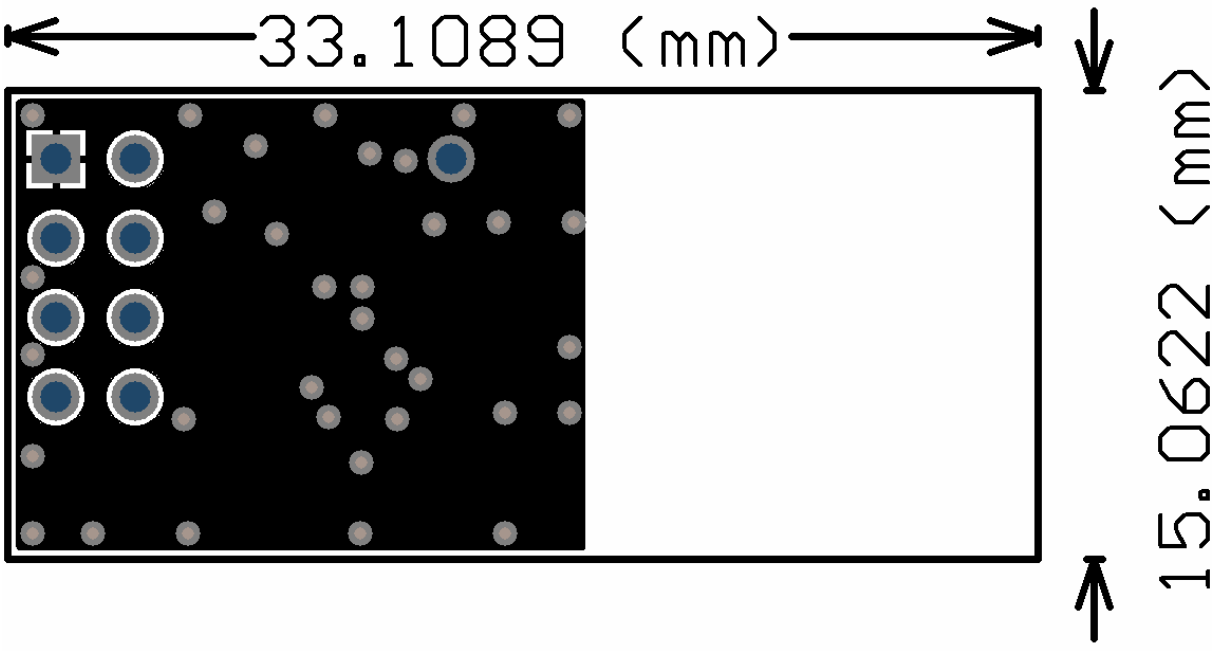
Figure 14 shows a PCB layout example for the application schematic in Figure 13. A double-sided FR-4 board of 1.6mm thickness is used. This PCB has a ground plane on the bottom layer. Additionally, there are ground areas on the component side of the board to ensure sufficient grounding of critical components. A large number of via holes connect the top layer ground areas to the bottom layer ground plane.



Top overlay



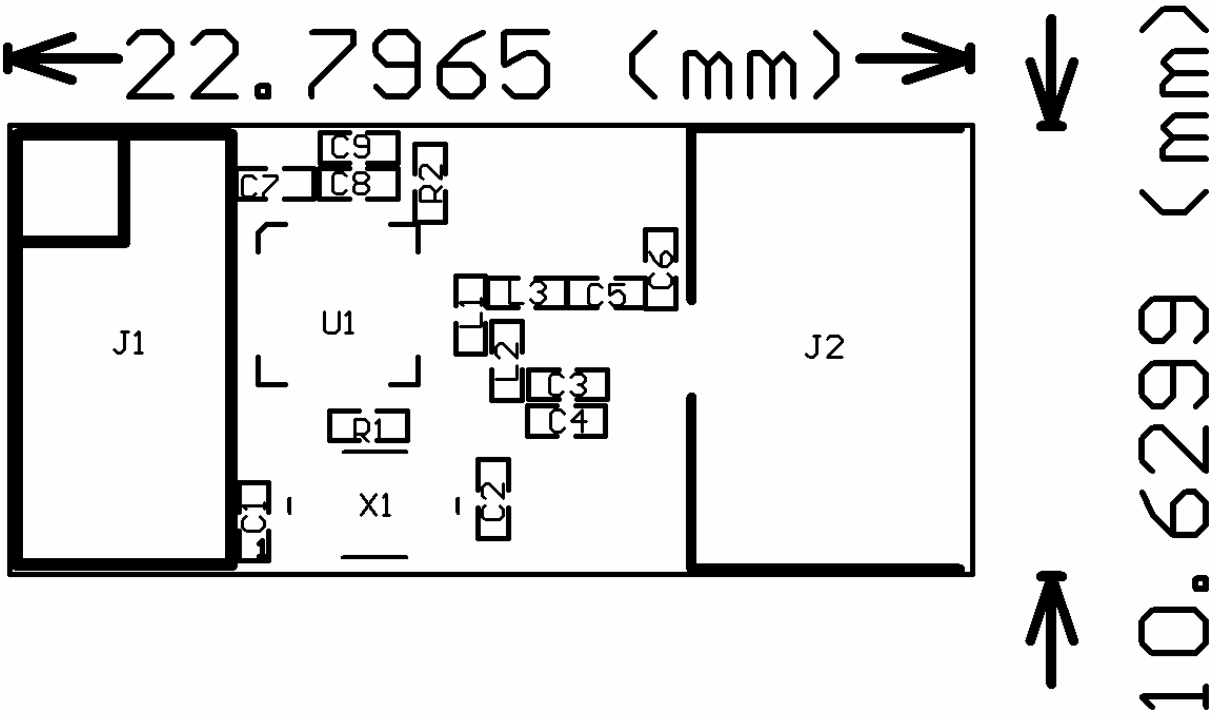
Top layer



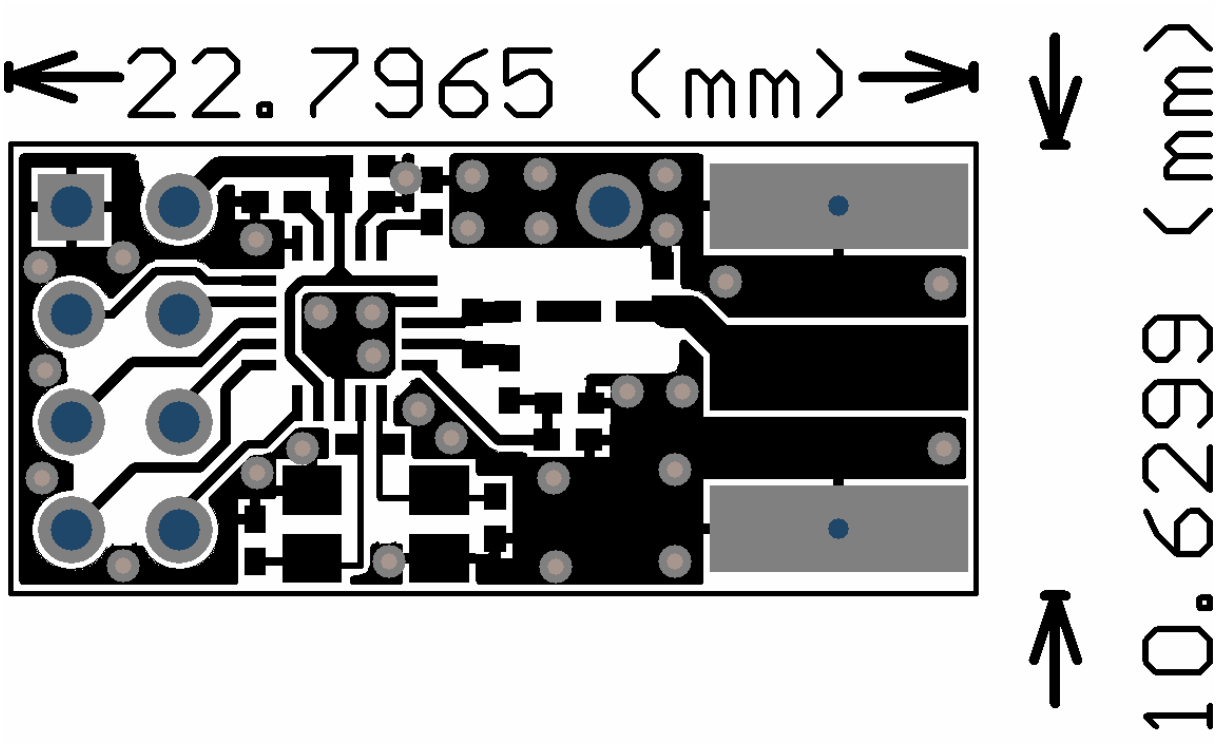
Bottom layer

Figure 14 nRF24L01 RF layout with single ended connection to PCB antenna and 0603 size passive components

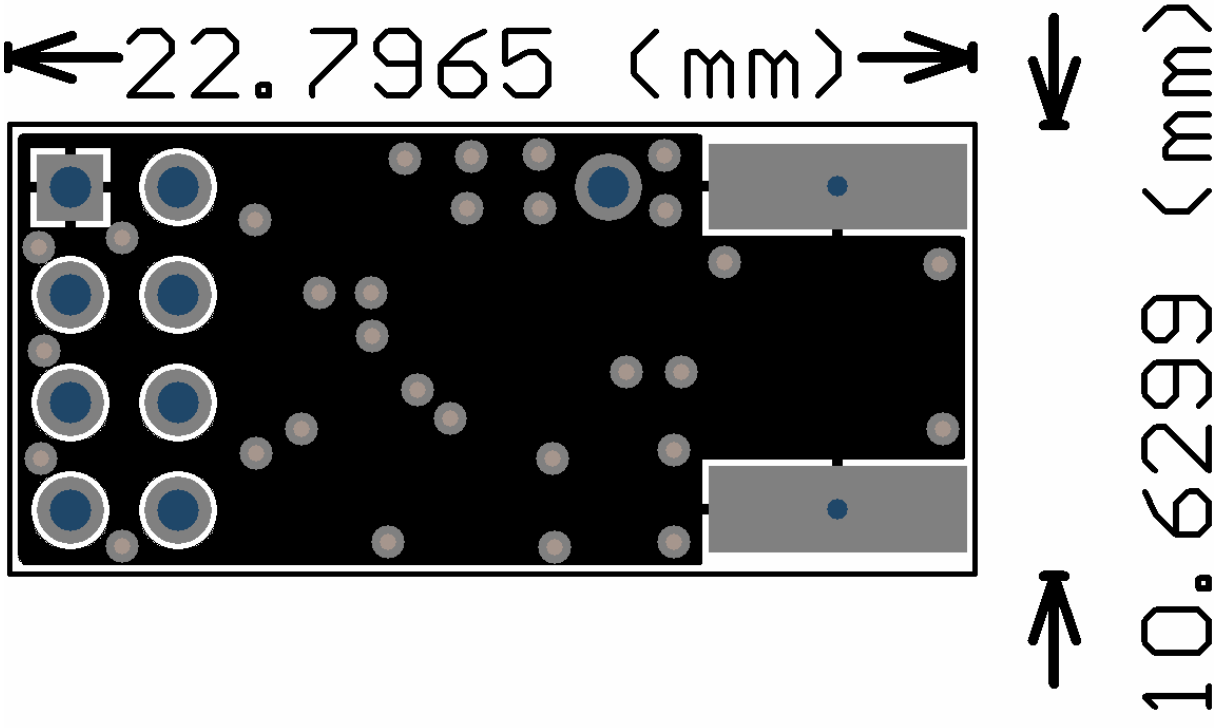
The nest figure (Figure 15) is for the SMA output to have a board for direct measurements at a 50Ω SMA connector.



Top Overlay



Top Layer



Bottom Layer

Figure 15 Module with OFM crystal and SMA connector

DEFINITIONS

Data sheet status	
Objective product specification	This data sheet contains target specifications for product development.
Preliminary product specification	This data sheet contains preliminary data; supplementary data may be published from Nordic Semiconductor ASA later.
Product specification	This data sheet contains final product specifications. Nordic Semiconductor ASA reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
Limiting values	
Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Specifications sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
Application information	
Where application information is given, it is advisory and does not form part of the specification.	

Table 17. Definitions

Nordic Semiconductor ASA reserves the right to make changes without further notice to the product to improve reliability, function or design. Nordic Semiconductor ASA does not assume any liability arising out of the application or use of any product or circuits described herein.

LIFE SUPPORT APPLICATIONS

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Nordic Semiconductor ASA customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Nordic Semiconductor ASA for any damages resulting from such improper use or sale.

Preliminary Product Specification: Revision Date: 08.03.2006.

Data sheet order code: 080306-nRF24L01

YOUR NOTES

Nordic Semiconductor ASA – World Wide Distributors

For Your nearest dealer, please see http://www.nordicsemi.no

Main Office:
Vestre Rosten 81, N-7075 Tiller, Norway
Phone: +47 72 89 89 00, Fax: +47 72 89 89 89

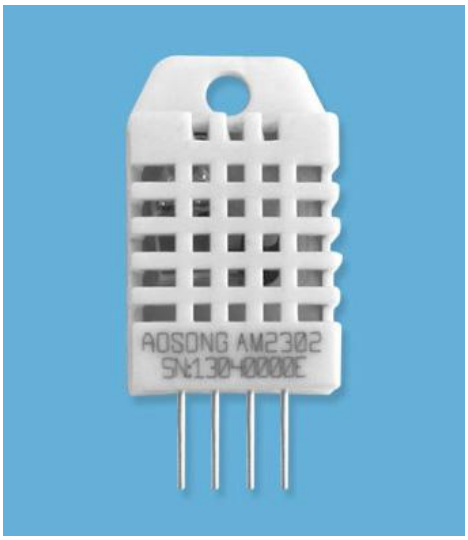
Visit the Nordic Semiconductor ASA website at http://www.nordicsemi.no

参考資料

AOSONG

Temperature and humidity module

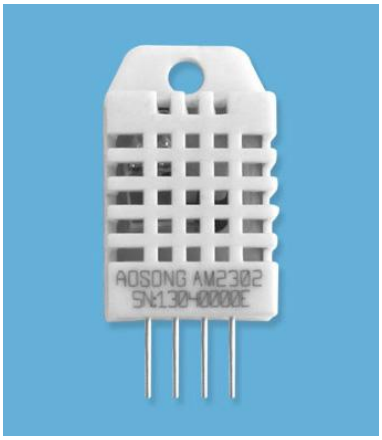
AM2302 Product Manual



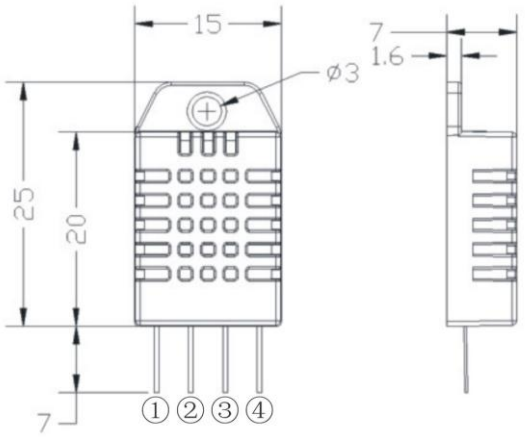
www.aosong.com

1、Product Overview

AM2302 capacitive humidity sensing digital temperature and humidity module is one that contains the compound has been calibrated digital signal output of the temperature and humidity sensors. Application of a dedicated digital modules collection technology and the temperature and humidity sensing technology, to ensure that the product has high reliability and excellent long-term stability. The sensor includes a capacitive sensor wet components and a high-precision temperature measurement devices, and connected with a high-performance 8-bit microcontroller. The product has excellent quality, fast response, strong anti-jamming capability, and high cost. Each sensor is extremely accurate humidity calibration chamber calibration. The form of procedures, the calibration coefficients stored in the microcontroller, the sensor within the processing of the heartbeat to call these calibration coefficients. Standard single-bus interface, system integration quick and easy. Small size, low power consumption, signal transmission distance up to 20 meters, making it the best choice of all kinds of applications and even the most demanding applications. Products for the 3-lead (single-bus interface) connection convenience. Special packages according to user needs.



Physical map



Dimensions (unit: mm)

2、Applications

HVAC, dehumidifier, testing and inspection equipment, consumer goods, automotive, automatic control, data loggers, home appliances, humidity regulator, medical, weather stations, and other humidity measurement and control and so on.

3、Features

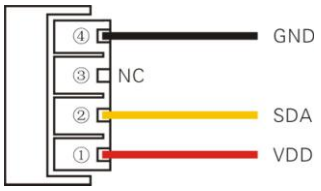
Ultra-low power, the transmission distance, fully automated calibration, the use of capacitive humidity sensor, completely interchangeable, standard digital single-bus output, excellent long-term stability, high accuracy temperature measurement devices.

4、The definition of single-bus interface

4.1 AM2302 Pin assignments

Table 1： AM2302 Pin assignments

Pin	Name	Description
①	VDD	Power (3.3V-5.5V)
②	SDA	Serial data, bidirectional port
③	NC	Empty
④	GND	Ground



PIC1： AM2302 Pin Assignment

4.2 Power supply pins（VDD GND）

AM2302 supply voltage range 3.3V － 5.5V, recommended supply voltage is 5V.

4.3 Serial data（SDA）

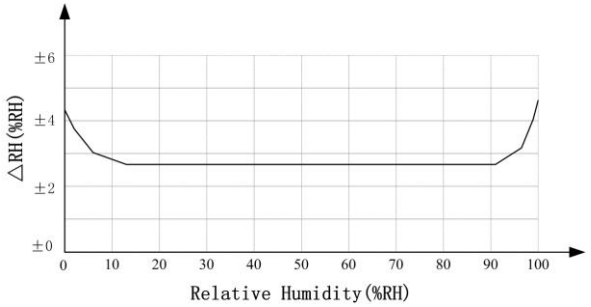
SDA pin is tri structure for reading, writing sensor data. Specific communication timing, see the detailed description of the communication protocol.

5、Sensor performance

5.1 Relative humidity

Table 2： AM2302 Relative humidity performance table

Parameter	Condition	min	typ	max	Unit
Resolution			0.1		%RH
Range		0		99.9	%RH
Accuracy [1]	25℃		± 2		%RH
Repeatability			± 0.3		%RH
Exchange		Completely interchangeable			
Response [2]	1/e(63%)		<5		S
Sluggish			<0.3		%RH
Drift [3]	Typical		<0.5		%RH/yr

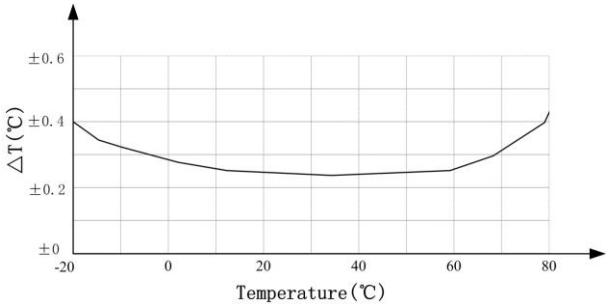


Pic2: At25℃ The error of relative humidity

5.2 Temperature

Table 3： AM2302 Relative temperature performance

Parameter	Condition	min	typ	max	Unit
Resolution			0.1		℃
			16		bit
Accuracy			± 0.5	± 1	℃
Range		-40		80	℃
Repeat			± 0.2		℃
Exchange		Completely interchangeable			
Response	1/e(63%)		<10		S
Drift			± 0.3		℃/yr



Pic3： The maximum temperature error

6、Electrical Characteristics

Electrical characteristics, such as energy consumption, high, low, input, output voltage, depending on the power supply. Table 4 details the electrical characteristics of the AM2302, if not identified, said supply voltage of 5V. To get the best results with the sensor, please design strictly in accordance with the conditions of design in Table 4.

Table 4： AM2302 DC Characteristics

Parameter	Condition	min	typ	max	Unit
Voltage		3.3	5	5.5	V
Power consumption [4]	Dormancy	10	15		μA
	Measuring		500		μA
	Average		300		μA
Low level output voltage	I_{OL}[5]	0		300	mV
High output voltage	Rp<25 kΩ	90%		100%	VDD
Low input voltage	Decline	0		30%	VDD
Input High Voltage	Rise	70%		100%	VDD
Rpu[6]	VDD = 5V VIN = VSS	30	45	60	kΩ
Output current	turn on		8		mA
	turn off	10	20		μA
Sampling period		2			S

[1] the accuracy of the factory inspection, the sensor 25℃ and 5V, the accuracy specification of test conditions, it does not include hysteresis and nonlinearity, and is only suitable for non-condensing environment.

[2] to achieve an order of 63% of the time required under the conditions of 25℃ and 1m / s airflow.

[3] in the volatile organic compounds, the values may be higher. See the manual application to store information.

[4] this value at VDD = 5.0V when the temperature is 25℃, 2S / time, under the conditions of the average.

[5] low output current.

[6] that the pull-up resistor.

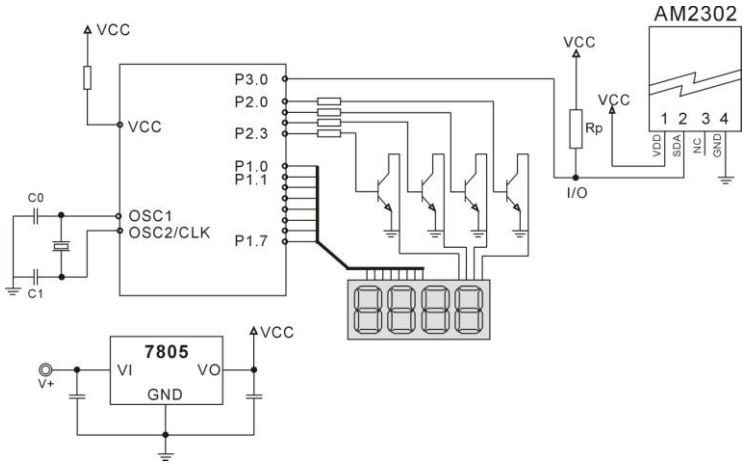
7、Single-bus communication（ONE-WIRE）

7.1 Typical circuits for single bus

Microprocessor and AM2302 connection typical application circuit is shown in Figure 4. Single bus communication mode, pull the SDA microprocessor I / O port is connected.

Special instructions of the single-bus communication：

1. Typical application circuit recommended in the short cable length of 30 meters on the 5.1K pull-up resistor pullup resistor according to the actual situation of lower than 30 m.
2. With 3.3V supply voltage, cable length shall not be greater than 100cm. Otherwise, the line voltage drop will lead to the sensor power supply, resulting in measurement error.
3. Read the sensor minimum time interval for the 2S; read interval is less than 2S, may cause the temperature and humidity are not allowed or communication is unsuccessful, etc..
4. Temperature and humidity values are each read out the results of the last measurement For real-time data that need continuous read twice, we recommend repeatedly to read sensors, and each read sensor interval is greater than 2 seconds to obtain accuratethe data.



Pic4： AM2302 Typical circuits for single bus

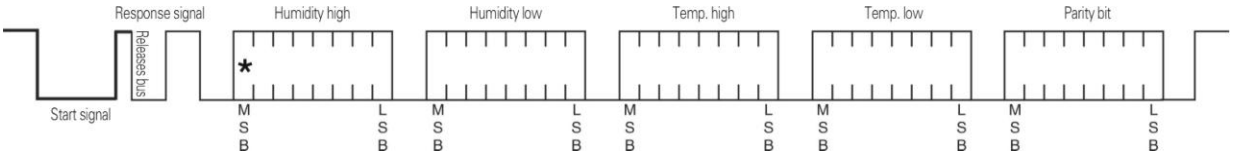
7.2、Single-bus communication protocol

◎Single bus Description

AM2302 device uses a simplified single-bus communication. Single bus that only one data line, data exchange system, controlled by the data line to complete. Equipment (microprocessor) through an open-drain or tri-state port connected to the data line to allow the device does not send data to release the bus, while other devices use the bus; single bus usually require an external about 5.1kΩ pull-up resistor, so when the bus is idle, its status is high. Because they are the master-slave structure, only the host calls the sensor, the sensor will answer, so the hosts to access the sensor must strictly follow the sequence of single bus, if there is a sequence of confusion, the sensor will not respond to the host.

◎Single bus to send data definition

SDA For communication and synchronization between the microprocessor and the AM2302, single-bus data format, a transmission of 40 data, the high first-out. Specific communication timing shown in Figure 5, the communication format is depicted in Table 5.



Pic5： AM2302 Single-bus communication protocol

Table 5： AM2302 Communication format specifier

Name	Single-bus format definition
Start signal	Microprocessor data bus (SDA) to bring down a period of time (at least 800μ s) [1] notify the sensor to prepare the data.
Response signal	Sensor data bus (SDA) is pulled down to 80μ s, followed by high-80μ s response to host the start signal.
Data format	Host the start signal is received, the sensor one-time string from the data bus (SDA) 40 data, the high first-out.
Humidity	Humidity resolution of 16Bit, the previous high; humidity sensor string value is 10 times the actual humidity values.
Temp.	Temperature resolution of 16Bit, the previous high; temperature sensor string value is 10 times the actual temperature value; The temperature is the highest bit (Bit15) is equal to 1 indicates a negative temperature, the temperature is the highest bit (Bit15) is equal to 0 indicates a positive temperature; Temperature in addition to the most significant bit (Bit14 ~ bit 0) temperature values.
Parity bit	Parity bit = humidity high + humidity low + temperature high + temperature low

◎**Single-bus data calculation example**

Example 1： 40 Data received：

0000 0010	1001 0010	0000 0001	0000 1101	1010 0010
High humidity 8	Low humidity 8	High temp. 8	Low temp. 8	Parity bit

Calculate：

0000 0010+1001 0010 +0000 0001+0000 1101= 1010 0010（Parity bit）

Received data is correct：

humidity： 0000 0010 1001 0010 = 0292H (Hexadecimal)= 2×256 + 9×16 + 2 = 658

=> Humidity = 65.8%RH

Temp.： 0000 0001 0000 1101 = 10DH(Hexadecimal) = 1×256 + 0×16 + 13 = 269

=> Temp.= 26.9℃

◎**Special Instructions：**

When the temperature is below 0 ℃, the highest position of the temperature data.

