UNIVERSIDAD POLITECNICA DE VALENCIA
CAMPUS DE ALCOY

ESCUELA POLITÉCNICA SUPERIOR DE ALCOY

DESARROLLO DE UN SISTEMA DE CONTROL REMOTO DE REGADÍO

Trabajo Fin de Grado

Ingeniería Eléctrica

Autor: Planes Tur, Salvador

Tutor: Miró Orozco, Ignacio

Curso: 2017-2018

Dedicatoria

Agradecer el apoyo que me ha dado durante todo este tiempo a María del Mar Torres Rosello que sin su apoyo muchas cosas no hubiesen sido realizadas.

RESUMEN

Este proyecto versará sobre un control de regadío para un huerto de naranjos con una Raspberry pi y un Arduino donde se automatizará todo lo relacionado con el regadío, así como la automatización de las averías. Además, se guardarán todos los datos relacionados con las averías o alertas en una memoria extraíble, y se creará una interfaz gráfica con una pantalla, donde se podrán observar los diferentes valores obtenidos por los sensores instalados.

Se utilizarán sensores para poder controlar y observar la humedad del terreno donde se encuentran arraigados los naranjos, y de esta forma se podrá utilizar el agua de un modo más eficiente regando solo cuando sea estrictamente necesario.

Todos los sensores instalados se comunicaran con el Arduino por módulos de radio frecuencia; la alimentación del Arduino y la Raspberry pi la proporcionarán una placa solar y una batería.

Se instalará un módulo GSM para comunicarse con el propietario del huerto inmediatamente a través del teléfono móvil cuando se produzca una alerta o avería.

Palabras clave: Arduino, Raspberry pi, humedad, huerto y automatización.

RESUM

Aquest projecte versarà sobre un control de regadiu per a un hort de tarongers mitjançant una Raspberry pi i un Arduino on s'automatitzarà tot el relacionat amb el regadiu, així com l'automatització de les averies. A més, es guardaran totes les dades relacionades amb les averies o alertes en una memòria extraïble, i es crearà una interfície gràfica amb una pantalla, on es podran observar els diferents valors obtinguts per els sensors instal·lats.S'utilitzaran sensors per a poder controlar i observar la humitat del terreny on es troben arrelats els tarongers, i d'aquesta manera es podrà emprar l'aigua d'una manera més eficient regant sols quan siga estrictament necessari.

Tots els sensors instal·lats es comunicaran amb el Arduino per mòduls de radio freqüència; l'alimentació del Arduino i la Rasberry pi la proporcionarà un panell solar i una bateria. S'instal·larà un mòdul GSM per a comunicar-se amb el propietari del hort immediatament a traves del telèfon mòbil quan es produïsca una alerta o averia.

Paraules clau: Arduino, Raspberry pi, humitat, hort i automatització

ABSTRACT

This project will focus on the control of irrigation for an orchard of orange trees with a Raspberry pi and an Arduino which will automate everything related to irrigation, as well as the automation of the information about the breakdowns. In addition, all data related to the breakdowns or alerts will be saved on a memory stick, and will create a graphic interface with a screen, where you can see the different values obtained by installed sensors.

Sensors are used to control and observe the soil moisture where orange trees are rooted and in this way the water in a more efficient way can be used watering only when it is strictly necessary.

All installed sensors will communicate with the Arduino by radio frequency modules; the power of the Arduino and the raspberry pi are will provide by a solar panel and a battery.

A GSM module will be installed to communicate with the owner of the garden immediately via mobile phone when there is an alert or a breakdown.

Key words: Arduino, Raspberry pi, moisture, orchard and automation.

Índice

Índice de Figuras

Índice de Tablas

1. Introducción

1.1. Antecedentes

Después de investigar el mercado y buscar la existencia de un control remoto de regadío comercial, que posera las mismas características que el desarrollado y mostrado en el presente Trabajo Fin de Grado y no encontrar ninguno que se comercializase, se decidió llevar a cabo el presente trabajo.

Así pues, la falta de dispositivos electrónicos que se puedan manejar a través de Internet por el usuario mediante cualquier dispositivo móvil con conexión a la red, bien sean Smartphones, Tablets u ordenadores, más allá de los simples ordenadores de control de regadío, nos ha llevado a desarrollar el siguiente dispositivo.

Este dispositivo permite al agricultor controlar desde cualquier lugar y sin necesidad de desplazarse al campo de cultivo las características del mismo como puedan ser la temperatura ambiental, la humedad del aire, la humedad del terreno y la posibilidad de encender y apagar el motor de regadío cuando sea necesario y el cultivo precise de agua.

Dada la falta de inversión en el sector primario y la escasez de tecnología avanzada se ha considerado necesario el desarrollo del siguiente proyecto para llegar a las personas que se dedican al cultivo de fruta y hortaliza a pequeña y gran escala y deseen compatibilizar el cultivo de los campos familiares con otro tipo de actividad laboral.

La agricultura a nivel global acumula prácticamente tres décadas de infrainversión al tiempo que tiene que hacer frente al reto mundial de alimentar más y mejor a la población creciente. Según el Instituto Nacional de Estadística en el primer trimestre del año 2018 solamente un total de 833.800 personas se encuentran trabajando en el sector agrícola español.

El rápido crecimiento económico y social de muchos países emergentes está cambiando la forma de alimentarse de la población, de manera que no únicamente hay que alimentar a más población, sino que en general las personas desean comer mejor, más variado y con productos de alta calidad. Para todo esto habrá que incrementar la producción en un 70% según (Barua, 2017) cita en la World Resources Institute.

Aunque en muchos países en desarrollo todavía hay mucho margen de mejora en los ratios de productividad en términos de mejoras técnicas de cultivo, utilización de mejores fertilizantes y pesticidas, sistemas de riego más eficientes, o mayor maquinación del campo, en muchos países ya se ha entrado en una fase de estancamiento por parte del sector y urge buscar nuevas fórmulas para aumentar la rentabilidad de los cultivos.

Por ello, la monitorización digital de cultivos, la agricultura de precisión, la utilización de robots, drones y otras mejoras en el campo de la maquinaria, en el sector agrícola, están configurando lo que se empieza a llamar la Cuarta Revolución Industrial ya que conforman campos de investigación y desarrollo de gran importancia.

1.2. Objetivos

A continuación se detallan los objetivos del presente trabajo:

El objetivo principal es obtener toda la información posible relativa a las condiciones climáticas y ambientales de un campo de cultivo, para optimizar las horas y el tiempo de riego, así como automatizarlo y controlarlo de manera remota.

Los datos que principalmente se desean obtener son la temperatura y la humedad del aire, así como la humedad del terreno para saber cuándo hay que tomar decisiones en el campo tales como el riego.

El siguiente objetivo es controlar las bombas de regadío con la finalidad de encenderlas y apagarlas siempre que sea necesario de forma remota evitando así desplazamientos al campo.

Mediante la ejecución de este proyecto se pretende mejorar el cuidado y mantenimiento de un campo de cultivo, ya que, a través de este proyecto se reducirá la dedicación que un individuo ha de emplear en las visitas a sus campos para verificar que la tierra está en las condiciones óptimas para que los cultivos den sus frutos.

Así pues, con el uso del proyecto diseñado se pretende beneficiar tanto a los individuos como a los campos de cultivo haciendo que éstos últimos estén controlados 24/7 y los dueños puedan recibir la información para saber cuándo necesitan regar, o abonar.

Por otra parte, otro de los objetivos a alcanzar sería procesar y almacenar toda la información obtenida en una base de datos. Con esto se conseguirá observar y analizar los datos con más detalle, pudiendo posteriormente proporcionarlos a los usuarios para que puedan realizar previsiones de producción.

El siguiente objetivo es crear una intranet online para poder procesar los datos y proporcionar los gráficos pertinentes de la información obtenida. De este modo no solamente se podrá acceder a los datos obtenidos a través de la pantalla instalada junto con el equipo, sino que, se haría accesible la información desde cualquier dispositivo con conexión a internet mediante la creación de la citada intranet online.

1.3.¿Qué es Telegram?

Para la consecución de los objetivos citados anteriormente, es necesario abordar la explicación de Telegram y sus funciones ya que esta aplicación es clave para el desarrollo del presente trabajo.

Según (Telegram, 2018), Telegram es una aplicación de mensajería instantánea, que destaca por su alta seguridad, además de su rapidez y gratuidad. Así pues, en la web de la organización se detalla que la aplicación se puede utilizar en cualquier dispositivo, bien sean ordenadores, Smartphones o tablets, ofreciendo la posibilidad de sincronización de todos los mensajes al mismo tiempo en cualquier dispositivo que esté instalado. Además, la aplicación se encuentra disponible para todos los sistemas operativos que se encuentran en el mercado actualmente, siendo éstos, macOS, Linux y Windows en ordenadores y Adroid, Windows Phone, e IOS en Smartphones. También tiene disponible una versión web mediante la cual se puede acceder a la aplicación a través de un navegador web sin la necesidad de descargarla.

La compañía Telegram es una organización sin ánimo de lucro, que se creó en el año 2013 y posee su sede principal en Berlín. A parte de la aplicación que hemos descrito, la compañía ofrece plataformas para bots, una nube privada de 1,5GB por archivo, chats secretos, llamadas, canales de difusión, supergrupos de hasta 100.000 personas. Además, permite buscar a otros usuarios sin tener su número de teléfono mediante la búsqueda por nombre de usuario.

La *application programming interface* o API que es la interfaz de programación de aplicaciones de Telegram, es abierta y permite a los usuarios desarrollar sus propias aplicaciones de Telegram. También poseen una Bot API , que es una plataforma para desarrolladores que permite a cualquier persona crear fácilmente herramientas especializadas para Telegram, integrar cualquier servicio e incluso aceptar pagos de usuarios de todo el mundo.

Así pues, el presente proyecto ha podido ser desarrollado mediante el uso de la Bot API, con la que se ha desarrollado el bot que emite y recibe la información. El control del proyecto desarrollado se realiza a través de ésta API en la que se encuentra el bot de Telegram que ha sido programado previamente y ambos se encuentran en la Raspberry pi. Toda la información que recibe y emite el bot que está situado en la Raspberry pi, pasa previamente por un Arduino que será el encargado de obtener los datos y transmitirlos mediante una conexión serie USB.

2. Memoria

En el siguiente apartado se van a describir los materiales que se han utilizado para la realización del presente proyecto, distinguiendo los materiales que han sido necesarios para el desarrollo del hardware o partes físicas y las del software o programación.

2.1. Desarrollo hardware

Seguidamente se van a describir todos los componentes que han sido utilizados para la elaboración de la maqueta:

2.1.1. Tarjeta microSD

Se ha elegido una tarjeta microSD Sandisk Ultra de 32Gb de clase 10 que trabaja a una velocidad de transferencia máxima de 98MB/s. Dicha tarjeta se va a introducir en la Raspberry pi y en ella se almacenará el sistema operativo, las librerías y la API de Telegram.

Ilustración 1. Tarjeta microSD seleccionada.

2.1.2. Raspberry pi 3

Según (RaspberryPiWordMark, 2018) describe en su web, la Raspberry pi es un ordenador del tamaño de una tarjeta de crédito que se conecta a un monitor y a un teclado. Es un pequeño ordenador de bajo coste que se puede utilizar en proyectos de electrónica y muchas más cosas, como por ejemplo: hojas de cálculo, procesador de texto, reproducción de videos, navegación por internet y aprender a programar en diferentes lenguajes.

Se ha elegido la Raspberry pi 3 porque posee una gran comunidad de desarrolladores y es open software. Esto facilita encontrar en internet una gran cantidad de información sobre las funciones y la aplicabilidad de la placa (Baker, 2015)

En el presente proyecto se utiliza la Raspberry pi 3 como servidor. Ésta se encargará de captar y guardar toda la información procedente del Arduino maestro, que a su vez, recibirá la información procedente de los sensores mediante otro Arduino esclavo.

La Raspberry pi 3 se conecta al Arduino maestro por puerto serie USB, y almacenará el bot, cosa que nos permitirá interactuar con él y obtener información de los sensores y actuadores.

Por otra parte, se utilizarán los puertos *general-purpose input/output* o GPIO de la Raspberry pi para encender un relé, que será el encargado de poner en funcionamiento la bomba de regadío y pararla cuando sea necesario.

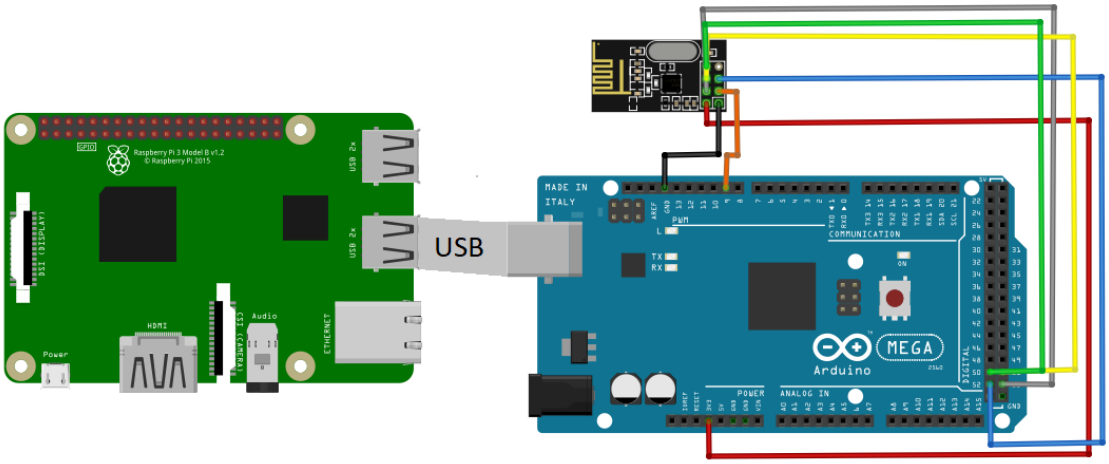


Ilustración 2. Conexión Raspberry pi 3 Arduino mediante USB

2.1.3. Arduino

Los arduinos seleccionados para la ejecución del proyecto han sido Arduino uno y Arduino mega. En concreto, el Arduino uno se ha utilizado como esclavo y se encargará de interpretar y enviar los datos del campo de cultivo, y el Arduino mega se ha utilizado como maestro, y se encargará de recibir y trasmitir los datos a la Raspberry pi mediante conexión serie USB.

Se han utilizado estos arduinos porque son los que se han adquirido, pero realmente se podría haber hecho con cualquiera de los arduinos que se comercializan actualmente.

Seguidamente se detallan las características de cada uno de los arduinos utilizados según (Arduino, 2018) describe en su web, y podemos observar cada uno de los ardunios en la Figura 3 y en la Figura 4:

Características Arduino Uno	Características Arduino Mega
Microcontrolador Atmega328. Voltaje de entrada: 7-12V. Voltaje de trabajo: 5V. Corriente por pin I/O: 40mA. 14 pines digitales I/O 6 canales PWM 6 Entradas analogicas 16MHz de velocidad de reloj. Memoria Flash: 32 KB	Microcontrolador Atmega2560. Voltaje de entrada: 7-12V. Voltaje de trabajo: 5V. Corriente por pin I/O: 40mA. 54 pines digitales I/O 14 de ellos son salidas PWM 16 entradas analógicas 16Mhz de velocidad de reloj Memoria Flash: 256 KB

Tabla 1. Características Arduino Uno y Arduino Mega

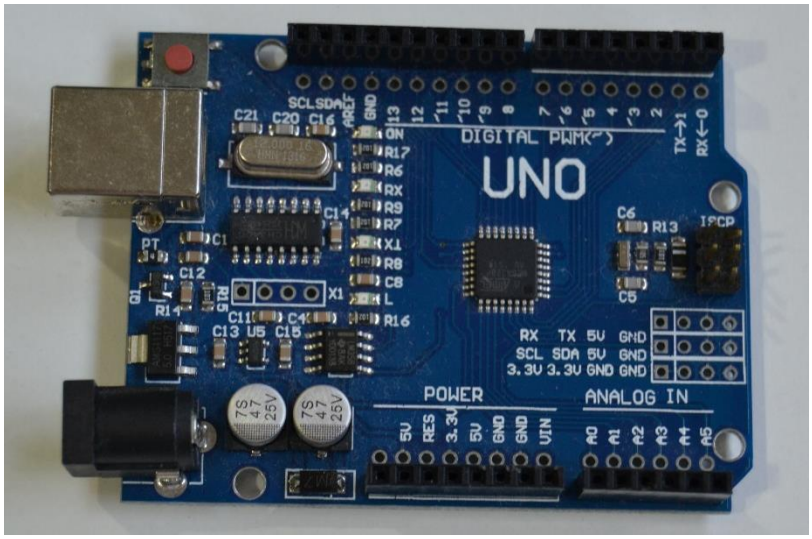


Ilustración 3. Arduino Uno

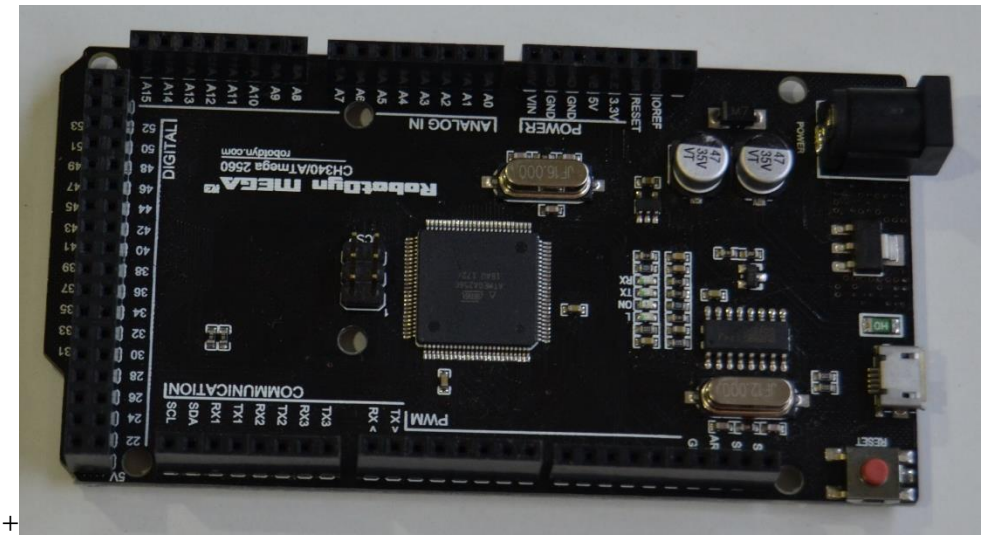


Ilustración 4. Arduino Mega

2.1.4. Sensores y módulos

Sensor humedad-temperatura del aire DHT22

El sensor de humedad y temperatura seleccionado es el DHT22, y éste funciona en un rango extenso. El sensor va conectado al Arduino uno, que es el esclavo, y le proporciona la información relativa a la humedad y la temperatura del aire. A continuación se describen las características del sensor que se puede observar en la Figura 5:

- Rango de temperatura; -40ºC a 125ºC con ± 0.5ºC
- Rango de humedad; 0% al 100% con 5% de precisión
- Realiza 2 muestras por segundo
- Alimentación a 3,3V

Más adelante, en el apartado titulado 2.1.5. Esquema de conexionado se explica cómo se conecta este sensor.

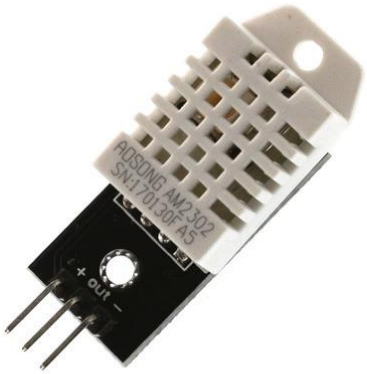


Ilustración 5. Sensor DHT-22

Sensor humedad del terreno FC-28

El sensor de la humedad del terreno que irá colocado en la tierra, incluye un módulo que se encarga de entregar la medición al Arduino de forma analógica o digital. La placa Arduino contiene un convertidor analógico a digital de 6 canales, 8 canales en Mini y Nano, y 16 en la Mega. Esto significa que mapeará voltajes de entrada entre 0 y 5 voltios en valores enteros entre 0 y 1023. Esto produce una resolución entre las lecturas de: 5 voltios / 1024 unidades o, 0,0049 voltios (4.9 mV) por unidad. El ajuste analógico funciona desde 0 a 1023, dónde 0 significaría que está sumergido en agua, por tanto, muy húmedo, y el 1023 significaría que está muy seco, por ejemplo el desierto.

Se necesitan aproximadamente 100 microsegundos (0,0001 s) para leer una entrada analógica, por lo que la velocidad máxima de lectura es aproximadamente 10.000 veces por segundo.

El ajuste digital se realiza mediante el potenciómetro que lleva el módulo del sensor. Una vez realizado el ajuste digital obtendremos o una señal LOW (nivel bajo) cuando el suelo no esté húmedo o una señal que indique HIGH (nivel alto) cuando esté húmedo. La señal oscilará entre los valores de 0 a 5 voltios.

Tal y como se ha nombrado anteriormente, en el apartado titulado 2.1.5. Esquema de conexionado se explica cómo se conecta este sensor.

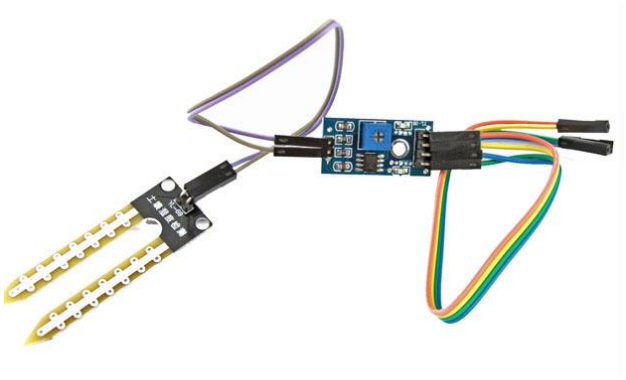


Ilustración 6. Sensor FC28

Modulo radio frecuencia NRF24L0

Este módulo realiza la emisión o recepción de los datos. En total, en el presente trabajo se han utilizado dos. Uno está colocado en el Arduino Uno y el otro en el Arduino Mega para conseguir la comunicación entre ambos. En la Figura 7 podemos ver el módulo utilizado y las especificaciones de éste son las siguientes:

- Alimentación entre 1,9 V y 3,3V
- Consumo de corriente 15 mA
- Tasa de transmisión +20 dBm
- Alcance del transmisor aproximado 1000 m
- Banda de 2,4 GHz
- Soporta hasta 6 canales de recepción

Así pues, la conexión del mismo se explicará en el apartado titulado 2.1.5. Esquema de conexionado.

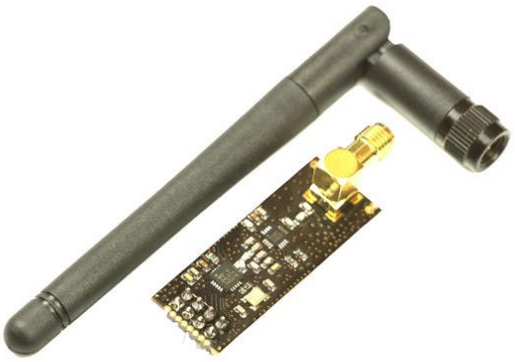


Ilustración 7. Módulo NRF24L0

Modulo relé

Éste módulo va conectado a la Raspberry pi y enciende o apaga la bomba o motor de regadío. Las características del relé son las siguientes:

- Alimentación de excitación 5V
- Puede ser activado por nivel alto o bajo
- 1 contacto normalmente abierto y 1 contacto normalmente cerrado
- 10A a 250 V AC
- 10A a 30V DC
- 10A a 125V AC
- 10A a 24V DC

En el apartado 2.1.5. Esquema de conexionado se explicará la conexión de éste módulo.

Ilustración 8. Módulo relé

2.1.5. Esquema conexionado

En el siguiente apartado se van a detallar las conexiones de los sensores y los módulos a las placas de arduino, Arduino Uno y Arduino Mega, y a la Raspberry pi.

2.1.5.1. Arduino Uno

Primero se van a detallar las conexiones del módulo NRF 24L0, el Sensor FC28 y Sensor DTH22 al Arduino Uno o esclavo.

Módulo de radio frecuencia NRF 24L0

Pin	NRF24L0	Arduino UNO
GND	1	GND
VCC	2	3,3V
CE	3	9
CSN	4	10

SCK	5	13
MOSI	6	11
MISO	7	12

Tabla 2. Conexiones NRF 24L0

Sensor de humedad del terreno FC28

Pin	FC 28	Arduino UNO
A0	A0	A0
5V	VCC	VCC
GND	GND	GND

Tabla 3. Conexiones FC28

Sensor de temperatura y humedad del aire DTH22

Pin	DHT 22	Arduino UNO
D2	OUT	2
3.3V	+	3.3V
GND	-	GND

Tabla 4. Conexiones DHT22

A continuación, en la Figura 9 se puede observar cómo quedan las conexiones realizadas en el Arduino Uno. Se detallan en el esquema los sensores y el módulo que se han instalado. El módulo de radio frecuencia NRF 24L0 es el de color negro, el sensor DTH22 de temperatura y humedad del aire es el de color blanco, y abajo se encuentra el sensor de humedad del terreno que es de color azul oscuro. Todos ellos van conectados al Arduino Uno que es la placa más grande de color azul claro.