Example： -10.1 ℃ Expressed as 1 000 0000 0110 0101

Temp.： 0000 0000 0110 0101 = 0065H(Hexadecimal)= 6×16 +5 = 101

=> Temp. = -10.1℃

Example 2： 40 received data：

0000 0010	1001 0010	0000 0001	0000 1101	1011 0010
High humidity 8	Low humidity 8	High temp. 8	Low temp. 8	Parity bit

Calculate：

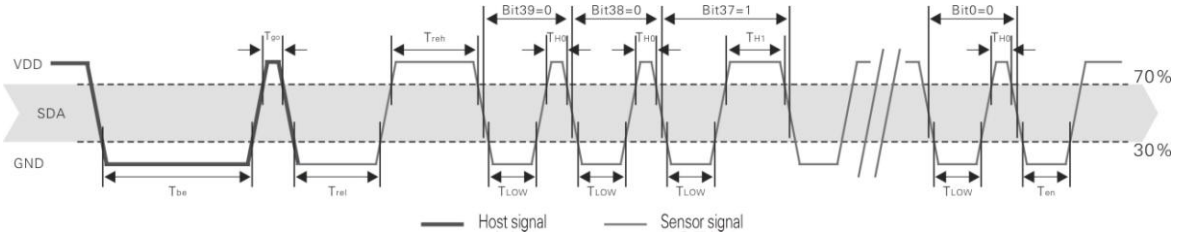
0000 0010+1001 0010 +0000 0001+0000 1101= 1010 0010 ≠ 1011 0010 （Validation error）

The received data is not correct, give up, to re-receive data.

7.3 Single-bus communication timing

User host (MCU) to send a start signal (data bus SDA line low for at least 800μ s) after AM2302 from Sleep mode conversion to high-speed mode.The host began to signal the end of the AM2302 send a response signal sent from the data bus SDA serial 40Bit's data, sends the byte high; data sent is followed by: Humidity high、Humidity low、Temperature high、Temperature low、Parity bit，Send data to the end of trigger information collection, the collection end of the sensor is automatically transferred to the sleep mode, the advent until the next communication.

Detailed timing signal characteristics in Table 6， Single-bus communication timing diagram Pic 6：



Pic 6： AM2302 Single-bus communication timing

Note: the temperature and humidity data read by the host from the AM2302 is always the last measured value, such as the two measurement interval is very long, continuous read twice to the second value of real-time temperature and humidity values, while two readtake minimum time interval be 2S.

Table 6: Single bus signal characteristics

Symbol	Parameter	min	typ	max	Unit
T_{be}	Host the start signal down time	0.8	1	20	mS
T_{go}	Bus master has released time	20	30	200	μS
T_{rel}	Response to low time	75	80	85	μS
T_{reh}	In response to high time	75	80	85	μS
T_{LOW}	Signal "0", "1" low time	48	50	55	μS
T_{H0}	Signal "0" high time	22	26	30	μS
T_{H1}	Signal "1" high time	68	70	75	μS
T_{en}	Sensor to release the bus time	45	50	55	μS

Note： To ensure the accurate communication of the sensor, the read signal, in strict accordance with the design parameters and timing in Table 6 and Figure 6.

7.4 Peripherals read step example

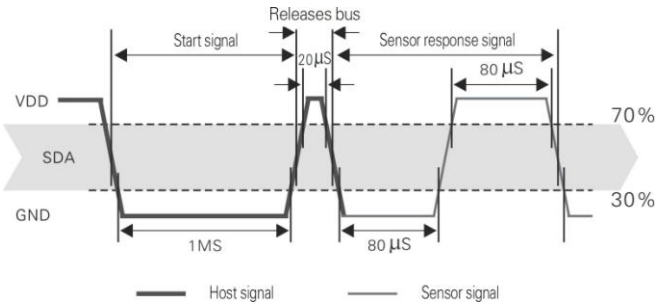
Communication between the host and the sensor can read data through the following three steps to complete.

Step 1

AM2302 have to wait for the power (on AM2302 power 2S crossed the unstable state, the device can not send any instructions to read during this period), the test environment temperature and humidity data, and record data, since the sensor into a sleep state automatically. AM2302 The SDA data line from the previous pull-up resistor pulled up is always high, the AM2302 the SDA pin is in input state, the time detection of external signal.

Step 2

Microprocessor I/O set to output, while output low, and low hold time can not be less than 800us, typical values are down 1MS, then the microprocessor I/O is set to input state, the release of the bus, due to the pull-up resistor, the microprocessor I/O AM2302 the SDA data line also will be high, the bus master has released the AM2302 send a response signal, that is, the output 80 microseconds low as the response signal, tightthen output high of 80 microseconds notice peripheral is ready to receive data signal transmission as shown to Pic7 :



Pic7： Single bus decomposition of the timing diagram

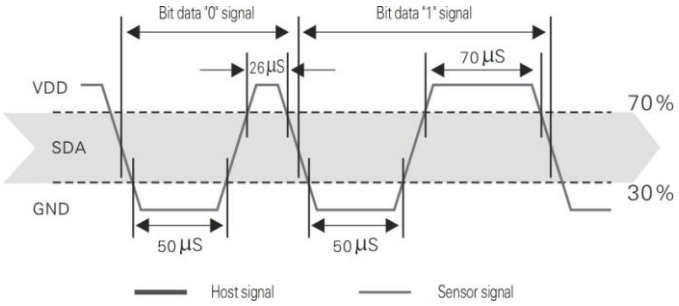
Step 3

AM2302 sending the response, followed by the data bus SDA continuous serial output 40 data, the microprocessor receives 40 data I/O level changes.

Bit data "0" format: 26-28 microseconds 50 microseconds low plus high;

Bit data "1" format: the high level of low plus, 50 microseconds to 70 microseconds;

Bit data "0" bit data "1" format signal shown to pic 8:

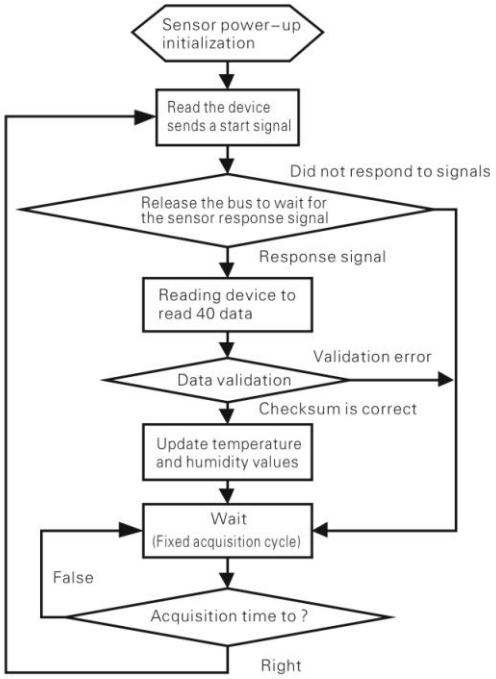


Pic 8： The single bus break down the timing diagram

AM2302 data bus SDA output 40 data continue to output the low 50 microseconds into the input state, followed by pull-up resistor goes high. AM2302 internal re-test environmental temperature and humidity data, and record the data, the end of the test records, the microcontroller automatically into hibernation. Microcontroller only after receipt of the start signal of the host wake-up sensor, into the working state.

7.5 Peripheral to read flow chart

AM2302 sensor read single bus flow chart diagram shown in Figure 9, we also provide the C51 read the code examples, customers need to download, please visit our website (www.aosong.com) related to downloadthis manual does not provide the code description.



Pic9：Single-bus to read the flow chart

8、Application of information

1. Work and storage conditions

Outside the sensor the proposed scope of work may lead to temporary drift of the signal up to 300% RH. Return to normal working conditions, sensor calibration status will slowly toward recovery. To speed up the recovery process may refer to "resume processing". Prolonged use of non-normal operating conditions, will accelerate the aging of the product.

Avoid placing the components on the long-term condensation and dry environment, as well as the following environment.

A, salt spray

B, acidic or oxidizing gases such as sulfur dioxide, hydrochloric acid

Recommended storage environment

Temperature: 10 ~ 40 ℃ Humidity: 60% RH or less

2. The impact of exposure to chemicals

The capacitive humidity sensor has a layer by chemical vapor interference, the proliferation of chemicals in the sensing layer may lead to drift and decreased sensitivity of the measured values. In a pure environment, contaminants will slowly be released. Resume processing as described below will accelerate this process. The high concentration of chemical pollution (such as ethanol) will lead to the complete damage of the sensitive layer of the sensor.

3. The temperature influence

Relative humidity of the gas to a large extent dependent on temperature. Therefore, in the measurement of humidity,

should be to ensure that the work of the humidity sensor at the same temperature. With the release of heat of electronic components share a printed circuit board, the installation should be as far as possible the sensor away from the electronic components and mounted below the heat source, while maintaining good ventilation of the enclosure. To reduce the thermal conductivity sensor and printed circuit board copper plating should be the smallest possible, and leaving a gap between the two.

4. Light impact

Prolonged exposure to sunlight or strong ultraviolet radiation, and degrade performance.

5. Resume processing

Placed under extreme working conditions or chemical vapor sensor, which allows it to return to the status of calibration by the following handler. Maintain two hours in the humidity conditions of 45℃ and <10% RH (dry); followed by 20-30℃ and> 70% RH humidity conditions to maintain more than five hours.

6. Wiring precautions

The quality of the signal wire will affect the quality of the voltage output, it is recommended to use high quality shielded cable.

7. Welding information

Manual welding, in the maximum temperature of 300℃ under the conditions of contact time shall be less than 3 seconds.

8. Product upgrades

Details, please the consultation Aosong electronics department.

9、The license agreement

Without the prior written permission of the copyright holder, shall not in any form or by any means, electronic or mechanical (including photocopying), copy any part of this manual, nor shall its contents be communicated to a third party. The contents are subject to change without notice.

The Company and third parties have ownership of the software, the user may use only signed a contract or software license.

10、Warnings and personal injury

This product is not applied to the safety or emergency stop devices, as well as the failure of the product may result in injury to any other application, unless a particular purpose or use authorized. Installation, handling, use or maintenance of the product refer to product data sheets and application notes. Failure to comply with this recommendation may result in death and serious personal injury. The Company will bear all damages resulting personal injury or death, and waive any claims that the resulting subsidiary company managers and employees and agents, distributors, etc. that may arise, including: a variety of costs, compensation costs, attorneys' fees, and so on.

11、Quality Assurance

The company and its direct purchaser of the product quality guarantee period of three months (from the date of delivery). Publishes the technical specifications of the product data sheet shall prevail. Within the warranty period, the product was confirmed that the quality is really defective, the company will provide free repair or replacement. The user must satisfy the following conditions:

① The product is found defective within 14 days written notice to the Company;

② The product shall be paid by mail back to the company;

③ The product should be within the warranty period.

The Company is only responsible for those used in the occasion of the technical condition of the product defective product. Without any guarantee, warranty or written statement of its products used in special applications. Company for its products applied to the reliability of the product or circuit does not make any commitment.

DS18B20

Programmable Resolution 1-Wire Digital Thermometer

General Description

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

Applications

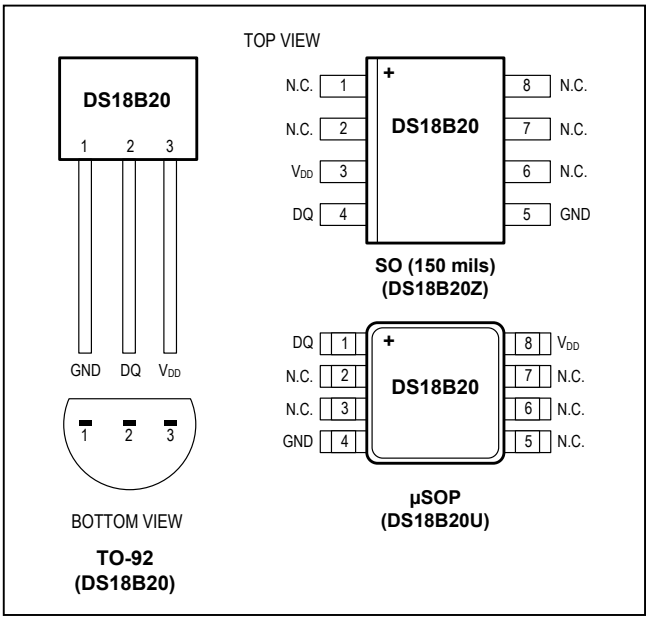
- Thermostatic Controls
- Industrial Systems
- Consumer Products
- Thermometers
- Thermally Sensitive Systems

Ordering Information **appears at end of data sheet.**

Benefits and Features

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Reduce Component Count with Integrated Temperature Sensor and EEPROM
  - Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
  - ±0.5°C Accuracy from -10°C to +85°C
  - Programmable Resolution from 9 Bits to 12 Bits
  - No External Components Required
- Parasitic Power Mode Requires Only 2 Pins for Operation (DQ and GND)
- Simplifies Distributed Temperature-Sensing Applications with Multidrop Capability
  - Each Device Has a Unique 64-Bit Serial Code Stored in On-Board ROM
- Flexible User-Definable Nonvolatile (NV) Alarm Settings with Alarm Search Command Identifies Devices with Temperatures Outside Programmed Limits
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages

Pin Configurations



19-7487; Rev 4; 1/15

maxim integrated™

Absolute Maximum Ratings

Voltage Range on Any Pin Relative to Ground ....-0.5V to +6.0V
Operating Temperature Range..... -55°C to +125°C
Storage Temperature Range..... -55°C to +125°C
Solder Temperature.....Refer to the IPC/JEDEC J-STD-020 Specification.

These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

DC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Supply Voltage	V_{DD}	Local power (Note 1)		+3.0		+5.5	V
Pullup Supply Voltage	V_{PU}	Parasite power	(Notes 1, 2)	+3.0		+5.5	V
		Local power		+3.0		V_{DD}	
Thermometer Error	t_{ERR}	-10°C to +85°C	(Note 3)			±0.5	°C
		-55°C to +125°C				±2	
Input Logic-Low	V_{IL}	(Notes 1, 4, 5)		-0.3		+0.8	V
Input Logic-High	V_{IH}	Local power	(Notes 1,6)	+2.2		The lower of 5.5 or V_{DD} + 0.3	V
		Parasite power		+3.0			
Sink Current	I_L	$V_{I/O}$ = 0.4V		4.0			mA
Standby Current	I_{DDS}	(Notes 7, 8)			750	1000	nA
Active Current	I_{DD}	V_{DD} = 5V (Note 9)			1	1.5	mA
DQ Input Current	I_{DQ}	(Note 10)			5		µA
Drift		(Note 11)			±0.2		°C

Note 1: All voltages are referenced to ground.
Note 2: The Pullup Supply Voltage specification assumes that the pullup device is ideal, and therefore the high level of the pullup is equal to V_{PU}. In order to meet the V_{IH} spec of the DS18B20, the actual supply rail for the strong pullup transistor must include margin for the voltage drop across the transistor when it is turned on; thus: $V_{PU\_ACTUAL} = V_{PU\_IDEAL} + V_{TRANSISTOR}$.
Note 3: See typical performance curve in Figure 1.
Note 4: Logic-low voltages are specified at a sink current of 4mA.
Note 5: To guarantee a presence pulse under low voltage parasite power conditions, V_{ILMAX} may have to be reduced to as low as 0.5V.
Note 6: Logic-high voltages are specified at a source current of 1mA.
Note 7: Standby current specified up to +70°C. Standby current typically is 3µA at +125°C.
Note 8: To minimize I_{DDS}, DQ should be within the following ranges: GND ≤ DQ ≤ GND + 0.3V or V_{DD} – 0.3V ≤ DQ ≤ V_{DD}.
Note 9: Active current refers to supply current during active temperature conversions or EEPROM writes.
Note 10: DQ line is high (“high-Z” state).
Note 11: Drift data is based on a 1000-hour stress test at +125°C with V_{DD} = 5.5V.

AC Electrical Characteristics–NV Memory

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
NV Write Cycle Time	t_{WR}			2	10	ms
EEPROM Writes	N_{EEWR}	-55°C to +55°C	50k			writes
EEPROM Data Retention	t_{EEDR}	-55°C to +55°C	10			years

AC Electrical Characteristics

(-55°C to +125°C; V_{DD} = 3.0V to 5.5V)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
Temperature Conversion Time	t_{CONV}	9-bit resolution	(Note 12)			93.75	ms
		10-bit resolution				187.5	
		11-bit resolution				375	
		12-bit resolution				750	
Time to Strong Pullup On	t_{SPON}	Start convert T command issued				10	µs
Time Slot	t_{SLOT}	(Note 12)		60		120	µs
Recovery Time	t_{REC}	(Note 12)		1			µs
Write 0 Low Time	t_{LOW0}	(Note 12)		60		120	µs
Write 1 Low Time	t_{LOW1}	(Note 12)		1		15	µs
Read Data Valid	t_{RDV}	(Note 12)				15	µs
Reset Time High	t_{RSTH}	(Note 12)		480			µs
Reset Time Low	t_{RSTL}	(Notes 12, 13)		480			µs
Presence-Detect High	t_{PDHIGH}	(Note 12)		15		60	µs
Presence-Detect Low	t_{PDLOW}	(Note 12)		60		240	µs
Capacitance	$C_{IN/OUT}$					25	pF

Note 12: See the timing diagrams in Figure 2.
Note 13: Under parasite power, if t_{RSTL} > 960µs, a power-on reset can occur.

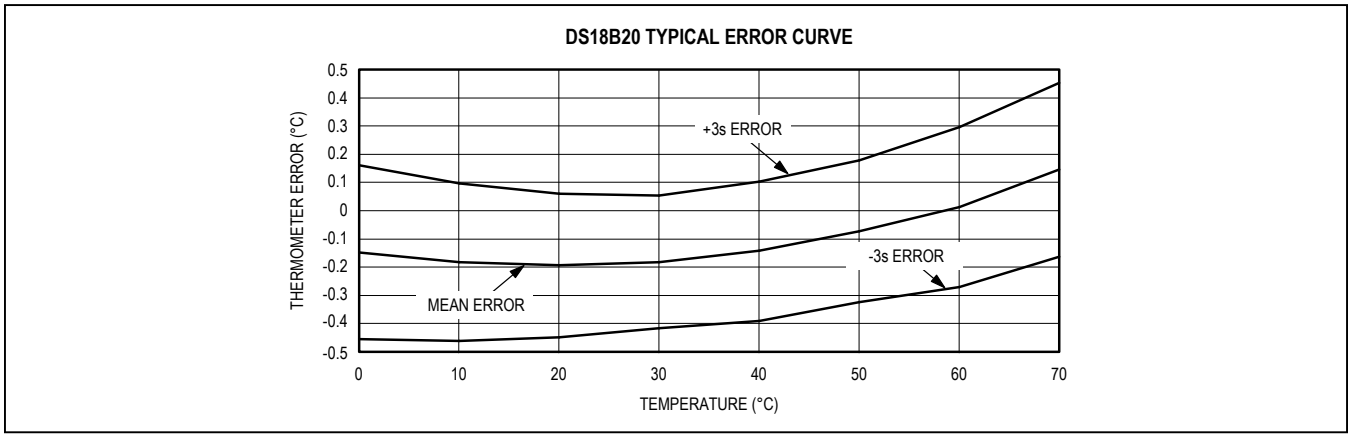


Figure 1. Typical Performance Curve

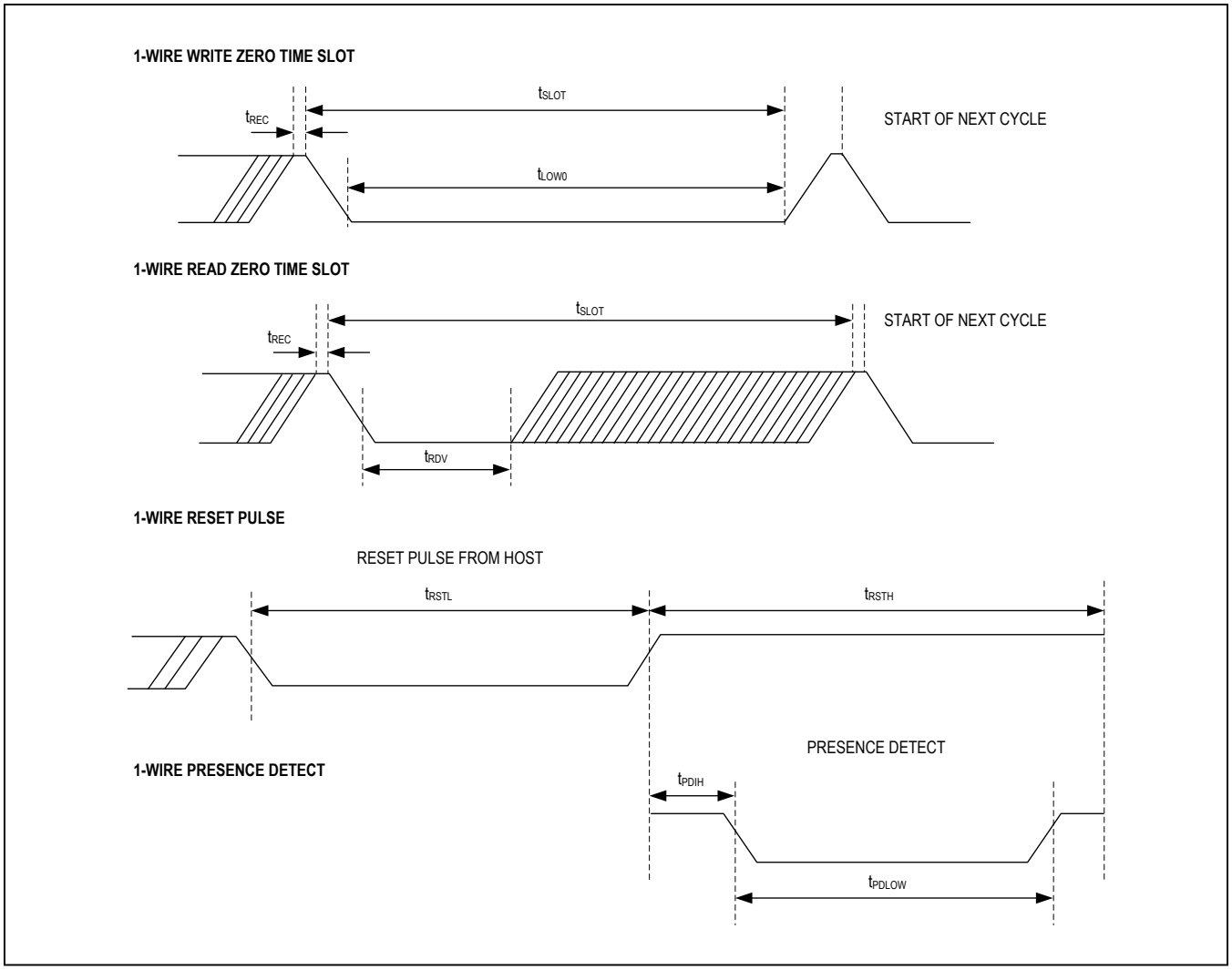


Figure 2. Timing Diagrams

Pin Description

PIN			NAME	FUNCTION
SO	µSOP	TO-92		
1, 2, 6, 7, 8	2, 3, 5, 6, 7	—	N.C.	No Connection
3	8	3	V_{DD}	Optional V_{DD}. V_{DD} must be grounded for operation in parasite power mode.
4	1	2	DQ	Data Input/Output. Open-drain 1-Wire interface pin. Also provides power to the device when used in parasite power mode (see the *Powering the DS18B20* section.)
5	4	1	GND	Ground

Overview

Figure 3 shows a block diagram of the DS18B20, and pin descriptions are given in the *Pin Description* table. The 64-bit ROM stores the device's unique serial code. The scratchpad memory contains the 2-byte temperature register that stores the digital output from the temperature sensor. In addition, the scratchpad provides access to the 1-byte upper and lower alarm trigger registers (T_H and T_L) and the 1-byte configuration register. The configuration register allows the user to set the resolution of the temperature-to-digital conversion to 9, 10, 11, or 12 bits. The T_H, T_L, and configuration registers are nonvolatile (EEPROM), so they will retain data when the device is powered down.

The DS18B20 uses Maxim's exclusive 1-Wire bus protocol that implements bus communication using one control signal. The control line requires a weak pullup resistor since all devices are linked to the bus via a 3-state or open-drain port (the DQ pin in the case of the DS18B20). In this bus system, the microprocessor (the master device) identifies and addresses devices on the bus using each device's unique 64-bit code. Because each device has a unique code, the number of devices that can be addressed on one bus is virtually unlimited. The 1-Wire bus protocol, including detailed explanations of the commands and "time slots," is covered in the *1-Wire Bus System* section.

Another feature of the DS18B20 is the ability to operate without an external power supply. Power is instead supplied through the 1-Wire pullup resistor through the DQ pin when the bus is high. The high bus signal also charges an internal capacitor (C_{PP}), which then supplies power to the device when the bus is low. This method of deriving power from the 1-Wire bus is referred to as "parasite power." As an alternative, the DS18B20 may also be powered by an external supply on V_{DD}.

Operation—Measuring Temperature

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The DS18B20 powers up in a low-power idle state. To initiate a temperature measurement and A-to-D conversion, the master must issue a Convert T [44h] command. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its idle state. If the DS18B20 is powered by an external supply, the master can issue "read time slots" (see the *1-Wire Bus System* section) after the Convert T command and the DS18B20 will respond by transmitting 0 while the temperature conversion is in progress and 1 when the conversion is done. If the DS18B20 is powered with parasite power, this notification technique cannot be used since the bus must be pulled high by a strong pullup during the entire temperature conversion. The bus requirements for parasite power are explained in detail in the *Powering the DS18B20* section.