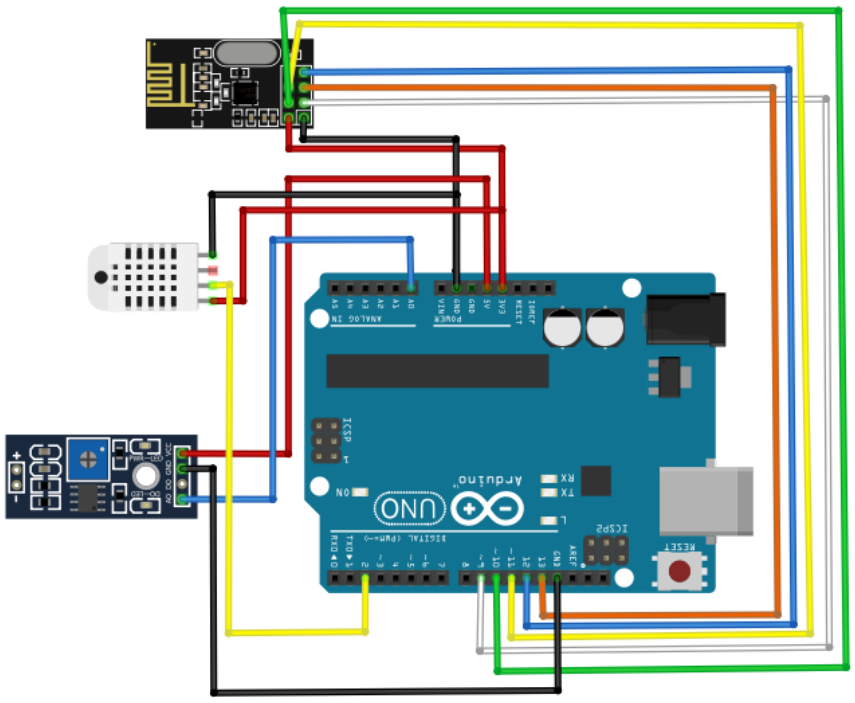


Ilustración 9. Esquema de conexiones Arduino Uno

2.1.5.2. Arduino mega

A continuación se va a detallar la conexión del módulo NRF 24L0 al Arduino Mega o maestro.

Módulo radio frecuencia NRF 24L0

Pin	NRF24L0	Arduino UNO
GND	1	GND
VCC	2	3,3V
CE	3	9
CSN	4	53
SCK	5	52
MOSI	6	51
MISO	7	50

Tabla 5. Conexión NRF24L0

En la Figura 10 se puede observar la conexión del módulo de radio frecuencia NRF 24L0 con el Arduino Mega. El Arduino Mega es la placa de color azul claro más grade, y el módulo de radio frecuencia NRF 24L0 la placa de color negro.

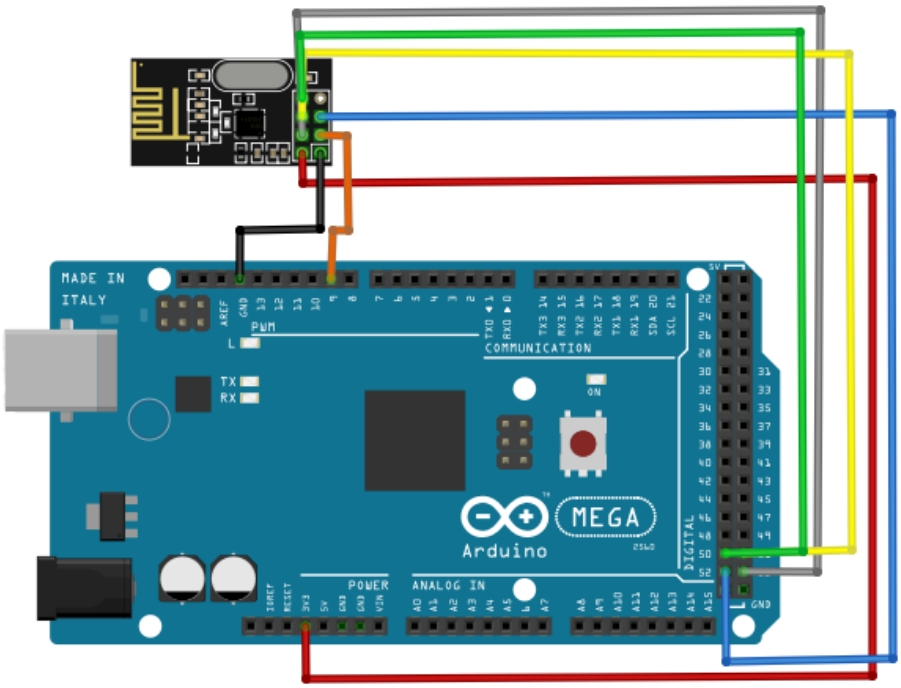


Ilustración 10. Conexiones Arduino Mega.

2.1.5.3. Raspberry pi

Seguidamente se detalla la conexión del módulo relé a la Raspberry pi.

Módulo relé

En la Figura 11 se puede observar la conexión del módulo relé, dónde el cable rojo que es el positivo, salida GPIO 17 de la Raspberry pi irá conectado al A1 del relé y el cable negro GND de la Raspberry pi irá al A2 del relé.

Los otros contactos NC, NO y Común son los que van conectados al motor de regadío o bomba.

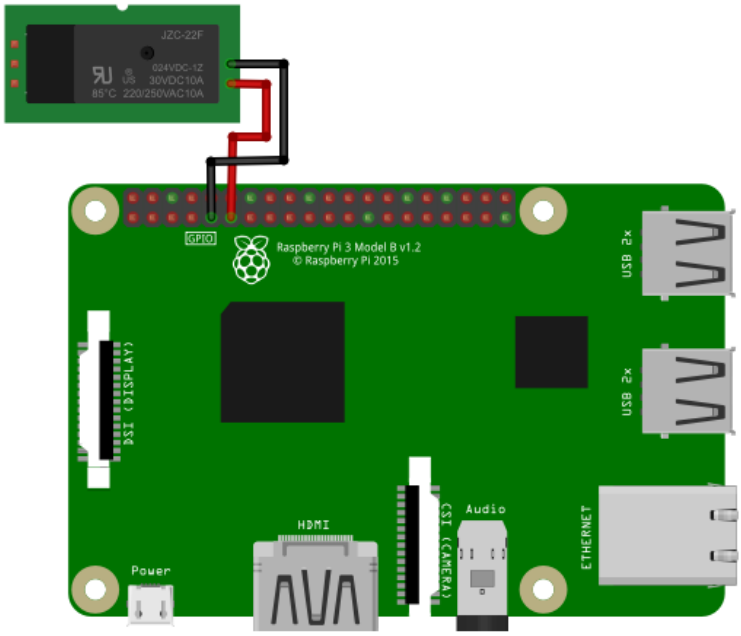


Ilustración 11. Conexión módulo relé

2.2. Desarrollo software

Seguidamente se van a describir las aplicaciones y los comandos utilizados para la programación:

2.2.1. Obtención del token y del bot

Los bots son unas cuentas especiales que se pueden crear en la aplicación Telegram y que no requieren de un número de teléfono adicional para configurarlo, es decir, se pueden crear como una cuenta a parte de la personal. El token se define como un número identificativo único para cada bot.

A continuación se describen los pasos para la creación de un bot y la obtención del token:

Primero, accedemos a la aplicación de Telegram, que se ha descargado previamente, y en el buscador vamos al símbolo de la lupa para buscar el bot que permite la generación de nuestro propio bot. Éste bot es el que aparece en la aplicación como BotFather, y es el bot oficial de Telegram que nos permitirá generar nuestro bot.

Ilustración 12. Búsqueda del bot

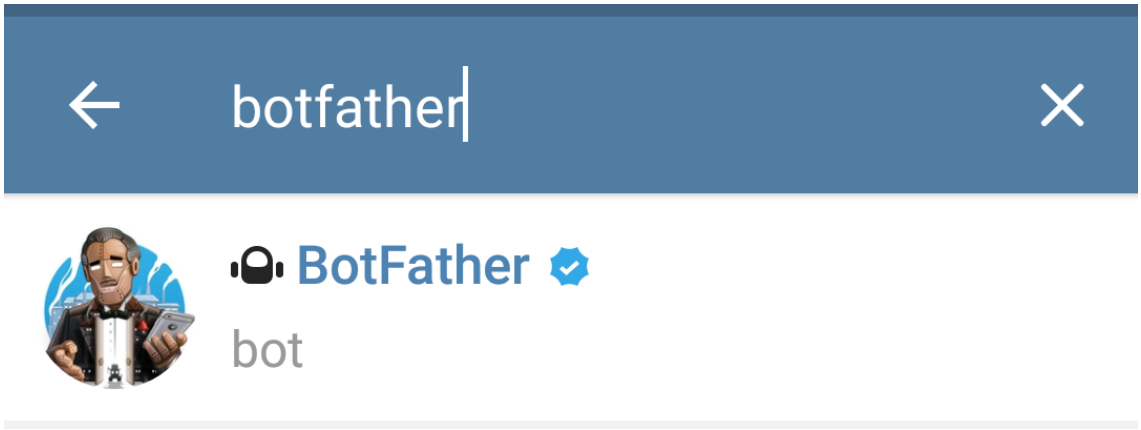


Ilustración 13. Bot oficial de Telegram

Una vez obtenida la búsqueda del BotFather en Telegram se inicia una conversación con el bot para proceder a la creación del bot propio.

En la conversación se nos proporcionarán una lista de comandos, y se debe de pulsar el siguiente:

```
/newbot
```

El comando seleccionado anteriormente realiza la creación de un nuevo bot, así como la creación automática de un token para el bot.

Una vez pulsado el comando, el BotFather pedirá que le asignemos un nombre a nuestro bot, que siempre ha de terminar con:

```
_bot
```

Una vez generado el nombre del bot, procederemos a crear un comando que cuando se lo mandemos a nuestro bot, éste se encenderá, y para ello tendremos que escribir el siguiente comando:

```
/setcommands
```

Introducido el comando anterior, para que nuestro bot se encienda, crearemos nuestro comando de iniciación al que llamaremos:

```
/start
```

Una vez escrito el comando de iniciación `/start`, se lo enviamos al BotFather. Y a continuación, solamente queda que le pidamos el token del bot generado. Para ello, se escribirá el siguiente comando:

```
/token
```

Seguidamente el BotFather nos pedirá que seleccionemos el nombre del bot del cual deseamos obtener el token. Y una vez seleccionado el token que nos proporcionará será similar al siguiente:

```
498559005:AAFSmj_0XRR_tQBRXVHmRz7B6G4hpFbmfoU
```

2.2.2. Instalación del sistema operativo de la Raspberry pi

Para empezar y poder hacer funcionar la Raspberry pi se deberá de descargar el sistema operativo de la Raspberry pi Sketch Desktop de la web oficial de (Raspberry pi, s.f.), que está basado en Linux. Una vez descargado el sistema operativo, éste se introducirá en la tarjeta microSD.

Para ello, primero se procederá a darle formato a la tarjeta microSD, que se formateará con el programa SD card formater que se puede obtener en (SD Memory Card Formatter, s.f.).

Seguidamente ejecutamos el programa SD card formater como administrador, y una vez arrancado el programa, seleccionaremos el puerto donde se encuentra la microSD conectada al PC tal y como se puede observar en la Figura 14.

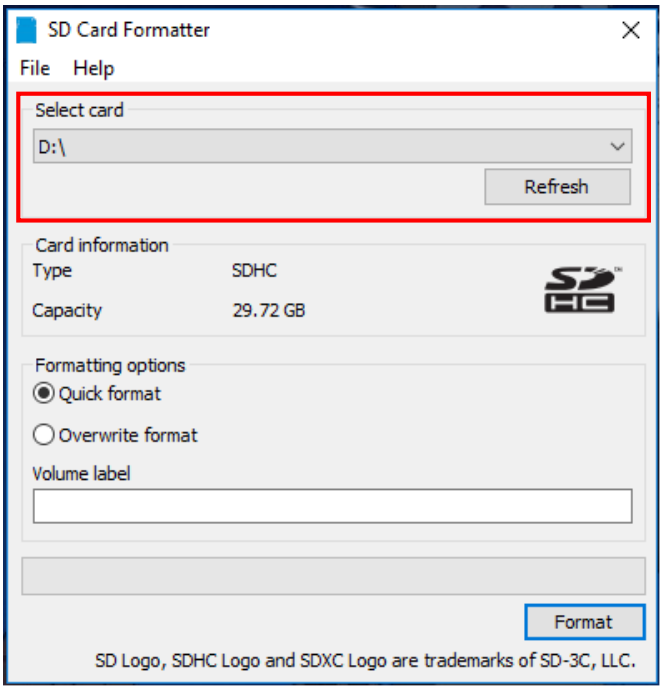


Ilustración 14. Selección del puerto

La siguiente opción que se tiene que configurar es la velocidad del formateo. Por defecto viene seleccionada la casilla de formateo rápido o Quick format tal y como podemos observar en la Figura 15, y ésta es la que dejaremos seleccionada. Al terminar de configurar la velocidad del formateo, ejecutaremos el formateo, y una vez terminado, podremos montar nuestro sistema operativo.

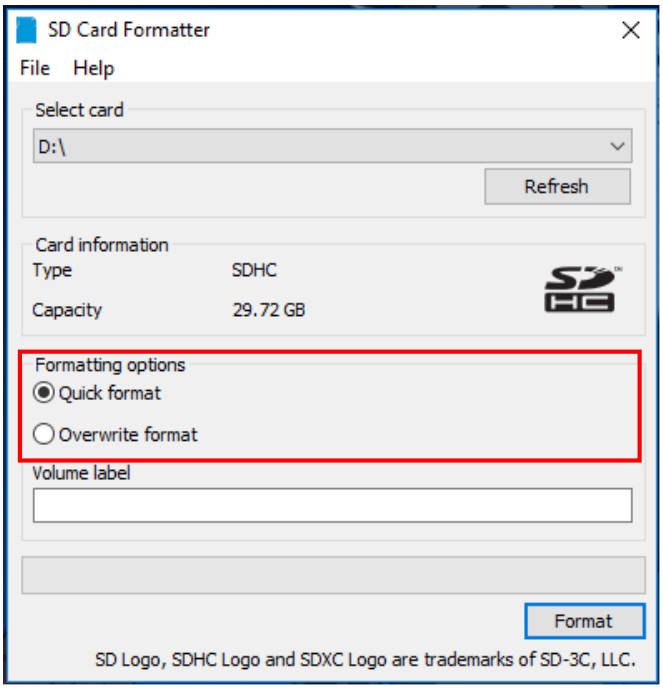


Ilustración 15. Tipo de formateo

Una formateada la tarjeta microSD, se procederá a instalar el sistema operativo descargado anteriormente en la misma. Se utilizará el programa Win32diskmager, que se podrá descargar de (win32diskimager, s.f.) para poder instalar el sistema operativo en la tarjeta.

Una vez iniciado el programa Win32diskmager, en la primera opción que aparece deberemos seleccionar donde se encuentra la imagen o sistema operativo descargado de la web de (Raspberry pi, s.f.) tal y como se muestra en la Figura 16, así pues, también se debe de seleccionar el puerto donde se encuentra la tarjeta microSD conectada.

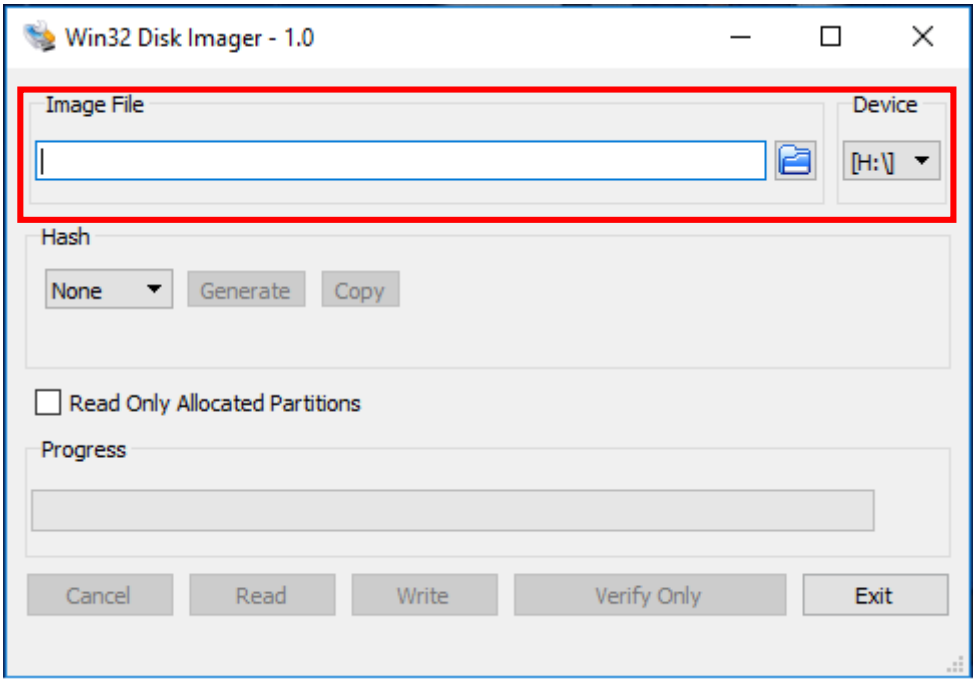


Ilustración 16. Selección de la imagen y del puerto

Posteriormente se montará la imagen descargada en la tarjeta microSD pulsando sobre el botón del programa que pone "Write" y que se encuentra en la parte inferior central como podemos ver en la Figura16.

Finalmente, el sistema operativo quedará instalado en la tarjeta microSD una vez desarrollados todos los pasos detallados en el presente punto. Seguidamente, se extraerá la tarjeta del PC y se introducirá en la Raspberry pi pudiendo ya operar con la misma.

2.2.3. Instalación de librerías en la Raspberry pi

Antes de explicar el proceso que hay que seguir para la instalación de las librerías en la Raspberry pi, conviene destacar que se puede programar la Raspberry pi mediante conexión ssh a través del programa Puty o conectando un monitor, un ratón y un teclado a la placa. De la última forma solamente se podrá trabajar si el sistema operativo que se ha instalado en la tarjeta microSD tiene un entorno gráfico o escritorio. En nuestro caso, el sistema operativo tiene escritorio y por ello se ha trabajado de las dos maneras. Los comandos descritos a continuación, se han introducido mediante ssh.

Primero, se actualizarán todos los repositorios de la Raspberry pi mediante la utilización del siguiente comando:

```
sudo apt-get -y update
```

Una vez terminada la actualización, se instalará la API de Telegram y las librerías necesarias que se van a detallar a continuación.

Se procederá a la instalación del módulo "pip", que facilitará la instalación posterior de otros paquetes. El comando para la instalación es el siguiente:

```
sudo apt-get install python3-pip
```

Una vez realizada la instalación del módulo “pip” se procederá a la instalación de la API de Telegram de la siguiente forma:

```
sudo python3 -m pip install python-telegram-bot
```

A continuación, instalamos la librería serie que es la que se encarga de la comunicación entre el Arduino y la Raspberry pi mediante conexión por puerto serie USB.

```
sudo apt-get install python3-serial
```

En el sistema operativo instalado tenemos la librería instalada de Python 3 que es la que vamos a utilizar; en caso de no tenerla instalada, procederemos a instalarla con el siguiente comando:

```
sudo apt-get install python3
```

Una vez llegados a este punto, comprobaremos si hay actualizaciones disponibles, poniendo el comando detallado aquí abajo, que debería actualizar todo lo que se ha instalado anteriormente si fuese necesario:

```
sudo apt-get -y upgrade
```

En el comando anterior, la función de la letra “-y” es omitir la pregunta de si se quiere realizar la actualización o no, ejecutándola directamente.
Una vez instaladas todas las librerías detalladas anteriormente, crearemos una carpeta donde almacenaremos nuestro bot. Para crear la carpeta se tendrán de poner los siguientes comandos según (RASPBERRY PI WEB INFO):

```
mkdir Nombre de la carpeta
```

El acceso a la carpeta se realiza de la siguiente forma, tal y como se describe en (RASPBERRY PI WEB INFO):

```
Cd Nombre de la carpeta
```

Una vez accedemos a la carpeta tenemos de crear el bot, tal y como se describe en (RASPBERRY PI WEB INFO). Como posteriormente se va a programar en Python 3, deberemos ponerle el nombre a nuestro archivo acabado con”**.py”**

```
sudo nano Nombre del archivo.py
```

Una vez ejecutados los pasos anteriores, se abrirá una ventana donde se escribirá el código del bot para programarlo. Más adelante, en el punto 2.2.4 Programación bot Raspberry pi se presenta el programa.

Por otra parte, se procederá a la instalación de la base de datos donde se almacenarán todos los valores obtenidos desde el campo de cultivo.

Siguiendo la página web de (PROMETEC, s.f.) se han instalado las siguientes librerías para poder instalar la base de datos.

En primer lugar se instalará el programa Apache:

```
sudo apt-get install apache
```

Después se instalará MySQL, un sistema de gestión de datos:

```
sudo apt-get install mysql-server mysql-client
```

Una vez instalados los programas Apache y MySQL, se debe reiniciar la Raspberry pi. Una vez esté reiniciada, accederemos al navegador que tengamos instalado en la Raspberry pi para acceder a la base de datos; se puede acceder a la base de datos poniendo en la URL del navegador:

```
127.0.0.1 o localhost
```

Más adelante, en el punto 2.2.5 Programación de la base de datos se presenta la programación de la base de datos.