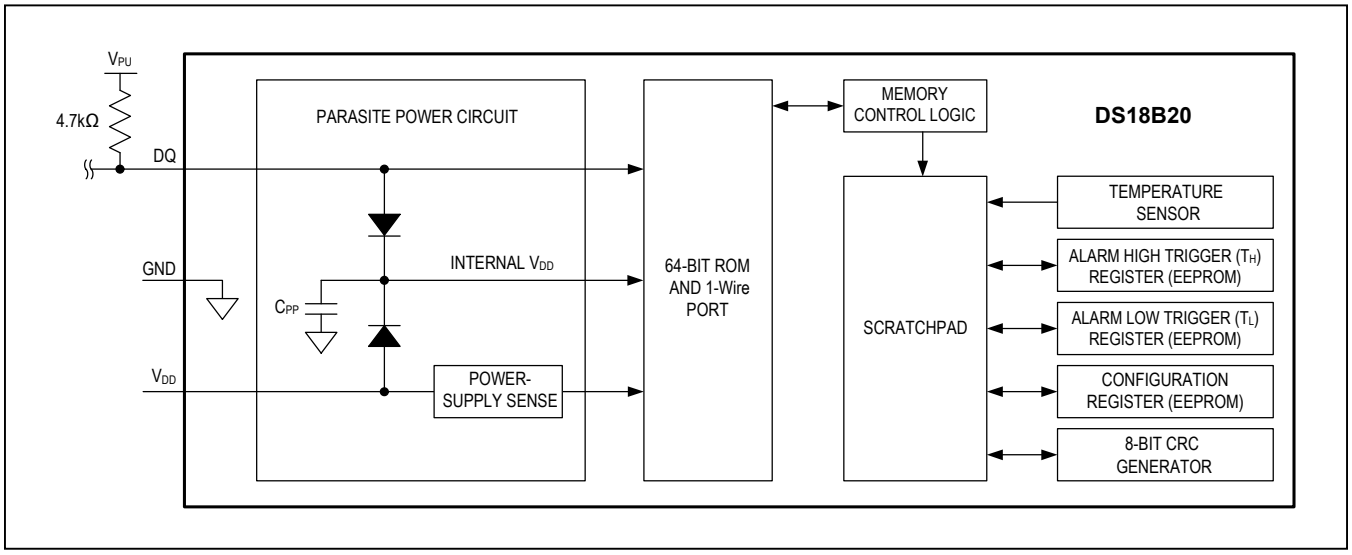


Figure 3. DS18B20 Block Diagram

The DS18B20 output temperature data is calibrated in degrees Celsius; for Fahrenheit applications, a lookup table or conversion routine must be used. The temperature data is stored as a 16-bit sign-extended two's complement number in the temperature register (see Figure 4). The sign bits (S) indicate if the temperature is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. If the DS18B20 is configured for 12-bit resolution, all bits in the temperature register will contain valid data. For 11-bit resolution, bit 0 is undefined. For 10-bit resolution, bits 1 and 0 are undefined, and for 9-bit resolution bits 2, 1, and 0 are undefined. Table 1 gives examples of digital output data and the corresponding temperature reading for 12-bit resolution conversions.

Operation—Alarm Signaling

After the DS18B20 performs a temperature conversion, the temperature value is compared to the user-defined two's complement alarm trigger values stored in the 1-byte T_H and T_L registers (see Figure 5). The sign bit (S) indicates if the value is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. The T_H and T_L registers are nonvolatile (EEPROM) so they will retain data when the device is powered down. T_H and T_L can be accessed through bytes 2 and 3 of the scratchpad as explained in the *Memory* section.

Only bits 11 through 4 of the temperature register are used in the T_H and T_L comparison since T_H and T_L are 8-bit registers. If the measured temperature is lower than

	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	**BIT 14**	**BIT 13**	**BIT 12**	**BIT 11**	**BIT 10**	**BIT 9**	**BIT 8**
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Figure 4. Temperature Register Format

Table 1. Temperature/Data Relationship

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	DIGITAL OUTPUT (HEX)
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+25.0625	0000 0001 1001 0001	0191h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-25.0625	1111 1110 0110 1111	FE6Fh
-55	1111 1100 1001 0000	FC90h

**The power-on reset value of the temperature register is +85°C.*

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figure 5. T_H and T_L Register Format

or equal to T_L or higher than or equal to T_H, an alarm condition exists and an alarm flag is set inside the DS18B20. This flag is updated after every temperature measurement; therefore, if the alarm condition goes away, the flag will be turned off after the next temperature conversion.

The master device can check the alarm flag status of all DS18B20s on the bus by issuing an Alarm Search [ECh] command. Any DS18B20s with a set alarm flag will respond to the command, so the master can determine exactly which DS18B20s have experienced an alarm condition. If an alarm condition exists and the T_H or T_L settings have changed, another temperature conversion should be done to validate the alarm condition.

Powering the DS18B20

The DS18B20 can be powered by an external supply on the V_{DD} pin, or it can operate in "parasite power" mode, which allows the DS18B20 to function without a local external supply. Parasite power is very useful for applications that require remote temperature sensing or that are very space constrained. Figure 3 shows the DS18B20's parasite-power control circuitry, which "steals" power from the 1-Wire bus via the DQ pin when the bus is high. The stolen charge powers the DS18B20 while the bus is high, and some of the charge is stored on the parasite power capacitor (C_{PP}) to provide power when the bus is low. When the DS18B20 is used in parasite power mode, the V_{DD} pin must be connected to ground.

In parasite power mode, the 1-Wire bus and CPP can provide sufficient current to the DS18B20 for most operations as long as the specified timing and voltage requirements are met (see the *DC Electrical Characteristics* and *AC Electrical Characteristics*). However, when the DS18B20 is performing temperature conversions or copying data from the scratchpad memory to EEPROM, the operating current can be as high as 1.5mA. This current can cause an unacceptable voltage drop across the weak 1-Wire pullup resistor and is more current than can be supplied by C_{PP}. To assure that the DS18B20 has sufficient supply current, it is necessary to provide a strong pullup on the 1-Wire bus whenever temperature conversions are taking place or data is being copied from the scratchpad to EEPROM. This can be accomplished by using a MOSFET to pull the bus directly to the rail as shown in Figure 6. The 1-Wire bus must be switched to the strong pullup within 10µs (max) after a Convert T [44h] or Copy Scratchpad [48h] command is issued, and the bus must be held high by the pullup for the duration of the conversion (t_{CONV}) or data transfer (t_{WR} = 10ms). No other activity can take place on the 1-Wire bus while the pullup is enabled.

The DS18B20 can also be powered by the conventional method of connecting an external power supply to the V_{DD} pin, as shown in Figure 7. The advantage of this method is that the MOSFET pullup is not required, and the 1-Wire bus is free to carry other traffic during the temperature conversion time.

The use of parasite power is not recommended for temperatures above +100°C since the DS18B20 may not be able to sustain communications due to the higher leakage currents that can exist at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that the DS18B20 be powered by an external power supply.

In some situations the bus master may not know whether the DS18B20s on the bus are parasite powered or powered by external supplies. The master needs this information to determine if the strong bus pullup should be used during temperature conversions. To get this information, the master can issue a Skip ROM [CCh] command followed by a Read Power Supply [B4h] command followed by a "read time slot". During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high. If the bus is pulled low, the master knows that it must supply the strong pullup on the 1-Wire bus during temperature conversions.

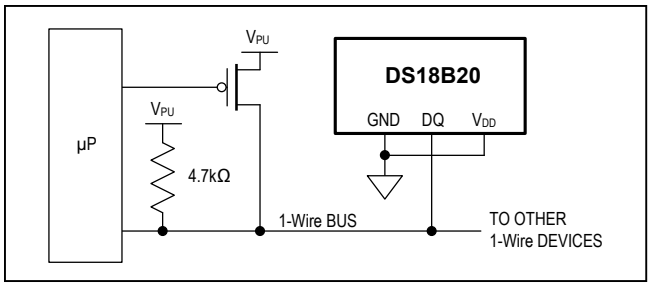


Figure 6. Supplying the Parasite-Powered DS18B20 During Temperature Conversions

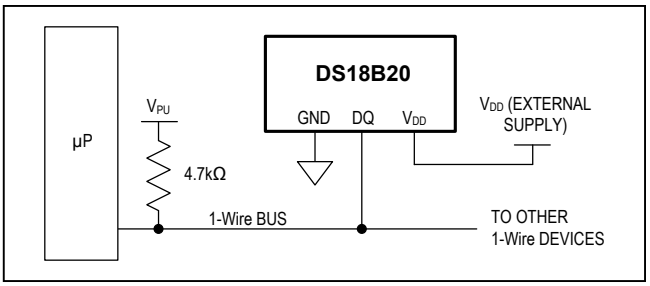


Figure 7. Powering the DS18B20 with an External Supply

64-BIT Lasered ROM code

Each DS18B20 contains a unique 64–bit code (see Figure 8) stored in ROM. The least significant 8 bits of the ROM code contain the DS18B20's 1-Wire family code: 28h. The next 48 bits contain a unique serial number. The most significant 8 bits contain a cyclic redundancy check (CRC) byte that is calculated from the first 56 bits of the ROM code. A detailed explanation of the CRC bits is provided in the *CRC Generation* section. The 64-bit ROM code and associated ROM function control logic allow the DS18B20 to operate as a 1-Wire device using the protocol detailed in the *1-Wire Bus System* section.

Memory

The DS18B20's memory is organized as shown in Figure 9. The memory consists of an SRAM scratchpad with nonvolatile EEPROM storage for the high and low alarm trigger registers (T_H and T_L) and configuration register. Note that if the DS18B20 alarm function is not used, the TH and TL registers can serve as general-purpose memory. All memory commands are described in detail in the *DS18B20 Function Commands* section.

Byte 0 and byte 1 of the scratchpad contain the LSB and the MSB of the temperature register, respectively. These bytes are read-only. Bytes 2 and 3 provide access to TH and TL registers. Byte 4 contains the configuration register data, which is explained in detail in the *Configuration Register* section. Bytes 5, 6, and 7 are reserved for internal use by the device and cannot be overwritten.

Byte 8 of the scratchpad is read-only and contains the CRC code for bytes 0 through 7 of the scratchpad. The DS18B20 generates this CRC using the method described in the *CRC Generation* section.

Data is written to bytes 2, 3, and 4 of the scratchpad using the Write Scratchpad [4Eh] command; the data must be transmitted to the DS18B20 starting with the least significant bit of byte 2. To verify data integrity, the scratchpad can be read (using the Read Scratchpad [BEh] command) after the data is written. When reading the scratchpad, data is transferred over the 1-Wire bus starting with the least significant bit of byte 0. To transfer the T_H, T_L and configuration data from the scratchpad to EEPROM, the master must issue the Copy Scratchpad [48h] command.

Data in the EEPROM registers is retained when the device is powered down; at power-up the EEPROM data is reloaded into the corresponding scratchpad locations. Data can also be reloaded from EEPROM to the scratchpad at any time using the Recall E^2 [B8h] command. The master can issue read time slots following the Recall E^2 command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done.

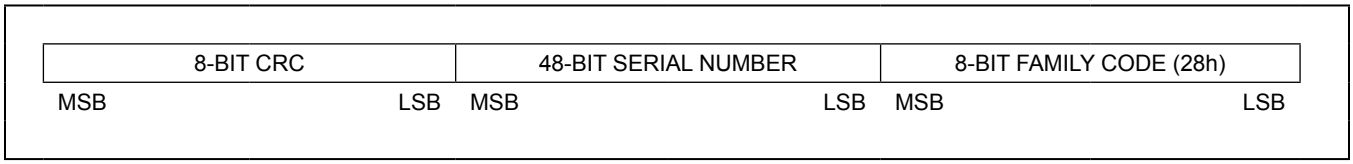


Figure 8. 64-Bit Lasered ROM Code

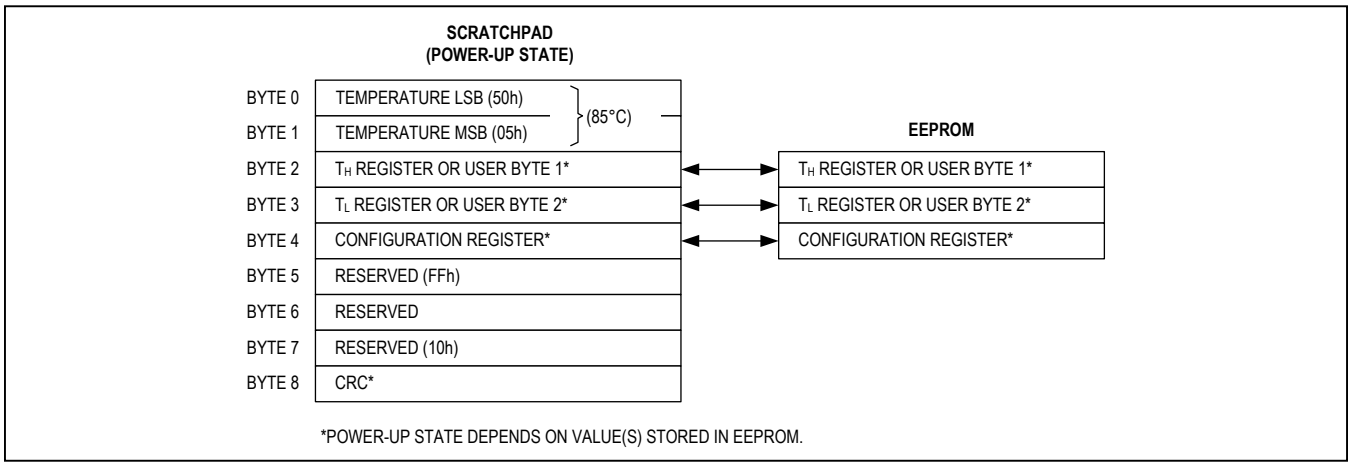


Figure 9. DS18B20 Memory Map

Configuration Register

Byte 4 of the scratchpad memory contains the configuration register, which is organized as illustrated in Figure 10. The user can set the conversion resolution of the DS18B20 using the R0 and R1 bits in this register as shown in Table 2. The power-up default of these bits is R0 = 1 and R1 = 1 (12-bit resolution). Note that there is a direct tradeoff between resolution and conversion time. Bit 7 and bits 0 to 4 in the configuration register are reserved for internal use by the device and cannot be overwritten.

CRC Generation

CRC bytes are provided as part of the DS18B20's 64-bit ROM code and in the 9th byte of the scratchpad memory. The ROM code CRC is calculated from the first 56 bits of the ROM code and is contained in the most significant byte of the ROM. The scratchpad CRC is calculated from the data stored in the scratchpad, and therefore it changes when the data in the scratchpad changes. The CRCs provide the bus master with a method of data validation when data is read from the DS18B20. To verify that data has been read correctly, the bus master must re-calculate the CRC from the received data and then compare this value to either the ROM code CRC (for ROM reads) or to the scratchpad CRC (for scratchpad reads). If the calculated CRC matches the read CRC, the data has been received error free. The comparison of CRC values and the decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS18B20 that prevents a command sequence from proceeding if the DS18B20 CRC (ROM or scratchpad) does not match the value generated by the bus master.

The equivalent polynomial function of the CRC (ROM or scratchpad) is:

$$CRC = X^8 + X^5 + X^4 + 1$$

The bus master can re-calculate the CRC and compare it to the CRC values from the DS18B20 using the polynomial generator shown in Figure 11. This circuit consists of a shift register and XOR gates, and the shift register bits are initialized to 0. Starting with the least significant bit of the ROM code or the least significant bit of byte 0 in the scratchpad, one bit at a time should shifted into the shift register. After shifting in the 56th bit from the ROM or the most significant bit of byte 7 from the scratchpad, the polynomial generator will contain the recalculated CRC. Next, the 8-bit ROM code or scratchpad CRC from the DS18B20 must be shifted into the circuit. At this point, if the re-calculated CRC was correct, the shift register will contain all 0s. Additional information about the Maxim 1-Wire cyclic redundancy check is available in *Application Note 27: Understanding and Using Cyclic Redundancy Checks with Maxim iButton Products*.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0	1	1	1	1	1

Figure 10. Configuration Register

Table 2. Thermometer Resolution Configuration

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME	
0	0	9	93.75ms	(t_{CONV}/8)
0	1	10	187.5ms	(t_{CONV}/4)
1	0	11	375ms	(t_{CONV}/2)
1	1	12	750ms	(t_{CONV})

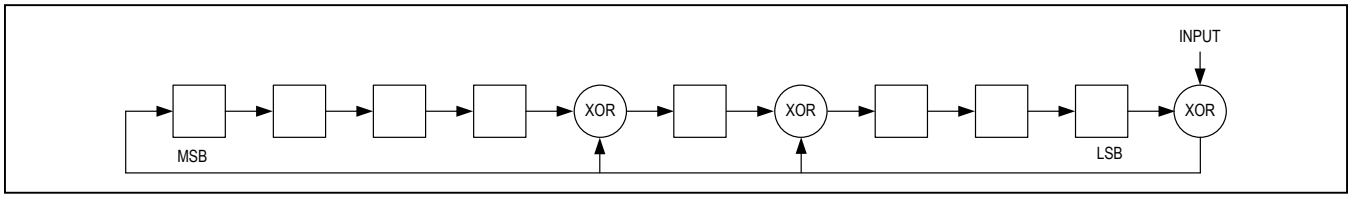


Figure 11. CRC Generator

1-Wire Bus System

The 1-Wire bus system uses a single bus master to control one or more slave devices. The DS18B20 is always a slave. When there is only one slave on the bus, the system is referred to as a “single-drop” system; the system is “multidrop” if there are multiple slaves on the bus.

All data and commands are transmitted least significant bit first over the 1-Wire bus.

The following discussion of the 1-Wire bus system is broken down into three topics: hardware configuration, transaction sequence, and 1-Wire signaling (signal types and timing).

Hardware Configuration

The 1-Wire bus has by definition only a single data line. Each device (master or slave) interfaces to the data line via an open-drain or 3-state port. This allows each device to “release” the data line when the device is not transmitting data so the bus is available for use by another device. The 1-Wire port of the DS18B20 (the DQ pin) is open drain with an internal circuit equivalent to that shown in Figure 12.

The 1-Wire bus requires an external pullup resistor of approximately 5kΩ; thus, the idle state for the 1-Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1-Wire bus is in the inactive (high) state during the recovery period. If the bus is held low for more than 480µs, all components on the bus will be reset.

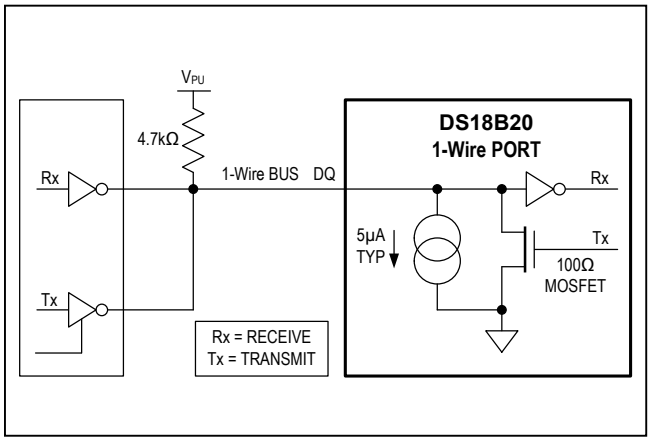


Figure 12. Hardware Configuration

Transaction Sequence

The transaction sequence for accessing the DS18B20 is as follows:

Step 1. Initialization

Step 2. ROM Command (followed by any required data exchange)

Step 3. DS18B20 Function Command (followed by any required data exchange)

It is very important to follow this sequence every time the DS18B20 is accessed, as the DS18B20 will not respond if any steps in the sequence are missing or out of order. Exceptions to this rule are the Search ROM [F0h] and Alarm Search [ECh] commands. After issuing either of these ROM commands, the master must return to Step 1 in the sequence.

Initialization

All transactions on the 1-Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s). The presence pulse lets the bus master know that slave devices (such as the DS18B20) are on the bus and are ready to operate. Timing for the reset and presence pulses is detailed in the *1-Wire Signaling* section.

ROM Commands

After the bus master has detected a presence pulse, it can issue a ROM command. These commands operate on the unique 64-bit ROM codes of each slave device and allow the master to single out a specific device if many are present on the 1-Wire bus. These commands also allow the master to determine how many and what types of devices are present on the bus or if any device has experienced an alarm condition. There are five ROM commands, and each command is 8 bits long. The master device must issue an appropriate ROM command before issuing a DS18B20 function command. A flowchart for operation of the ROM commands is shown in Figure 13.

Search Rom [F0h]

When a system is initially powered up, the master must identify the ROM codes of all slave devices on the bus, which allows the master to determine the number of slaves and their device types. The master learns the ROM codes through a process of elimination that requires the master to perform a Search ROM cycle (i.e., Search ROM command followed by data exchange) as many times as necessary to identify all of the slave devices.

If there is only one slave on the bus, the simpler Read ROM [33h] command can be used in place of the Search ROM process. For a detailed explanation of the Search ROM procedure, refer to *Application Note 937: Book of iButton® Standards*. After every Search ROM cycle, the bus master must return to Step 1 (Initialization) in the transaction sequence.

Read Rom [33h]

This command can only be used when there is one slave on the bus. It allows the bus master to read the slave's 64-bit ROM code without using the Search ROM procedure. If this command is used when there is more than one slave present on the bus, a data collision will occur when all the slaves attempt to respond at the same time.

Match Rom [55H]

The match ROM command followed by a 64-bit ROM code sequence allows the bus master to address a specific slave device on a multidrop or single-drop bus. Only the slave that exactly matches the 64-bit ROM code sequence will respond to the function command issued by the master; all other slaves on the bus will wait for a reset pulse.

Skip Rom [CCh]

The master can use this command to address all devices on the bus simultaneously without sending out any ROM code information. For example, the master can make all DS18B20s on the bus perform simultaneous temperature conversions by issuing a Skip ROM command followed by a Convert T [44h] command.

Note that the Read Scratchpad [BEh] command can follow the Skip ROM command only if there is a single slave device on the bus. In this case, time is saved by allowing the master to read from the slave without sending the device's 64-bit ROM code. A Skip ROM command followed by a Read Scratchpad command will cause a data collision on the bus if there is more than one slave since multiple devices will attempt to transmit data simultaneously.

Alarm Search [ECh]

The operation of this command is identical to the operation of the Search ROM command except that only slaves with a set alarm flag will respond. This command allows the master device to determine if any DS18B20s experienced an alarm condition during the most recent temperature conversion. After every Alarm Search cycle (i.e., Alarm Search command followed by data exchange), the bus master must return to Step 1 (Initialization) in the transaction sequence. See the *Operation—Alarm Signaling* section for an explanation of alarm flag operation.

DS18B20 Function Commands

After the bus master has used a ROM command to address the DS18B20 with which it wishes to communicate, the master can issue one of the DS18B20 function commands. These commands allow the master to write to and read from the DS18B20's scratchpad memory, initiate temperature conversions and determine the power supply mode. The DS18B20 function commands, which are described below, are summarized in Table 3 and illustrated by the flowchart in Figure 14.

Convert T [44h]

This command initiates a single temperature conversion. Following the conversion, the resulting thermal data is stored in the 2-byte temperature register in the scratchpad memory and the DS18B20 returns to its low-power idle state. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for the duration of the conversion (t_{CONV}) as described in the *Powering the DS18B20* section. If the DS18B20 is powered by an external supply, the master can issue read time slots after the Convert T command and the DS18B20 will respond by transmitting a 0 while the temperature conversion is in progress and a 1 when the conversion is done. In parasite power mode this notification technique cannot be used since the bus is pulled high by the strong pullup during the conversion.

Write Scratchpad [4Eh]

This command allows the master to write 3 bytes of data to the DS18B20's scratchpad. The first data byte is written into the T_H register (byte 2 of the scratchpad), the second byte is written into the T_L register (byte 3), and the third byte is written into the configuration register (byte 4). Data must be transmitted least significant bit first. All three bytes MUST be written before the master issues a reset, or the data may be corrupted.

Read Scratchpad [BEh]

This command allows the master to read the contents of the scratchpad. The data transfer starts with the least significant bit of byte 0 and continues through the scratchpad until the 9th byte (byte 8 – CRC) is read. The master may issue a reset to terminate reading at any time if only part of the scratchpad data is needed.

¡Button is a registered trademark of Maxim Integrated Products, Inc.

Copy Scratchpad [48h]

This command copies the contents of the scratchpad T_H, T_L and configuration registers (bytes 2, 3 and 4) to EEPROM. If the device is being used in parasite power mode, within 10µs (max) after this command is issued the master must enable a strong pullup on the 1-Wire bus for at least 10ms as described in the *Powering the DS18B20* section.

Recall E^2 [B8h]

This command recalls the alarm trigger values (T_H and T_L) and configuration data from EEPROM and places the data in bytes 2, 3, and 4, respectively, in the scratchpad memory. The master device can issue read time slots following the Recall E^2 command and the DS18B20 will indicate the status of the recall by transmitting 0 while the recall is in progress and 1 when the recall is done. The recall operation happens automatically at power-up, so valid data is available in the scratchpad as soon as power is applied to the device.

Read Power Supply [B4h]

The master device issues this command followed by a read time slot to determine if any DS18B20s on the bus are using parasite power. During the read time slot, parasite powered DS18B20s will pull the bus low, and externally powered DS18B20s will let the bus remain high. See the *Powering the DS18B20* section for usage information for this command.

Table 3. DS18B20 Function Command Set

COMMAND	DESCRIPTION	PROTOCOL	1-Wire BUS ACTIVITY AFTER COMMAND IS ISSUED	NOTES
TEMPERATURE CONVERSION COMMANDS				
Convert T	Initiates temperature conversion.	44h	DS18B20 transmits conversion status to master (not applicable for parasite-powered DS18B20s).	1
MEMORY COMMANDS				
Read Scratchpad	Reads the entire scratchpad including the CRC byte.	BEh	DS18B20 transmits up to 9 data bytes to master.	2
Write Scratchpad	Writes data into scratchpad bytes 2, 3, and 4 (T_H, T_L, and configuration registers).	4Eh	Master transmits 3 data bytes to DS18B20.	3
Copy Scratchpad	Copies T_H, T_L, and configuration register data from the scratchpad to EEPROM.	48h	None	1
Recall E^2	Recalls T_H, T_L, and configuration register data from EEPROM to the scratchpad.	B8h	DS18B20 transmits recall status to master.	
Read Power Supply	Signals DS18B20 power supply mode to the master.	B4h	DS18B20 transmits supply status to master.	

Note 1: For parasite-powered DS18B20s, the master must enable a strong pullup on the 1-Wire bus during temperature conversions and copies from the scratchpad to EEPROM. No other bus activity may take place during this time.
Note 2: The master can interrupt the transmission of data at any time by issuing a reset.
Note 3: All three bytes must be written before a reset is issued.

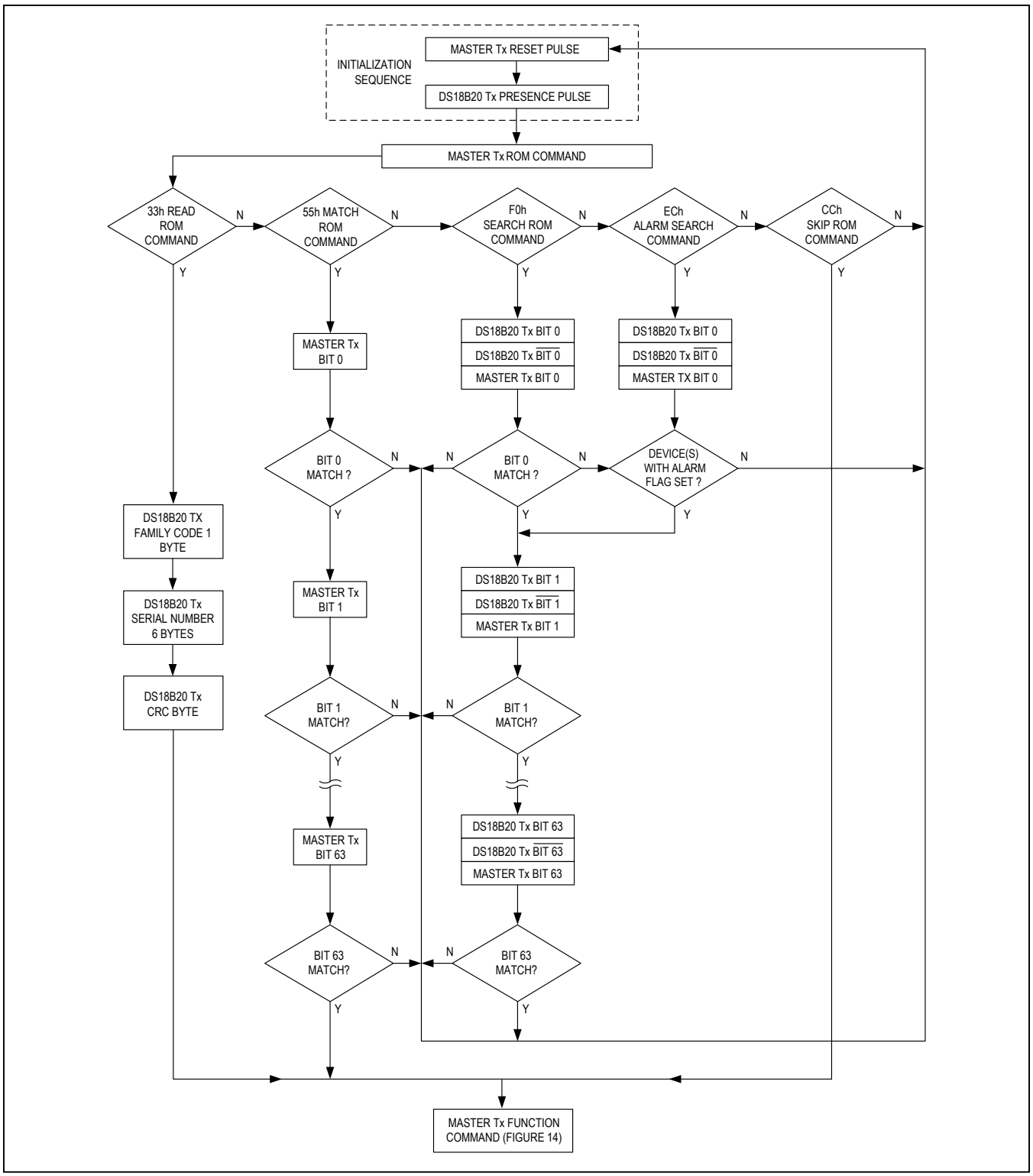


Figure 13. ROM Commands Flowchart

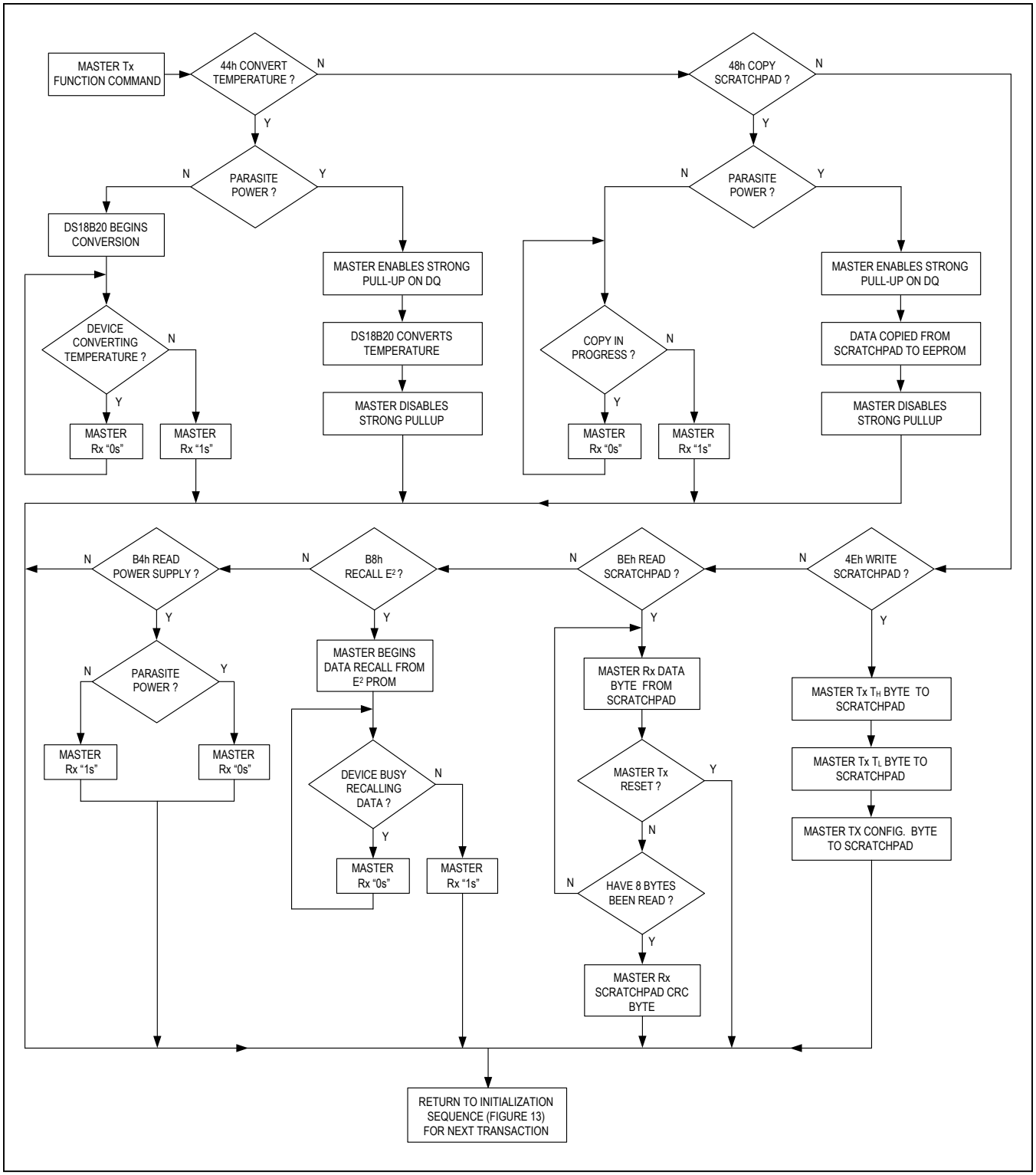


Figure 14. DS18B20 Function Commands Flowchart

1-Wire Signaling

The DS18B20 uses a strict 1-Wire communication protocol to ensure data integrity. Several signal types are defined by this protocol: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. The bus master initiates all these signals, with the exception of the presence pulse.

Initialization Procedure—Reset And Presence Pulses

All communication with the DS18B20 begins with an initialization sequence that consists of a reset pulse from the master followed by a presence pulse from the DS18B20. This is illustrated in Figure 15. When the DS18B20 sends the presence pulse in response to the reset, it is indicating to the master that it is on the bus and ready to operate.

During the initialization sequence the bus master transmits (T_X) the reset pulse by pulling the 1-Wire bus low for a minimum of 480µs. The bus master then releases the bus and goes into receive mode (R_X). When the bus is released, the 5kΩ pullup resistor pulls the 1-Wire bus high. When the DS18B20 detects this rising edge, it waits 15µs to 60µs and then transmits a presence pulse by pulling the 1-Wire bus low for 60µs to 240µs.

Read/Write Time Slots

The bus master writes data to the DS18B20 during write time slots and reads data from the DS18B20 during read time slots. One bit of data is transmitted over the 1-Wire bus per time slot.

Write Time Slots

There are two types of write time slots: “Write 1” time slots and “Write 0” time slots. The bus master uses a Write 1 time slot to write a logic 1 to the DS18B20 and a Write 0 time slot to write a logic 0 to the DS18B20. All write time slots must be a minimum of 60µs in duration with a minimum of a 1µs recovery time between individual write slots. Both types of write time slots are initiated by the master pulling the 1-Wire bus low (see Figure 14).

To generate a Write 1 time slot, after pulling the 1-Wire bus low, the bus master must release the 1-Wire bus within 15µs. When the bus is released, the 5kΩ pullup resistor will pull the bus high. To generate a Write 0 time slot, after pulling the 1-Wire bus low, the bus master must continue to hold the bus low for the duration of the time slot (at least 60µs).

The DS18B20 samples the 1-Wire bus during a window that lasts from 15µs to 60µs after the master initiates the write time slot. If the bus is high during the sampling window, a 1 is written to the DS18B20. If the line is low, a 0 is written to the DS18B20.

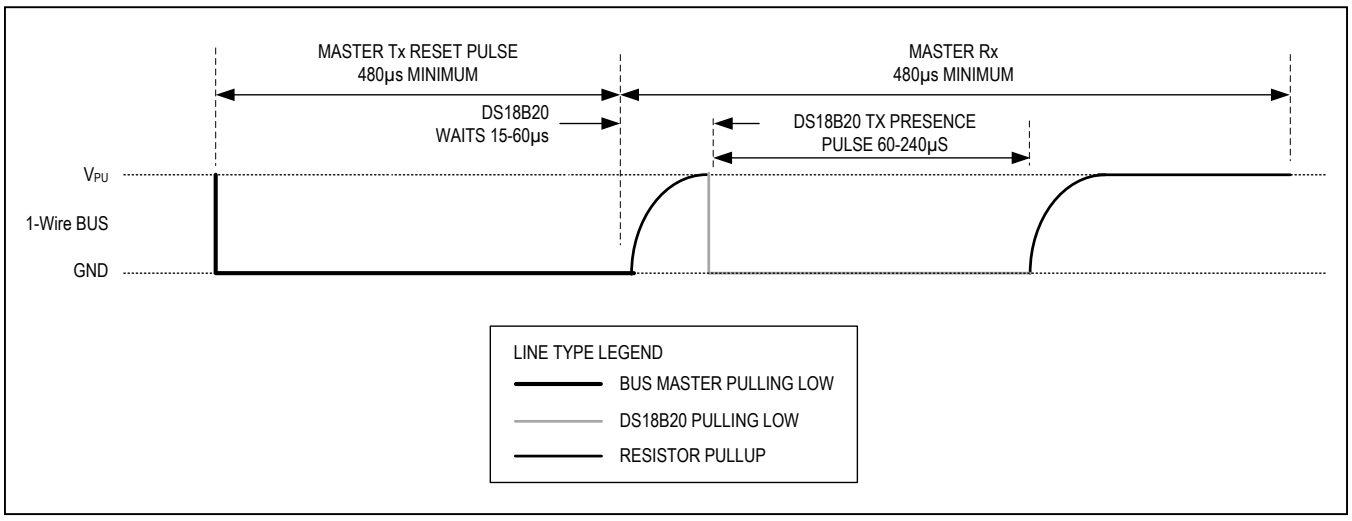


Figure 15. Initialization Timing

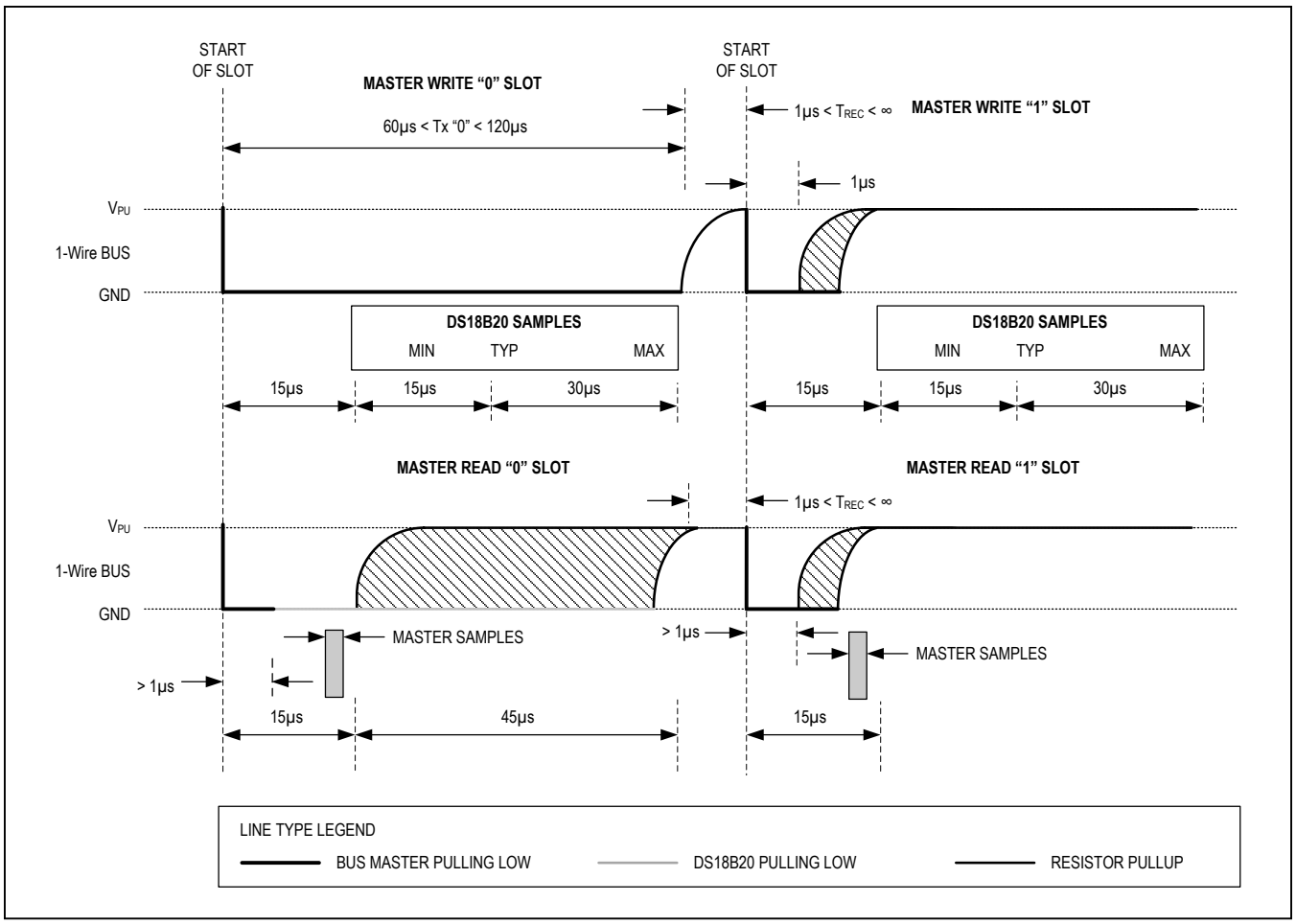


Figure 16. Read/Write Time Slot Timing Diagram

Read Time Slots

The DS18B20 can only transmit data to the master when the master issues read time slots. Therefore, the master must generate read time slots immediately after issuing a Read Scratchpad [BEh] or Read Power Supply [B4h] command, so that the DS18B20 can provide the requested data. In addition, the master can generate read time slots after issuing Convert T [44h] or Recall E^2 [B8h] commands to find out the status of the operation as explained in the *DS18B20 Function Commands* section.

All read time slots must be a minimum of 60µs in duration with a minimum of a 1µs recovery time between slots. A read time slot is initiated by the master device pulling the 1-Wire bus low for a minimum of 1µs and then releasing the bus (see Figure 16). After the master initiates the read time slot, the DS18B20 will begin transmitting a 1 or 0 on bus. The DS18B20 transmits a 1 by leaving the bus high and transmits a 0 by pulling the bus low. When transmitting a 0, the DS18B20 will release the bus by the end of the time slot, and the bus will be pulled back to its high idle state by the pullup resister. Output data from the DS18B20 is valid for 15µs after the falling edge that initiated the read time slot. Therefore, the master must release the bus and then sample the bus state within 15µs from the start of the slot.

Figure 17 illustrates that the sum of T_{INIT}, T_{RC}, and T_{SAMPLE} must be less than 15µs for a read time slot. Figure 18 shows that system timing margin is maximized by keeping T_{INIT} and T_{RC} as short as possible and by locating the master sample time during read time slots towards the end of the 15µs period.

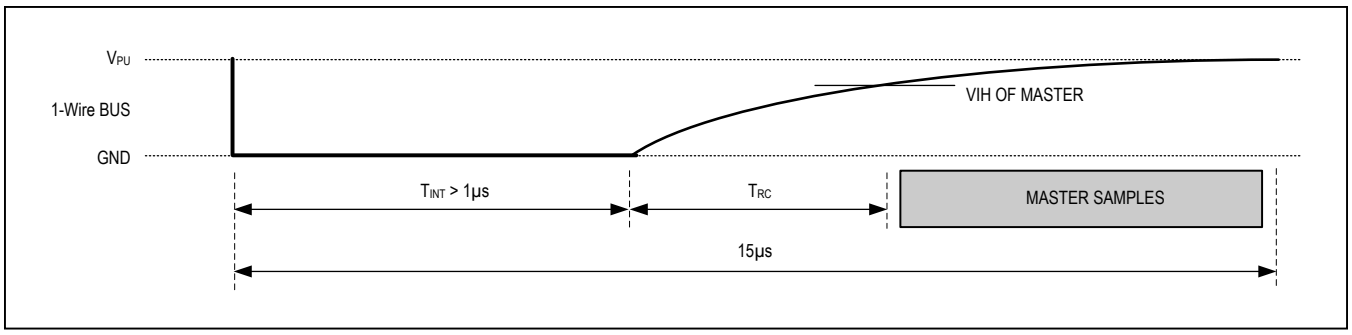


Figure 17. Detailed Master Read 1 Timing

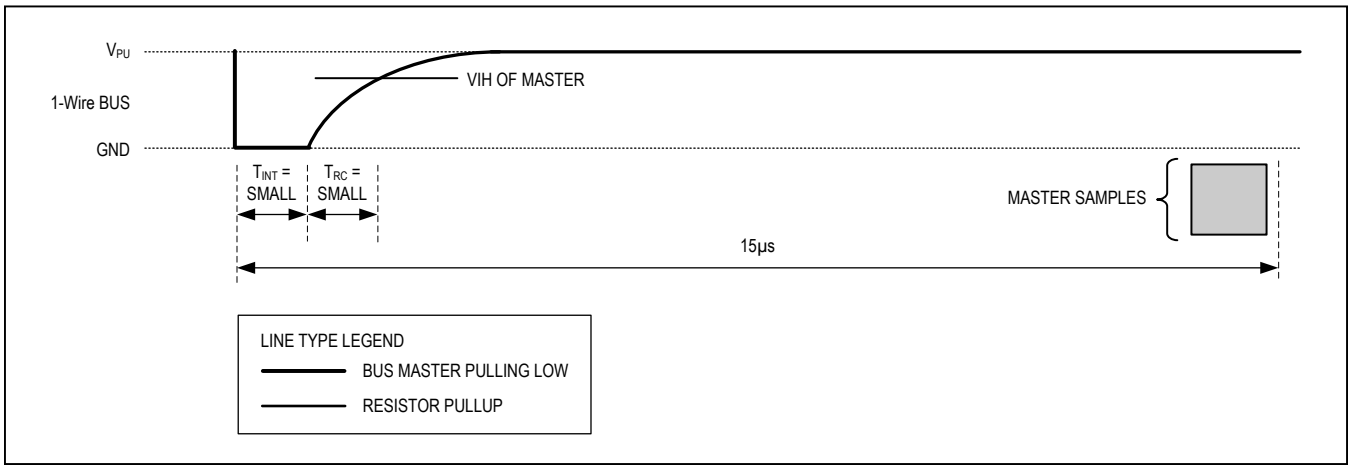


Figure 18. Recommended Master Read 1 Timing

Related Application Notes

The following application notes can be applied to the DS18B20 and are available at www.maximintegrated.com.

Application Note 27: Understanding and Using Cyclic Redundancy Checks with Maxim iButton Products

Application Note 122: Using Dallas' 1-Wire ICs in 1-Cell Li-Ion Battery Packs with Low-Side N-Channel Safety FETs Master

Application Note 126: 1-Wire Communication Through Software

Application Note 162: Interfacing the DS18x20/DS1822 1-Wire Temperature Sensor in a Microcontroller Environment

Application Note 208: Curve Fitting the Error of a Bandgap-Based Digital Temperature Sensor

Application Note 2420: 1-Wire Communication with a Microchip PICmicro Microcontroller

Application Note 3754: Single-Wire Serial Bus Carries Isolated Power and Data

Sample 1-Wire subroutines that can be used in conjunction with A*pplication Note 74: Reading and Writing iButtons via Serial Interfaces* can be downloaded from the Maxim website.

DS18B20 Operation Example 1

In this example there are multiple DS18B20s on the bus and they are using parasite power. The bus master initiates a temperature conversion in a specific DS18B20 and then reads its scratchpad and recalculates the CRC to verify the data.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20s respond with presence pulse.
Tx	55h	Master issues Match ROM command.
Tx	64-bit ROM code	Master sends DS18B20 ROM code.
Tx	44h	Master issues Convert T command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for the duration of the conversion (t_{CONV}).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20s respond with presence pulse.
Tx	55h	Master issues Match ROM command.
Tx	64-bit ROM code	Master sends DS18B20 ROM code.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.

DS18B20 Operation Example 2

In this example there is only one DS18B20 on the bus and it is using parasite power. The master writes to the TH, TL, and configuration registers in the DS18B20 scratchpad and then reads the scratchpad and recalculates the CRC to verify the data. The master then copies the scratchpad contents to EEPROM.

MASTER MODE	DATA (LSB FIRST)	COMMENTS
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	4Eh	Master issues Write Scratchpad command.
Tx	3 data bytes	Master sends three data bytes to scratchpad (T_H, T_L, and config).
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	BEh	Master issues Read Scratchpad command.
Rx	9 data bytes	Master reads entire scratchpad including CRC. The master then recalculates the CRC of the first eight data bytes from the scratchpad and compares the calculated CRC with the read CRC (byte 9). If they match, the master continues; if not, the read operation is repeated.
Tx	Reset	Master issues reset pulse.
Rx	Presence	DS18B20 responds with presence pulse.
Tx	CCh	Master issues Skip ROM command.
Tx	48h	Master issues Copy Scratchpad command.
Tx	DQ line held high by strong pullup	Master applies strong pullup to DQ for at least 10ms while copy operation is in progress.

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	TOP MARK
DS18B20	-55°C to +125°C	3 TO-92	18B20
DS18B20+	-55°C to +125°C	3 TO-92	18B20
DS18B20/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)	18B20
DS18B20-SL/T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20-SL+T&R	-55°C to +125°C	3 TO-92 (2000 Piece)*	18B20
DS18B20U	-55°C to +125°C	8 FSOP	18B20
DS18B20U+	-55°C to +125°C	8 FSOP	18B20
DS18B20U/T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20U+T&R	-55°C to +125°C	8 FSOP (3000 Piece)	18B20
DS18B20Z	-55°C to +125°C	8 SO	DS18B20
DS18B20Z+	-55°C to +125°C	8 SO	DS18B20
DS18B20Z/T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20
DS18B20Z+T&R	-55°C to +125°C	8 SO (2500 Piece)	DS18B20

+Denotes a lead-free package. A "+" will appear on the top mark of lead-free packages.

T&R = Tape and reel.

**TO-92 packages in tape and reel can be ordered with straight or formed leads. Choose "SL" for straight leads. Bulk TO-92 orders are straight leads only.*

Revision History

REVISION DATE	DESCRIPTION	PAGES CHANGED
030107	In the Absolute Maximum Ratings section, removed the reflow oven temperature value of +220°C. Reference to JEDEC specification for reflow remains.	19
101207	In the *Operation—Alarm Signaling* section, added “or equal to” in the description for a TH alarm condition	5
	In the *Memory* section, removed incorrect text describing memory.	7
	In the *Configuration Register* section, removed incorrect text describing configuration register.	8
042208	In the *Ordering Information* table, added TO-92 straight-lead packages and included a note that the TO-92 package in tape and reel can be ordered with either formed or straight leads.	2
1/15	Updated *Benefits and Features* section	1

MQ135 Semiconductor Sensor for Air Quality Control

Sensitive material of MQ135 gas sensor is SnO_2, which with lower conductivity in clean air. When the target combustible gas exist, The sensor's conductivity is more higher along with the gas concentration rising. Please use simple electrocircuit, Convert change of conductivity to correspond output signal of gas concentration.

MQ135 gas sensor has high sensitity to Ammonia, Sulfide and Benze steam, also sensitive to smoke and other harmful gases. It is with low cost and suitable for different application.