2.2.4. Programa Bot Raspberry pi

```
1. import telegram    #Definición de la librería de Telegram
2. import serial      #Definición de la librería Serie
3. import time        #Definición de la librería Tiempo
4. import logging
5. import RPi.GPIO as GPIO #Definición de la librería de los
   puertos GPIO

6. from telegram import (ReplyKeyboardMarkup,
   ReplyKeyboardRemove)
7. from telegram.ext import(Updater, CommandHandler,
   MessageHandler, Filters, RegexHandler, ConversationHandler)

8. ardu = serial.Serial('/dev/ttyUSB0', 9600) #Configuración vía
   serie con el Arduino
9. token = "452734232:AAGvuCt7bP3RF9aa-ydsRuEbD7cYpQSBZ_4"
   #Número de identificación del bot
10.   temperatura=[]        #Guardar valor de la variable
11.   humedad=[]       #Guardar valor de la variable
12.   hterreno=[]           #Guardar valor de la variable

13.   logging.basicConfig(format='%(asctime)s - %(name)s -
   %(levelname)s -%(message) -%(chat_id)', level=logging.INFO)
14.   ELIGE, SALIR, ATRAS = range(3)
15.   logger = logging.getLogger(__name__)
16.   reply_keyboard = [['Temperatura y
   Humedad'],['Motor'],['Estado del motor'],['Salir']]
17.   salir_keyboard = [['Salir']]
18.   motor_keyboard = [['On'], ['Off'],['Atras']]
19.   atras_keyboard = ['Atras']
20.   GPIO.setmode(GPIO.BCM)
21.   GPIO.setup(17,GPIO.OUT)     #Definición del pin que
   encenderá la bomba de regadío
22.   def start(bot, update):     #Definición del comando Start
23.   update.message.reply_text('Buenas se ha pulsado el comando
   /start se desplegara el menu \n\n'
                  +'Resumen de los botones; \n\n'
                  +'Temperaura y Humeda: muestra la temperatura
                  y la humedad del terreno y aire \n\n'
```

```
                    +'Bomba: Para encender y apagar la bomba de
                    regadio \n\n'
                    +'Estado de la bomba: saber si está encendida
                    o apagada la bomba \n\n'
                    +'Salir: Desactivamos el bot, no se resetean
                    las variables sino se quedan guardadas con el
                    ultimo valor',
        reply_markup=ReplyKeyboardMarkup(reply_keyboard,
        one_time_keyboard=False))

        #Mensaje de bienvenida e información de lo que realizan
        los comandos

24.   return ELIGE
   #En el siguiente comando al pulsar el botón Temperatura y
   Humedad en la pantalla de telegram mandara la información al
   arduino maestro para que capte la información del arduino
   esclavo y que nos responda.
25.   def temp1(bot, update):
26.   ardu.write(b'T')
27.   temperatura = ardu.readline()
28.   ardu.write(b'H')
29.   humedad = ardu.readline()
30.   ardu.write(b'A')
31.   hterreno = ardu.readline()
32.   update.message.reply_text('La temperatura es de {} °C
   '.format(temperatura.decode('ascii',errors='replace'),end='')
   ,
33.   reply_markup=ReplyKeyboardMarkup(reply_keyboard,
   one_time_keyboard=False))
34.   update.message.reply_text('La humedad del aire es del  {} %
   '.format(humedad.decode('ascii',errors='replace'),end=''),
35.   reply_markup=ReplyKeyboardMarkup(reply_keyboard,
   one_time_keyboard=False))
36.   update.message.reply_text('La humedad del terreno es del
   {}
   %'.format(hterreno.decode('ascii',errors='replace'),end=''),
37.   reply_markup=ReplyKeyboardMarkup(reply_keyboard,
   one_time_keyboard=False))
38.   return ELIGE
   #El comando cancelar es el que se encarga de apagar el bot.
39.   def cancel(bot,update):
40.   update.message.reply_text('Saliste del teclado, volver al
   teclado introduzca  el comando /start',
   reply_markup=ReplyKeyboardRemove())
41.   return ConversationHandler.END
   #Al pulsar sobre el botón motor abre un submenú con dos
   opciones encender o apagar el motor.
42.   def motor(bot,update):
43.   update.message.reply_text('Encienda o pague el motor el
   motor se encuentra :',
```

```
        reply_markup=ReplyKeyboardMarkup(motor_keyboard,
        one_time_keyboard=False))
44.   return ELIGE
   #Boton del menú ON, al pulsar el botón el motor se encenderá.
45.   def motoron(bot,update):
46.   f=open("k.txt","w")   #Creamos un archivo .txt y escribimos
47.   f.write("1") #Se escribe un 1 si encendemos el motor
48.   f.close()  #Cerramos el fichero que habíamos creado
49.   GPIO.output(17, True) #Activamos el pin 17 de la raspberry
   pi
50.   update.message.reply_text('Motor encendido
   ',reply_markup=ReplyKeyboardMarkup(motor_keyboard,
   one_time_keyboard=False))
51.   return
   #Botón del menú OFF, al pulsar el botón se apaga el motor
52.   def motoroff(bot, update):
53.   f=open("k.txt","w")   #Abrimos el archivo creado
   anteriormente y escribimos
54.   f.write("0")    #Se escribe un 0 si apagamos el motor
55.   f.close()  #Cerramos el fichero que habíamos creado
56.   GPIO.output(17, False) #Desactivamos el pin 17 de la
   raspberry pi
57.   update.message.reply_text('Motor apagado
   ',reply_markup=ReplyKeyboardMarkup(motor_keyboard,
   one_time_keyboard=False))
58.   return
   #Botón estado, al pulsar el botón informara del estado en que
   se encuentra el motor si es ON o OFF
59.   def estado(bot, update):   #Comprobación del estado de la
   bomba
60.   f=open("k.txt","r")   # Abrimos el archivo creado
   anteriormente y lo leemos
61.   a=f.read() #Leemos el fichero y lo almacenamos en la
   variable "a"
62.   f.close()  #Cerramos el fichero
63.   if int(a) == 1: #Comparación de encendido o apagado de la
   bomba
64.   bot.send_message(chat_id=update.message.chat_id,
        text="La bomba está ENCENDIDA")
        #Mensaje que nos avisa de que el motor esta encendido
65.   if int(a) == 0: #Comparación de encendido o apagado de la
   bomba
66.   bot.send_message(chat_id=update.message.chat_id,
        text="La bomba está APAGADA")
        #Mensaje que nos avisa de que el motor esta encendido
67.   def error(bot, update, error):
68.   logger.warning('Update "%s" caused error "%s" ', update,
   error)
69.   updater= Updater(token)
70.   dp = updater.dispatcher
71.   conv_handler = ConversationHandler(
```

```
72.   entry_points=[CommandHandler('start', start)],  #Comando
   que inicializa el teclado
   #Estructura del teclado, nombre que se muestra en el teclado
   y la variable que se le ha asignado
73.   states={ELIGE: [RegexHandler('^(Temperatura y Humedad)$',
   temp1),
               RegexHandler('^(Salir)$', cancel),
               RegexHandler('^(Motor)$',motor),
               RegexHandler('^(On)$', motoron),
               RegexHandler('^(Off)$', motoroff),
               RegexHandler('^(Estado del motor)$', estado),
               RegexHandler('^(Atras)$', start)],

         SALIR: [RegexHandler('^(Salir)$', cancel)],
         ATRAS: [RegexHandler('^(Atras)$', start)],
         },
74.   fallbacks=[RegexHandler('^(Salir)$', cancel)]
75.   )
76.   dp.add_handler(conv_handler)
77.   dp.add_error_handler(error)
78.   updater.start_polling()
79.   updater.idle()
```

2.2.5. Programación de la base de datos

```
1. import serial  #Importamos la librería comunicación serie
2. import time    #Importamos la librería Tiempo
3. import MySQLdb #Importamos la librería de la base de datos
4. ardu = serial.Serial('/dev/ttyUSB0', 9600)    #Puerto de
   comunicación del arduino
5. db =
   MySQLdb.connect("localhost","pi","raspberry","Temperaturas")
      #Dirección donde se encuentra la base de datos

6. while True: #Enpieza el buble al ser verdadero y no termina
   nunca
7.  time.sleep(1800)  #Cuenta 30min pero en segundos
8.  ardu.write(b'T')  #Envía la letra T por el puerto serie
9.  temperatura = ardu.readline() #Lee el valor respondido del
   arduino por el puerto serie
10.    ardu.write(b'H') #Envía la letra H por el puerto serie
11.    haire = ardu.readline()     #Lee el valor respondido del
   arduino por el puerto serie
12.    ardu.write(b'A') #Envía la letra A por el puerto serie
13.    hterreno = ardu.readline()  #Lee el valor respondido del
   arduino por el puerto serie
14.    fecha = time.strftime("%d/%m/%y")+"
   "+time.strftime("%H:%M:%S")  #Añadimos en la base de datos
   día, mes, año, hora, minuto y segundo
```

```
15.    #print(fecha)
16.    cursor = db.cursor()
17.    sql = 'INSERT INTO
   valores(fecha,haire,hterreno,temperatura) VALUES
   ("%s","%s","%s","%s")'%(fecha, haire, hterreno, temperatura)
   #Añadimos a la base de datos los datos obtenidos de los
   sensores
18.    #print(sql+"\n")
19.    cursor.execute(sql)
20.    db.commit()
21.   #db.close()
```

2.2.6. Programación receptor Arduino Mega

```
1. #include <SPI.h>         //Definimos la librería
2. #include <nRF24L01.h>    //Definimos la librería del módulo RF
3. #include <RF24.h>        //Definimos la librería del módulo RF
4. const int pinCE = 9;     // Definimos el pin 9 como constante
5. const int pinCSN = 53;   // Definimos el pin 53 como constante
6. RF24 radio(pinCE, pinCSN);
7. const uint64_t pipe = 0xE8E8F0F0E1LL;
8. float datos[3];    //Creamos un array con los datos de los
   sensores
9. unsigned int dato;
10.   void setup()
11.   {
12.   radio.begin();              //Activamos la comunicación
13.   Serial.begin(9600);         //Activamos el puerto serie
14.   radio.openReadingPipe(1, pipe); //Abrimos el canal de la
   radio y leemos los valores
15.   radio.startListening();
16.   }
17.   void loop()
18.   {
19.   if (radio.available())
20.   {
21.   radio.read(datos,sizeof(datos));
22.   Serial.print(datos[0]);
23.   if (dato == 'T') Serial.print(datos[0],"°C");
24.   Serial.println("");
25.   }
26.   delay(1000);                //Esperamos 1 segundo
27.   }
```

2.2.7. Programa emisor Arduino Uno

```
1. #include <SPI.h>   //Definimos la librería de comunicación
   SPI
2. #include <nRF24L01.h>    //Definimos la librería del módulo RF
3. #include <RF24.h>        //Definimos la librería del módulo RF
4. #include "DHT.h"   //Definimos la librería del sensor dht22
```

```
#define DHTPIN 2   //Definimos el Pin 3 para la lectura del sensor.
#define DHTTYPE DHT22         //Definimos el Tipo de Sensor.
DHT dht(DHTPIN, DHTTYPE);     //Creamos un objeto dht con las variables definidas anteriormente.
const int pinCE = 9;
const int pinCSN = 10;
  RF24 radio(pinCE, pinCSN);
  const int sensorPin = A0;       // Sensor de humedad del terreno
  // Single radio pipe address for the 2 nodes to communicate.
  const uint64_t pipe = 0xE8E8F0F0E1LL;
  float datos[3];
  void setup()
  {
  radio.begin();              //Activa el modulo radio
  radio.openWritingPipe(pipe);
  }
  void loop()
  {
  int h= map(analogRead(sensorPin), 0, 1023, 100, 0);   // Comprueba el porcentaje de humedad tiene si un 100% o un 0% según su rango de 1023 a 0
  datos[0]=(dht.readTemperature());     //Almacenamiento del dato de temperatura
  datos[1]=(dht.readHumidity());  //Almacenamiento del dato de humedad aire
  datos[2]=(h);               //Almacenamiento del dato de humedad terreno
  radio.write(datos, sizeof(datos));
  }
```

2.2.8. Diseño de la página web

A continuación se van a detallar los pasos que se han seguido para diseñar la página web en la que se presentarán los datos gráficamente. A la página web diseñada solamente se puede acceder desde una intranet. La información se presenta de forma gráfica y se actualiza cada 30 minutos a partir de la base de datos creada con MSYQL.

En primer lugar, se creará el programa que enlaza la interfaz gráfica con la base de datos. Dicho programa se titula Index.php y se describe en el siguiente apartado titulado de la misma forma.

Lo que realiza el programa es la representación de una gráfica con las siguientes características, temperatura del aire, humedad del aire y humedad del suelo, dicha grafica se crea utilizando una de las API de Google que se pueden encontrar en Internet.

La programación de la intranet se encuentra escrita en PHP y HTM es una intranet básica.

En el código escrito se filtran los datos obtenidos por los sensores y eliminando los valores que no son de interés.

En el siguiente punto llamado 2.2.9. Index.php se puede encontrar el código que se ha utilizado.

2.2.9. Index.php

```
1. <?php
2. $con=mysqli_connect("localhost","pi","raspberry","Temperaturas"); //Para conectar con la base de datos se tiene que tener acceso como administrador se tiene de poner en las comillas donde se encuentra la base de datos, usuario, contraseña, tabla de la base de datos donde estan los valores
3. if (mysqli_connect_errno())

4. {
5. echo "Failed to connect to MySQL: " . mysqli_connect_error();
6. }
7. $sql="SELECT * FROM valores"; // Consulta sql, con ella recuperamos la informacion de la tabla valores de la base de datos
8. $result=mysqli_query($con,$sql); // ejecutamos la consulta
9. $tiempo = 0;
10.   ?>
11.   <html>  <!--Este apartado del codigo es la api de google.-->
12.   <head>
13.   <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

14.   <script type="text/javascript">
15.   google.charts.load('current', {'packages':['corechart']});
16.   google.charts.setOnLoadCallback(drawChart);
```

```
17.  function drawChart() {
18.  var data = google.visualization.arrayToDataTable([
19.  ['Tiempo', 'haire', 'hterreno', 'Temperatura'],

20.  <?php
21.  while($row=mysqli_fetch_array($result))
22.  {
23.  $tiempo++; // Esta es la variable que se le suma 1 en cada
     pasada del bucle para que al superar el número de registros
     que recuperamos en la sql el bucle termine
24.  if($tiempo<=24) // en esta condicion solo permites hasta un
     maximo de 24
25.  {
26.  echo "['" .substr($row["fecha"],9). "',".
     substr(substr($row["haire"],2),0,-1) . " , ".
     substr(substr($row["hterreno"],2),0,-1) . " , ".
     substr(substr($row["temperatura"],2),0,-1) . "],";
27.  } // Aquí mostramos los datos
28.  }
29.  mysqli_free_result($result); // liberas los resultados para
     que no ocupen memoria
30.  mysqli_close($con); // cierras la conexión
31.  ?>
32.  ]);
33.  var options = {  // esto son opciones configurables de la
     gráfica de google
34.  title: 'Grafica',
35.  curveType: 'function',
36.  legend: { position: 'bottom' }
37.  };
38.  var chart = new
     google.visualization.LineChart(document.getElementById('curve
     _chart'));
39.  chart.draw(data, options);
40.  }
41.  </script>

42.  </head>
43.  <body>

   44.     <div id="curve_chart" style="width: 100%; height:
     100%"></div> // <!-- la forma en que se llama a la gráfica
     -->
45.  </body>
46.  </html>
```

3. Resultados

Los resultados obtenidos en la ejecución del presente proyecto han sido favo, se pretendía realizar una maqueta donde se pudiera controlar de forma remota un motor de regadío, la obtención de la temperatura y humedades.

El principal objetivo era la creación de una maqueta que realizara la obtención de datos, y posteriormente el envío de ellos a través de la red social Telegram.

El segundo objetivo marcado, era la creación de una intranet que mostrara en una gráfica los datos obtenidos durante un período de tiempo. La obtención de datos ha sido cada 30 minutos en las pruebas que se han realizado.

Dicha maqueta ha estado en funcionamiento durante 7 días, de esta forma se ha testeado y comprobado que todo funcione de forma correcta, obteniendo los datos y almacenándolos en su base de datos correspondiente.

Algunos objetivos no han podido realizarse debido a los bajos conocimientos de programación y al leguaje Python 3.

4. Conclusiones y perspectivas

No se han llegado a obtener los principales objetivos debido a la falta de conocimientos de programación.

Los principales problemas que se encuentran en este proyecto son:

- La falta de seguridad a la hora de acceder al bot, cualquier persona ajena puede tener acceso, se debería establecer un usuario y contraseña para tener privacidad a la hora de mantener una conversación con el bot; de esta forma, ningún usuario podría acceder y manipular los comandos.
- Creación de una web online donde se pueda acceder desde cualquier sitio con solo tener conexión a internet.
- Creación de un temporizador de apagado o encendido del motor de regadío, principalmente para apagarlo una vez esté encendido y su posible configuración con el tiempo que se pretenda.
- Creación de una variable para programar, si se quiere que se encienda el motor en caso de que baje la humedad del terreno por debajo de unos parámetros marcados.
- Encriptación de la información de los módulos de radio frecuencia.
- Actualizaciones automáticas de la API de Telegram.
- Incorporación de un módulo gsm a la raspberry pi, de esta forma se puede obtener conexión a internet desde cualquier sitio.
- Creación de una variable para posibles errores o alertas debido a un fallo, errores de lectura de los sensores, encendido del motor de regadío u otro

5. Bibliografía

3, A. a. (19 de octubre de 2017). *Aprenda a programar con Python 3 .* ANAYA.

Arduino. (2018). *ARDUINO*. Obtenido de http://arduino.cl/arduino-uno/

Baker, J. (2 de Febrero de 2015). *Opensource.com*. Obtenido de https://opensource.com/resources/raspberry-pi

Barua, P. (2017). Implementation Guide for utilities: Designing renewable energy products to meet large energy costumer needs. *World Resources Institute*, 1-20.

ESTESO, M. P. (s.f.). *geekytheory.* Obtenido de https://geekytheory.com/tutorial-raspberry-pi-configurar-wif/

Mechatronics, N. (s.f.). *Naylamp Mechatronics*. Obtenido de Naylamp Mechatronics: http://www.naylampmechatronics.com/blog/16_Tutorial-b%C3%A1sico-NRF24L01-con-Arduino.html

PROMETEC. (s.f.). Obtenido de PROMETEC: https://www.prometec.net/raspberry-pi-servidor/

Raspberry pi. (s.f.). Obtenido de https://www.raspberrypi.org/downloads/raspbian/

RASPBERRY PI WEB INFO. (s.f.). Obtenido de http://rpi.uroboros.es/comand.html

RaspberryPiWordMark. (2018). *raspberrypi.org*. Obtenido de https://www.raspberrypi.org/help/faqs/#introWhatIs

SD Memory Card Formatter. (s.f.). Obtenido de SD Memory Card Formatter: https://www.sdcard.org/downloads/formatter_4/

Telegram. (2018). *Telegram*. Obtenido de https://telegram.org/

Tur, S. P. (s.f.). *Archivos de interes*. Obtenido de https://drive.google.com/open?id=1WcUo0VIt1LsvcGr-dDxBbf_2QCeZkn_O

Wikipedia. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Telegram_Messenger#Interfaz_e_interacciones

win32diskimager. (s.f.). Obtenido de https://win32-disk-imager.uptodown.com/windows

3, A. a. (19 de octubre de 2017). *Aprenda a programar con Python 3 .* ANAYA.

Arduino. (2018). *ARDUINO*. Obtenido de http://arduino.cl/arduino-uno/

Baker, J. (2 de Febrero de 2015). *Opensource.com*. Obtenido de https://opensource.com/resources/raspberry-pi

Barua, P. (2017). Implementation Guide for utilities: Designing renewable energy products to meet large energy costumer needs. *World Resources Institute*, 1-20.

ESTESO, M. P. (s.f.). *geekytheory.* Obtenido de https://geekytheory.com/tutorial-raspberry-pi-configurar-wif/

Mechatronics, N. (s.f.). *Naylamp Mechatronics*. Obtenido de Naylamp Mechatronics: http://www.naylampmechatronics.com/blog/16_Tutorial-b%C3%A1sico-NRF24L01-con-Arduino.html

PROMETEC. (s.f.). Obtenido de PROMETEC: https://www.prometec.net/raspberry-pi-servidor/

Raspberry pi. (s.f.). Obtenido de https://www.raspberrypi.org/downloads/raspbian/

RASPBERRY PI WEB INFO. (s.f.). Obtenido de http://rpi.uroboros.es/comand.html

RaspberryPiWordMark. (2018). *raspberrypi.org*. Obtenido de https://www.raspberrypi.org/help/faqs/#introWhatIs

SD Memory Card Formatter. (s.f.). Obtenido de SD Memory Card Formatter: https://www.sdcard.org/downloads/formatter_4/

Telegram. (2018). *Telegram*. Obtenido de https://telegram.org/

Tur, S. P. (s.f.). *Archivos de interes*. Obtenido de https://drive.google.com/open?id=1WcUo0VIt1LsvcGr-dDxBbf_2QCeZkn_O

Wikipedia. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Telegram_Messenger#Interfaz_e_interacciones

win32diskimager. (s.f.). Obtenido de https://win32-disk-imager.uptodown.com/windows

6. Anexos

ANEXO 1
Diseños 3D

En este anexo se detallan los archivos que se han creado.

Los archivos en 3D han sido diseñados en el programa Fusion 360 de autodesk, dichos archivos se podrán encontrar en el enlace donde se adjuntara al final.

Los diseños en 3D se imprimirán en una impresora 3D, una vez terminados se podrán montar en el prototipo.

Los detalles de las piezas impresas son los siguientes:

- Material PLA
- Altura de capa 0.2mm
- Capas solidas inferiores 6
- Capas solidas superiores 6
- Contorno del perímetro 2
- Relleno 12%
- Temperatura del extrusor 215ºC
- Temperatura cama 0ºC
- Velocidad de impresión 80mm/s
- Velocidad de contorno 50%
- Velocidad de relleno 50%

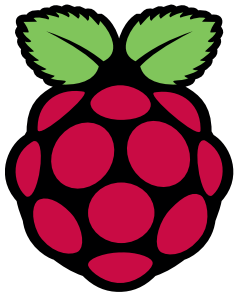
En el enlace que se adjuntara se podrá visualizar un video explicativo de las conexiones realizadas y su funcionamiento.

Enlace;

https://drive.google.com/open?id=1WcUo0VIt1LsvcGr-dDxBbf_2QCeZkn_O

ANEXO 2
Raspberry pi

DATASHEET



Raspberry Pi Compute Module (CM1)

Raspberry Pi Compute Module 3 (CM3)

Raspberry Pi Compute Module 3 Lite (CM3L)

Version 1.0, October 2016

Table 1: Revision History

Revision	Date	Description
1.0	13/10/2016	First release

Contents

List of Figures

List of Tables

1 Introduction

The Raspberry Pi Compute Module (CM1), Compute Module 3 (CM3) and Compute Module 3 Lite (CM3L) are DDR2-SODIMM-mechanically-compatible System on Modules (SoMs) containing processor, memory, eMMC Flash (for CM1 and CM3) and supporting power circuitry. These modules allow a designer to leverage the Raspberry Pi hardware and software stack in their own custom systems and form factors. In addition these module have extra IO interfaces over and above what is available on the Raspberry Pi model A/B boards opening up more options for the designer.

The CM1 contains a BCM2835 processor (as used on the original Raspberry Pi and Raspberry Pi B+ models), 512MByte LPDDR2 RAM and 4Gbytes eMMC Flash. The CM3 contains a BCM2837 processor (as used on the Raspberry Pi 3), 1Gbyte LPDDR2 RAM and 4Gbytes eMMC Flash. Finally the CM3L product is the same as CM3 except the eMMC Flash is not fitted, and the SD/eMMC interface pins are available for the user to connect their own SD/eMMC device.

Note that the BCM2837 processor is an evolution of the BCM2835 processor. The only real differences are that the BCM2837 can address more RAM (up to 1Gbyte) and the ARM CPU complex has been upgraded from a single core ARM11 in BCM2835 to a Quad core Cortex A53 with dedicated 512Kbyte L2 cache in BCM2837. All IO interfaces and peripherals stay the same and hence the two chips are largely software and hardware compatible.

The pinout of CM1 and CM3 are identical. Apart from the CPU upgrade and increase in RAM the other significant hwardware differences to be aware of are that CM3 has grown from 30mm to 31mm in height, the VBAT supply can now draw significantly more power under heavy CPU load, and the HDMI_HPD_N_1V8 (GPIO46_1V8 on CM1) and EMMC_EN_N_1V8 (GPIO47_1V8 on CM1) are now driven from an IO expander rather than the processor. If a designer of a CM1 product has a suitably specified VBAT, can accomodate the extra 1mm module height increase and has followed the design rules with respect to GPIO46_1V8 and GPIO47_1V8 then a CM3 should work fine in a board designed for a CM1.

2 Features

2.1 Hardware

- Low cost
- Low power
- High availability
- High reliability
  - Tested over millions of Raspberry Pis Produced to date
  - Module IO pins have 35u hard gold plating

2.2 Peripherals

- 48x GPIO
- 2x I2C
- 2x SPI
- 2x UART
- 2x SD/SDIO
- 1x HDMI 1.3a
- 1x USB2 HOST/OTG
- 1x DPI (Parallel RGB Display)
- 1x NAND interface (SMI)
- 1x 4-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 2-lane CSI Camera Interface (up to 1Gbps per lane)
- 1x 4-lane DSI Display Interface (up to 1Gbps per lane)
- 1x 2-lane DSI Display Interface (up to 1Gbps per lane)

2.3 Software

- ARMv6 (CM1) or ARMv7 (CM3, CM3L) Instruction Set
- Mature and stable Linux software stack
  - Latest Linux Kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Full availability of GPU functions using standard APIs

3 Block Diagram

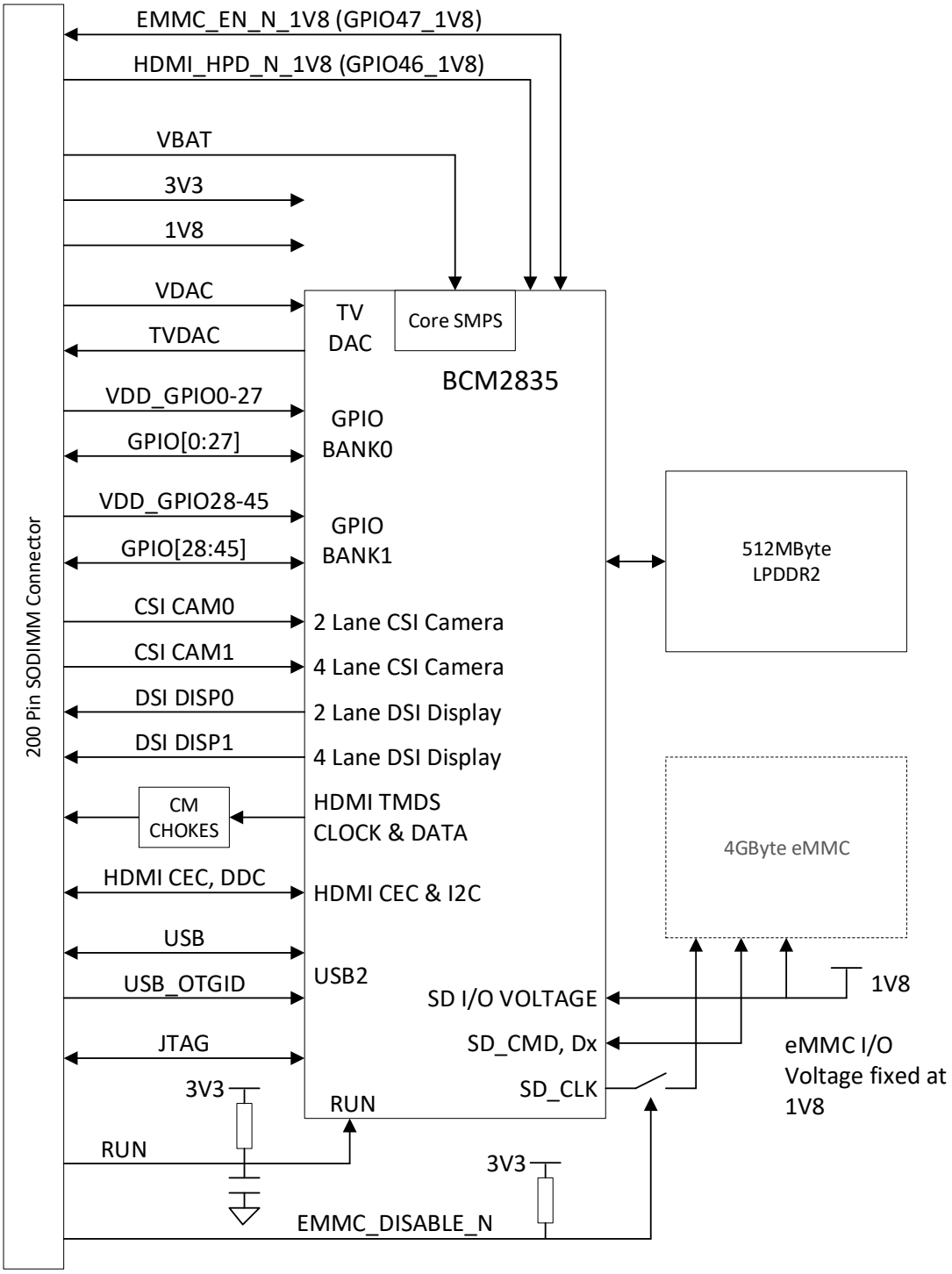


Figure 1: CM1 Block Diagram

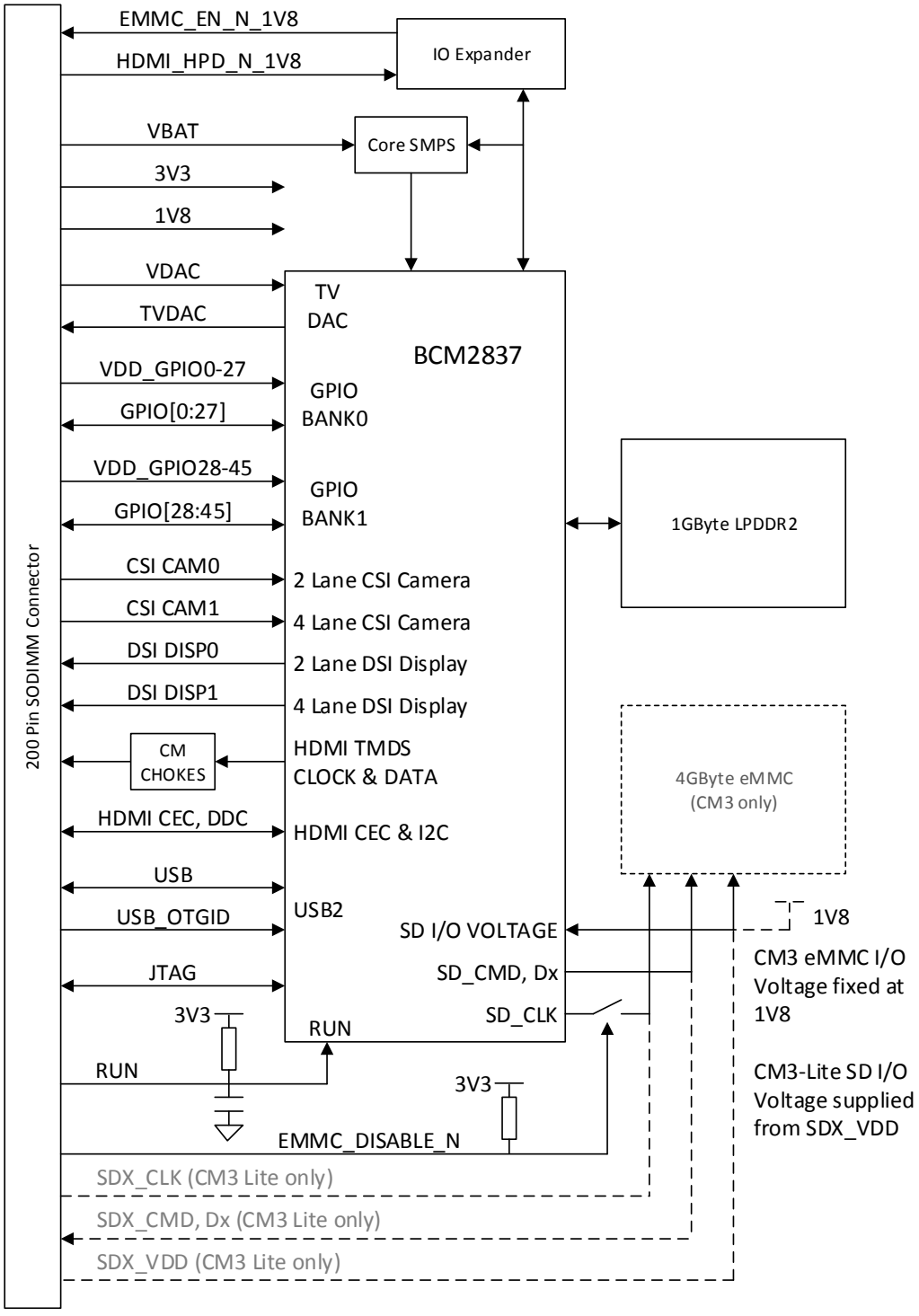


Figure 2: CM3/CM3L Block Diagram

4 Mechanical Specification

The Compute Modules conform to JEDEC MO-224 mechanical specification for 200 pin DDR2 (1.8V) SODIMM modules (with the exception that the CM3, CM3L modules are 31mm in height rather than 30mm of CM1) and therefore should work with the many DDR2 SODIMM sockets available on the market. **(Please note that the pinout of the Compute Module is not the same as a DDR2 SODIMM module; they are not electrically compatible.)**

The SODIMM form factor was chosen as a way to provide the 200 pin connections using a standard, readily available and low cost connector compatible with low cost PCB manufacture.

The maximum component height on the underside of the Compute Module is 1.2mm.

The maximum component height on the top side of the Compute Module is 1.5mm.

The Compute Module PCB thickness is 1.0mm +/- 0.1mm.

Note that the location and arrangement of components on the Compute Module may change slightly over time due to revisions for cost and manufacturing considerations; however, maximum component heights and PCB thickness will be kept as specified.

Figure 3 gives the CM1 mechanical dimensions. Figure 4 gives the CM3 and CM3L mechanical dimensions.

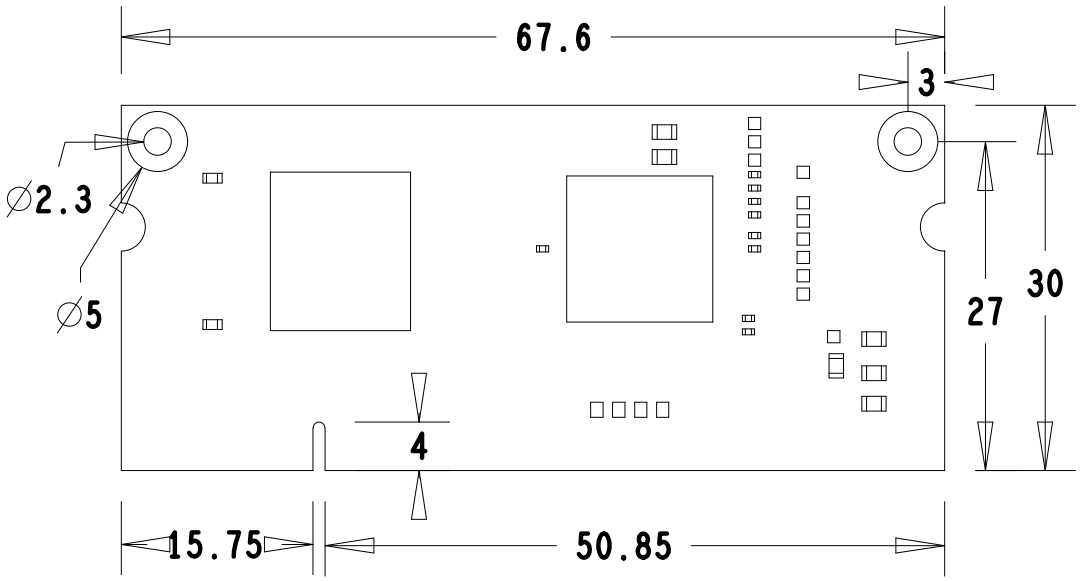


Figure 3: CM1 Mechanical Dimensions

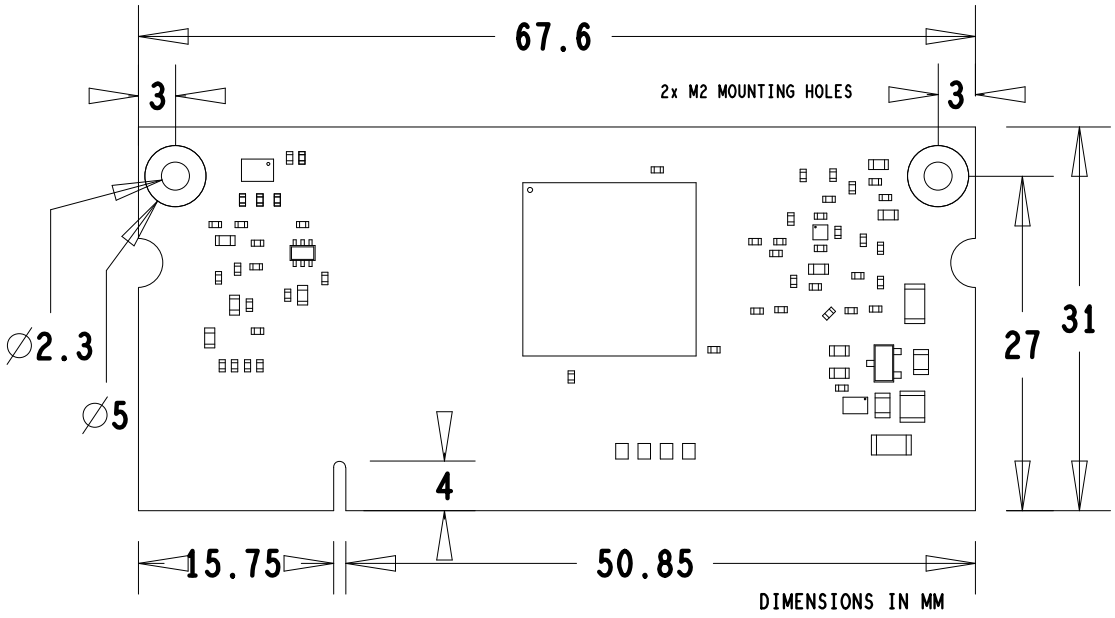


Figure 4: CM3 and CM3L Mechanical Dimensions

5 Pin Assignments

CM1	CM3-Lite	CM3	PIN		PIN	CM3	CM3-Lite	CM1
GND			1		2	EMMC_DISABLE_N		
GPIO0			3		4	NC	SDX_VDD	NC
GPIO1			5		6	NC	SDX_VDD	NC
GND			7		8	GND		NC
GPIO2			9		10	NC	SDX_CLK	NC
GPIO3			11		12	NC	SDX_CMD	NC
GND			13		14	GND		NC
GPIO4			15		16	NC	SDX_D0	NC
GPIO5			17		18	NC	SDX_D1	NC
GND			19		20	GND		NC
GPIO6			21		22	NC	SDX_D2	NC
GPIO7			23		24	NC	SDX_D3	NC
GND			25		26	GND		
GPIO8			27		28	GPIO28		
GPIO9			29		30	GPIO29		
GND			31		32	GND		
GPIO10			33		34	GPIO30		
GPIO11			35		36	GPIO31		
GND			37		38	GND		
GPIO0-27_VDD			39		40	GPIO0-27_VDD		
			KEY					
GPIO28-45_VDD			41		42	GPIO28-45_VDD		
GND			43		44	GND		
GPIO12			45		46	GPIO32		
GPIO13			47		48	GPIO33		
GND			49		50	GND		
GPIO14			51		52	GPIO34		
GPIO15			53		54	GPIO35		
GND			55		56	GND		
GPIO16			57		58	GPIO36		
GPIO17			59		60	GPIO37		
GND			61		62	GND		
GPIO18			63		64	GPIO38		
GPIO19			65		66	GPIO39		
GND			67		68	GND		
GPIO20			69		70	GPIO40		
GPIO21			71		72	GPIO41		
GND			73		74	GND		
GPIO22			75		76	GPIO42		
GPIO23			77		78	GPIO43		
GND			79		80	GND		
GPIO24			81		82	GPIO44		
GPIO25			83		84	GPIO45		
GND			85		86	GND		
GPIO26			87		88	HDMI_HPD_N_1V8		GPIO46_1V8
GPIO27			89		90	EMMC_EN_N_1V8		GPIO47_1V8
GND			91		92	GND		
DSI0_DN1			93		94	DSI1_DP0		
DSI0_DP1			95		96	DSI1_DN0		
GND			97		98	GND		
DSI0_DN0			99		100	DSI1_CP		
DSI0_DP0			101		102	DSI1_CN		
GND			103		104	GND		
DSI0_CN			105		106	DSI1_DP3		
DSI0_CP			107		108	DSI1_DN3		
GND			109		110	GND		
HDMI_CLK_N			111		112	DSI1_DP2		
HDMI_CLK_P			113		114	DSI1_DN2		
GND			115		116	GND		
HDMI_D0_N			117		118	DSI1_DP1		
HDMI_D0_P			119		120	DSI1_DN1		
GND			121		122	GND		
HDMI_D1_N			123		124	NC		
HDMI_D1_P			125		126	NC		
GND			127		128	NC		
HDMI_D2_N			129		130	NC		
HDMI_D2_P			131		132	NC		
GND			133		134	GND		
CAM1_DP3			135		136	CAM0_DP0		
CAM1_DN3			137		138	CAM0_DN0		
GND			139		140	GND		
CAM1_DP2			141		142	CAM0_CP		
CAM1_DN2			143		144	CAM0_CN		
GND			145		146	GND		
CAM1_CP			147		148	CAM0_DP1		
CAM1_CN			149		150	CAM0_DN1		
GND			151		152	GND		
CAM1_DP1			153		154	NC		
CAM1_DN1			155		156	NC		
GND			157		158	NC		
CAM1_DP0			159		160	NC		
CAM1_DN0			161		162	NC		
GND			163		164	GND		
USB_DP			165		166	TVDAC		
USB_DM			167		168	USB_OTGID		
GND			169		170	GND		
HDMI_CEC			171		172	VC_TRST_N		
HDMI_SDA			173		174	VC_TDI		
HDMI_SCL			175		176	VC_TMS		
RUN			177		178	VC_TDO		
VDD_CORE (DO NOT CONNECT)			179		180	VC_TCK		
GND			181		182	GND		
1V8			183		184	1V8		
1V8			185		186	1V8		
GND			187		188	GND		
VDAC			189		190	VDAC		
3V3			191		192	3V3		
3V3			193		194	3V3		
GND			195		196	GND		
VBAT			197		198	VBAT		
VBAT			199		200	VBAT		

Table 2: Compute Module SODIMM Connector Pinout

Table 2 gives the Compute Module pinout and Table 3 gives the Compute Module pin functions.

Pin Name	DIR	Voltage Ref	PDN[a] State	If Unused	Description/Notes
RUN and Boot Control (see text for usage guide)					
RUN	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_DISABLE_N	I	3V3[b]	Pull High	Leave open	Has internal 10k pull up
EMMC_EN_N_1V8	O	1V8	Pull High	Leave open	Has internal 2k2 pull up
GPIO					
GPIO[27:0]	I/O	GPIO0-27_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 0
GPIO[45:28]	I/O	GPIO28-45_VDD	Pull or Hi-Z[c]	Leave open	GPIO Bank 1
Primary SD Interface[d,e]					
SDX_CLK	O	SDX_VDD	Pull High	Leave open	Primary SD interface CLK
SDX_CMD	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface CMD
SDX_Dx	I/O	SDX_VDD	Pull High	Leave open	Primary SD interface DATA
USB Interface					
USB_Dx	I/O	-	Z	Leave open	Serial interface
USB_OTGID	I	3V3		Tie to GND	OTG pin detect
HDMI Interface					
HDMI_SCL	I/O	3V3[b]	Z[f]	Leave open	DDC Clock (5.5V tolerant)
HDMI_SDA	I/O	3V3[b]	Z[f]	Leave open	DDC Data (5.5V tolerant)
HDMI_CEC	I/O	3V3	Z	Leave open	CEC (has internal 27k pull up)
HDMI_CLKx	O	-	Z	Leave open	HDMI serial clock
HDMI_Dx	O	-	Z	Leave open	HDMI serial data
HDMI_HPD_N_1V8	I	1V8	Pull High	Leave open	HDMI hotplug detect
CAM0 (CSI0) 2-lane Interface					
CAM0_Cx	I	-	Z	Leave open	Serial clock
CAM0_Dx	I	-	Z	Leave open	Serial data
CAM1 (CSI1) 4-lane Interface					
CAM1_Cx	I	-	Z	Leave open	Serial clock
CAM1_Dx	I	-	Z	Leave open	Serial data
DSI0 (Display 0) 2-lane Interface					
DSI0_Cx	O	-	Z	Leave open	Serial clock
DSI0_Dx	O	-	Z	Leave open	Serial data
DSI1 (Display 1) 4-lane Interface					
DSI1_Cx	O	-	Z	Leave open	Serial clock
DSI1_Dx	O	-	Z	Leave open	Serial data
TV Out					
TVDAC	O	-	Z	Leave open	Composite video DAC output
JTAG Interface					
TMS	I	3V3	Z	Leave open	Has internal 50k pull up
TRST_N	I	3V3	Z	Leave open	Has internal 50k pull up
TCK	I	3V3	Z	Leave open	Has internal 50k pull up
TDI	I	3V3	Z	Leave open	Has internal 50k pull up
TDO	O	3V3	O	Leave open	Has internal 50k pull up

[a] The PDN column indicates power-down state (when RUN pin LOW)
[b] Must be driven by an open-collector driver
[c] GPIO have software enabled pulls which keep state over power-down
[d] Only available on Lite variants
[e] The CM will always try to boot from this interface first
[f] Requires external pull-up resistor to 5V as per HDMI spec

Table 3: Pin Functions

6 Electrical Specification

Caution! Stresses above those listed in Table 4 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VBAT	Core SMPS Supply	-0.5	6.0	V
3V3	3V3 Supply Voltage	-0.5	4.10	V
1V8	1V8 Supply Voltage	-0.5	2.10	V
VDAC	TV DAC Supply	-0.5	4.10	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	-0.5	4.10	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	-0.5	4.10	V
SDX_VDD	Primary SD/eMMC Supply Voltage	-0.5	4.10	V

Table 4: Absolute Maximum Ratings

DC Characteristics are defined in Table 5

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 1.8V	-	-	0.6	V
		VDD_IO = 2.7V	-	-	0.8	V
V_{IH}	Input high voltage[a]	VDD_IO = 1.8V	1.0	-	-	V
		VDD_IO = 2.7V	1.3	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	5	μA
C_{IN}	Input capacitance	-	-	5	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 1.8V, IOL = -2mA	-	-	0.2	V
		VDD_IO = 2.7V, IOL = -2mA	-	-	0.15	V
V_{OH}	Output high voltage[b]	VDD_IO = 1.8V, IOH = 2mA	1.6	-	-	V
		VDD_IO = 2.7V, IOH = 2mA	2.5	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 1.8V, VO = 0.4V	12	-	-	mA
		VDD_IO = 2.7V, VO = 0.4V	17	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 1.8V, VO = 1.4V	10	-	-	mA
		VDD_IO = 2.7V, VO = 2.3V	16	-	-	mA
R_{PU}	Pullup resistor	-	50	-	65	kΩ
R_{PD}	Pulldown resistor	-	50	-	65	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 5: DC Characteristics

AC Characteristics are defined in Table 6 and Fig. 5.

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	1.6	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	1.7	-	ns
GPCLK	t_{JOSC}	Oscillator-derived GPCLK cycle-cycle jitter (RMS)	-	-	20	ps
GPCLK	t_{JPLL}	PLL-derived GPCLK cycle-cycle jitter (RMS)	-	-	48	ps

[a] Default drive strength, CL = 5pF, VDD_IOx = 3.3V

Table 6: Digital I/O Pin AC Characteristics

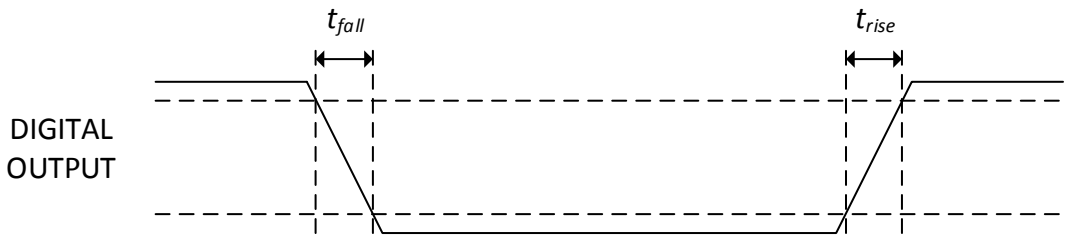


Figure 5: Digital IO Characteristics

7 Power Supplies

The Compute Module has six separate supplies that must be present and powered at all times; you cannot leave any of them unpowered, even if a specific interface or GPIO bank is unused. The six supplies are as follows:

1. VBAT is used to power the BCM283x processor core. It feeds the SMPS that generates the chip core voltage.
2. 3V3 powers various BCM283x PHYs, IO and the eMMC Flash.
3. 1V8 powers various BCM283x PHYs, IO and SDRAM.
4. VDAC powers the composite (TV-out) DAC.
5. GPIO0-27_VREF powers the GPIO 0-27 IO bank.
6. GPIO28-45_VREF powers the GPIO 28-45 IO bank.

Supply	Descripion	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply[a]	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

[a] Requires a clean 2.5-2.8V supply if TV DAC is used, else connect to 3V3

Table 7: Power Supply Operating Ranges

7.1 Supply Sequencing

Supplies should be staggered so that the highest voltage comes up first, then the remaining voltages in descending order. This is to avoid forward biasing internal (on-chip) diodes between supplies, and causing latch-up. Alternatively supplies can be synchronised to come up at exactly the same time as long as at no point a lower voltage supply rail voltage exceeds a higher voltage supply rail voltage.

7.2 Power Requirements

Exact power requirements will be heavily dependent upon the individual use case. If an on-chip subsystem is unused, it is usually in a low power state or completely turned off. For instance, if your application does not use 3D graphics then a large part of the core digital logic will never turn on and need power. This is also the case for camera and display interfaces, HDMI, USB interfaces, video encoders and decoders, and so on.

Powerchain design is critical for stable and reliable operation of the Compute Module. We strongly recommend that designers spend time measuring and verifying power requirements for their particular use case and application, as well as paying careful attention to power supply sequencing and maximum supply voltage tolerance.

Table 8 specifies the recommneded minimum power supply outputs required to power the Compute Module.

Supply	Minimum Requirement	Unit
VBAT (CM1)	2000[a]	mW
VBAT (CM3,3L)	3500[a]	mW
3V3	250	mA
1V8	250	mA
VDAC	25	mA
GPIO0-27_VDD	50[b]	mA
GPIO28-45_VDD	50[b]	mA
SDX_VDD	50[b]	mA

[a] Recommended minimum. Actual power drawn is very dependent on use-case
[b] Each GPIO can supply up to 16mA, aggregate current per bank must not exceed 50mA

Table 8: Mimimum Power Supply Requirements

8 Booting

The 4GB eMMC Flash device on CM3 is directly connected to the primary BCM2837 SD/eMMC interface. These connections are not accessible on the module pins. On CM3L this SD interface is available on the SDX_ pins.

When initially powered on, or after the RUN pin has been held low and then released, the BCM2837 will try to access the primary SD/eMMC interface. It will then look for a file called bootcode.bin on the primary partition (which must be FAT) to start booting the system. If it cannot access the SD/eMMC device or the boot code cannot be found, it will fall back to waiting for boot code to be written to it over USB; in other words, its USB port is in slave mode waiting to accept boot code from a suitable host.

A USB boot tool is available on Github which allows a host PC running Linux to write the BCM2837 boot code over USB to the module. That boot code then runs and provides access to the SD/eMMC as a USB mass storage device, which can then be read and written using the host PC. Note that a Raspberry Pi can be used as the host machine. For those using Windows a precompiled and packeged tool is available. For more information see here.