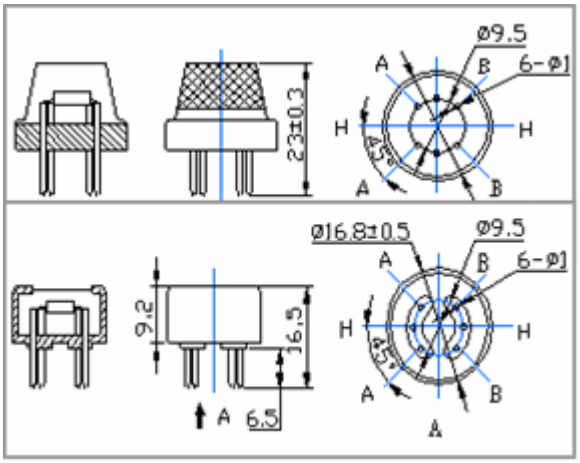
Character

* Good sensitivity to Harmful gases in wide range
* High sensitivity to Ammonia, Sulfide and Benze
* Long life and low cost
* Simple drive circuit

Application

* Domestic air pollution detector
* Industrial air pollution detector
* Portable air pollution detector

Configuration

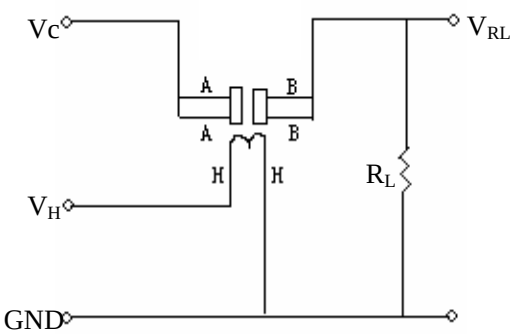


Technical Data

Model No.			MQ135
Sensor Type			Semiconductor
Standard Encapsulation			Bakelite (Black Bakelite)
Detection Gas			Ammonia, Sulfide, Benze steam
Concentration			10-10000ppm (Ammonia, Benze, Hydrogen)
Circuit	Loop Voltage	V_c	≤24V DC
	Heater Voltage	V_H	5.0V±0.2V AC or DC
	Load Resistance	R_L	Adjustable
Character	Heater Resistance	R_H	31Ω±3Ω（Room Tem.）
	Heater consumption	P_H	≤900mW
	Sensing Resistance	R_s	2KΩ-20KΩ(in 100ppm NH_3)
	Sensitivity	S	Rs(in air)/Rs(100ppm NH_3)≥5
	Slope	α	≤0.6 (R_{100ppm}/R_{50ppm} NH_3)
Condition	Tem. Humidity		20℃±2℃；65%±5%RH
	Standard test circuit		Vc:5.0V±0.1V; V_H: 5.0V±0.1V
	Preheat time		Over 48 hours

$Ps=Vc^2 \times Rs/(Rs+RL)^2$

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):

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

Sensitivity Characteristics

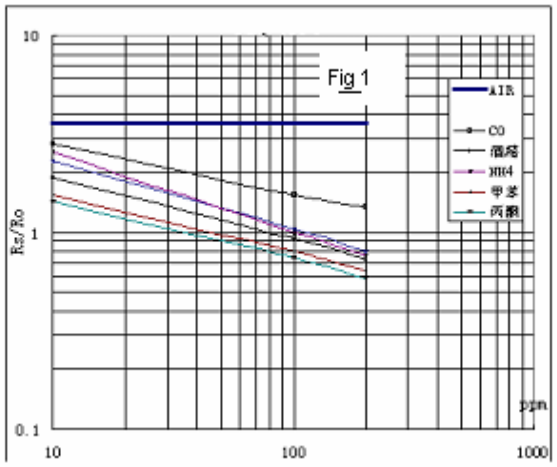


Fig.1 shows the typical sensitivity characteristics of the MQ135, ordinate means resistance ratio of the sensor (Rs/Ro), abscissa is concentration of gases. Rs means resistance in different gases, Ro means resistance of sensor in 100ppm Ammonia. All test are under standard test conditions.

Influence of Temperature/Humidity

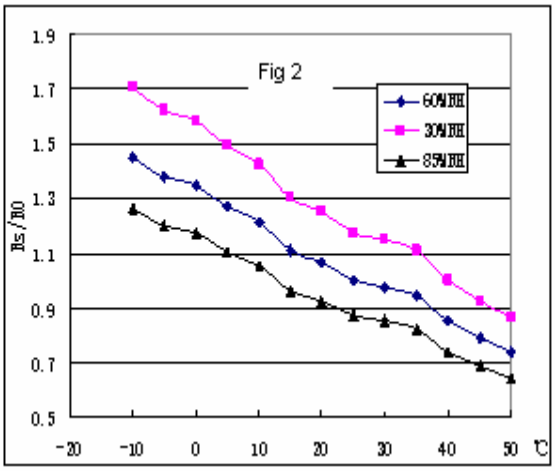
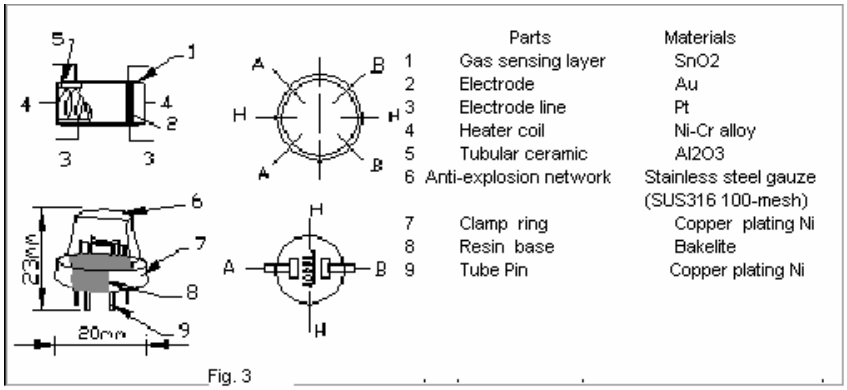


Fig.2 shows the typical temperature and humidity characteristics. Ordinate means resistance ratio of the sensor (Rs/Ro), Rs means resistance of sensor in 100ppm Ammonia under different tem. and humidity. Ro means resistance of the sensor in environment of 100ppm Ammonia, 20℃/65%RH

Structure and configuration



	Parts	Materials
1	Gas sensing layer	SnO2
2	Electrode	Au
3	Electrode line	Pt
4	Heater coil	Ni-Cr alloy
5	Tubular ceramic	Al2O3
6	Anti-explosion network	Stainless steel gauze (SUS316 100-mesh)
7	Clamp ring	Copper plating Ni
8	Resin base	Bakelite
9	Tube Pin	Copper plating Ni

Structure and configuration of MQ135 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-4 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

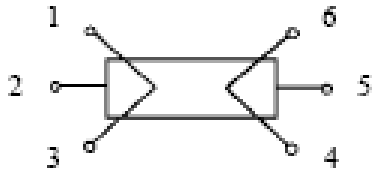
Do avoid icing on sensor'surface, otherwise sensor would lose sensitivity.

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

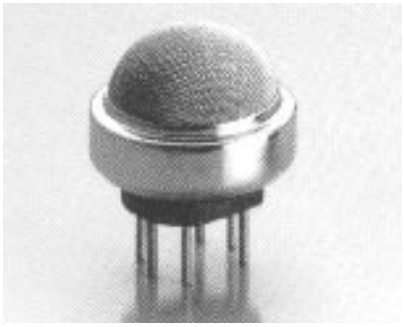
TGS 823 - for the detection of Organic Solvent Vapors

Features:

* High sensitivity to organic solvent vapors such as ethanol
* High stability and reliability over a long period
* Long life and low cost
* Ceramic base resistant to extreme environments

Applications:

* Breath alcohol detectors
* Gas leak detectors/alarms
* Solvent detectors for factories, dry cleaners, and semiconductor industries



The sensing element of Figaro gas sensors is a tin dioxide (SnO_2) semiconductor which has low conductivity in clean air. In the presence of a detectable gas, the sensor's conductivity increases depending on the gas concentration in the air. A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration.

The **TGS 823** has high sensitivity to the vapors of organic solvents as well as other volatile vapors. It also has sensitivity to a variety of combustible gases such as carbon monoxide, making it a good general purpose sensor. Its ceramic base can withstand severe environments as high as 200°C.

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 is indicated as sensor resistance ratio (Rs/Ro) which is defined as follows:

Rs = Sensor resistance of displayed gases at various concentrations

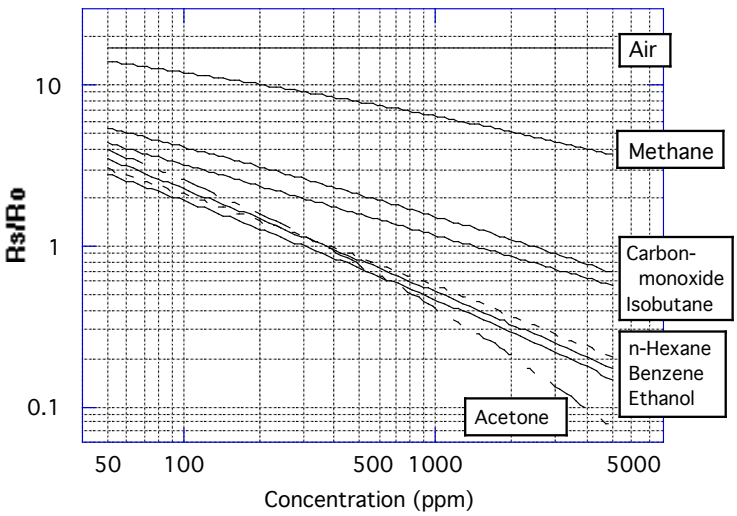
Ro = Sensor resistance in 300ppm ethanol

The figure below represents typical temperature and humidity dependency characteristics. Again, the Y-axis is indicated as sensor resistance ratio (Rs/Ro), defined as follows:

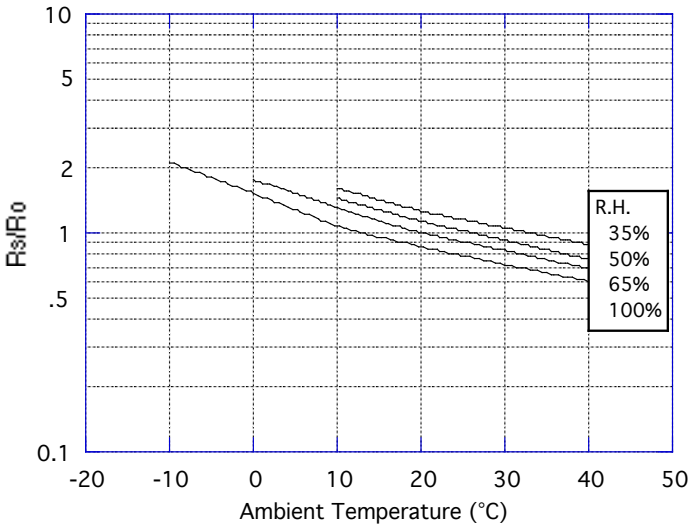
Rs = Sensor resistance at 300ppm of ethanol at various temperatures/humidities

Ro = Sensor resistance at 300ppm of ethanol at 20°C and 65% R.H.

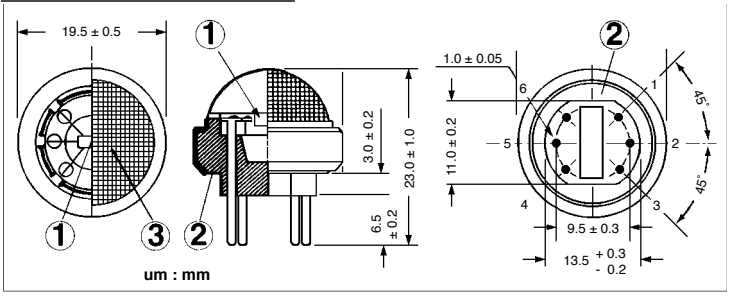
Sensitivity Characteristics:



Temperature/Humidity Dependency:



Structure and Dimensions:



(1) Sensing Element:
SnO_2 is sintered to form a thick film on the surface of an alumina ceramic tube which contains an internal heater.

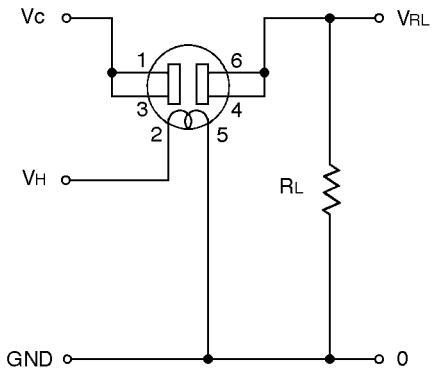
(2) Sensor Base:
Alumina ceramic

(3) Flame Arrestor:
100 mesh SUS 316 double gauze

Pin Connection and Basic Measuring Circuit:

The numbers shown around the sensor symbol in the circuit diagram at the right correspond with the pin numbers shown in the sensor's structure drawing (above). When the sensor is connected as shown in the basic circuit, output across the Load Resistor (V_{RL}) increases as the sensor's resistance (Rs) decreases, depending on gas concentration.

Basic Measuring Circuit:



Standard Circuit Conditions:

Item	Symbol	Rated Values	Remarks
Heater Voltage	V_H	5.0±0.2V	AC or DC
Circuit Voltage	V_C	Max. 24V	DC only Ps≤15mW
Load Resistance	R_L	Variable	0.45kΩ min.

Electrical Characteristics:

Item	Symbol	Condition	Specification
Sensor Resistance	Rs	Ethanol at 300ppm/air	1kΩ ~ 10kΩ
Change Ratio of Sensor Resistance	Rs/Ro	$\frac{Rs(\text{Ethanol at 300ppm/air})}{Rs(\text{Ethanol at 50ppm/air})}$	0.40 ± 0.10
Heater Resistance	R_H	Room temperature	38.0 ± 3.0Ω
Heater Power Consumption	P_H	V_H=5.0V	660mW (typical)

Standard Test Conditions:

TGS 823 complies with the above electrical characteristics when the sensor is tested in standard conditions as specified below:

Test Gas Conditions: 20°±2°C, 65±5%R.H.
Circuit Conditions: V_C = 10.0±0.1V (AC or DC),
V_H = 5.0±0.05V (AC or DC),
R_L = 10.0kΩ±1%
Preheating period before testing: More than 7 days

Sensor Resistance (Rs) is calculated by the following formula:

$$Rs = \left(\frac{V_C}{V_{RL}} - 1\right) \times R_L$$

Power dissipation across sensor electrodes (Ps) is calculated by the following formula:

$$Ps = \frac{V_C^2 \times Rs}{(Rs + R_L)^2}$$

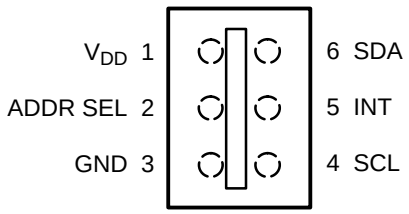
FIGARO USA, INC.
121 S. Wilke Rd. Suite 300
Arlington Heights, IL 60005
Phone: (847)-832-1701
Fax: (847)-832-1705
email: figarousa@figarosensor.com

For information on warranty, please refer to Standard Terms and Conditions of Sale of Figaro USA Inc.

REV: 09/02

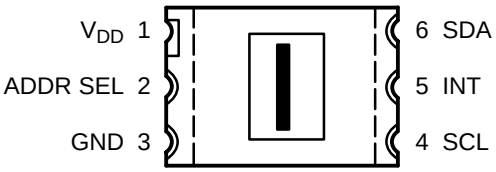
- **Approximates Human Eye Response to Control Display Backlight and Keyboard Illumination**
- **Precisely Measures Illuminance in Diverse Lighting Conditions Providing Exposure Control in Cameras**
- **Programmable Interrupt Function with User-Defined Upper and Lower Threshold Settings**
- **16-Bit Digital Output with SMBus (TSL2560) or I²C (TSL2561) Fast-Mode at 400 KHz**
- **Programmable Analog Gain and Integration Time Supporting 1,000,000-to-1 Dynamic Range**
- **Available in Ultra-Small 1.25 mm × 1.75 mm Chipscale Package**
- **Automatically Rejects 50/60-Hz Lighting Ripple**
- **Low Active Power (0.75 mW Typical) with Power Down Mode**
- **RoHS Compliant**

PACKAGE CS
6-LEAD CHIPSCALE
(TOP VIEW)



Package Drawings are Not to Scale

PACKAGE T
6-LEAD TMB
(TOP VIEW)



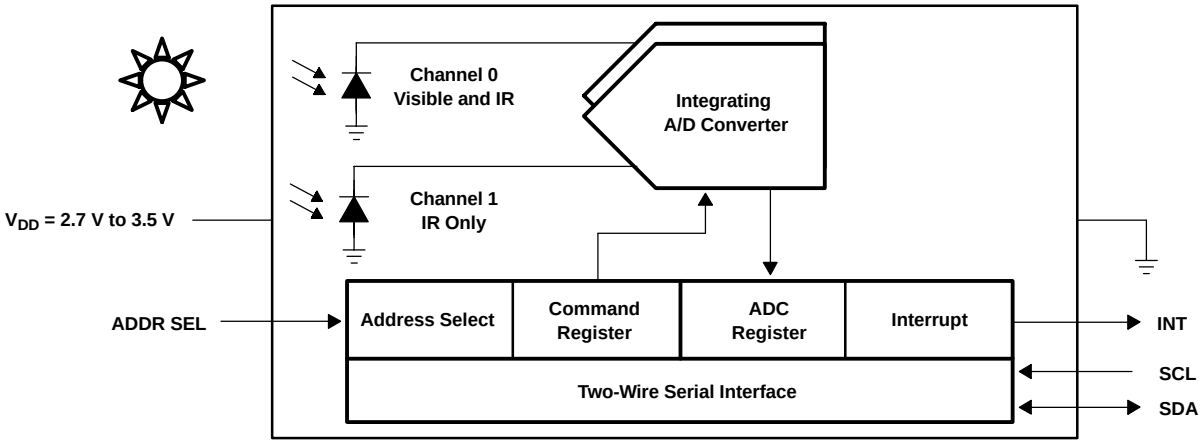
Description

The TSL2560 and TSL2561 are light-to-digital converters that transform light intensity to a digital signal output capable of direct I²C (TSL2561) or SMBus (TSL2560) interface. Each device combines one broadband photodiode (visible plus infrared) and one infrared-responding photodiode on a single CMOS integrated circuit capable of providing a near-photopic response over an effective 20-bit dynamic range (16-bit resolution). Two integrating ADCs convert the photodiode currents to a digital output that represents the irradiance measured on each channel. This digital output can be input to a microprocessor where illuminance (ambient light level) in lux is derived using an empirical formula to approximate the human eye response. The TSL2560 device permits an SMB-Alert style interrupt, and the TSL2561 device supports a traditional level style interrupt that remains asserted until the firmware clears it.

While useful for general purpose light sensing applications, the TSL2560/61 devices are designed particularly for display panels (LCD, OLED, etc.) with the purpose of extending battery life and providing optimum viewing in diverse lighting conditions. Display panel backlighting, which can account for up to 30 to 40 percent of total platform power, can be automatically managed. Both devices are also ideal for controlling keyboard illumination based upon ambient lighting conditions. Illuminance information can further be used to manage exposure control in digital cameras. The TSL2560/61 devices are ideal in notebook/tablet PCs, LCD monitors, flat-panel televisions, cell phones, and digital cameras. In addition, other applications include street light control, security lighting, sunlight harvesting, machine vision, and automotive instrumentation clusters.

Texas Advanced Optoelectronic Solutions Inc.
800 Jupiter Road, Suite 205 • Plano, TX 75074 • (972) 673-0759
www.taosinc.com

Functional Block Diagram



Detailed Description

The TSL2560 and TSL2561 are second-generation ambient light sensor devices. Each contains two integrating analog-to-digital converters (ADC) that integrate currents from two photodiodes. Integration of both channels occurs simultaneously. Upon completion of the conversion cycle, the conversion result is transferred to the Channel 0 and Channel 1 data registers, respectively. The transfers are double-buffered to ensure that the integrity of the data is maintained. After the transfer, the device automatically begins the next integration cycle.

Communication to the device is accomplished through a standard, two-wire SMBus or I^2C serial bus. Consequently, the TSL256x device can be easily connected to a microcontroller or embedded controller. No external circuitry is required for signal conditioning, thereby saving PCB real estate as well. Since the output of the TSL256x device is digital, the output is effectively immune to noise when compared to an analog signal.

The TSL256x devices also support an interrupt feature that simplifies and improves system efficiency by eliminating the need to poll a sensor for a light intensity value. The primary purpose of the interrupt function is to detect a meaningful change in light intensity. The concept of a *meaningful change* can be defined by the user both in terms of light intensity and time, or persistence, of that change in intensity. The TSL256x devices have the ability to define a threshold above and below the current light level. An interrupt is generated when the value of a conversion exceeds either of these limits.

Terminal Functions

TERMINAL NAME	CS PKG NO.	T PKG NO.	TYPE	DESCRIPTION
ADDR SEL	2	2	I	SMBus device select — three-state
GND	3	3		Power supply ground. All voltages are referenced to GND.
INT	5	5	O	Level or SMB Alert interrupt.
SCL	4	4	I	SMBus serial clock input terminal — clock signal for SMBus serial data.
SDA	6	6	I/O	SMBus serial data I/O terminal — serial data I/O for SMBus.
V_{DD}	1	1		Supply voltage.

Available Options

DEVICE	INTERFACE	PACKAGE − LEADS	PACKAGE DESIGNATOR	ORDERING NUMBER
TSL2560	SMBus	Chipscale	CS	TSL2560CS
TSL2560	SMBus	TMB-6	T	TSL2560T
TSL2561	I^2C	Chipscale	CS	TSL2561CS
TSL2561	I^2C	TMB-6	T	TSL2561T

Absolute Maximum Ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{DD} (see Note 1) . . . . . 3.8 V
Digital output voltage range, V_O . . . . . −0.5 V to 3.8 V
Digital output current, I_O . . . . . −1 mA to 20 mA
Storage temperature range, T_{stg} . . . . . −40°C to 85°C
ESD tolerance, human body model . . . . . 2000 V

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltages are with respect to GND.

Recommended Operating Conditions

	MIN	NOM	MAX	UNIT
Supply voltage, V_{DD}	2.7	3	3.6	V
Operating free-air temperature, T_A	−30		70	°C
SCL, SDA input low voltage, V_{IL}	−0.5		0.8	V
SCL, SDA input high voltage, V_{IH}	2.1		3.6	V

Electrical Characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
I_{DD}	Supply current	Active		0.24	0.6	mA
		Power down		3.2	15	µA
V_{OL}	INT, SDA output low voltage	3 mA sink current	0		0.4	V
		6 mA sink current	0		0.6	V
I_{LEAK}	Leakage current		−5		5	µA

Operating Characteristics, High Gain (16×), V_{DD} = 3 V, T_A = 25°C, (unless otherwise noted) (see Notes 2, 3, 4, 5)

PARAMETER		TEST CONDITIONS	CHANNEL	TSL2560T, TSL2561T			TSL2560CS, TSL2561CS			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
f_{osc}	Oscillator frequency			690	735	780	690	735	780	kHz
	Dark ADC count value	E_e = 0, T_{int} = 402 ms	Ch0	0		4	0		4	counts
			Ch1	0		4	0		4	
	Full scale ADC count value (Note 6)	T_{int} > 178 ms	Ch0			65535			65535	counts
			Ch1			65535			65535	
		T_{int} = 101 ms	Ch0			37177			37177	
			Ch1			37177			37177	
		T_{int} = 13.7 ms	Ch0			5047			5047	
			Ch1			5047			5047	
	ADC count value	λ_p = 640 nm, T_{int} = 101 ms, E_e = 36.3 µW/cm^2	Ch0	750	1000	1250				counts
			Ch1		200					
		λ_p = 940 nm, T_{int} = 101 ms, E_e = 119 µW/cm^2	Ch0	700	1000	1300				
			Ch1		820					
		λ_p = 640 nm, T_{int} = 101 ms, E_e = 41 µW/cm^2	Ch0				750	1000	1250	counts
			Ch1					190		
		λ_p = 940 nm, T_{int} = 101 ms, E_e = 135 µW/cm^2	Ch0				700	1000	1300	
			Ch1					850		
	ADC count value ratio: Ch1/Ch0	λ_p = 640 nm, T_{int} = 101 ms		0.15	0.20	0.25	0.14	0.19	0.24	
		λ_p = 940 nm, T_{int} = 101 ms		0.69	0.82	0.95	0.70	0.85	1	
R_e	Irradiance responsivity	λ_p = 640 nm, T_{int} = 101 ms	Ch0		27.5			24.4		counts/(µW/cm^2)
			Ch1		5.5			4.6		
		λ_p = 940 nm, T_{int} = 101 ms	Ch0		8.4			7.4		
			Ch1		6.9			6.3		
R_v	Illuminance responsivity	Fluorescent light source: T_{int} = 402 ms	Ch0		36			35		counts/lux
			Ch1		4			3.8		
		Incandescent light source: T_{int} = 402 ms	Ch0		144			129		
			Ch1		72			67		
	ADC count value ratio: Ch1/Ch0	Fluorescent light source: T_{int} = 402 ms			0.11			0.11		
		Incandescent light source: T_{int} = 402 ms			0.5			0.52		
R_v	Illuminance responsivity, low gain mode (Note 7)	Fluorescent light source: T_{int} = 402 ms	Ch0		2.3			2.2		counts/lux
			Ch1		0.25			0.24		
		Incandescent light source: T_{int} = 402 ms	Ch0		9			8.1		
			Ch1		4.5			4.2		
	(Sensor Lux) / (actual Lux), high gain mode (Note 8)	Fluorescent light source: T_{int} = 402 ms		0.65	1	1.35	0.65	1	1.35	
		Incandescent light source: T_{int} = 402 ms		0.60	1	1.40	0.60	1	1.40	

NOTES:
2. Optical measurements are made using small-angle incident radiation from light-emitting diode optical sources. Visible 640 nm LEDs and infrared 940 nm LEDs are used for final product testing for compatibility with high-volume production.
3. The 640 nm irradiance E_e is supplied by an AlInGaP light-emitting diode with the following characteristics: peak wavelength λp = 640 nm and spectral halfwidth $\Delta\lambda$½ = 17 nm.
4. The 940 nm irradiance E_e is supplied by a GaAs light-emitting diode with the following characteristics: peak wavelength λp = 940 nm and spectral halfwidth $\Delta\lambda$½ = 40 nm.
5. Integration time T_{int}, is dependent on internal oscillator frequency (f_{osc}) and on the integration field value in the timing register as described in the *Register Set* section. For nominal f_{osc} = 735 kHz, nominal T_{int} = (number of clock cycles)/f_{osc}.
   Field value 00: $T_{int} = (11 \times 918)/f_{osc}$ = 13.7 ms
   Field value 01: $T_{int} = (81 \times 918)/f_{osc}$ = 101 ms
   Field value 10: $T_{int} = (322 \times 918)/f_{osc}$ = 402 ms
   Scaling between integration times vary proportionally as follows: 11/322 = 0.034 (field value 00), 81/322 = 0.252 (field value 01), and 322/322 = 1 (field value 10).
6. Full scale ADC count value is limited by the fact that there is a maximum of one count per two oscillator frequency periods and also by a 2-count offset.
   Full scale ADC count value = ((number of clock cycles)/2 − 2)
   Field value 00: Full scale ADC count value = $((11 \times 918)/2 - 2)$ = 5047
   Field value 01: Full scale ADC count value = $((81 \times 918)/2 - 2)$ = 37177
   Field value 10: Full scale ADC count value = 65535, which is limited by 16 bit register. This full scale ADC count value is reached for 131074 clock cycles, which occurs for T_{int} = 178 ms for nominal f_{osc} = 735 kHz.
7. Low gain mode has 16× lower gain than high gain mode: (1/16 = 0.0625).
8. The sensor Lux is calculated using the empirical formula shown on p. 22 of this data sheet based on measured Ch0 and Ch1 ADC count values for the light source specified. Actual Lux is obtained with a commercial luxmeter. The range of the (sensor Lux) / (actual Lux) ratio is estimated based on the variation of the 640 nm and 940 nm optical parameters. Devices are not 100% tested with fluorescent or incandescent light sources.