The Compute Module has a pin called EMMC_DISABLE_N which when shorted to GND will disable the SD/eMMC interface (by physically disconnecting the SD_CMD pin), forcing BCM2837 to boot from USB. Note that when the eMMC is disabled in this way, it takes a couple of seconds from powering up for the processor to stop attempting to talk to the SD/eMMC device and fall back to booting from USB.

Note that once booted over USB, BCM2837 needs to re-enable the SD/eMMC device (by releasing EMMC_DISABLE_N) to allow access to it as mass storage. It expects to be able to do this by driving the EMMC_EN_N_1V8 pin LOW, which at boot is initially an input with a pull up to 1V8. If an end user wishes to add the ability to access the SD/eMMC over USB in their product, similar circuitry to that used on the Compute Module IO Board to enable/disable the USB boot and SD/eMMC must be used; that is, EMMC_DISABLE_N pulled low via MOSFET(s) and released again by MOSFET, with the gate controlled by EMMC_EN_N_1V8. Ensure you use MOSFETs suitable for switching at 1.8V (i.e. use a device with gate threshold voltage, Vt, suitable for 1.8V switching).

9 Peripherals

9.1 GPIO

BCM283x has in total 54 GPIO lines in 3 separate voltage banks. All GPIO pins have at least two alternative functions within the SoC. When not used for the alternate peripheral function, each GPIO pin may be set as an input (optionally as an interrupt) or an output. The alternate functions are usually peripheral I/Os, and most peripherals appear twice to allow flexibility on the choice of I/O voltage.

On CM1, CM3 and CM3L bank2 is used on the module to connect to the eMMC device and, on CM3 and CM3L, for an on-board I2C bus (to talk to the core SMPS and control the special function pins). On CM3L most of bank 2 is exposed to allow a user to connect their choice of SD card or eMMC device (if required).

Bank0 and 1 GPIOs are available for general use. GPIO0 to GPIO27 are bank 0 and GPIO28-45 make up bank1. GPIO0-27_VDD is the power supply for bank0 and GPIO28-45_VDD is the power supply for bank1. SDX_VDD is the supply for bank2 on CM3L. These supplies can be in the range 1.8V-3.3V (see Table 7) and are not optional; each bank must be powered, even when none of the GPIOs for that bank are used.

Note that the HDMI_HPD_N_1V8 and EMMC_EN_N_1V8 pins (on CM1 these were called GPIO46_1V8 and GPIO47_1V8 respectively) are 1.8V IO and are used for special functions (HDMI hot plug detect and boot control respectively). Please do not use these pins for any other purpose, as the software for the Compute Module will always expect these pins to have these special functions. If they are unused please leave them unconnected.

All GPIOs except GPIO28, 29, 44 and 45 have weak in-pad pull-ups or pull-downs enabled when the device is powered on. It is recommended to add off-chip pulls to GPIO28, 29, 44 and 45 to make sure they never float during power on and initial boot.

9.1.1 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	-	-	-
1	High	SCL0	SA4	DE	-	-	-
2	High	SDA1	SA3	LCD_VSYNC	-	-	-
3	High	SCL1	SA2	LCD_HSYNC	-	-	-
4	High	GPCLK0	SA1	DPI_D0	-	-	ARM_TDI
5	High	GPCLK1	SA0	DPI_D1	-	-	ARM_TDO
6	High	GPCLK2	SOE_N	DPI_D2	-	-	ARM_RTCK
7	High	SPI0_CE1_N	SWE_N	DPI_D3	-	-	-
8	High	SPI0_CE0_N	SD0	DPI_D4	-	-	-
9	Low	SPI0_MISO	SD1	DPI_D5	-	-	-
10	Low	SPI0_MOSI	SD2	DPI_D6	-	-	-
11	Low	SPI0_SCLK	SD3	DPI_D7	-	-	-
12	Low	PWM0	SD4	DPI_D8	-	-	ARM_TMS
13	Low	PWM1	SD5	DPI_D9	-	-	ARM_TCK
14	Low	TXD0	SD6	DPI_D10	-	-	TXD1
15	Low	RXD0	SD7	DPI_D11	-	-	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	-	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	-	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	-	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	-	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	-
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	-
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	-
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	-
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	-
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	-

Table 9: GPIO Bank0 Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
28	None	SDA0	SA5	PCM_CLK	FL0	-	-
29	None	SCL0	SA4	PCM_FS	FL1	-	-
30	Low	TE0	SA3	PCM_DIN	CTS0	-	CTS1
31	Low	FL0	SA2	PCM_DOUT	RTS0	-	RTS1
32	Low	GPCLK0	SA1	RING_OCLK	TXD0	-	TXD1
33	Low	FL1	SA0	TE1	RXD0	-	RXD1
34	High	GPCLK0	SOE_N	TE2	SD1_CLK	-	-
35	High	SPI0_CE1_N	SWE_N	-	SD1_CMD	-	-
36	High	SPI0_CE0_N	SD0	TXD0	SD1_DAT0	-	-
37	Low	SPI0_MISO	SD1	RXD0	SD1_DAT1	-	-
38	Low	SPI0_MOSI	SD2	RTS0	SD1_DAT2	-	-
39	Low	SPI0_SCLK	SD3	CTS0	SD1_DAT3	-	-
40	Low	PWM0	SD4	-	SD1_DAT4	SPI2_MISO	TXD1
41	Low	PWM1	SD5	TE0	SD1_DAT5	SPI2_MOSI	RXD1
42	Low	GPCLK1	SD6	TE1	SD1_DAT6	SPI2_SCLK	RTS1
43	Low	GPCLK2	SD7	TE2	SD1_DAT7	SPI2_CE0_N	CTS1
44	None	GPCLK1	SDA0	SDA1	TE0	SPI2_CE1_N	-
45	None	PWM1	SCL0	SCL1	TE1	SPI2_CE2_N	-

Table 10: GPIO Bank1 Alternate Functions

Table 9 and Table 10 detail the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the Broadcom Peripherals Specification document and have Linux drivers available.

9.1.2 Secondary Memory Interface (SMI)

The SMI peripheral is an asynchronous NAND type bus supporting Intel mode80 type transfers at 8 or 16 bit widths and available in the ALT1 positions on GPIO banks 0 and 1 (see Table 9 and Table 10). It is not publicly documented in the Broadcom Peripherals Specification but a Linux driver is available in the Raspberry Pi Github Linux repository (bcm2835_smi.c in linux/drivers/misc).

9.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available on bank 0 GPIOs. This up-to-24-bit parallel interface can support a secondary display. Again this interface is not documented in the Broadcom Peripherals Specification but documentation can be found here.

9.1.4 SD/SDIO Interface

The BCM283x supports two SD card interfaces, SD0 and SD1.

The first (SD0) is a proprietary Broadcom controller that does not support SDIO and is the primary interface used to boot and talk to the eMMC or SDX_x signals.

The second interface (SD1) is standards compliant and can interface to SD, SDIO and eMMC devices; for example on a Raspberry Pi 3 it is used to talk to the on-board BCM43438 WiFi device in SDIO mode.

Both interfaces can support speeds up to 50MHz single ended (SD High Speed Mode).

9.2 CSI (MIPI Serial Camera)

Currently the CSI interface is not openly documented and only CSI camera sensors supported by the official Raspberry Pi firmware will work with this interface. Supported sensors are the OmniVision OV5647 and Sony IMX219.

It is recommended to attach other cameras via USB.

9.3 DSI (MIPI Serial Display)

Currently the DSI interface is not openly documented and only DSI displays supported by the official Raspberry Pi firmware will work with this interface.

Displays can also be added via the parallel DPI interface which is available as a GPIO alternate function - see Table 9 and Section 9.1.3

9.4 USB

The BCM283x USB port is On-The-Go (OTG) capable. If using either as a fixed slave or fixed master, please tie the USB_OTGID pin to ground.

The USB port (Pins USB_DP and USB_DM) must be routed as 90 ohm differential PCB traces.

Note that the port is capable of being used as a true OTG port however there is no official documentation. Some users have had success making this work.

9.5 HDMI

BCM283x supports HDMI V1.3a.

It is recommended that users follow a similar arrangement to the Compute Module IO Board circuitry for HDMI output.

The HDMI CK_P/N (clock) and D0-D2_P/N (data) pins must each be routed as matched length 100 ohm differential PCB traces. It is also important to make sure that each differential pair is closely phase matched. Finally, keep HDMI traces well away from other noise sources and as short as possible.

Failure to observe these design rules is likely to result in EMC failure.

9.6 Composite (TV Out)

The TVDAC pin can be used to output composite video (PAL or NTSC). Please route this signal away from noise sources and use a 75 ohm PCB trace.

Note that the TV DAC is powered from the VDAC supply which must be a clean supply of 2.5-2.8V. It is recommended users generate this supply from 3V3 using a low noise LDO.

If the TVDAC output is not used VDAC can be connected to 3V3, but it must be powered even if the TV-out functionality is unused.

10 Thermals

The BCM283x SoC employs DVFS (Dynamic Voltage and Frequency Scaling) on the core voltage. When the processor is idle (low CPU utilisation), it will reduce the core frequency and voltage to reduce current draw and heat output. When the core utilisation exceeds a certain threshold the core votlage is increased and the core frequency is boosted to the maximum working frerquency. The voltage and frequency are throttled back when the CPU load reduces back to an 'idle' level OR when the silicon temperature as mesured by the on-chip temperature sensor exceeds 85C (thermal throttling).

A designer must pay careful attention to the thermal design of products using the CM3/CM3L so that performance is not artificially curtailed due to the processor thermal throttling, as the Quad ARM complex in the BCM2837 can generate significant heat output.

10.1 Temperature Range

The operating temperature range of the module is set by the lowest maximum and highest minimum of any of the components used.

The eMMC and LPDDR2 have the narrowest range, these are rated for -25 to +80 degrees Celsius. Therefore the nominal range for the CM3 and CM3L is -25C to +80C.

However, this range is the maximum for the silicon die; therefore, users would have to take into account the heat generated when in use and make sure this does not cause the temperature to exceed 80 degrees Celsius.

11 Availability

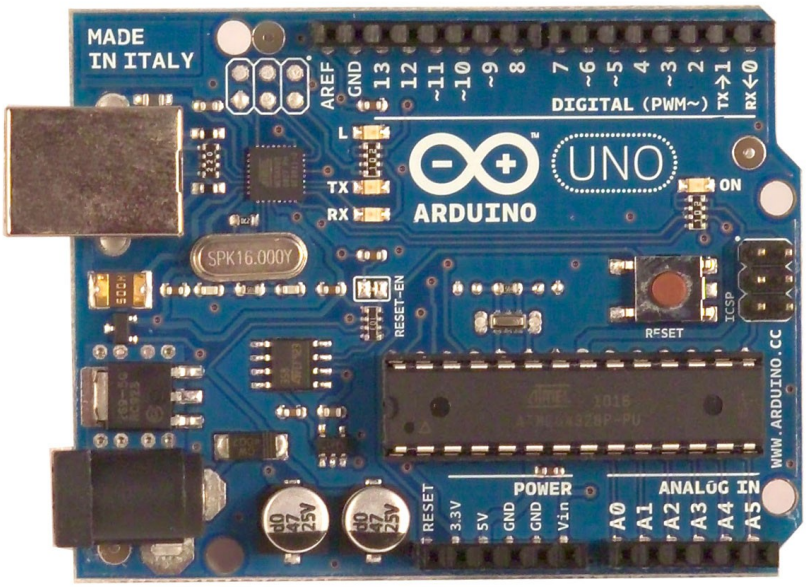
Raspberry Pi guarantee availability of CM1, CM3 and CM3 Lite until at least January 2023.

12 Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

ANEXO 3
Arduino Uno

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

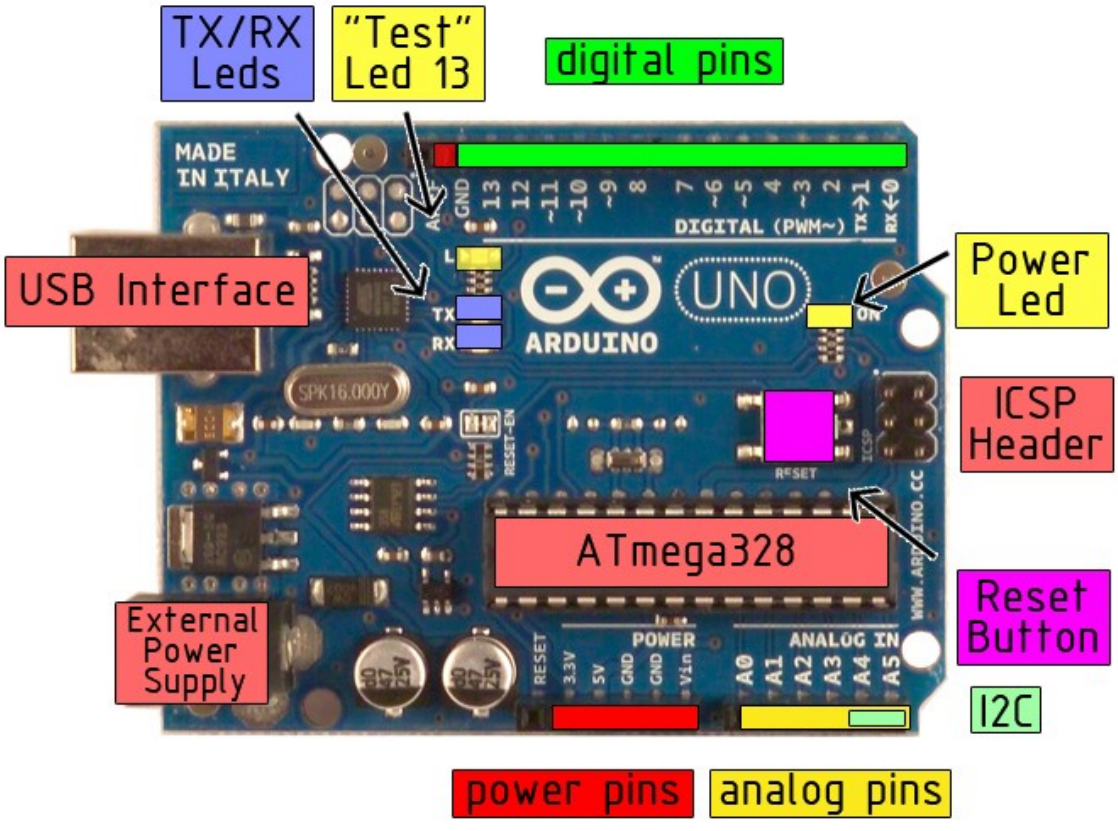
Technical Specification

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.

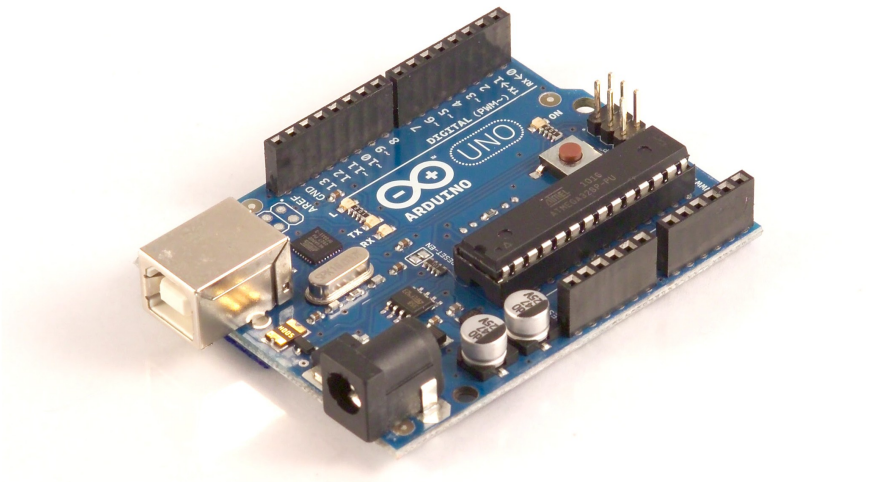
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread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

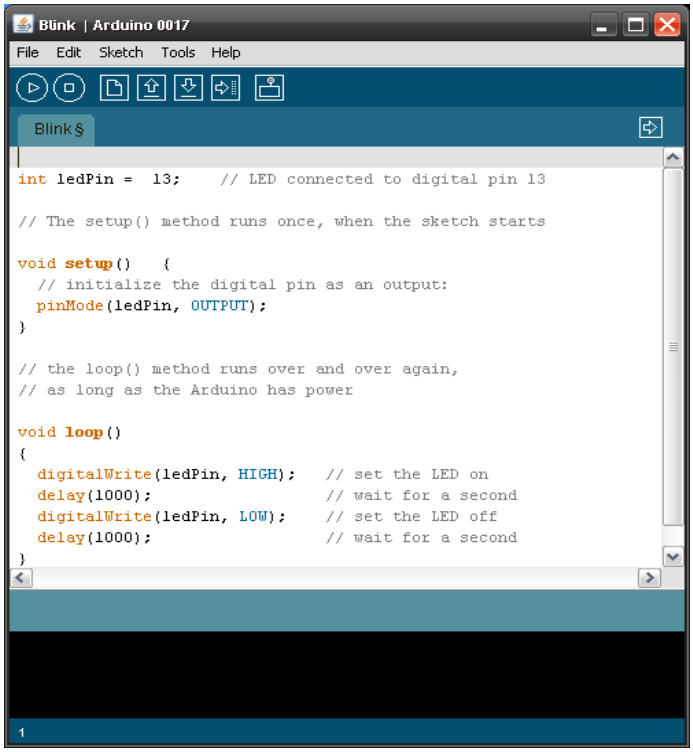
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

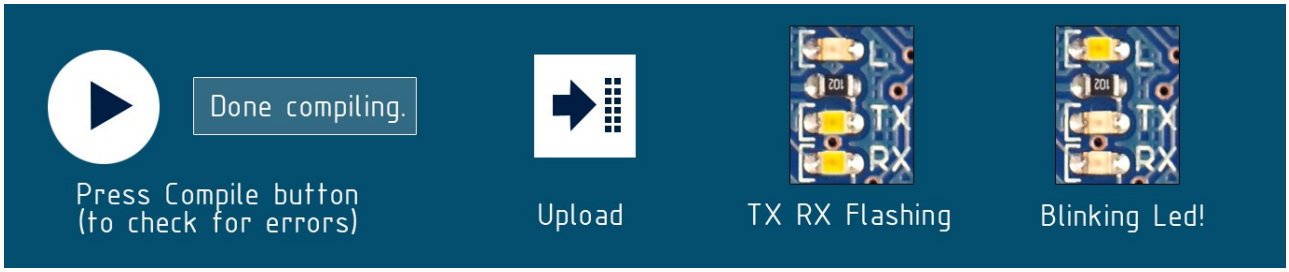
File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

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

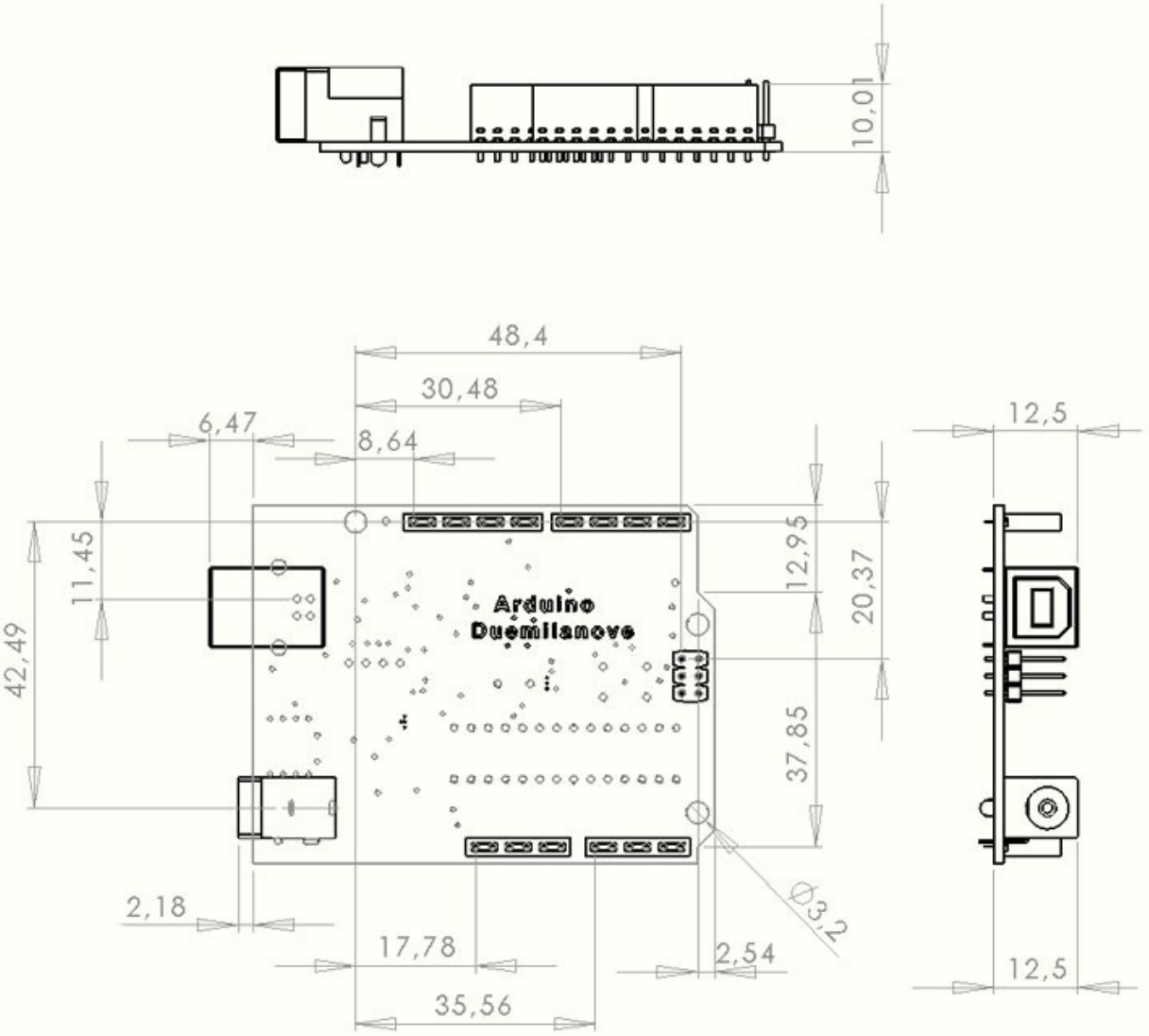
In **Tools>Board** select

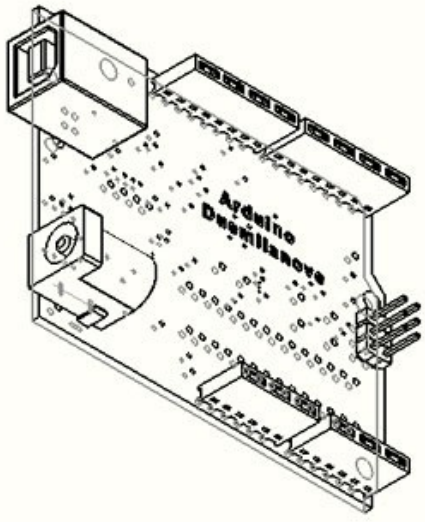
Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.





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

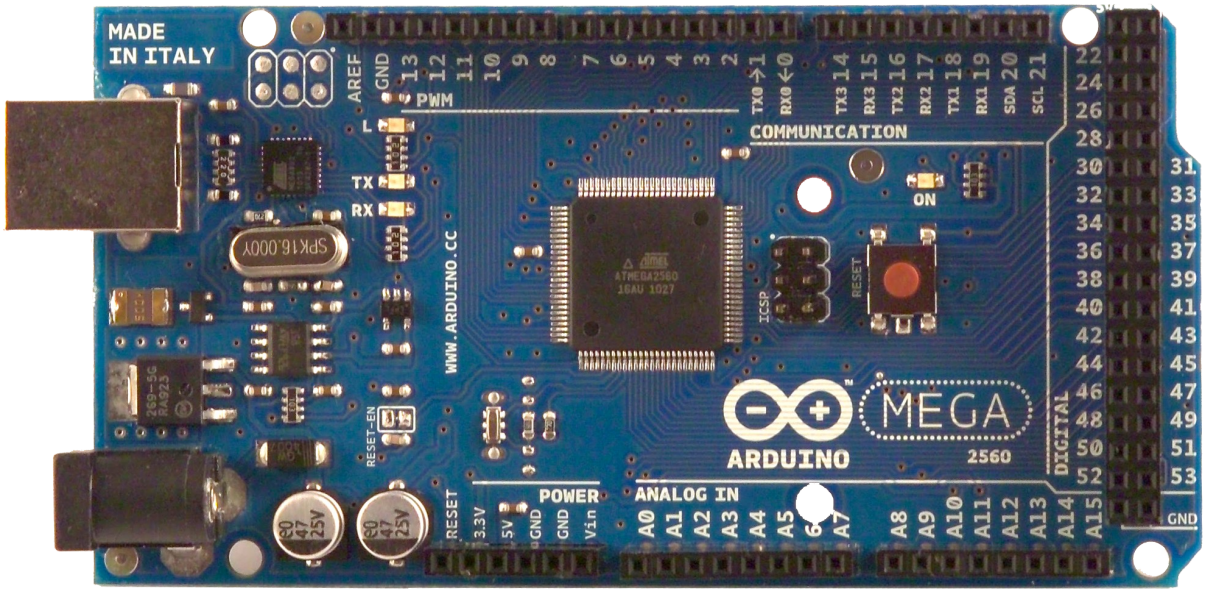
The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility 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.

ANEXO 4
Arduino Mega

Arduino MEGA 2560



Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), 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 Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

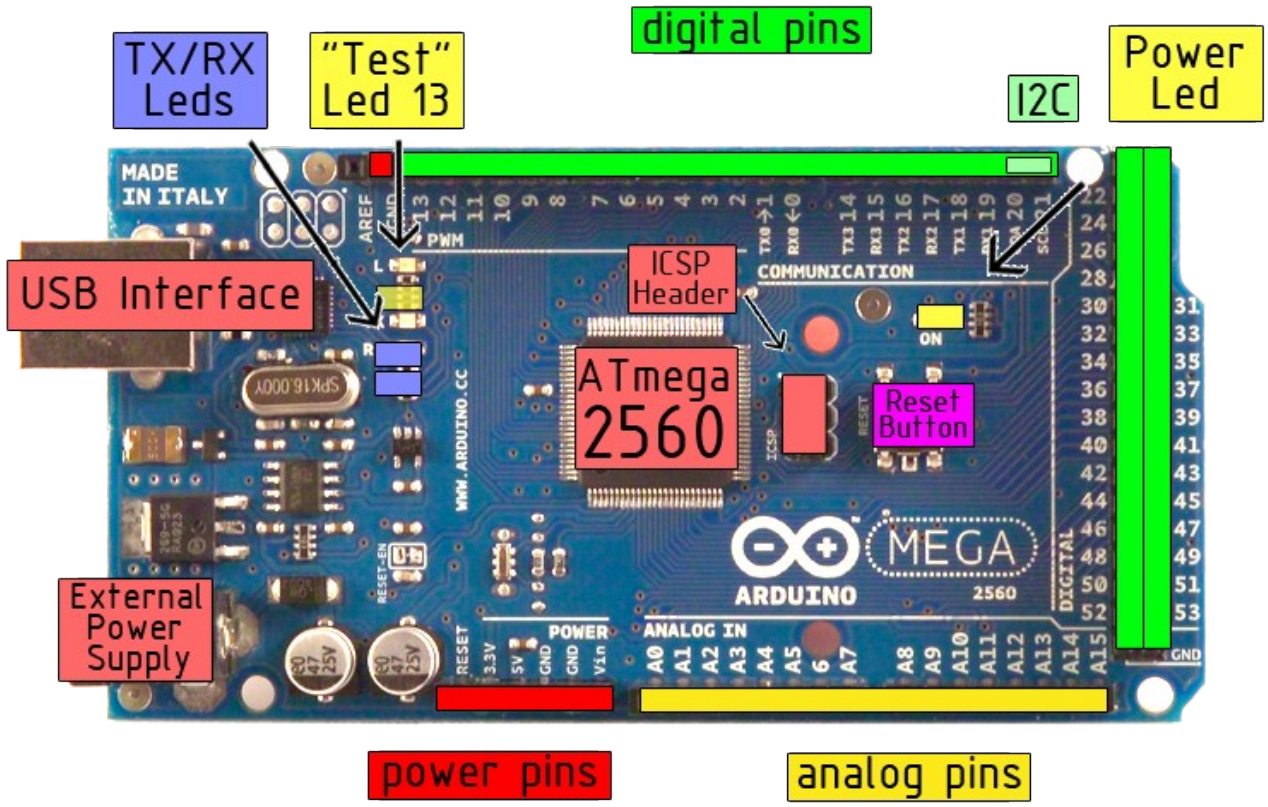
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 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 Mega2560 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.

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 ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega 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); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** 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: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **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.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 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 analogReference() function.

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.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 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 Mega's digital pins.

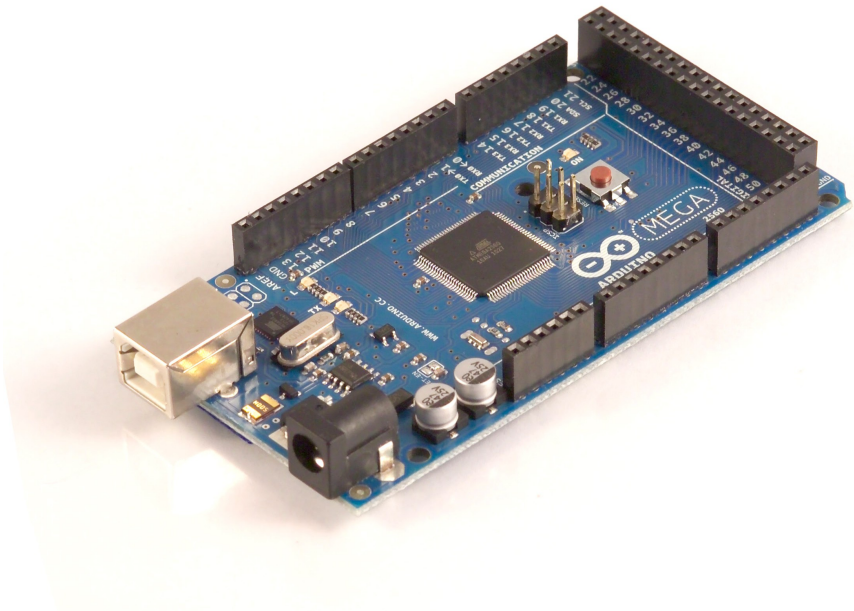
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega 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.



Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 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 ATmega2560 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 Mega2560 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 Mega2560. 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 Mega 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 Mega 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 and Shield Compatibility

The maximum length and width of the Mega PCB are 4 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.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

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

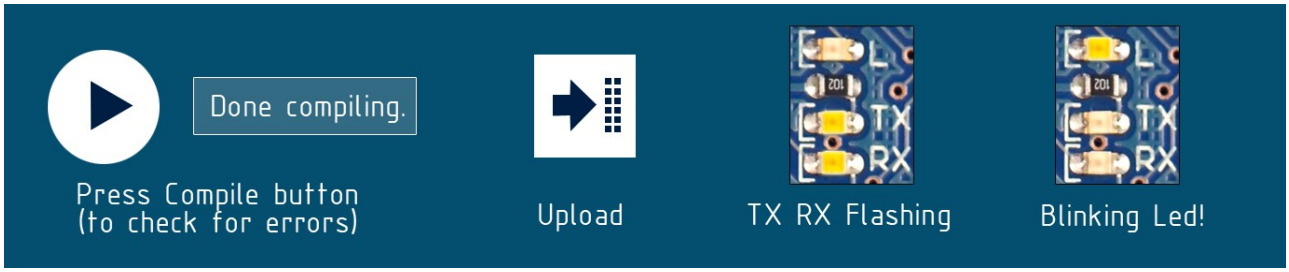
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

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

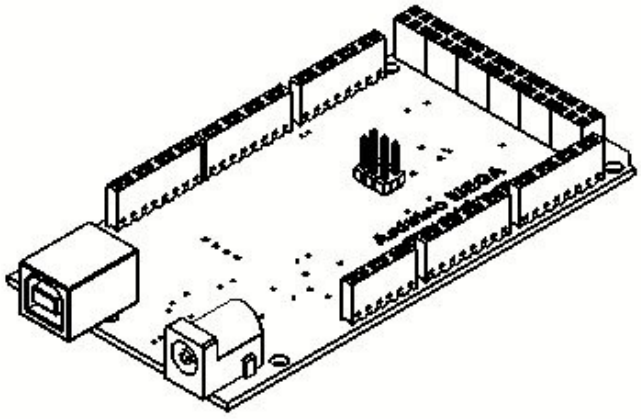
Once you have your skecth you'll see something very close to the screenshot on the right.

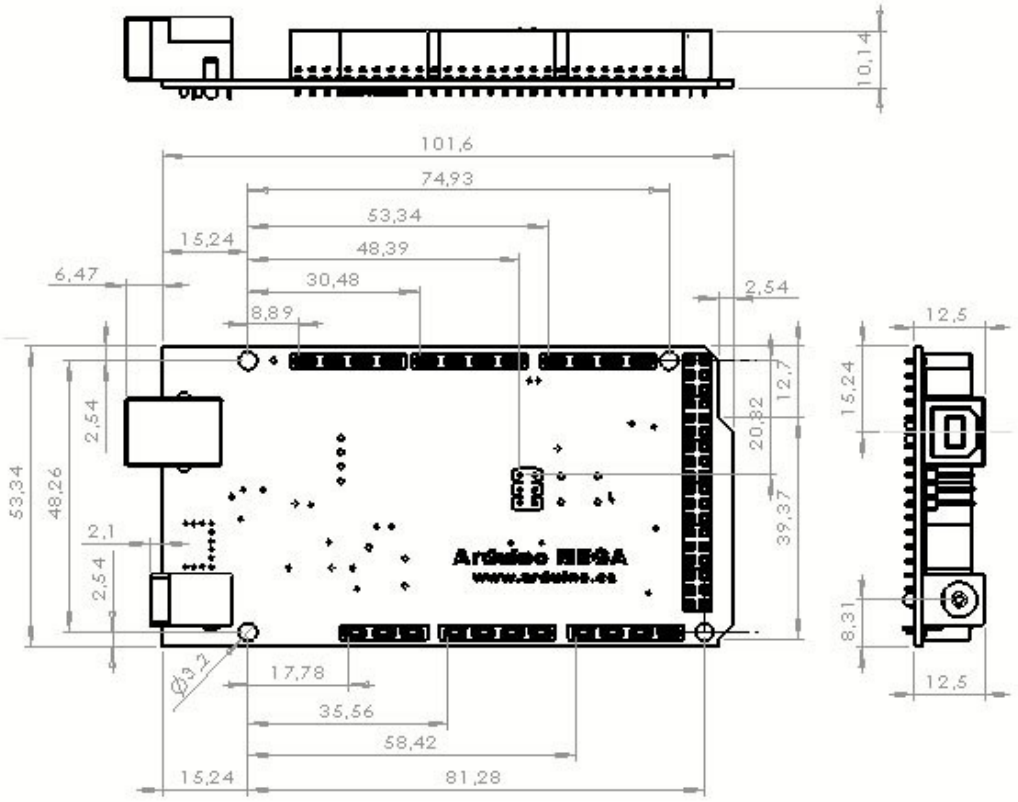
In **Tools>Board** select MEGA

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.

ANEXO 5
Módulo nRF24L01

nRF24L01
Single Chip 2.4GHz Transceiver

Product Specification

Key Features

- Worldwide 2.4GHz ISM band operation
- Up to 2Mbps on air data rate
- Ultra low power operation
- 11.3mA TX at 0dBm output power
- 12.3mA RX at 2Mbps air data rate
- 900nA in power down
- 22µA in standby-I
- On chip voltage regulator
- 1.9 to 3.6V supply range
- Enhanced ShockBurst™
- Automatic packet handling
- Auto packet transaction handling
- 6 data pipe MultiCeiver™
- Air compatible with nRF2401A, 02, E1 and E2
- Low cost BOM
- ±60ppm 16MHz crystal
- 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

Applications

- Wireless PC Peripherals
- Mouse, keyboards and remotes
- 3-in-one desktop bundles
- Advanced Media center remote controls
- VoIP headsets
- Game controllers
- Sports watches and sensors
- RF remote controls for consumer electronics
- Home and commercial automation
- Ultra low power sensor networks
- Active RFID
- Asset tracing systems
- Toys

July 2007

Liability disclaimer

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.

All application information is advisory and does not form part of the specification.

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 are not implied. Exposure to limiting values for extended periods may affect device reliability.

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.

Data sheet status	
Objective product specification	This product specification contains target specifications for product development.
Preliminary product specification	This product specification contains preliminary data; supplementary data may be published from Nordic Semiconductor ASA later.
Product specification	This product specification 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.

Contact details

Visit www.nordicsemi.no for Nordic Semiconductor sales offices and distributors worldwide

Main office:

Otto Nielsens vei 12
7004 Trondheim
Phone: +47 72 89 89 00
Fax: +47 72 89 89 89
www.nordicsemi.no

Writing Conventions

This product specification follows a set of typographic rules that makes the document consistent and easy to read. The following writing conventions are used:

- Commands, bit state conditions, and register names are written in `Courier`.
- Pin names and pin signal conditions are written in **`Courier bold`**.
- Cross references are underlined and highlighted in blue.

Revision History

Date	Version	Description
July 2007	2.0	• Restructured layout in a new template • Added details of the following features: ▸ Dynamic Payload Length (DPL) ▸ Acknowledgement Payload (`ACK_PLD`) ▸ Feature register ▸ ACTIVATE SPI command ▸ Selective Auto Acknowledgement (`NO_ACK`)

Contents

1 Introduction

The nRF24L01 is a single chip 2.4GHz transceiver with an embedded baseband protocol engine (Enhanced ShockBurst™), designed for ultra low power wireless applications. The nRF24L01 is designed for operation in the world wide ISM frequency band at 2.400 - 2.4835GHz. An MCU (microcontroller) and very few external passive components are needed to design a radio system with the nRF24L01.

The nRF24L01 is configured and operated through a Serial Peripheral Interface (SPI.) Through this interface the register map is available. The register map contains all configuration registers in the nRF24L01 and is accessible in all operation modes of the chip.

The embedded baseband protocol engine (Enhanced ShockBurst™) is based on packet communication and supports various modes from manual operation to advanced autonomous protocol operation. Internal FIFOs ensure a smooth data flow between the radio front end and the system's MCU. Enhanced ShockBurst™ reduces system cost by handling all the high-speed link layer operations.

The radio front end uses GFSK modulation. It has user configurable parameters like frequency channel, output power and air data rate.

The air data rate supported by the nRF24L01 is configurable to 2Mbps. The high air data rate combined with two power saving modes makes the nRF24L01 very suitable for ultra low power designs.

Internal voltage regulators ensure a high Power Supply Rejection Ratio (PSRR) and a wide power supply range.

1.1 Features

Features of the nRF24L01 include:

- Radio
  - Worldwide 2.4GHz ISM band operation
  - 126 RF channels
  - Common RX and TX pins
  - GFSK modulation
  - 1 and 2Mbps air data rate
  - 1MHz non-overlapping channel spacing at 1Mbps
  - 2MHz non-overlapping channel spacing at 2Mbps
- Transmitter
  - Programmable output power: 0, -6, -12 or -18dBm
  - 11.3mA at 0dBm output power
- Receiver
  - Integrated channel filters
  - 12.3mA at 2Mbps
  - -82dBm sensitivity at 2Mbps
  - -85dBm sensitivity at 1Mbps
  - Programmable LNA gain
- RF Synthesizer
  - Fully integrated synthesizer
  - No external loop filer, VCO varactor diode or resonator
  - Accepts low cost ±60ppm 16MHz crystal
- Enhanced ShockBurst™
  - 1 to 32 bytes dynamic payload length
  - Automatic packet handling
  - Auto packet transaction handling
  - 6 data pipe MultiCeiver™ for 1:6 star networks
- Power Management
  - Integrated voltage regulator
  - 1.9 to 3.6V supply range
  - Idle modes with fast start-up times for advanced power management
  - 22uA Standby-I mode, 900nA power down mode
  - Max 1.5ms start-up from power down mode
  - Max 130us start-up from standby-I mode
- Host Interface
  - 4-pin hardware SPI
  - Max 8Mbps
  - 3 separate 32 bytes TX and RX FIFOs
  - 5V tolerant inputs
- Compact 20-pin 4x4mm QFN package

1.2 Block diagram

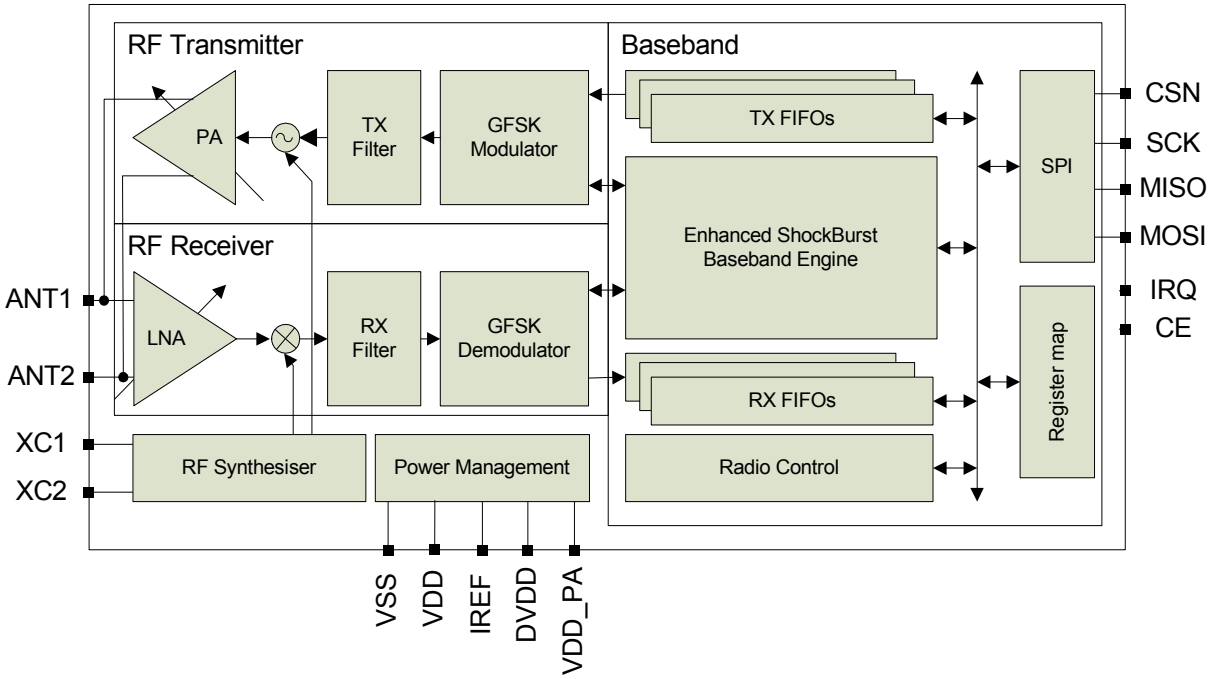


Figure 1. nRF24L01 block diagram

2 Pin Information

2.1 Pin assignment

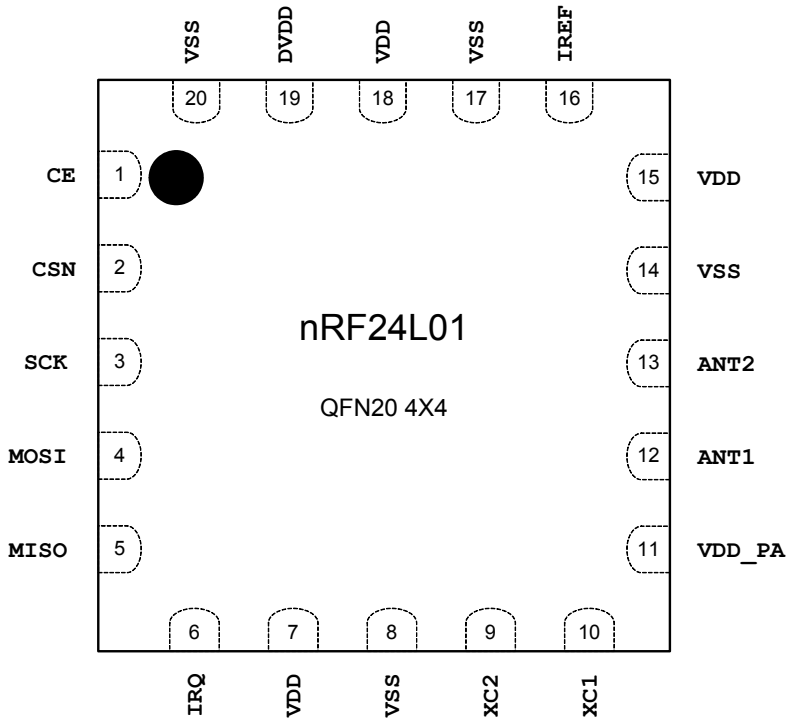


Figure 2. nRF24L01 pin assignment (top view) for the QFN20 4x4 package

2.2 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. Active low
7	VDD	Power	Power Supply (+1.9V - +3.6V 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 Output(+1.8V) for the internal nRF24L01 Power Amplifier. Must be connected to ANT1 and ANT2 as shown in Figure 30.
12	ANT1	RF	Antenna interface 1
13	ANT2	RF	Antenna interface 2
14	VSS	Power	Ground (0V)
15	VDD	Power	Power Supply (+1.9V - +3.6V DC)
16	IREF	Analog Input	Reference current. Connect a 22kΩ resistor to ground. See: Figure 30.
17	VSS	Power	Ground (0V)
18	VDD	Power	Power Supply (+1.9V - +3.6V DC)
19	DVDD	Power Output	Internal digital supply output for de-coupling purposes. See: Figure 30.
20	VSS	Power	Ground (0V)

Table 1. nRF24L01 pin function

3 Absolute maximum ratings

Note: Exceeding one or more of the limiting values may cause permanent damage to nRF24L01.

Operating conditions	Minimum	Maximum	Units
Supply voltages			
VDD	-0.3	3.6	V
VSS		0	V
Input voltage			
V_I	-0.3	5.25	V
Output voltage			
V_O	VSS to VDD	VSS to VDD	
Total Power Dissipation			
P_D (T_A=85°C)		60	mW
Temperatures			
Operating Temperature	-40	+85	°C
Storage Temperature	-40	+125	°C

Table 2. Absolute maximum ratings

4 Operating conditions

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
VDD	Supply voltage		1.9	3.0	3.6	V
VDD	Supply voltage if input signals >3.6V		2.7	3.0	3.3	V
TEMP	Operating Temperature		-40	+27	+85	ºC

Table 3. Operating conditions

5 Electrical specifications

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

5.1 Power consumption

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
	Idle modes					
I_{VDD_PD}	Supply current in power down			900		nA
I_{VDD_ST1}	Supply current in standby-I mode	a		22		µA
I_{VDD_ST2}	Supply current in standby-II mode			320		µA
I_{VDD_SU}	Average current during 1.5ms crystal oscillator startup			285		µA
	Transmit					
I_{VDD_TX0}	Supply current @ 0dBm output power	b		11.3		mA
I_{VDD_TX6}	Supply current @ -6dBm output power	b		9.0		mA
I_{VDD_TX12}	Supply current @ -12dBm output power	b		7.5		mA
I_{VDD_TX18}	Supply current @ -18dBm output power	b		7.0		mA
I_{VDD_AVG}	Average Supply current @ -6dBm output power, Enhanced ShockBurst™	c		0.12		mA
I_{VDD_TXS}	Average current during TX settling	d		8.0		mA
	Receive					
I_{VDD_2M}	Supply current 2Mbps			12.3		mA
I_{VDD_LC}	Supply current 2Mbps LNA low current			11.5		mA
I_{VDD_1M}	Supply current 1Mbps			11.8		mA
I_{VDD_LC}	Supply current 1Mbps LNA low current			11.1		mA
I_{VDD_RXS}	Average current during RX settling	e		8.4		mA

a. Current is given for a 12pF crystal. Current when using external clock is dependent on signal swing.
b. Antenna load impedance = 15Ω+j88Ω.
c. Antenna load impedance = 15Ω+j88Ω. Average data rate 10kbps and full packets
d. Average current consumption for TX startup (130µs) and when changing mode from RX to TX (130µs).
e. Average current consumption for RX startup (130µs) and when changing mode from TX to RX (130µs).