AC Electrical Characteristics, V_{DD} = 3 V, T_A = 25°C (unless otherwise noted)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{(CONV)}$	Conversion time		12	100	400	ms
$f_{(SCL)}$	Clock frequency				400	kHz
$t_{(BUF)}$	Bus free time between start and stop condition		4.7			µs
$t_{(HDSTA)}$	Hold time after (repeated) start condition. After this period, the first clock is generated.		4			µs
$t_{(SUSTA)}$	Repeated start condition setup time		4.7			µs
$t_{(SUSTO)}$	Stop condition setup time		4			µs
$t_{(HDDAT)}$	Data hold time		300			ns
$t_{(SUDAT)}$	Data setup time		250			ns
$t_{(LOW)}$	SCL clock low period		4.7			µs
$t_{(HIGH)}$	SCL clock high period		4			µs
$t_{(TIMEOUT)}$	Detect clock/data low timeout		25		35	ms
t_F	Clock/data fall time				300	ns
t_R	Clock/data rise time				1000	ns
C_i	Input pin capacitance				10	pF

PARAMETER MEASUREMENT INFORMATION

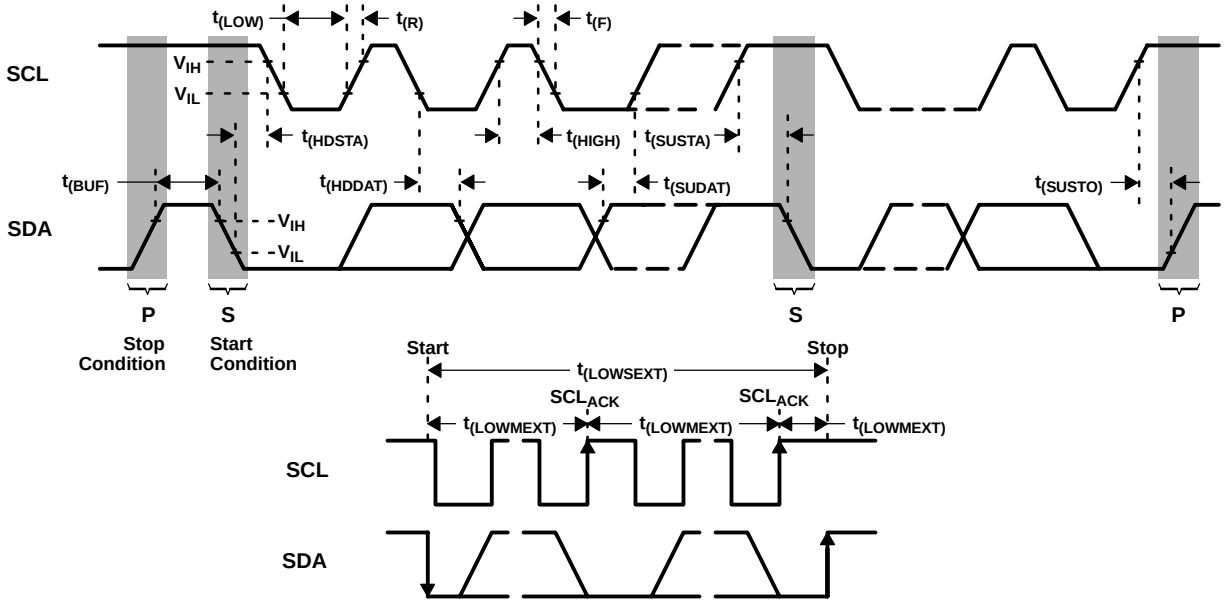


Figure 1. Timing Diagrams

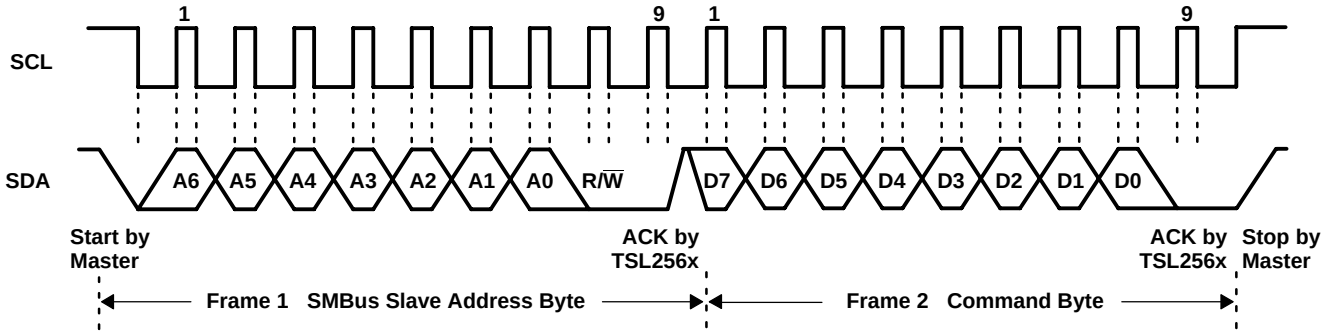


Figure 2. Example Timing Diagram for SMBus Send Byte Format

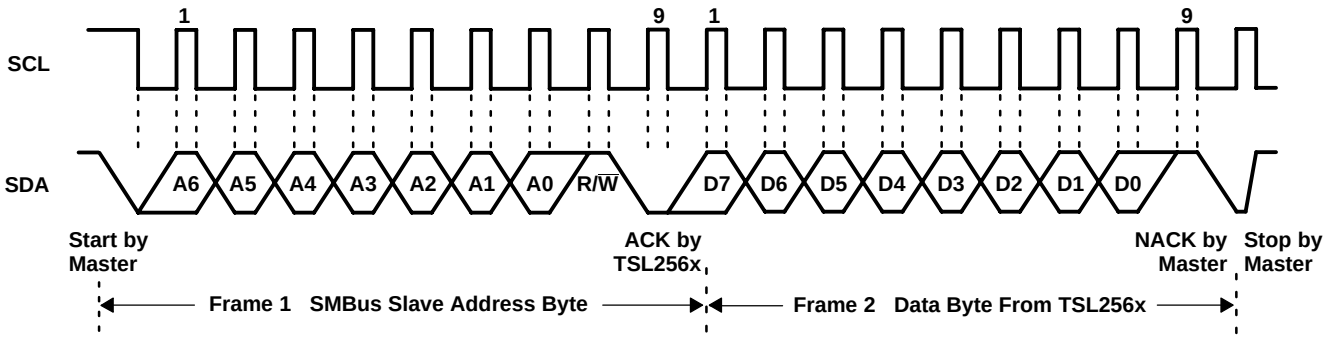


Figure 3. Example Timing Diagram for SMBus Receive Byte Format

TYPICAL CHARACTERISTICS

SPECTRAL RESPONSIVITY

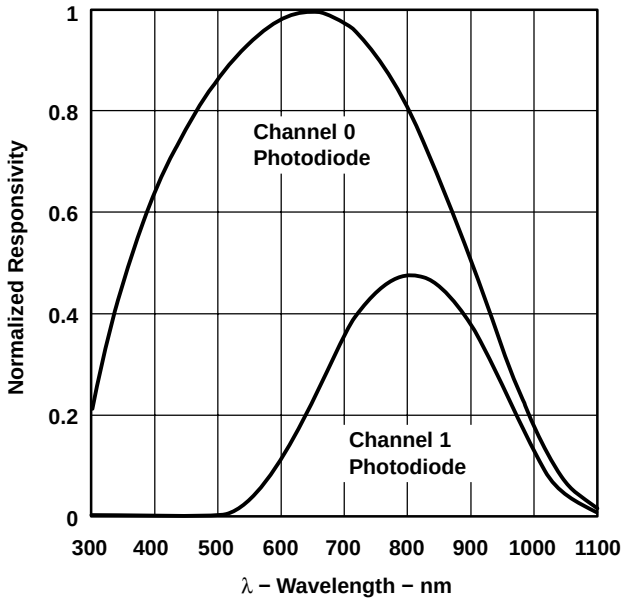


Figure 4

NORMALIZED RESPONSIVITY vs. ANGULAR DISPLACEMENT — CS PACKAGE

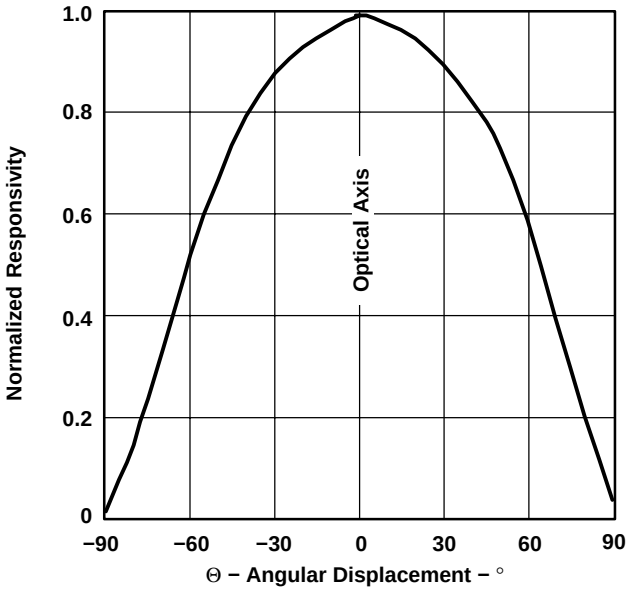


Figure 5

NORMALIZED RESPONSIVITY vs. ANGULAR DISPLACEMENT — TMB PACKAGE

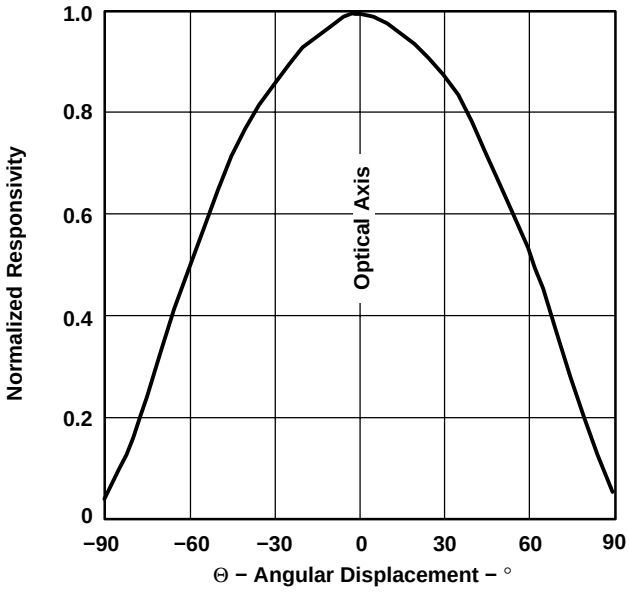


Figure 6

PRINCIPLES OF OPERATION

Analog-to-Digital Converter

The TSL256x contains two integrating analog-to-digital converters (ADC) that integrate the currents from the channel 0 and channel 1 photodiodes. Integration of both channels occurs simultaneously, and upon completion of the conversion cycle the conversion result is transferred to the channel 0 and channel 1 data registers, respectively. The transfers are double buffered to ensure that invalid data is not read during the transfer. After the transfer, the device automatically begins the next integration cycle.

Digital Interface

Interface and control of the TSL256x is accomplished through a two-wire serial interface to a set of registers that provide access to device control functions and output data. The serial interface is compatible with System Management Bus (SMBus) versions 1.1 and 2.0, and I^2C bus Fast-Mode. The TSL256x offers three slave addresses that are selectable via an external pin (ADDR SEL). The slave address options are shown in Table 1.

Table 1. Slave Address Selection

ADDR SEL TERMINAL LEVEL	SLAVE ADDRESS	SMB ALERT ADDRESS
GND	0101001	0001100
Float	0111001	0001100
VDD	1001001	0001100

NOTE: The Slave and SMB Alert Addresses are 7 bits. Please note the SMBus and I^2C protocols on pages 9 through 12. A read/write bit should be appended to the slave address by the master device to properly communicate with the TSL256X device.

SMBus and I^2C Protocols

Each *Send* and *Write* protocol is, essentially, a series of bytes. A byte sent to the TSL256x with the most significant bit (MSB) equal to 1 will be interpreted as a COMMAND byte. The lower four bits of the COMMAND byte form the register select address (see Table 2), which is used to select the destination for the subsequent byte(s) received. The TSL256x responds to any Receive Byte requests with the contents of the register specified by the stored register select address.

The TSL256X implements the following protocols of the SMB 2.0 specification:

- Send Byte Protocol
- Receive Byte Protocol
- Write Byte Protocol
- Write Word Protocol
- Read Word Protocol
- Block Write Protocol
- Block Read Protocol

The TSL256X implements the following protocols of the Philips Semiconductor I^2C specification:

- I^2C Write Protocol
- I^2C Read (Combined Format) Protocol

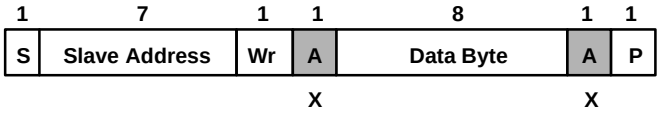
When an SMBus Block Write or Block Read is initiated (see description of COMMAND Register), the byte following the COMMAND byte is ignored but is a requirement of the SMBus specification. This field contains the byte count (i.e. the number of bytes to be transferred). The TSL2560 (SMBus) device ignores this field and extracts this information by counting the actual number of bytes transferred before the Stop condition is detected.

When an I^2C Write or I^2C Read (Combined Format) is initiated, the byte count is also ignored but follows the SMBus protocol specification. Data bytes continue to be transferred from the TSL2561 (I^2C) device to Master until a NACK is sent by the Master.

The data formats supported by the TSL2560 and TSL2561 devices are:

- Master transmitter transmits to slave receiver (SMBus and I^2C):
  - The transfer direction in this case is not changed.
- Master reads slave immediately after the first byte (SMBus only):
  - At the moment of the first acknowledgment (provided by the slave receiver) the master transmitter becomes a master receiver and the slave receiver becomes a slave transmitter.
- Combined format (SMBus and I^2C):
  - During a change of direction within a transfer, the master repeats both a START condition and the slave address but with the R/W bit reversed. In this case, the master receiver terminates the transfer by generating a NACK on the last byte of the transfer and a STOP condition.

For a complete description of SMBus protocols, please review the SMBus Specification at http://www.smbus.org/specs. For a complete description of I^2C protocols, please review the I^2C Specification at http://www.semiconductors.philips.com.



A Acknowledge (this bit position may be 0 for an ACK or 1 for a NACK)

P Stop Condition

Rd Read (bit value of 1)

S Start Condition

Sr Repeated Start Condition

Wr Write (bit value of 0)

X Shown under a field indicates that that field is required to have a value of X

... Continuation of protocol

Master-to-Slave

Slave-to-Master

Figure 7. SMBus and I^2C Packet Protocol Element Key

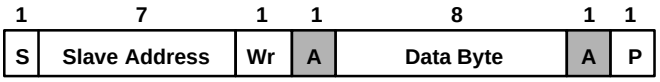


Figure 8. SMBus Send Byte Protocol

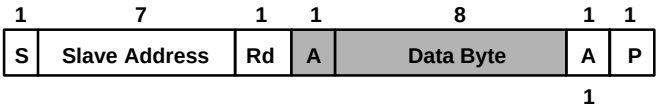


Figure 9. SMBus Receive Byte Protocol

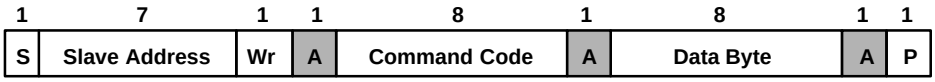


Figure 10. SMBus Write Byte Protocol

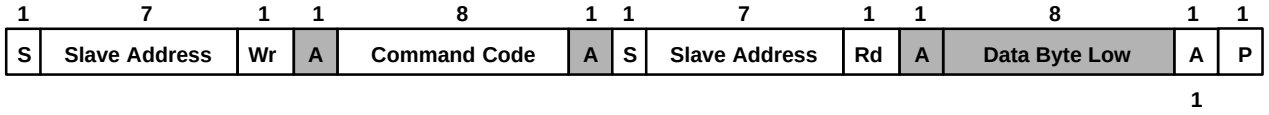


Figure 11. SMBus Read Byte Protocol

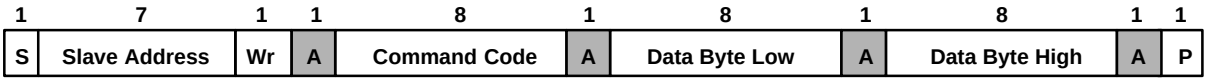


Figure 12. SMBus Write Word Protocol

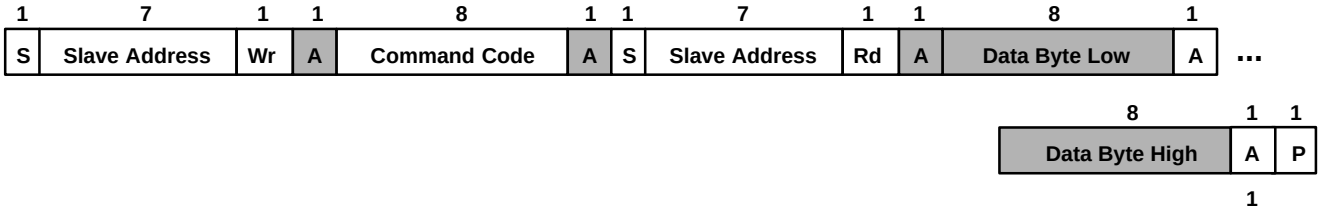


Figure 13. SMBus Read Word Protocol

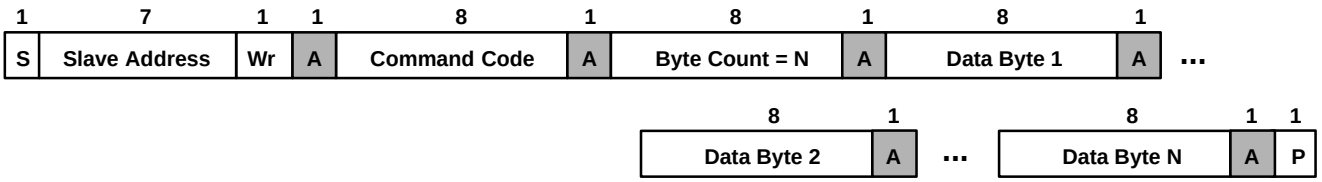


Figure 14. SMBus Block Write or I²C Write Protocols

NOTE: The I²C write protocol does not use the Byte Count packet, and the Master will continue sending Data Bytes until the Master initiates a Stop condition. See the Command Register on page 13 for additional information regarding the Block Read/Write protocol.

Figure 15. SMBus Block Read or I²C Read (Combined Format) Protocols

NOTE: The I²C read protocol does not use the Byte Count packet, and the Master will continue receiving Data Bytes until the Master initiates a Stop Condition. See the Command Register on page 13 for additional information regarding the Block Read/Write protocol.

Register Set

The TSL256x is controlled and monitored by sixteen registers (three are reserved) and a command register accessed through the serial interface. These registers provide for a variety of control functions and can be read to determine results of the ADC conversions. The register set is summarized in Table 2.

Table 2. Register Address

ADDRESS	RESISTER NAME	REGISTER FUNCTION
−−	COMMAND	Specifies register address
0h	CONTROL	Control of basic functions
1h	TIMING	Integration time/gain control
2h	THRESHLOWLOW	Low byte of low interrupt threshold
3h	THRESHLOWHIGH	High byte of low interrupt threshold
4h	THRESHHIGHLOW	Low byte of high interrupt threshold
5h	THRESHHIGHHIGH	High byte of high interrupt threshold
6h	INTERRUPT	Interrupt control
7h	−−	Reserved
8h	CRC	Factory test — not a user register
9h	−−	Reserved
Ah	ID	Part number/ Rev ID
Bh	−−	Reserved
Ch	DATA0LOW	Low byte of ADC channel 0
Dh	DATA0HIGH	High byte of ADC channel 0
Eh	DATA1LOW	Low byte of ADC channel 1
Fh	DATA1HIGH	High byte of ADC channel 1

The mechanics of accessing a specific register depends on the specific SMB protocol used. Refer to the section on SMBus protocols. In general, the COMMAND register is written first to specify the specific control/status register for following read/write operations.

Command Register

The command register specifies the address of the target register for subsequent read and write operations. The Send Byte protocol is used to configure the COMMAND register. The command register contains eight bits as described in Table 3. The command register defaults to 00h at power on.

Table 3. Command Register

	7	6	5	4	3	2	1	0	
	CMD	CLEAR	WORD	BLOCK	ADDRESS				COMMAND
Reset Value:	0	0	0	0	0	0	0	0	

FIELD	BIT	DESCRIPTION
CMD	7	Select command register. Must write as 1.
CLEAR	6	Interrupt clear. Clears any pending interrupt. This bit is a write-one-to-clear bit. It is self clearing.
WORD	5	SMB Write/Read Word Protocol. 1 indicates that this SMB transaction is using either the SMB Write Word or Read Word protocol.
BLOCK	4	Block Write/Read Protocol. 1 indicates that this transaction is using either the Block Write or the Block Read protocol. See Note below.
ADDRESS	3:0	Register Address. This field selects the specific control or status register for following write and read commands according to Table 2.

NOTE: An I^2C block transaction will continue until the Master sends a stop condition. See Figure 14 and Figure 15. Unlike the I2C protocol, the SMBus read/write protocol requires a Byte Count. All four ADC Channel Data Registers (Ch through Fh) can be read simultaneously in a single SMBus transaction. This is the only 32-bit data block supported by the TSL2560 SMBus protocol. The BLOCK bit must be set to 1, and a read condition should be initiated with a COMMAND CODE of 9Bh. By using a COMMAND CODE of 9Bh during an SMBus Block Read Protocol, the TSL2560 device will automatically insert the appropriate Byte Count (Byte Count = 4) as illustrated in Figure 15. A write condition should not be used in conjunction with the Bh register.

Control Register (0h)

The CONTROL register contains two bits and is primarily used to power the TSL256x device up and down as shown in Table 4.

Table 4. Control Register

	7	6	5	4	3	2	1	0	
0h	Resv	Resv	Resv	Resv	Resv	Resv	POWER		CONTROL
Reset Value:	0	0	0	0	0	0	0	0	

FIELD	BIT	DESCRIPTION
Resv	7:2	Reserved. Write as 0.
POWER	1:0	Power up/power down. By writing a 03h to this register, the device is powered up. By writing a 00h to this register, the device is powered down. ***NOTE:*** *If a value of 03h is written, the value returned during a read cycle will be 03h. This feature can be used to verify that the device is communicating properly.*

Timing Register (1h)

The TIMING register controls both the integration time and the gain of the ADC channels. A common set of control bits is provided that controls both ADC channels. The TIMING register defaults to 02h at power on.

Table 5. Timing Register

	7	6	5	4	3	2	1	0	
1h	Resv	Resv	Resv	GAIN	Manual	Resv	INTEG		TIMING
Reset Value:	0	0	0	0	0	0	1	0	

FIELD	BIT	DESCRIPTION
Resv	7−5	Reserved. Write as 0.
GAIN	4	Switches gain between low gain and high gain modes. Writing a 0 selects low gain (1×); writing a 1 selects high gain (16×).
Manual	3	Manual timing control. Writing a 1 begins an integration cycle. Writing a 0 stops an integration cycle. ***NOTE:*** *This field only has meaning when INTEG = 11. It is ignored at all other times.*
Resv	2	Reserved. Write as 0.
INTEG	1:0	Integrate time. This field selects the integration time for each conversion.

Integration time is dependent on the INTEG FIELD VALUE and the internal clock frequency. Nominal integration times and respective scaling between integration times scale proportionally as shown in Table 6. See Note 5 and Note 6 on page 5 for detailed information regarding how the scale values were obtained; see page 22 for further information on how to calculate lux.

Table 6. Integration Time

INTEG FIELD VALUE	SCALE	NOMINAL INTEGRATION TIME
00	0.034	13.7 ms
01	0.252	101 ms
10	1	402 ms
11	−−	N/A

The manual timing control feature is used to manually start and stop the integration time period. If a particular integration time period is required that is not listed in Table 6, then this feature can be used. For example, the manual timing control can be used to synchronize the TSL256x device with an external light source (e.g. LED). A start command to begin integration can be initiated by writing a 1 to this bit field. Correspondingly, the integration can be stopped by simply writing a 0 to the same bit field.

Interrupt Threshold Register (2h − 5h)

The interrupt threshold registers store the values to be used as the high and low trigger points for the comparison function for interrupt generation. If the value generated by channel 0 crosses below or is equal to the low threshold specified, an interrupt is asserted on the interrupt pin. If the value generated by channel 0 crosses above the high threshold specified, an interrupt is asserted on the interrupt pin. Registers THRESHLOWLOW and THRESHLOWHIGH provide the low byte and high byte, respectively, of the lower interrupt threshold. Registers THRESHHIGHLOW and THRESHHIGHHIGH provide the low and high bytes, respectively, of the upper interrupt threshold. The high and low bytes from each set of registers are combined to form a 16-bit threshold value. The interrupt threshold registers default to 00h on power up.