Table 4.Power consumption

5.2 General RF conditions

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
f_{OP}	Operating frequency	a	2400		2525	MHz
PLL_{res}	PLL Programming resolution			1		MHz
f_{XTAL}	Crystal frequency			16		MHz
Δf_{1M}	Frequency deviation @ 1Mbps			±160		kHz
Δf_{2M}	Frequency deviation @ 2Mbps			±320		kHz
R_{GFSK}	Air Data rate	b	1000		2000	kbps
$F_{CHANNEL\ 1M}$	Non-overlapping channel spacing @ 1Mbps	c		1		MHz
$F_{CHANNEL\ 2M}$	Non-overlapping channel spacing @ 2Mbps	c		2		MHz

a. Usable band is determined by local regulations
b. Data rate in each burst on-air
c. The minimum channel spacing is 1Mhz

Table 5. General RF conditions

5.3 Transmitter operation

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
P_{RF}	Maximum Output Power	a		0	+4	dBm
P_{RFC}	RF Power Control Range		16	18	20	dB
P_{RFCR}	RF Power Accuracy				±4	dB
P_{BW2}	20dB Bandwidth for Modulated Carrier (2Mbps)			1800	2000	kHz
P_{BW1}	20dB Bandwidth for Modulated Carrier (1Mbps)			900	1000	kHz
P_{RF1}	1^{st} Adjacent Channel Transmit Power 2MHz				-20	dBm
P_{RF2}	2^{nd} Adjacent Channel Transmit Power 4MHz				-50	dBm

a. Antenna load impedance = 15Ω+j88Ω

Table 6.Transmitter operation

5.4 Receiver operation

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
RX_{max}	Maximum received signal at <0.1% BER			0		dBm
RX_{SENS}	Sensitivity (0.1%BER) @2Mbps			-82		dBm
RX_{SENS}	Sensitivity at (0.1%BER) @1Mbps			-85		dBm
RX selectivity according to ETSI EN 300 440-1 V1.3.1 (2001-09) page 27						
C/I_{CO}	C/I Co-channel (@2Mbps)	a		7		dB
C/I_{1ST}	1^{st} Adjacent Channel Selectivity C/I 2MHz			1		dB
C/I_{2ND}	2^{nd} Adjacent Channel Selectivity C/I 4MHz			-21		dB
C/I_{3RD}	3^{rd} Adjacent Channel Selectivity C/I 6MHz			-27		dB
C/I_{CO}	C/I Co-channel (@1Mbps)	b		9		dB
C/I_{1ST}	1^{st} Adjacent Channel Selectivity C/I 1MHz			8		dB
C/I_{2ND}	2^{nd} Adjacent Channel Selectivity C/I 2MHz			-22		dB
C/I_{3RD}	3^{rd} Adjacent Channel Selectivity C/I 3MHz			-30		dB
RX selectivity with nRF24L01 equal modulation on interfering signal						
C/I_{CO}	C/I Co-channel (@2Mbps) (Modulated carrier)	a		11		dB
C/I_{1ST}	1^{st} Adjacent Channel Selectivity C/I 2MHz			4		dB
C/I_{2ND}	2^{nd} Adjacent Channel Selectivity C/I 4MHz			-20		dB
C/I_{3RD}	3^{rd} Adjacent Channel Selectivity C/I 6MHz			-27		dB
C/I_{CO}	C/I Co-channel (@1Mbps)	b		12		dB
C/I_{1ST}	1^{st} Adjacent Channel Selectivity C/I 1MHz			8		dB
C/I_{2ND}	2^{nd} Adjacent Channel Selectivity C/I 2MHz			-21		dB
C/I_{3RD}	3^{rd} Adjacent Channel Selectivity C/I 3MHz			-30		dB

a. Data rate is 2Mbps for the following C/I measurements
b. Data rate is 1Mbps for the following C/I measurements

Table 7. Receiver operation

5.5 Crystal specifications

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
Fxo	Crystal Frequency			16		MHz
ΔF	Tolerance	a b		±60		ppm
C_0	Equivalent parallel capacitance			1.5	7.0	pF
C_L	Load capacitance		8	12	16	pF
ESR	Equivalent Series Resistance				100	Ω

a. Frequency accuracy including; tolerance at 25°C, temperature drift, aging and crystal loading.
b. Frequency regulations in certain regions sets tighter requirements to frequency tolerance (Ex: Japan and Korea max. +/- 50ppm)

Table 8. Crystal specifications

5.6 DC characteristics

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
V_{IH}	HIGH level input voltage		0.7VDD		5.25[a]	V
V_{IL}	LOW level input voltage		VSS		0.3VDD	V

a. If the input signal >3.6V, the VDD of the nRF24L01 must be between 2.7V and 3.3V (3.0V±10%)

Table 9. Digital input pin

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
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)				0.3	V

Table 10. Digital output pin

5.7 Power on reset

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
T_{PUP}	Power ramp up time	a			100	ms
T_{POR}	Power on reset	b	1.6	5.3	10.3	ms

a. From 0V to 1.9V
b. Measured when the VDD reaches 1.9V to when the reset finishes

Table 11. Power on reset

6 Radio Control

This chapter describes the different modes the nRF24L01 radio transceiver can operate in and the parameters used to control the radio.

The nRF24L01 has a built-in state machine that controls the transitions between the different operating modes of the chip. The state machine takes input from user defined register values and internal signals.

6.1 Operational Modes

The nRF24L01 can be configured in four main modes of operation. This section describes these modes.

6.1.1 State diagram

The state diagram (Figure 3.) shows the modes the nRF24L01 can operate in and how they are accessed. The nRF24L01 is undefined until the **VDD** becomes 1.9V or higher. When this happens nRF24L01 enters the Power on reset state where it remains in reset until it enters the Power Down mode. Even when the nRF24L01 enters Power Down mode the MCU can control the chip through the SPI and the Chip Enable (**CE**) pin Three types of states are used in the state diagram. “Recommended operating mode” is a state that is used during normal operation. “Possible operating mode” is a state that is allowed to use, but it is not used during normal operation. “Transition state” is a time limited state used during start up of the oscillator and settling of the PLL.

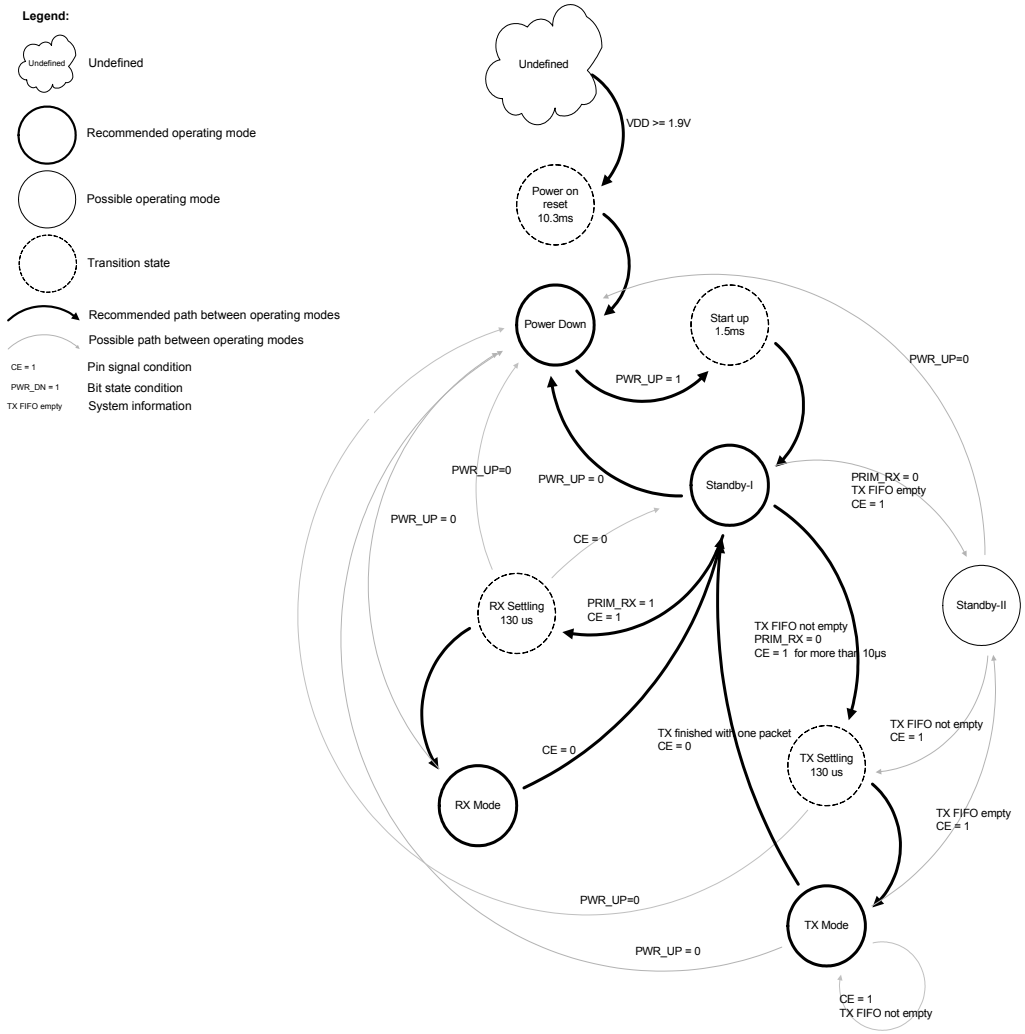


Figure 3. Radio control state diagram

6.1.2 Power Down Mode

In power down mode nRF24L01 is disabled with minimal current consumption. In power down mode all the register values available from the SPI are maintained and the SPI can be activated. For start up time see Table 13. on page 22. Power down mode is entered by setting the `PWR_UP` bit in the `CONFIG` register low.

6.1.3 Standby Modes

By settting the `PWR_UP` bit in the `CONFIG` register to 1, the device enters standby-I mode. 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. This is the mode the nRF24L01 returns to from TX or RX mode when **`CE`** is set low.

In standby-II mode extra clock buffers are active compared to standby-I mode and much more current is used compared to standby-I mode. Standby-II occurs when **`CE`** is held high on a PTX device with empty TX FIFO. If a new packet is uploaded to the TX FIFO, the PLL starts and the packet is transmitted.

The register values are maintained during standby modes and the SPI may be activated. For start up time see Table 13. on page 22.

6.1.4 RX mode

The RX mode is an active mode where the nRF24L01 radio is a receiver. To enter this mode, the nRF24L01 must have the PWR_UP bit set high, PRIM_RX bit set high and the **CE** pin set high.

In this mode the receiver demodulates the signals from the RF channel, constantly presenting the demodulated data to the baseband protocol engine. The baseband protocol engine constantly searches for a valid packet. If a valid packet is found (by a matching address and a valid CRC) the payload of the packet is presented in a vacant slot in the RX FIFO. If the RX FIFO is full, the received packet is discarded.

The nRF24L01 remains in RX mode until the MCU configures it to standby-I mode or power down mode. If the automatic protocol features (Enhanced ShockBurst™) in the baseband protocol engine are enabled, the nRF24L01 can enter other modes in order to execute the protocol.

In RX mode a carrier detect signal is avaliable. The carrier detect is a signal that is set high when a RF signal is detected inside the receiving frequency channel. The signal must be FSK modulated for a secure detection. Other signals can also be detected. The Carrier Detect (CD) is set high when an RF signal is detected in RX mode, otherwise CD is low. The internal CD signal is filtered before presented to CD register. The RF signal must be present for at least 128µs before the CD is set high. How to use the CD is described in Appendix E on page 74.

6.1.5 TX mode

The TX mode is an active mode where the nRF24L01 transmits a packet. To enter this mode, the nRF24L01 must have the PWR_UP bit set high, PRIM_RX bit set low, a payload in the TX FIFO and, a high pulse on the **CE** for more than 10µs.

The nRF24L01 stays in TX mode until it finishes transmitting a current packet. If **CE** = 0 nRF24L01 returns to standby-I mode. If **CE** = 1, the next action is determined by the status of the TX FIFO. If the TX FIFO is not empty the nRF24L01 remains in TX mode, transmitting the next packet. If the TX FIFO is empty the nRF24L01 goes into standby-II mode.The nRF24L01 transmitter PLL operates in open loop when in TX mode. It is important to never keep the nRF24L01 in TX mode for more than 4ms at a time. If the auto retransmit is enabled, the nRF24L01 is never in TX mode long enough to disobey this rule.

6.1.6 Operational modes configuration

The following table (Table 12.) describes how to configure the operational modes.

Mode	PWR_UP register	PRIM_RX register	CE	FIFO state
RX mode	1	1	1	-
TX mode	1	0	1	Data in TX FIFO. Will empty all levels in TX FIFO[a].
TX mode	1	0	minimum 10µs high pulse	Data in TX FIFO.Will empty one level in TX FIFO[b].
Standby-II	1	0	1	TX FIFO empty
Standby-I	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

a. In this operating mode if the **CE** is held high the TX FIFO is emptied and all necessary ACK and possible retransmits are carried out. The transmission continues as long as the TX FIFO is refilled. If the TX FIFO is empty when the **CE** is still high, nRF24L01 enters standby-II mode. In this mode the transmission of a packet is started as soon as the **CSN** is set high after a upload (UL) of a packet to TX FIFO.
b. This operating mode pulses the **CE** high for at least 10µs. This allows one packet to be transmitted. This is the normal operating mode. After the packet is transmittet, the nRF24L01 enters standby-I mode.

Table 12. nRF24L01 main modes

6.1.7 Timing Information

The timing information in this section is related to the transitions between modes and the timing for the **CE** pin. The transition from TX mode to RX mode or vice versa is the same as the transition from standby-I to TX mode or RX mode,Tstby2a.

Name	nRF24L01	Max.	Min.	Comments
Tpd2stby	Power Down ➔ Standby mode	1.5ms		Internal crystal oscillator
Tpd2stby	Power Down ➔ Standby mode	150µs		With external clock
Tstby2a	Standby modes ➔ TX/RX mode	130µs		
Thce	Minimum **CE** high		10µs	
Tpece2csn	Delay from **CE** pos. edge to **CSN** low		4µs	

Table 13. Operational timing of nRF24L01

When nRF24L01 is in power down mode it must settle for 1.5ms before it can enter the TX or RX modes. If an *external clock* is used this delay is reduced to 150µs, see Table 13. on page 22. The settling time must be controlled by the MCU.

Note: The register value is lost if **VDD** is turned off. In this case, nRF24L01 must be configured before entering the TX or RX modes.

6.2 Air data rate

The air data rate is the modulated signaling rate the nRF24L01 uses when transmitting and receiving data.

The air data rate can be 1Mbps or 2Mbps. The 1Mbps data rate gives 3dB better receiver sensitivity compared to 2Mbps. High air data rate means lower average current consumption and reduced probability of on-air collisions.

The air data rate is set by the `RF_DR` bit in the `RF_SETUP` register.

A transmitter and a receiver must be programmed with the same air data rate to be able to communicate with each other.

For compatibility with nRF2401A, nRF24E1, nRF2402 and nRF24E2 the air data rate must be set to 1Mbps.

6.3 RF channel frequency

The RF channel frequency determines the center of the channel used by the nRF24L01. The channel occupies a bandwidth of 1MHz at 1Mbps and 2MHz at 2Mbps. nRF24L01 can operate on frequencies from 2.400GHz to 2.525GHz. The resolution of the RF channel frequency setting is 1MHz.

At 2Mbps the channel occupies a bandwidth wider than the resolution of the RF channel frequency setting. To ensure non-overlapping channels in 2Mbps mode, the channel spacing must be 2MHz or more. At 1Mbps the channel bandwidth is the same as the resolution of the RF frequency setting.

The RF channel frequency is set by the RF_CH register according to the following formula:

F_0= 2400 + RF_CH [MHz]

A transmitter and a receiver must be programmed with the same RF channel frequency to be able to communicate with each other.

6.4 PA control

The PA control is used to set the output power from the nRF24L01 power amplifier (PA). In TX mode PA control has four programmable steps, see Table 14.

The PA control is set by the RF_PWR bits in the RF_SETUP register.

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

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

Table 14. RF output power setting for the nRF24L01

6.5 LNA gain

The gain in the Low Noise Amplifier (LNA) in the nRF24L01 receiver is controlled by the LNA gain setting. The LNA gain makes it possible to reduce the current consumption in RX mode with 0.8mA at the cost of 1.5dB reduction in receiver sensitivity.

The LNA gain has two steps and is set by the LNA_HCURR bit in the RF_SETUP register.

6.6 RX/TX control

The RX/TX control is set by PRIM_RX bit in the CONFIG register and sets the nRF24L01 in transmit/receive.

7 Enhanced ShockBurst™

Enhanced ShockBurst™ is a packet based data link layer. It features automatic packet assembly and timing, automatic acknowledgement and re-transmissions of packets. Enhanced ShockBurst™ enables the implementation of ultra low power, high performance communication with low cost host microcontrollers. The features enable significant improvements of power efficiency for bi-directional and uni-directional systems, without adding complexity on the host controller side.

7.1 Features

The main features of Enhanced ShockBurst™ are:

- 1 to 32 bytes dynamic payload length
- Automatic packet handling
- Auto packet transaction handling
  - Auto Acknowledgement
  - Auto retransmit
- 6 data pipe MultiCeiver™ for 1:6 star networks

7.2 Enhanced ShockBurst™ overview

Enhanced ShockBurst™ uses ShockBurst™ for automatic packet handling and timing. During transmit, ShockBurst™ assembles the packet and clocks the bits in the data packet into the transmitter for transmission. During receive, ShockBurst™ constantly searches for a valid address in the demodulated signal. When ShockBurst™ finds a valid address, it processes the rest of the packet and validates it by CRC. If the packet is valid the payload is moved into the RX FIFO. The high speed bit handling and timing is controlled by ShockBurst™.

Enhanced ShockBurst™ features automatic packet transaction handling that enables the implementation of a reliable bi-directional data link. An Enhanced ShockBurst™ packet transaction is a packet exchange between to transceivers, where one transceiver is the Primary Receiver (PRX) and the other is the Primary Transmitter (PTX). An Enhanced ShockBurst™ packet transaction is always initiated by a packet transmission from the PTX, the transaction is complete when the PTX has received an acknowledgment packet (ACK packet) from the PRX.

The automatic packet transaction handling works as follows:

- The user initiates the transaction by transmitting a data packet from the PTX to the PRX. Enhanced ShockBurst™ automatically sets the PTX in receive mode to wait for the ack packet.
- If the packet is received by the PRX, Enhanced ShockBurst™ automatically assembles and transmits an acknowledgment packet (ACK packet) to the PTX before returning to receive mode
- If the PTX does not receive the ACK packet within a set time, Enhanced ShockBurst™ will automatically retransmit the original data packet and set the PTX in receive mode to wait for the ACK packet

The PRX can attach user data to the ACK packet enabling a bi-directional data link. The Enhanced ShockBurst™ is highly configurable; it is possible to configure parameters such as maximum number of retransmits and the delay from one transmission to the next retransmission. All automatic handling is done without involvement of the MCU.

Section 7.3 on page 25 gives a description of the Enhanced ShockBurst packet format, section 7.4 on page 26 describes autmatic packet handling, section 7.5 on page 28 describes automatic packet transaction handling, section 7.6 on page 31 provides flowcharts for PTX and PRX operation.

7.3 Enhanced Shockburst™ packet format

The format of the Enhanced ShockBurst™ packet is described in this chapter. The Enhanched Shock-Burst™ packet contains a preamble field, address field, packet control field, payload field and a CRC field. Figure 4. on page 25 shows the packet format with MSB to the left.

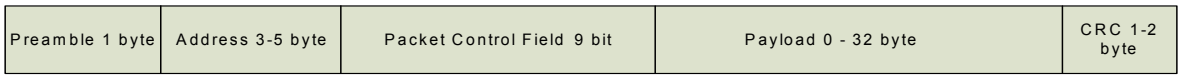


Figure 4. An Enhanced ShockBurst™ packet with payload (0-32 bytes)

7.3.1 Preamble

The preamble is a bit sequence used to detect 0 and 1 levels in the receiver. The preamble is one byte long and is either 01010101 or 10101010. If the first bit in the address is 1 the preamble is automatically set to 10101010 and if the first bit is 0 the preamble is automatically set to 01010101. This is done to ensure there are enough transitions in the preamble to stabilize the receiver.

7.3.2 Address

This is the address for the receiver. An address ensures that the correct packet are detected by the receiver. The address field can be configured to be 3, 4 or, 5 bytes long with the `AW` register.

Note: Addresses where the level shifts only one time (that is, 000FFFFFFF) can often be detected in noise and can give a false detection, which may give a raised Packet-Error-Rate. Addresses as a continuation of the preamble (hi-low toggling) raises the Packet-Error-Rate.

7.3.3 Packet Control Field

Figure 5 shows the format of the 9 bit packet control field, MSB to the left.

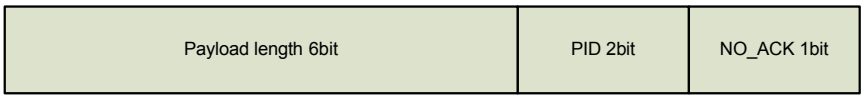


Figure 5. Packet control field

The packet control field contains a 6 bit payload length field, a 2 bit PID (Packet Identity) field and, a 1 bit `NO_ACK` flag.

7.3.3.1 Payload length

This 6 bit field specifies the length of the payload in bytes. The length of the payload can be from 0 to 32 bytes.

Coding: 000000 = 0 byte (only used in empty ACK packets.) 100000 = 32 byte, 100001 = Don't care.

This field is only used if the Dynamic Payload Length function is enabled.

7.3.3.2 PID (Packet identification)

The 2 bit PID field is used to detect if the received packet is new or retransmitted. PID prevents the PRX device from presenting the same payload more than once to the MCU. The PID field is incremented at the TX side for each new packet received through the SPI. The PID and CRC fields (see section 7.3.5 on page 26) are used by the PRX device to determine if a packet is retransmitted or new. When several data packets are lost on the link, the PID fields may become equal to the last received PID. If a packet has the same PID as the previous packet, nRF24L01 compares the CRC sums from both packets. If the CRC sums are also equal, the last received packet is considered a copy of the previously received packet and discarded.

7.3.3.3 No Acknowledgment flag(NO_ACK)

The Selective Auto Acknowledgement feature controls the NO_ACK flag.

This flag is only used when the auto acknowledgement feature is used. Setting the flag high, tells the receiver that the packet is not to be auto acknowledged.

7.3.4 Payload

The payload is the user defined content of the packet. It can be 0 to 32 bytes wide and is transmitted on-air as it is uploaded (unmodified) to the device.

7.3.5 CRC (Cyclic Redundancy Check)

The CRC is the error detection mechanism in the packet. It may either be 1 or 2 bytes and is calculated over the address, Packet Control Field, and Payload.

The polynomial for 1 byte CRC is $X^8 + X^2 + X + 1$. Initial value 0xFF

The polynomial for 2 byte CRC is $X^{16} + X^{12} + X^5 + 1$. Initial value 0xFFFF

No packet is accepted by Enhanced ShockBurst™ if the CRC fails.

7.4 Automatic packet handling

Enhanced ShockBurst™ uses ShockBurst™ for automatic packet handling. The functions are static and dynamic payload length, automatic packet assembly, automatic packet validation and automatic packet disassembly.

7.4.1 Static and Dynamic Payload Length

The Enhanced ShockBurst™ provides two alternatives for handling payload lengths, static and dynamic.

The default alternative is static payload length. With static payload length all packets between a transmitter and a receiver have the same length. Static payload length is set by the RX_PW_Px registers on the receiver side. The payload length on the transmitter side is set by the number of bytes clocked into the TX_FIFO and must equal the value in the RX_PW_Px register on the receiver side

Dynamic Payload Length(DPL) is an alternative to static payload length.DPL enables the transmitter to send packets with variabel payload length to the receiver. This means for a system with different payload lenghts it is not necessary to scale the packet length to the longest payload.

With DPL feature the nRF24L01 can decode the payload length of the received packet automatically instead of using the RX_PW_Px registers. The MCU can read the length of the received payload by using the R_RX_PL_WID command.

In order to enable DPL the EN_DPL bit in the FEATURE register must be set. In RX mode the DYNPD register has to be set. A PTX that transmits to a PRX with DPL enabled must have the DPL_P0 bit in DYNPD set.

7.4.2 Automatic packet assembly

The automatic packet assembly assembles the preamble, address, packet control field, payload and CRC to make a complete packet before it is transmitted.

7.4.2.1 Preamble

The preamble is automaticly generated based on the address field.

7.4.2.2 Address

The address is fetched from the TX_ADDR register. The address field can be configured to be 3, 4 or 5 bytes long with the AW register.

7.4.2.3 Packet control field

For the static packet lenght option the payload length field is not used. With DPL enabled, the value in the payload length field is automaticly set to the number of bytes in the payload clocked into the TX FIFO.

The transmitter increments the PID field each time it generates a new packet and uses the same PID on packets that are retransmitted. Refer to the left flow chart in Figure 6. on page 28

The PTX can set the NO_ACK flag bit in the Packet Control Field with this command:

W_TX_PAYLOAD_NOACK

However, the function must first be enabled in the FEATURE register by setting the EN_DYN_ACK bit. When you use this option the PTX goes directly to standby-I mode after transmitting the packet and the PRX does not transmit an ACK packet when it receives the packet.

7.4.2.4 Payload

The payload is fetched from the TX FIFO.

7.4.2.5 CRC

The CRC is automaticly calculated based on the packet content with the polynomials in 7.3.5 on page 26.

The number of bytes in the CRC is set by the CRCO bit in the CONFIG register.

7.4.3 Automatic packet validation

Enhanced ShockBurst™ features automatic packet validation. In receive mode the nRF24L01 is constanly searching for a valid address (given in the RX_ADDR registers.) If a valid address is detected the Enhanched ShockBurst™ will start to validate the packet.