Table 7. Interrupt Threshold Register

REGISTER	ADDRESS	BITS	DESCRIPTION
THRESHLOWLOW	2h	7:0	ADC channel 0 lower byte of the low threshold
THRESHLOWHIGH	3h	7:0	ADC channel 0 upper byte of the low threshold
THRESHHIGHLOW	4h	7:0	ADC channel 0 lower byte of the high threshold
THRESHHIGHHIGH	5h	7:0	ADC channel 0 upper byte of the high threshold

NOTE: Since two 8-bit values are combined for a single 16-bit value for each of the high and low interrupt thresholds, the Send Byte protocol should not be used to write to these registers. Any values transferred by the Send Byte protocol with the MSB set would be interpreted as the COMMAND field and stored as an address for subsequent read/write operations and not as the interrupt threshold information as desired. The Write Word protocol should be used to write byte-paired registers. For example, the THRESHLOWLOW and THRESHLOWHIGH registers (as well as the THRESHHIGHLOW and THRESHHIGHHIGH registers) can be written together to set the 16-bit ADC value in a single transaction.

Interrupt Control Register (6h)

The INTERRUPT register controls the extensive interrupt capabilities of the TSL256x. The TSL256x permits both SMB-Alert style interrupts as well as traditional level-style interrupts. The interrupt persist bit field (PERSIST) provides control over when interrupts occur. A value of 0 causes an interrupt to occur after every integration cycle regardless of the threshold settings. A value of 1 results in an interrupt after one integration time period outside the threshold window. A value of *N* (where *N* is 2 through15) results in an interrupt only if the value remains outside the threshold window for *N* consecutive integration cycles. For example, if *N* is equal to 10 and the integration time is 402 ms, then the total time is approximately 4 seconds.

When a level Interrupt is selected, an interrupt is generated whenever the last conversion results in a value outside of the programmed threshold window. The interrupt is active-low and remains asserted until cleared by writing the COMMAND register with the CLEAR bit set.

In SMBAlert mode, the interrupt is similar to the traditional level style and the interrupt line is asserted low. To clear the interrupt, the host responds to the SMBAlert by performing a modified Receive Byte operation, in which the Alert Response Address (ARA) is placed in the slave address field, and the TSL256x that generated the interrupt responds by returning its own address in the seven most significant bits of the receive data byte. If more than one device connected on the bus has pulled the SMBAlert line low, the highest priority (lowest address) device will win communication rights via standard arbitration during the slave address transfer. If the device loses this arbitration, the interrupt will not be cleared. The Alert Response Address is 0Ch.

When INTR = 11, the interrupt is generated immediately following the SMBus write operation. Operation then behaves in an SMBAlert mode, and the *software set* interrupt may be cleared by an SMBAlert cycle.

NOTE: Interrupts are based on the value of Channel 0 only.

Table 8. Interrupt Control Register

	7	6	5	4	3	2	1	0	
6h	Resv	Resv	INTR		PERSIST				INTERRUPT
Reset Value:	0	0	0	0	0	0	0	0	

FIELD	BITS	DESCRIPTION
Resv	7:6	Reserved. Write as 0.
INTR	5:4	INTR Control Select. This field determines mode of interrupt logic according to Table 9, below.
PERSIST	3:0	Interrupt persistence. Controls rate of interrupts to the host processor as shown in Table 10, below.

Table 9. Interrupt Control Select

INTR FIELD VALUE	READ VALUE
00	Interrupt output disabled
01	Level Interrupt
10	SMBAlert compliant
11	Test Mode: Sets interrupt and functions as mode 10

NOTE: Field value of 11 may be used to test interrupt connectivity in a system or to assist in debugging interrupt service routine software.

Table 10. Interrupt Persistence Select

PERSIST FIELD VALUE	INTERRUPT PERSIST FUNCTION
0000	Every ADC cycle generates interrupt
0001	Any value outside of threshold range
0010	2 integration time periods out of range
0011	3 integration time periods out of range
0100	4 integration time periods out of range
0101	5 integration time periods out of range
0110	6 integration time periods out of range
0111	7 integration time periods out of range
1000	8 integration time periods out of range
1001	9 integration time periods out of range
1010	10 integration time periods out of range
1011	11 integration time periods out of range
1100	12 integration time periods out of range
1101	13 integration time periods out of range
1110	14 integration time periods out of range
1111	15 integration time periods out of range

ID Register (Ah)

The ID register provides the value for both the part number and silicon revision number for that part number. It is a read-only register, whose value never changes.

Table 11. ID Register

	7	6	5	4	3	2	1	0	
Ah	PARTNO				REVNO				ID
Reset Value:	–	–	–	–	–	–	–	–	

FIELD	BITS	DESCRIPTION
PARTNO	7:4	Part Number Identification: field value 0000 = TSL2560, field value 0001 = TSL2561
REVNO	3:0	Revision number identification

ADC Channel Data Registers (Ch − Fh)

The ADC channel data are expressed as 16-bit values spread across two registers. The ADC channel 0 data registers, DATA0LOW and DATA0HIGH provide the lower and upper bytes, respectively, of the ADC value of channel 0. Registers DATA1LOW and DATA1HIGH provide the lower and upper bytes, respectively, of the ADC value of channel 1. All channel data registers are read-only and default to 00h on power up.

Table 12. ADC Channel Data Registers

REGISTER	ADDRESS	BITS	DESCRIPTION
DATA0LOW	Ch	7:0	ADC channel 0 lower byte
DATA0HIGH	Dh	7:0	ADC channel 0 upper byte
DATA1LOW	Eh	7:0	ADC channel 1 lower byte
DATA1HIGH	Fh	7:0	ADC channel 1 upper byte

The upper byte data registers can only be read following a read to the corresponding lower byte register. When the lower byte register is read, the upper eight bits are strobed into a shadow register, which is read by a subsequent read to the upper byte. The upper register will read the correct value even if additional ADC integration cycles end between the reading of the lower and upper registers.

NOTE: The Read Word protocol can be used to read byte-paired registers. For example, the DATA0LOW and DATA0HIGH registers (as well as the DATA1LOW and DATA1HIGH registers) may be read together to obtain the 16-bit ADC value in a single transaction

APPLICATION INFORMATION: SOFTWARE

Basic Operation

After applying V_{DD}, the device will initially be in the power-down state. To operate the device, issue a command to access the CONTROL register followed by the data value 03h to power up the device. At this point, both ADC channels will begin a conversion at the default integration time of 400 ms. After 400 ms, the conversion results will be available in the DATA0 and DATA1 registers. Use the following pseudo code to read the data registers:

```
// Read ADC Channels Using Read Word Protocol − RECOMMENDED
      Address = 0x39                          //Slave addr – also 0x29 or 0x49

      //Address the Ch0 lower data register and configure for Read Word
      Command = 0xAC                          //Set Command bit and Word bit

      //Reads two bytes from sequential registers 0x0C and 0x0D
      //Results are returned in DataLow and DataHigh variables
      ReadWord (Address, Command, DataLow, DataHigh)
      Channel0 = 256 * DataHigh + DataLow

      //Address the Ch1 lower data register and configure for Read Word
      Command = 0xAE                          //Set bit fields 7 and 5

      //Reads two bytes from sequential registers 0x0E and 0x0F
      //Results are returned in DataLow and DataHigh variables
      ReadWord (Address, Command, DataLow, DataHigh)
      Channel1 = 256 * DataHigh + DataLow     //Shift DataHigh to upper byte


// Read ADC Channels Using Read Byte Protocol
      Address = 0x39                          //Slave addr − also 0x29 or 0x49
      Command = 0x8C                          //Address the Ch0 lower data register
      ReadByte (Address, Command, DataLow)    //Result returned in DataLow
      Command = 0x8D                          //Address the Ch0 upper data register
      ReadByte (Address, Command, DataHigh)   //Result returned in DataHigh
      Channel0 = 256 * DataHigh + DataLow     //Shift DataHigh to upper byte

      Command = 0x8E                          //Address the Ch1 lower data register
      ReadByte (Address, Command, DataLow)    //Result returned in DataLow
      Command = 0x8F                          //Address the Ch1 upper data register
      ReadByte (Address, Command, DataHigh)   //Result returned in DataHigh
      Channel1 = 256 * DataHigh + DataLow     //Shift DataHigh to upper byte
```

APPLICATION INFORMATION: SOFTWARE

Configuring the Timing Register

The command, timing, and control registers are initialized to default values on power up. Setting these registers to the desired values would be part of a normal initialization or setup procedure. In addition, to maximize the performance of the device under various conditions, the integration time and gain may be changed often during operation. The following pseudo code illustrates a procedure for setting up the timing register for various options:

```
// Set up Timing Register
       //Low Gain (1x), integration time of 402ms (default value)
       Address = 0x39
       Command = 0x81
       Data = 0x02
       WriteByte(Address, Command, Data)

       //Low Gain (1x), integration time of 101ms
       Data = 0x01
       WriteByte(Address, Command, Data)

       //Low Gain (1x), integration time of 13.7ms
       Data = 0x00
       WriteByte(Address, Command, Data)

       //High Gain (16x), integration time of 101ms
       Data = 0x11
       WriteByte(Address, Command, Data)

//Read data registers (see Basic Operation example)


//Perform Manual Integration
       //Set up for manual integration with Gain of 1x
       Data = 0x03
       //Set manual integration mode – device stops converting
       WriteByte(Address, Command, Data)

       //Begin integration period
       Data = 0x0B
       WriteByte(Address, Command, Data)

       //Integrate for 50ms
       Sleep (50)                                  //Wait for 50ms

       //Stop integrating
       Data = 0x03
       WriteByte(Address, Command, Data)

//Read data registers (see Basic Operation example)
```

APPLICATION INFORMATION: SOFTWARE

Interrupts

The interrupt feature of the TSL256x device simplifies and improves system efficiency by eliminating the need to poll the sensor for a light intensity value. Interrupt styles are determined by the INTR field in the Interrupt Register. The interrupt feature may be disabled by writing a field value of 00h to the Interrupt Control Register so that polling can be performed.

The versatility of the interrupt feature provides many options for interrupt configuration and usage. The primary purpose of the interrupt function is to provide a meaningful change in light intensity. However, it also be used as an end-of-conversion signal. The concept of a *meaningful change* can be defined by the user both in terms of light intensity and time, or persistence, of that change in intensity. The TSL256x device implements two 16-bit-wide interrupt threshold registers that allow the user to define a threshold above and below the current light level. An interrupt will then be generated when the value of a conversion exceeds either of these limits. For simplicity of programming, the threshold comparison is accomplished only with Channel 0. This simplifies calculation of thresholds that are based, for example, on a percent of the current light level. It is adequate to use only one channel when calculating light intensity differences since, for a given light source, the channel 0 and channel 1 values are linearly proportional to each other and thus both values scale linearly with light intensity.

To further control when an interrupt occurs, the TSL256x device provides an interrupt persistence feature. This feature allows the user to specify a number of conversion cycles for which a light intensity exceeding either interrupt threshold must persist before actually generating an interrupt. This can be used to prevent transient changes in light intensity from generating an unwanted interrupt. With a value of 1, an interrupt occurs immediately whenever either threshold is exceeded. With values of *N*, where *N* can range from 2 to 15, *N* consecutive conversions must result in values outside the interrupt window for an interrupt to be generated. For example, if *N* is equal to 10 and the integration time is 402 ms, then an interrupt will not be generated unless the light level persists for more than 4 seconds outside the threshold.

Two different interrupt styles are available: Level and SMBus Alert. The difference between these two interrupt styles is how they are cleared. Both result in the interrupt line going active low and remaining low until the interrupt is cleared. A level style interrupt is cleared by setting the CLEAR bit (bit 6) in the COMMAND register. The SMBus Alert style interrupt is cleared by an Alert Response as described in the Interrupt Control Register section and SMBus specification.

To configure the interrupt as an end-of-conversion signal, the interrupt PERSIST field is set to 0. Either Level or SMBus Alert style can be used. An interrupt will be generated upon completion of each conversion. The interrupt threshold registers are ignored. The following example illustrates the configuration of a level interrupt:

```
// Set up end−of−conversion interrupt, Level style
      Address = 0x39                          //Slave addr also 0x29 or 0x49
      Command = 0x86                          //Address Interrupt Register
      Data = 0x10                             //Level style, every ADC cycle
      WriteByte(Address, Command, Data)
```

APPLICATION INFORMATION: SOFTWARE

The following example pseudo code illustrates the configuration of an SMB Alert style interrupt when the light intensity changes 20% from the current value, and persists for 3 conversion cycles:

```
// Read current light level
       Address = 0x39                              //Slave addr also 0x29 or 0x49
       Command = 0xAC                              //Set Command bit and Word bit
       ReadWord (Address, Command, DataLow, DataHigh)
       Channel0 = (256 * DataHigh) + DataLow

       //Calculate upper and lower thresholds
       T_Upper = Channel0 + (0.2 * Channel0)
       T_Lower = Channel0 – (0.2 * Channel0)

       //Write the lower threshold register
       Command = 0xA2                              //Addr lower threshold reg, set Word Bit
       WriteWord (Address, Command, T_Lower.LoByte, T_Lower.HiByte)

       //Write the upper threshold register
       Command = 0xA4                              //Addr upper threshold reg, set Word bit
       WriteWord (Address, Command, T_Upper.LoByte, T_Upper.HiByte)

       //Enable interrupt
       Command = 0x86                              //Address interrupt register
       Data = 0x23                                 //SMBAlert style, PERSIST = 3
       WriteByte(Address, Command, Data)
```

In order to generate an interrupt on demand during system test or debug, a test mode (INTR = 11) can be used. The following example illustrates how to generate an interrupt on demand:

```
// Generate an interrupt
       Address = 0x39                              //Slave addr also 0x29 or 0x49
       Command = 0x86                              //Address Interrupt register
       Data = 0x30                                 //Test interrupt
       WriteByte(Address, Command, Data)

       //Interrupt line should now be low
```

APPLICATION INFORMATION: SOFTWARE

Calculating Lux

The TSL256x is intended for use in ambient light detection applications such as display backlight control, where adjustments are made to display brightness or contrast based on the brightness of the ambient light, as perceived by the human eye. Conventional silicon detectors respond strongly to infrared light, which the human eye does not see. This can lead to significant error when the infrared content of the ambient light is high, such as with incandescent lighting, due to the difference between the silicon detector response and the brightness perceived by the human eye.

This problem is overcome in the TSL256x through the use of two photodiodes. One of the photodiodes (channel 0) is sensitive to both visible and infrared light, while the second photodiode (channel 1) is sensitive primarily to infrared light. An integrating ADC converts the photodiode currents to digital outputs. Channel 1 digital output is used to compensate for the effect of the infrared component of light on the channel 0 digital output. The ADC digital outputs from the two channels are used in a formula to obtain a value that approximates the human eye response in the commonly used Illuminance unit of Lux:

Chipscale Package

For $0 < CH1/CH0 \leq 0.52$	$Lux = 0.0315 \times CH0 - 0.0593 \times CH0 \times ((CH1/CH0)^{1.4})$
For $0.52 < CH1/CH0 \leq 0.65$	$Lux = 0.0229 \times CH0 - 0.0291 \times CH1$
For $0.65 < CH1/CH0 \leq 0.80$	$Lux = 0.0157 \times CH0 - 0.0180 \times CH1$
For $0.80 < CH1/CH0 \leq 1.30$	$Lux = 0.00338 \times CH0 - 0.00260 \times CH1$
For $CH1/CH0 > 1.30$	$Lux = 0$

TMB Package

For $0 < CH1/CH0 \leq 0.50$	$Lux = 0.0304 \times CH0 - 0.062 \times CH0 \times ((CH1/CH0)^{1.4})$
For $0.50 < CH1/CH0 \leq 0.61$	$Lux = 0.0224 \times CH0 - 0.031 \times CH1$
For $0.61 < CH1/CH0 \leq 0.80$	$Lux = 0.0128 \times CH0 - 0.0153 \times CH1$
For $0.80 < CH1/CH0 \leq 1.30$	$Lux = 0.00146 \times CH0 - 0.00112 \times CH1$
For $CH1/CH0 > 1.30$	$Lux = 0$

The formulas shown above were obtained by optical testing with fluorescent and incandescent light sources, and apply only to open-air applications. Optical apertures (e.g. light pipes) will affect the incident light on the device.

Simplified Lux Calculation

Below is the argument and return value including source code (shown on following page) for calculating lux. The source code is intended for embedded and/or microcontroller applications. Two individual code sets are provided, one for the chipscale package and one for the TMB package. All floating point arithmetic operations have been eliminated since embedded controllers and microcontrollers generally do not support these types of operations. Since floating point has been removed, scaling must be performed prior to calculating illuminance if the integration time is not 402 ms and/or if the gain is not 16× as denoted in the source code on the following pages. This sequence scales first to mitigate rounding errors induced by decimal math.

```
extern unsigned int CalculateLux(unsigned int iGain, unsigned int tInt, unsigned int
    ch0, unsigned int ch1, int iType)
```

```
//**************************************************************************
//
//  Copyright © 2004−2005 TAOS, Inc.
//
//  THIS CODE AND INFORMATION IS PROVIDED ″AS IS″ WITHOUT WARRANTY OF ANY
//  KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
//  IMPLIED WARRANTIES OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR
//  PURPOSE.
//
//     Module Name:
//            lux.cpp
//
//**************************************************************************

#define LUX_SCALE   14     // scale by 2^14
#define RATIO_SCALE 9      // scale ratio by 2^9

//−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
// Integration time scaling factors
//−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

#define CH_SCALE           10     // scale channel values by 2^10
#define CHSCALE_TINT0      0x7517 // 322/11 * 2^CH_SCALE
#define CHSCALE_TINT1      0x0fe7 // 322/81 * 2^CH_SCALE

//−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
// T Package coefficients
//−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
// For Ch1/Ch0=0.00 to 0.50
//            Lux/Ch0=0.0304−0.062*((Ch1/Ch0)^1.4)
//            piecewise approximation
//                   For Ch1/Ch0=0.00 to 0.125:
//                          Lux/Ch0=0.0304−0.0272*(Ch1/Ch0)
//
//                   For Ch1/Ch0=0.125 to 0.250:
//                          Lux/Ch0=0.0325−0.0440*(Ch1/Ch0)
//
//                   For Ch1/Ch0=0.250 to 0.375:
//                          Lux/Ch0=0.0351−0.0544*(Ch1/Ch0)
//
//                   For Ch1/Ch0=0.375 to 0.50:
//                          Lux/Ch0=0.0381−0.0624*(Ch1/Ch0)
//
// For Ch1/Ch0=0.50 to 0.61:
//            Lux/Ch0=0.0224−0.031*(Ch1/Ch0)
//
// For Ch1/Ch0=0.61 to 0.80:
//            Lux/Ch0=0.0128−0.0153*(Ch1/Ch0)
//
// For Ch1/Ch0=0.80 to 1.30:
//            Lux/Ch0=0.00146−0.00112*(Ch1/Ch0)
//
// For Ch1/Ch0>1.3:
//            Lux/Ch0=0
//−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
#define K1T   0x0040       // 0.125 * 2^RATIO_SCALE
#define B1T   0x01f2       // 0.0304 * 2^LUX_SCALE
#define M1T   0x01be       // 0.0272 * 2^LUX_SCALE

#define K2T   0x0080       // 0.250 * 2^RATIO_SCALE
```

```
#define B2T   0x0214        // 0.0325 * 2^LUX_SCALE
#define M2T   0x02d1        // 0.0440 * 2^LUX_SCALE

#define K3T   0x00c0        // 0.375 * 2^RATIO_SCALE
#define B3T   0x023f        // 0.0351 * 2^LUX_SCALE
#define M3T   0x037b        // 0.0544 * 2^LUX_SCALE

#define K4T   0x0100        // 0.50 * 2^RATIO_SCALE
#define B4T   0x0270        // 0.0381 * 2^LUX_SCALE
#define M4T   0x03fe        // 0.0624 * 2^LUX_SCALE
#define K5T   0x0138        // 0.61 * 2^RATIO_SCALE
#define B5T   0x016f        // 0.0224 * 2^LUX_SCALE
#define M5T   0x01fc        // 0.0310 * 2^LUX_SCALE

#define K6T   0x019a        // 0.80 * 2^RATIO_SCALE
#define B6T   0x00d2        // 0.0128 * 2^LUX_SCALE
#define M6T   0x00fb        // 0.0153 * 2^LUX_SCALE

#define K7T   0x029a        // 1.3 * 2^RATIO_SCALE
#define B7T   0x0018        // 0.00146 * 2^LUX_SCALE
#define M7T   0x0012        // 0.00112 * 2^LUX_SCALE

#define K8T   0x029a        // 1.3 * 2^RATIO_SCALE
#define B8T   0x0000        // 0.000 * 2^LUX_SCALE
#define M8T   0x0000        // 0.000 * 2^LUX_SCALE

//------------------------------------------------
// CS package coefficients
//------------------------------------------------
// For 0 <= Ch1/Ch0 <= 0.52
//           Lux/Ch0 = 0.0315−0.0593*((Ch1/Ch0)^1.4)
//           piecewise approximation
//                 For 0 <= Ch1/Ch0 <= 0.13
//                       Lux/Ch0 = 0.0315−0.0262*(Ch1/Ch0)
//                 For 0.13 <= Ch1/Ch0 <= 0.26
//                       Lux/Ch0 = 0.0337−0.0430*(Ch1/Ch0)
//                 For 0.26 <= Ch1/Ch0 <= 0.39
//                       Lux/Ch0 = 0.0363−0.0529*(Ch1/Ch0)
//                 For 0.39 <= Ch1/Ch0 <= 0.52
//                       Lux/Ch0 = 0.0392−0.0605*(Ch1/Ch0)
// For 0.52 < Ch1/Ch0 <= 0.65
//     Lux/Ch0 = 0.0229−0.0291*(Ch1/Ch0)
// For 0.65 < Ch1/Ch0 <= 0.80
//     Lux/Ch0 = 0.00157−0.00180*(Ch1/Ch0)
// For 0.80 < Ch1/Ch0 <= 1.30
//     Lux/Ch0 = 0.00338−0.00260*(Ch1/Ch0)
// For Ch1/Ch0 > 1.30
//     Lux = 0
//------------------------------------------------
#define K1C   0x0043 // 0.130 * 2^RATIO_SCALE
#define B1C   0x0204 // 0.0315 * 2^LUX_SCALE
#define M1C   0x01ad // 0.0262 * 2^LUX_SCALE

#define K2C   0x0085 // 0.260 * 2^RATIO_SCALE
#define B2C   0x0228 // 0.0337 * 2^LUX_SCALE
#define M2C   0x02c1 // 0.0430 * 2^LUX_SCALE

#define K3C   0x00c8 // 0.390 * 2^RATIO_SCALE
#define B3C   0x0253 // 0.0363 * 2^LUX_SCALE
#define M3C   0x0363 // 0.0529 * 2^LUX_SCALE
```

```
#define K4C   0x010a // 0.520 * 2^RATIO_SCALE
#define B4C   0x0282 // 0.0392 * 2^LUX_SCALE
#define M4C   0x03df // 0.0605 * 2^LUX_SCALE

#define K5C   0x014d // 0.65 * 2^RATIO_SCALE
#define B5C   0x0177 // 0.0229 * 2^LUX_SCALE
#define M5C   0x01dd // 0.0291 * 2^LUX_SCALE

#define K6C   0x019a // 0.80 * 2^RATIO_SCALE
#define B6C   0x0101 // 0.0157 * 2^LUX_SCALE
#define M6C   0x0127 // 0.0180 * 2^LUX_SCALE

#define K7C   0x029a // 1.3 * 2^RATIO_SCALE
#define B7C   0x0037 // 0.00338 * 2^LUX_SCALE
#define M7C   0x002b // 0.00260 * 2^LUX_SCALE

#define K8C   0x029a // 1.3 * 2^RATIO_SCALE
#define B8C   0x0000 // 0.000 * 2^LUX_SCALE
#define M8C   0x0000 // 0.000 * 2^LUX_SCALE

// lux equation approximation without floating point calculations
//////////////////////////////////////////////////////////////////////////////
//     Routine:     unsigned int CalculateLux(unsigned int ch0, unsigned int ch0, int iType)
//
//     Description: Calculate the approximate illuminance (lux) given the raw
//                  channel values of the TSL2560. The equation if implemented
//                  as a piece−wise linear approximation.
//
//     Arguments:   unsigned int iGain − gain, where 0:1X, 1:16X
//                  unsigned int tInt − integration time, where 0:13.7mS, 1:100mS, 2:402mS,
//                   3:Manual
//                  unsigned int ch0 − raw channel value from channel 0 of TSL2560
//                  unsigned int ch1 − raw channel value from channel 1 of TSL2560
//                  unsigned int iType − package type (T or CS)
//
//     Return:      unsigned int − the approximate illuminance (lux)
//
//////////////////////////////////////////////////////////////////////////////
unsigned int CalculateLux(unsigned int iGain, unsigned int tInt, unsigned int ch0,
      unsigned int ch1, int iType)
{

      //−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
      // first, scale the channel values depending on the gain and integration time
      // 16X, 402mS is nominal.
      // scale if integration time is NOT 402 msec
      unsigned long chScale;
      unsigned long channel1;
      unsigned long channel0;
      switch (tInt)
      {
          case 0:    // 13.7 msec
                chScale = CHSCALE_TINT0;
                break;
          case 1:    // 101 msec
                chScale = CHSCALE_TINT1;
                break;
          default:   // assume no scaling
                chScale = (1 << CH_SCALE);
```

```
            break;
      }

      // scale if gain is NOT 16X
      if (!iGain) chScale = chScale << 4;   // scale 1X to 16X

      // scale the channel values
      channel0 = (ch0 * chScale) >> CH_SCALE;
      channel1 = (ch1 * chScale) >> CH_SCALE;
      //--------------------------------------------------------------------

      // find the ratio of the channel values (Channel1/Channel0)
      // protect against divide by zero
      unsigned long ratio1 = 0;
      if (channel0 != 0) ratio1 = (channel1 << (RATIO_SCALE+1)) / channel0;

      // round the ratio value
      unsigned long ratio = (ratio1 + 1) >> 1;

      // is ratio <= eachBreak ?
      unsigned int b, m;
      switch (iType)
      {



          case 0: // T package
                if ((ratio >= 0) && (ratio <= K1T))
                      {b=B1T; m=M1T;}
                else if (ratio <= K2T)
                      {b=B2T; m=M2T;}
                else if (ratio <= K3T)
                      {b=B3T; m=M3T;}
                else if (ratio <= K4T)
                      {b=B4T; m=M4T;}
                else if (ratio <= K5T)
                      {b=B5T; m=M5T;}
                else if (ratio <= K6T)
                      {b=B6T; m=M6T;}
                else if (ratio <= K7T)
                      {b=B7T; m=M7T;}
                else if (ratio > K8T)
                      {b=B8T; m=M8T;}
                break;

          case 1:// CS package
                if ((ratio >= 0) && (ratio <= K1C))
                      {b=B1C; m=M1C;}
                else if (ratio <= K2C)
                      {b=B2C; m=M2C;}
                else if (ratio <= K3C)
                      {b=B3C; m=M3C;}
                else if (ratio <= K4C)
                      {b=B4C; m=M4C;}
                else if (ratio <= K5C)
                      {b=B5C; m=M5C;}
                else if (ratio <= K6C)
                      {b=B6C; m=M6C;}
                else if (ratio <= K7C)
                      {b=B7C; m=M7C;}
```

```
                else if (ratio > K8C)
                       {b=B8C; m=M8C;}
                break;
        }

        unsigned long temp;
        temp = ((channel0 * b) − (channel1 * m));

        // do not allow negative lux value
        if (temp < 0) temp = 0;

        // round lsb (2^(LUX_SCALE−1))
        temp += (1 << (LUX_SCALE−1));

        // strip off fractional portion
        unsigned long lux = temp >> LUX_SCALE;

        return(lux);

}
```

APPLICATION INFORMATION: HARDWARE

Power Supply Decoupling

The power supply lines must be decoupled with a 0.1 μF capacitor placed as close to the device package as possible (Figure 16). The bypass capacitor should have low effective series resistance (ESR) and low effective series inductance (ESI), such as the common ceramic types, which provide a low impedance path to ground at high frequencies to handle transient currents caused by internal logic switching.

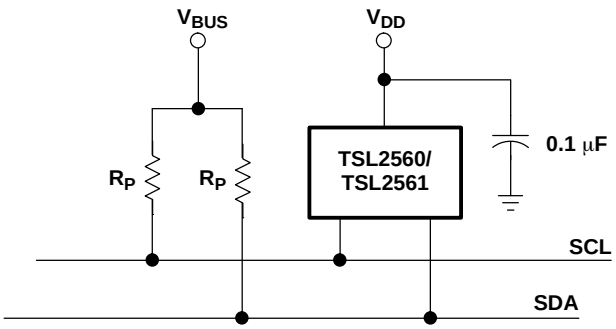


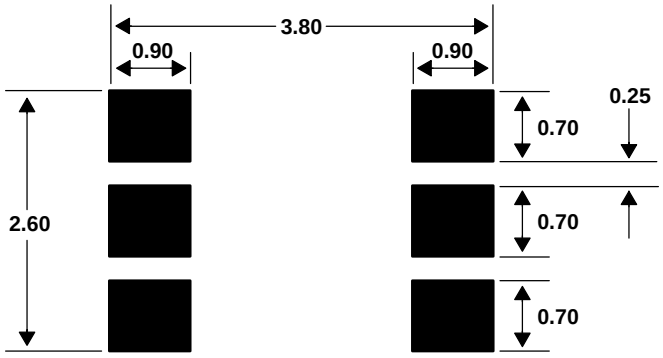
Figure 16. Bus Pull-Up Resistors

Pull-up resistors (Rp) maintain the SDAH and SCLH lines at a *high* level when the bus is free and ensure the signals are pulled up from a low to a high level within the required rise time. For a complete description of the SMBus maximum and minimum Rp values, please review the SMBus Specification at http://www.smbus.org/specs. For a complete description of I^2C maximum and minimum Rp values, please review the I^2C Specification at http://www.semiconductors.philips.com.

APPLICATION INFORMATION: HARDWARE

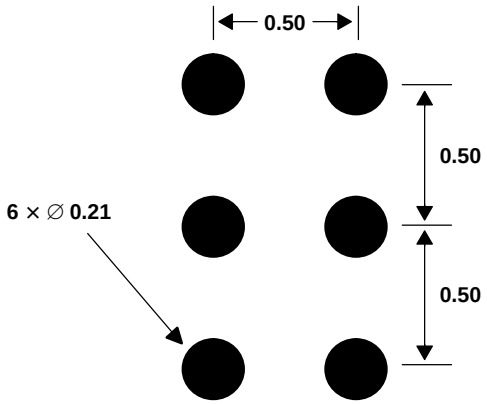
PCB Pad Layout

Suggested PCB pad layout guidelines for the TMB-6 surface mount package and CS chipscale package are shown in Figure 17 and Figure 18.



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.

Figure 17. Suggested TMB-6 Package PCB Layout



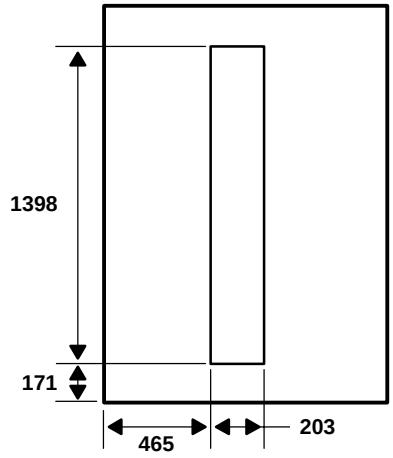
NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.

Figure 18. Suggested Chipscale Package PCB Layout

MECHANICAL DATA

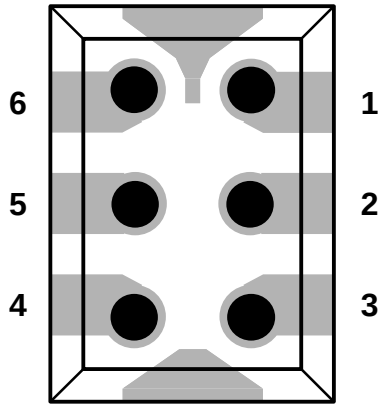
PACKAGE CS — **Six-Lead Chipscale Device**

TOP VIEW

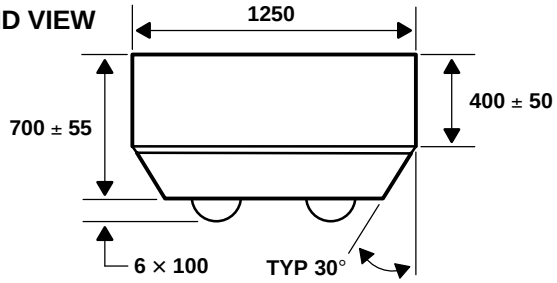


PIN OUT

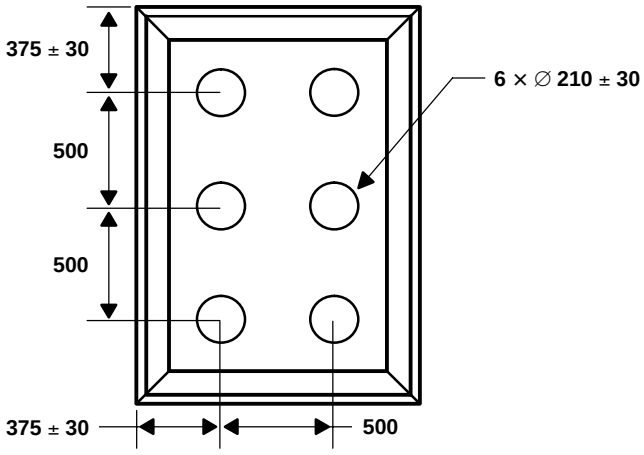
BOTTOM VIEW



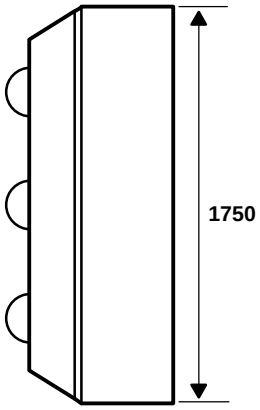
END VIEW



BOTTOM VIEW



SIDE VIEW



Lead Free

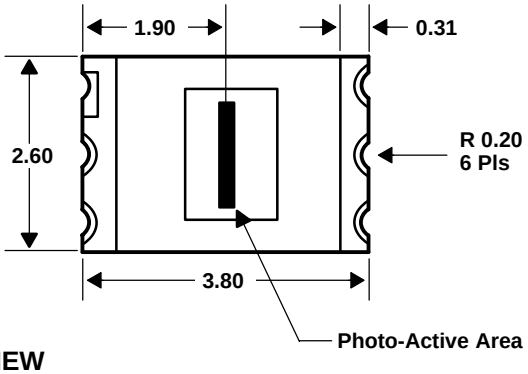
NOTES: A. All linear dimensions are in micrometers. Dimension tolerance is ± 25 μm unless otherwise noted.
B. Solder bumps are formed of Sn (96%), Ag (3.5%), and Cu (0.5%).
C. The top of the photodiode active area is 410 μm below the top surface of the package.
D. The layer above the photodiode is glass and epoxy with an index of refraction of 1.53.
E. This drawing is subject to change without notice.

Figure 19. Package CS — Six-Lead Chipscale Packaging Configuration

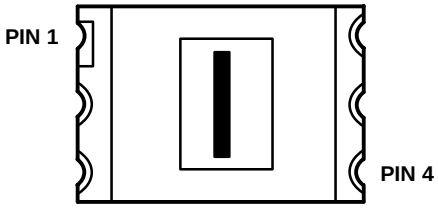
MECHANICAL DATA

PACKAGE TMB-6 **Six-Lead Surface Mount Device**

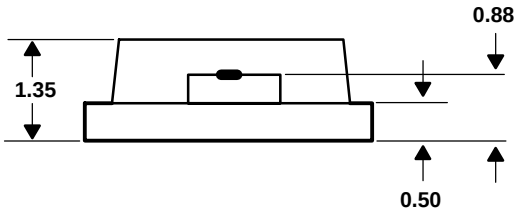
TOP VIEW



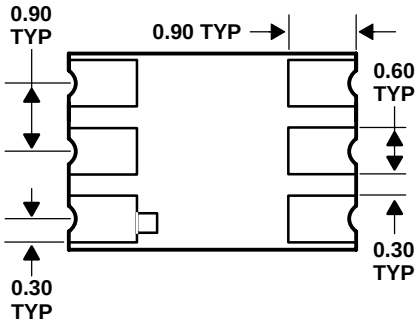
TOP VIEW



END VIEW



BOTTOM VIEW



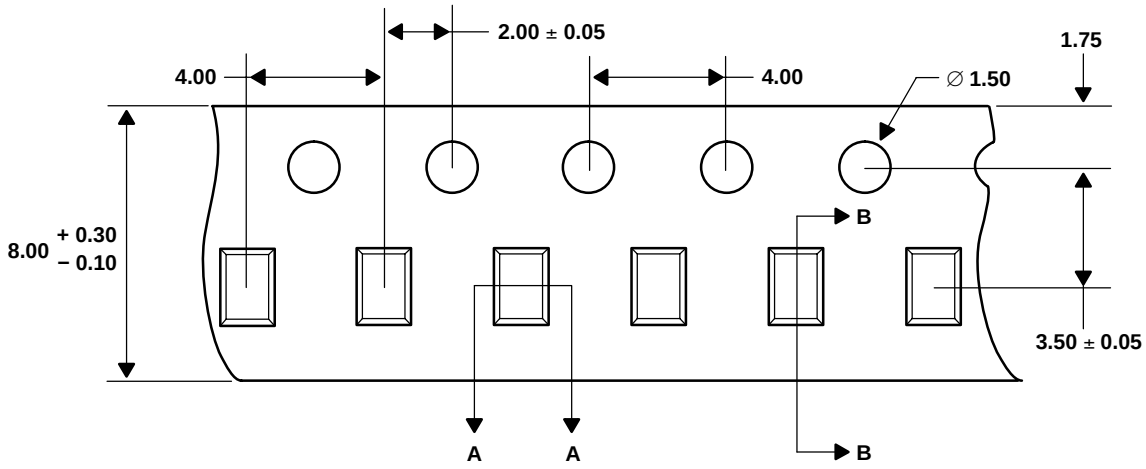
Lead Free

NOTES: A. All linear dimensions are in millimeters. Dimension tolerance is ± 0.20 mm unless otherwise noted.
B. The photo-active area is 1398 µm by 203 µm.
C. Package top surface is molded with an electrically nonconductive clear plastic compound having an index of refraction of 1.55.
D. Contact finish is 0.5 µm minimum of soft gold plated over a 18 µm thick copper foil pattern with a 5 µm to 9 µm nickel barrier.
E. The plastic overmold material has an index of refraction of 1.55.
F. This package contains no lead (Pb).
G. This drawing is subject to change without notice.

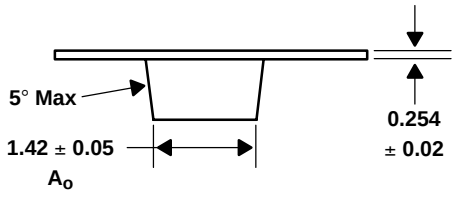
Figure 20. Package T — Six-Lead TMB Plastic Surface Mount Packaging Configuration

MECHANICAL DATA

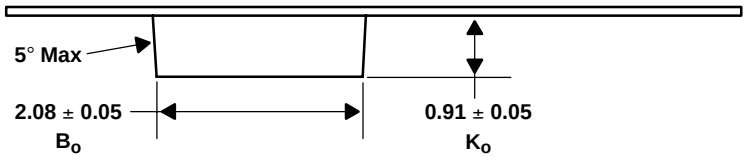
TOP VIEW

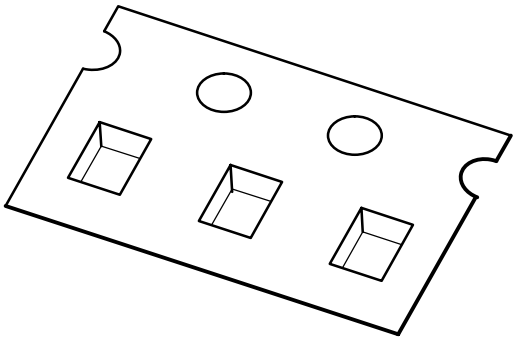


DETAIL A



DETAIL B

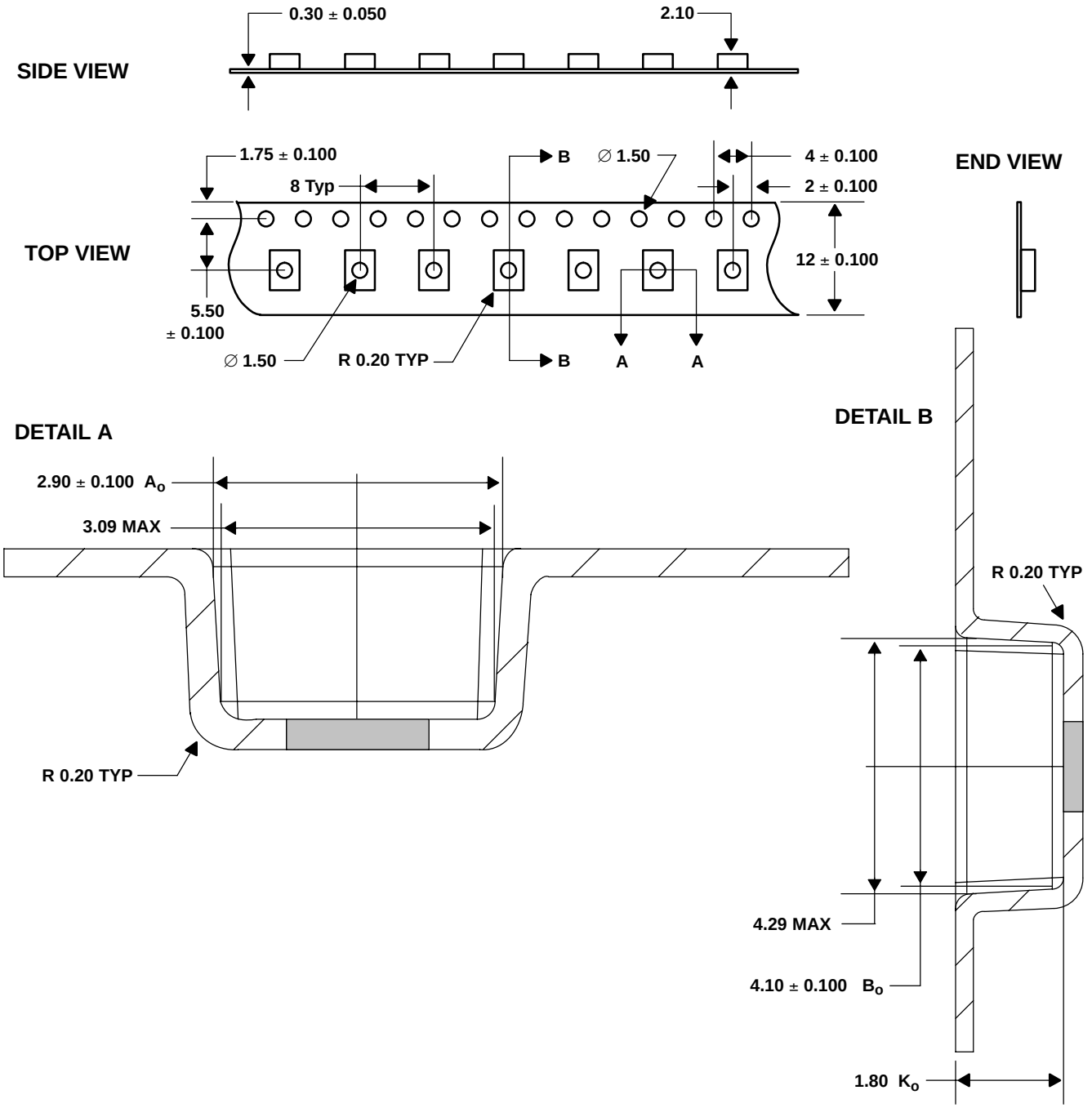




NOTES: A. All linear dimensions are in millimeters. Dimension tolerance is ± 0.10 mm unless otherwise noted.
B. The dimensions on this drawing are for illustrative purposes only. Dimensions of an actual carrier may vary slightly.
C. Symbols on drawing A_o, B_o, and K_o are defined in ANSI EIA Standard 481−B 2001.
D. Each reel is 178 millimeters in diameter and contains 3500 parts.
E. TAOS packaging tape and reel conform to the requirements of EIA Standard 481−B.
F. This drawing is subject to change without notice.

Figure 21. TSL2560/TSL2561 Chipscale Carrier Tape

MECHANICAL DATA



NOTES: A. All linear dimensions are in millimeters.
B. The dimensions on this drawing are for illustrative purposes only. Dimensions of an actual carrier may vary slightly.
C. Symbols on drawing A_o, B_o, and K_o are defined in ANSI EIA Standard 481−B 2001.
D. Each reel is 178 millimeters in diameter and contains 1000 parts.
E. TAOS packaging tape and reel conform to the requirements of EIA Standard 481−B.
F. This drawing is subject to change without notice.

Figure 22. TSL2560/TSL2561 TMB Carrier Tape

MANUFACTURING INFORMATION

The CS and T packages have been tested and have demonstrated an ability to be reflow soldered to a PCB substrate. The process, equipment, and materials used in these test are detailed below.

The solder reflow profile describes the expected maximum heat exposure of components during the solder reflow process of product on a PCB. Temperature is measured on top of component. The components should be limited to a maximum of three passes through this solder reflow profile.

Table 13. TSL2560/61 Solder Reflow Profile

PARAMETER	REFERENCE	TSL2560/61
Average temperature gradient in preheating		2.5°C/sec
Soak time	t_{soak}	2 to 3 minutes
Time above 217°C	t_1	Max 60 sec
Time above 230°C	t_2	Max 50 sec
Time above T_{peak} −10°C	t_3	Max 10 sec
Peak temperature in reflow	T_{peak}	260° C (−0°C/+5°C)
Temperature gradient in cooling		Max −5°C/sec

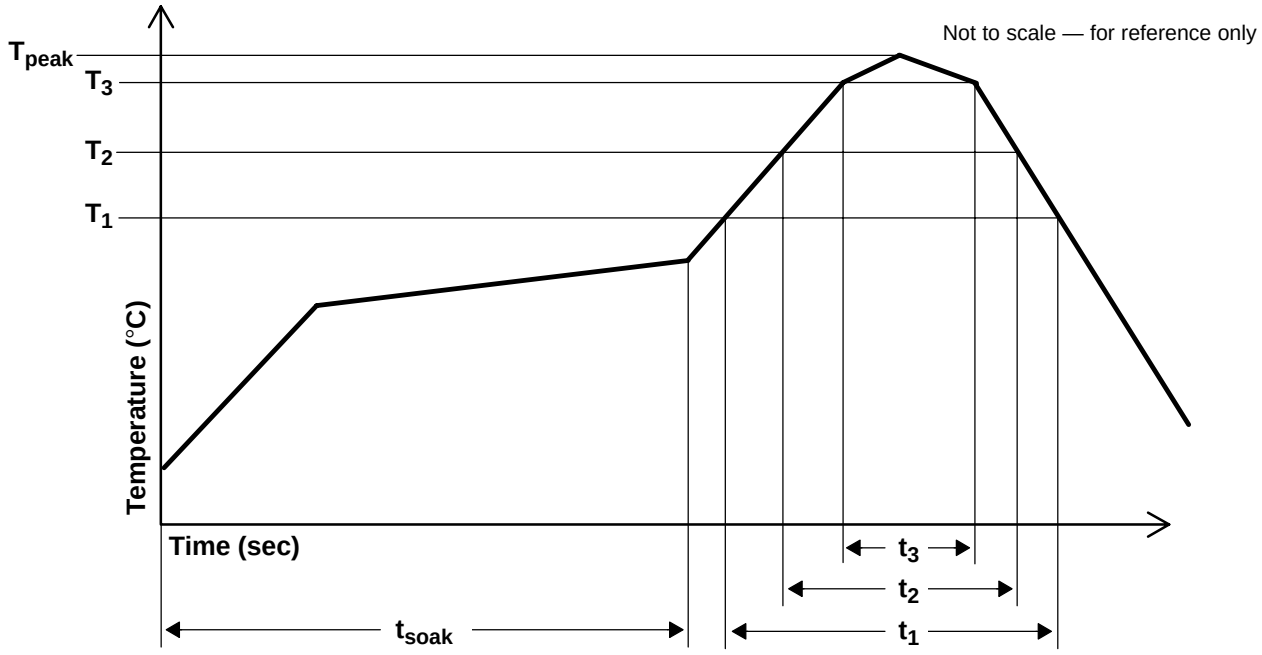


Figure 23. TSL2560/TSL2561 Solder Reflow Profile Graph

MANUFACTURING INFORMATION

Tooling Required

- Chipscale
  - Solder stencil (square aperture size 0.210 mm, stencil thickness of 152 μm)
- TMB
  - Solder stencil (aperture size 0.70 mm x 0.90 mm, stencil thickness of 152 μm)

Process

1. Apply solder paste using stencil
2. Place component
3. Reflow solder/cure
4. X-Ray verify (recommended for chipscale only)

Additional Notes for Chipscale

Placement of the TSL2560/TSL2561 chipscale device onto the gold immersion substrate is accomplished using a standard surface mount manufacturing process. Using a 152-μm stencil with a 0.21 mm square aperture, print solder paste onto the substrate. Machine-place the TSL2560/TSL2561 from the tape onto the substrate. A suggest pick-up tool is the Siemens Vacuum Pickup tool nozzle number 912. This nozzle has a rubber tip with a diameter of approximately 0.75 mm. The part is picked up from the center of the body.

It is important to use a substrate that has an immersion plating surface. This may be immersion gold, solder, or white tin. Hot air solder leveled (HASL) substrates are not coplanar, making them difficult to work with.

Qualified Equipment

- EKRA E5 — Stencil Printer
- ASYMTEC Century — Dispensing system
- SIEMENS F5 — Placement system
  - SIEMENS 912 — Vacuum Pickup Tool Nozzle
- VITRONICS 820 — Oven
- PHOENIX — Inspector X-Ray system

Qualified Materials

- Microbond solder paste, part number NC421

PRODUCTION DATA — information in this document is current at publication date. Products conform to specifications in accordance with the terms of Texas Advanced Optoelectronic Solutions, Inc. standard warranty. Production processing does not necessarily include testing of all parameters.

LEAD-FREE (Pb-FREE) and GREEN STATEMENT

Pb-Free (RoHS) TAOS' terms *Lead-Free* or *Pb-Free* mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TAOS Pb-Free products are suitable for use in specified lead-free processes.

Green (RoHS & no Sb/Br) TAOS defines *Green* to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material).

Important Information and Disclaimer The information provided in this statement represents TAOS' knowledge and belief as of the date that it is provided. TAOS bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TAOS has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TAOS and TAOS suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

NOTICE

Texas Advanced Optoelectronic Solutions, Inc. (TAOS) reserves the right to make changes to the products contained in this document to improve performance or for any other purpose, or to discontinue them without notice. Customers are advised to contact TAOS to obtain the latest product information before placing orders or designing TAOS products into systems.

TAOS assumes no responsibility for the use of any products or circuits described in this document or customer product design, conveys no license, either expressed or implied, under any patent or other right, and makes no representation that the circuits are free of patent infringement. TAOS further makes no claim as to the suitability of its products for any particular purpose, nor does TAOS assume any liability arising out of the use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages.

TEXAS ADVANCED OPTOELECTRONIC SOLUTIONS, INC. PRODUCTS ARE NOT DESIGNED OR INTENDED FOR USE IN CRITICAL APPLICATIONS IN WHICH THE FAILURE OR MALFUNCTION OF THE TAOS PRODUCT MAY RESULT IN PERSONAL INJURY OR DEATH. USE OF TAOS PRODUCTS IN LIFE SUPPORT SYSTEMS IS EXPRESSLY UNAUTHORIZED AND ANY SUCH USE BY A CUSTOMER IS COMPLETELY AT THE CUSTOMER'S RISK.