With static packet length the Enhanced ShockBurst™ will capture the packet according to the length given by the RX_PW register. With DPL Enhanced ShockBurst™ captures the packet according to the payload lenght field in the packet control field. After capturing the packet Enhanced ShockBurst™ will perform CRC.

If the CRC is valid, Enhanced ShockBurst™ will check PID. The received PID is compared with the previous received PID. If the PID fields are different, the packet is considered new. If the PID fields are equal the receiver must check if the received CRC is equal to the previous CRC. If the CRCs are equal, the packet is defined as equal to the previous packet and is discarded. Refer to the right flow chart in Figure 6. on page 28

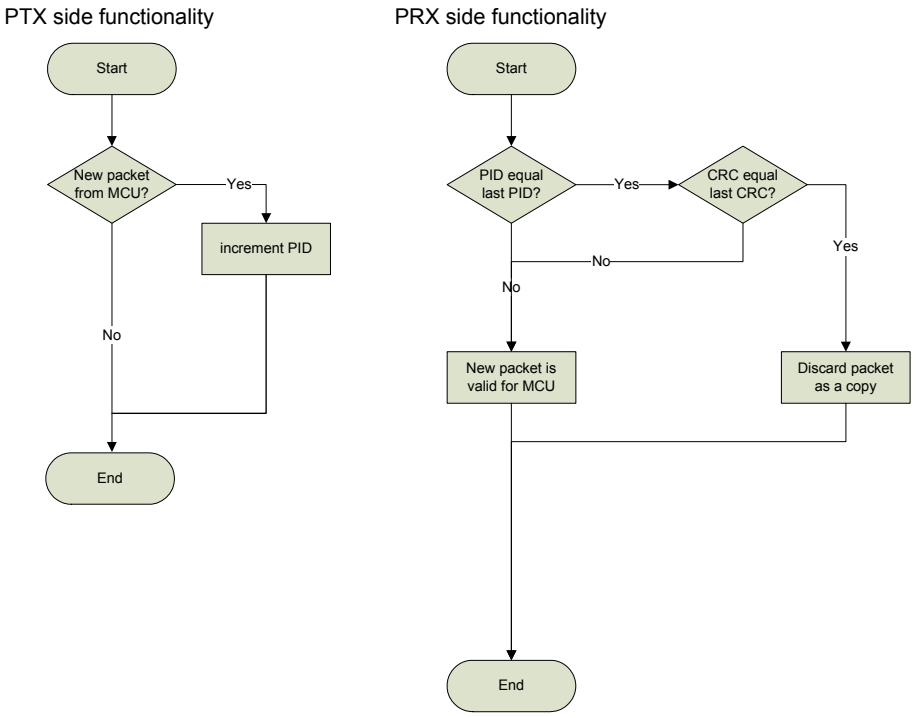


Figure 6. PID generation/detection

7.4.4 Automatic packet disassembly

After the packet is validated,Enhanched ShockBurst™ disassembles the packet and loads the payload into the RX FIFO, and assert the RX_DR IRQ

7.5 Automatic packet transaction handling

Enhanced ShockBurst™ features two functions for automatic packet transaction handling; auto acknowledgement and auto re-transmit.

7.5.1 Auto Acknowledgement

Auto acknowledgment is a function that automatically transmits an ACK packet to the PTX after it has received and validated a packet. The auto acknowledgement function reduces the load of the system MCU and can remove the need for dedicated SPI hardware. This also reduces cost and average current consumption. The Auto Acknowledgement feature is enabled by setting the EN_AA register.

Note: If the received packet has the NO_ACK flag set, the auto acknowledgement is not executed.

An ACK packet can contain an optional payload from PRX to PTX. In order to use this feature, the dynamic payload length feature mus be enabled. The MCU on the PRX side has to upload the payload by clocking it into the TX FIFO by using the W_ACK_PAYLOAD command. The payload is pending in the TX FIFO (PRX) until a new packet is received from the PTX. nRF24L01 can have three ACK packet payloads pending in the TX FIFO (PRX) at the same time.

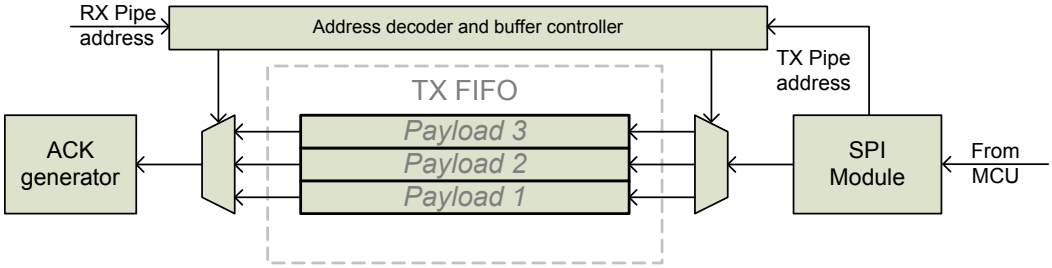


Figure 7. TX FIFO (PRX) with pending payloads

Figure 7. shows how the TX FIFO (PRX) is operated when handling pending ACK packet payloads. From the MCU the payload is clocked in with the W_ACK_PAYLOAD command. The address decoder and buffer controller ensure that the payload is stored in a vacant slot in the TX FIFO (PRX). When a packet is received, the address decoder and buffer controller are notified with the PTX address. This ensures that the right payload is presented to the ACK generator.

If the TX FIFO (PRX) contains more than one payload to a PTX, payloads are handled using the first in – first out principle. The TX FIFO (PRX) is blocked if all pending payloads are addressed to a PTX where the link is lost. In this case, the MCU can flush the TX FIFO (PRX) by using the FLUSH_TX command.

In order to enable Auto Acknowledgement with payload the EN_ACK_PAY bit in the FEATURE register must be set.

7.5.2 Auto Retransmission (ART)

The auto retransmission is a function that retransmits a packet if an ACK packet is not received. It is used at the PTX side in an auto acknowledgement system. You can set up the number of times a packet is allowed to be retransmitted if a packet is not acknowledged with the ARC bits in the SETUP_RETR register. PTX enters RX mode and waits a time period for an ACK packet each time a packet is transmitted. The time period the PTX is in RX mode is based on the following conditions:

- Auto Retransmit Delay (ARD) elapsed or
- No address match within 250µs or
- After received packet (CRC correct or not) if address match within 250µs

nRF24L01 asserts the TX_DS IRQ when the ACK packet is received

nRF24L01 enters standby-I mode if there is no more untransmitted data in the TX FIFO and the **CE** pin is low. If the ACK packet is not received, nRF24L01 goes back to TX mode after a delay defined by ARD and retransmits the data. This continues until acknowledgment is received, or the maximum number of retransmits is reached. Set PWR_UP =0 to abort auto retransmission. Two packet loss counters are incremented each time a packet is lost, ARC_CNT and PLOS_CNT in the OBSERVE_TX register. The ARC_CNT counts the number of retransmissions for the current transaction. The PLOS_CNT counts the total number of retransmissions since the last channel change. You reset ARC_CNT by initiating a new transaction. You reset PLOS_CNT by writing to the RF_CH register. It is possible to use the information in the OBSERVE_TX register to make a overall assessment of the channel quality.

The ARD defines the time from the end of a transmitted packet to a retransmit starts on the PTX side. ARD is set in SETUP_RETR register in steps of 250µs. A retransmit is made if no ACK packet is received by the PTX.

There is a restriction for the length of ARD when using ACK packets with payload. The ARD time must never be shorter than the sum of the startup time and the time on-air for the ACK packet.

For 1Mbps data rate and 5 byte address; 5 byte is maximum ACK packet payload length for ARD=250µs (reset value).

For 2Mbps data rate and 5 byte address; 15 byte is maximum ACK packet payload length for ARD=250µs (reset value).

ARD=500µs will be long enough for any payload length.

As an alternative to Auto Retransmit it is possible to manually set the nRF24L01 to retransmit a packet a number of times. This is done by the REUSE_TX_PL command. The MCU must initiate each transmission of the packet with the **CE** pin after this command has been used.

7.6 Enhanced ShockBurst flowcharts

This section shows flowcharts for PTX and PRX operation in Enhanced ShockBurst™. ShockBurst™ operation is marked with a dashed square in the flow charts.

7.6.1 PTX operation

The flowchart in Figure 8. shows how a nRF24L01 configured as a PTX behaves after entering standby-I mode.

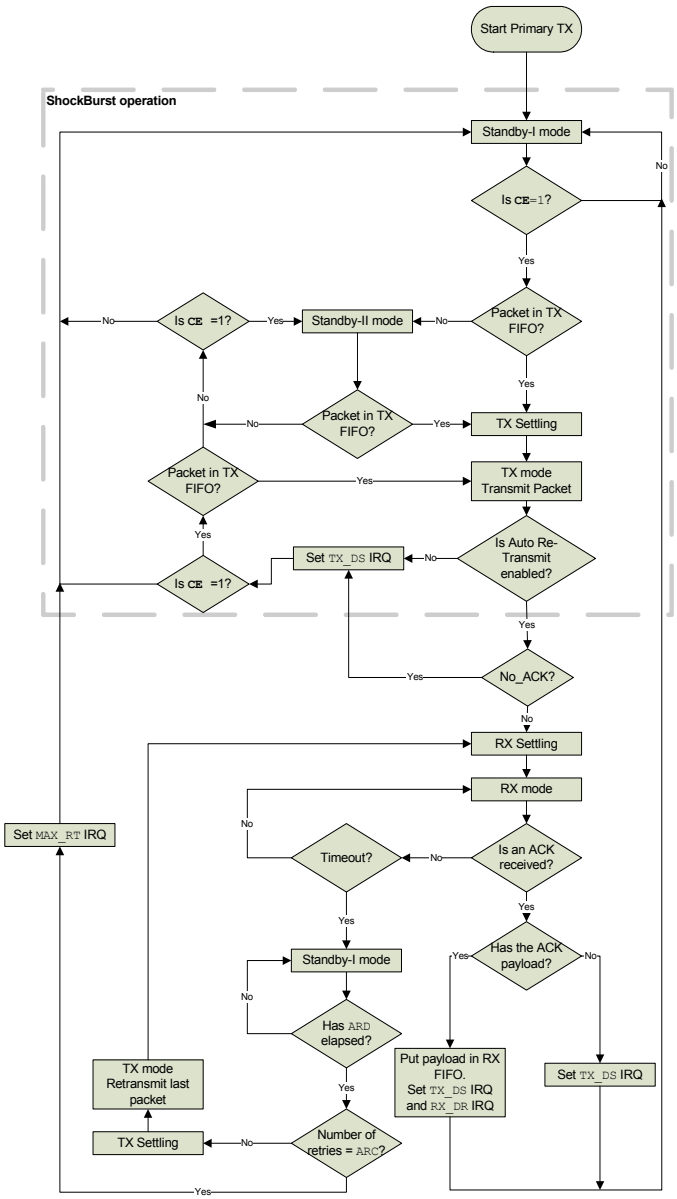


Figure 8. PTX operations in Enhanced ShockBurst™

You activate PTX mode by setting the CE pin high. If there is a packet present in the TX FIFO the nRF24L01 enters TX mode and transmits the packet. If Auto Retransmit is enabled, the state machine

checks if the NO_ACK flag is set. If it is not set, the nRF24L01 enters RX mode to receive an ACK packet. If the received ACK packet is empty, only the TX_DS IRQ is asserted. If the ACK packet contains a payload, both TX_DS IRQ and RX_DR IRQ are asserted simultaneously before nRF24L01 returns to standby-I mode.

If the ACK packet is not received before timeout occurs, the nRF24L01 returns to standby-I mode. It stays in standby-I mode until the ARD has elapsed. If the number of retransmits has not reached the ARC, the nRF24L01 enters TX mode and transmits the last packet once more.

While executing the Auto Retransmit feature, the number of retransmits can reach the maximum number defined in ARC. If this happens, the nRF24L01 asserts the MAX_RT IRQ and returns to standby-I mode.

If the **CE** is high and the TX FIFO is empty, the nRF24L01 enters Standby-II mode.

7.6.2 PRX operation

The flowchart in Figure 9. shows how a nRF24L01 configured as a PRX behaves after entering standby-I mode.

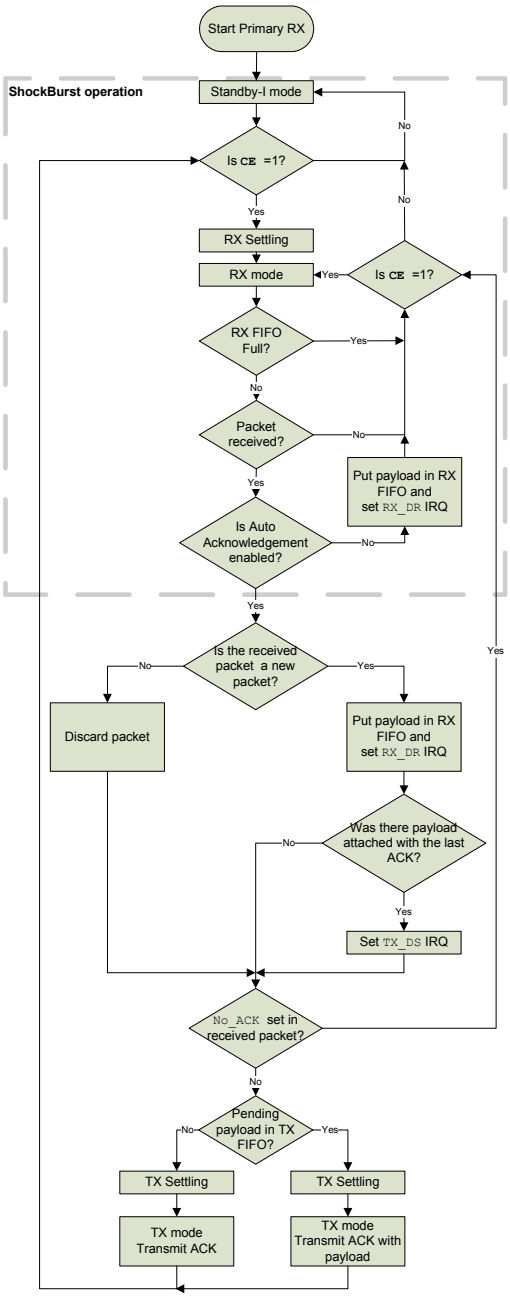


Figure 9. PRX operations in Enhanced ShockBurst™

You activate PRX mode by setting the **CE** pin high. The nRF24L01 enters RX mode and starts searching for packets. If a packet is received and Auto Acknowledgement is enabled the nRF24L01 decides if this is a new packet or a copy of a previously received packet. If the packet is new the payload is made available in the RX FIFO and the RX_DR IRQ is asserted. If the last received packet from the transmitter is acknowledged with an ACK packet with payload, the TX_DS IRQ indicates that the PTX received the ACK packet

with payload. If the No_ACK flag is not set in the received packet, the PRX enters TX mode. If there is a pending payload in the TX FIFO it is attached to the ACK packet. After the ACK packet is transmitted, the nRF24L01 returns to RX mode.

A copy of a previously received packet might be received if the ACK packet is lost. In this case, the PRX discards the received packet and transmits an ACK packet before it returns to RX mode.

7.7 Multiceiver

Multiceiver is a feature used in RX mode that contains a set of 6 parallel data pipes with unique addresses. A data pipe is a logical channel in the physical RF channel. Each data pipe has its own physical address decoding in the nRF24L01.

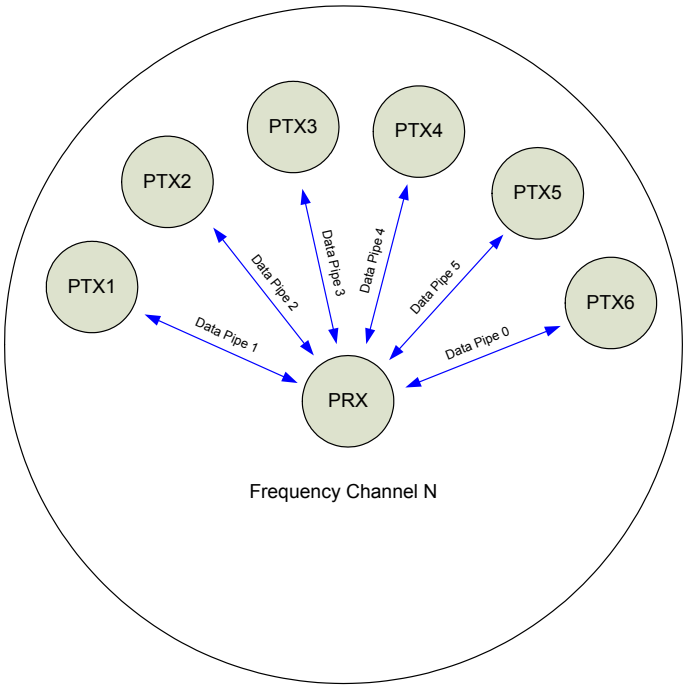


Figure 10. PRX using multiceiver

nRF24L01 configured as PRX (primary receiver) can receive data addressed to six different data pipes in one frequency channel as shown in Figure 10. Each data pipe has its own unique address and can be configured for individual behavior.

Up to six nRF24L01s configured as PTX can communicate with one nRF24L01 configured as PRX. All data pipe addresses are searched for simultaneously. Only one data pipe can receive a packet at a time. All data pipes can perform Enhanced ShockBurst™ functionality.

The following settings are common to all data pipes:

- CRC enabled/disabled (CRC always enabled when Enhanced ShockBurst™ is enabled)
- CRC encoding scheme
- RX address width
- Frequency channel
- Air data rate
- LNA gain

The data pipes are enabled with the bits in the `EN_RXADDR` register. By default only data pipe 0 and 1 are enabled.

Each data pipe address is configured in the `RX_ADDR_PX` registers.

Note: Always ensure that none of the data pipes have the same address.

Each pipe can have up to 5 byte configurable address. Data pipe 0 has a unique 5 byte address. Data pipes 1-5 share the 4 most significant address bytes. The LSByte must be unique for all 6 pipes. Figure 11. is an example of how data pipes 0-5 are addressed.

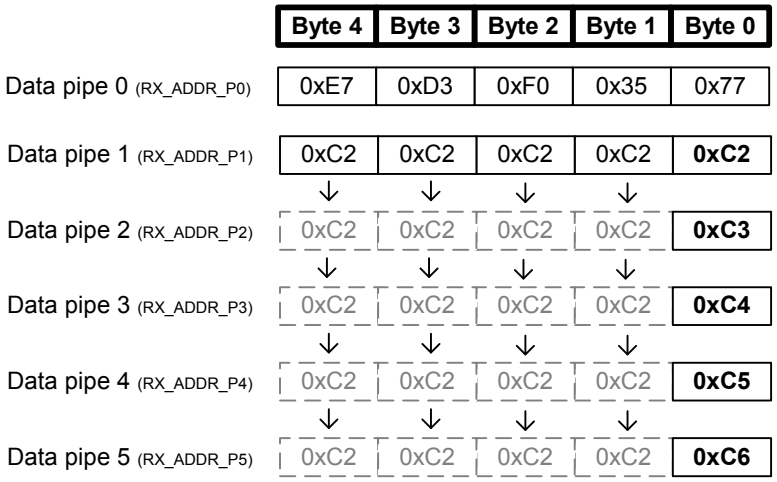


Figure 11. Addressing data pipes 0-5

The PRX, using multiceiver and Enhanced ShockBurst™, receives packets from more than one PTX. To ensure that the ACK packet from the PRX is transmitted to the correct PTX, the PRX takes the data pipe address where it received the packet and uses it as the TX address when transmitting the ACK packet. Figure 12. is an example of how address configuration could be for the PRX and PTX. On the PRX the RX_ADDR_Pn, defined as the pipe address, must be unique. On the PTX the TX_ADDR must be the same as the RX_ADDR_P0 and as the pipe address for the designated pipe.

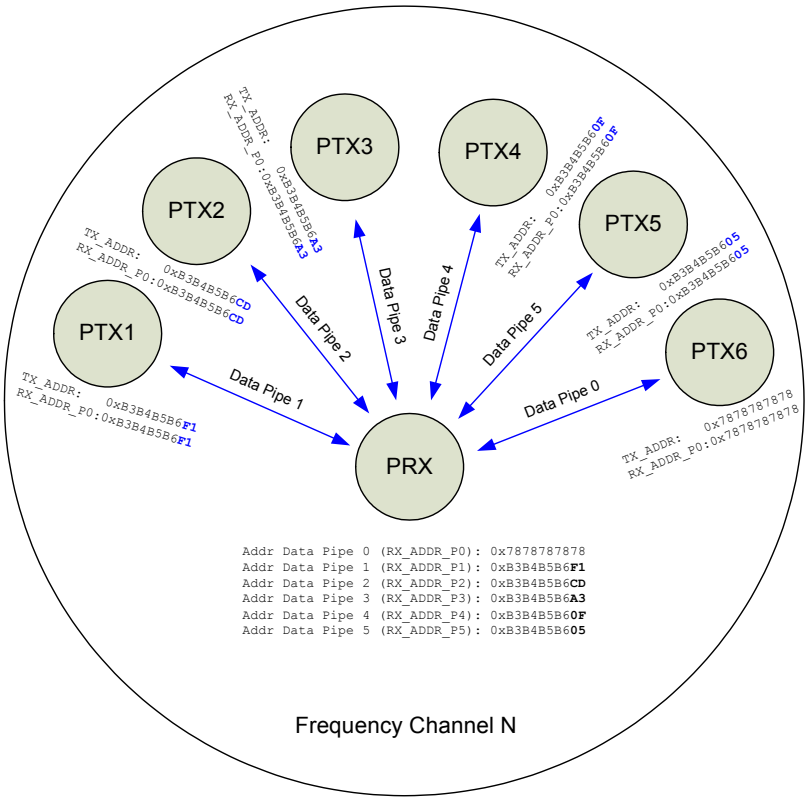
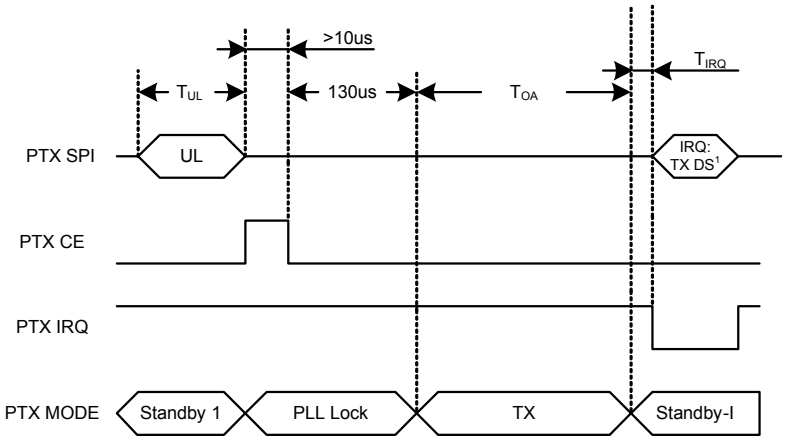


Figure 12. Example of data pipe addressing in multiceiver

No other data pipe can receive data until a complete packet is received by a data pipe that has detected its address. When multiple PTXs are transmitting to a PRX, the ARD can be used to skew the auto retransmission so that they only block each other once.

7.8 Enhanced ShockBurst™ timing

This section describes the timing sequence of Enhanced ShockBurst™ and how all modes are initiated and operated. The Enhanced ShockBurst™ timing is controlled through the Data and Control interface. The nRF24L01 can be set to static modes or autonomous modes where the internal state machine controls the events. Each autonomous mode/sequence is ended with an interrupt at the **IRQ** pin. All the interrupts are indicated as IRQ events in the timing diagrams.



1 IRQ if No Ack is on.

T_{IRQ} = 8.2ηs @ 1Mbps, T_{IRQ} = 6.0ηs @ 2Mbps

Figure 13. Transmitting one packet with `NO_ACK` on

The following equations calculate various timing measurements:

Symbol	Description	Equation
T_{OA}	Time on-air	$T_{OA} = \frac{packet\ length}{air\ data\ rate} = \frac{8\left[\frac{bit}{byte}\right]\cdot\left(1[byte]_{preamble} + 3,4\ or\ 5[bytes]_{address} + N[bytes]_{payload} + 1\ or\ 2[bytes]_{CRC}\right) + 9[bit]_{packet\ control\ field}}{air\ data\ rate\left[\frac{bit}{s}\right]}$
T_{ACK}	Time on-air Ack	$T_{ACK} = \frac{packet\ length}{air\ data\ rate} = \frac{8\left[\frac{bit}{byte}\right]\cdot\left(1[byte]_{preamble} + 3,4\ or\ 5[bytes]_{address} + N[bytes]_{payload} + 1\ or\ 2[bytes]_{CRC}\right) + 9[bit]_{packet\ control\ field}}{air\ data\ rate\left[\frac{bit}{s}\right]}$
T_{UL}	Time Upload	$T_{UL} = \frac{payload\ length}{SPI\ data\ rate} = \frac{8\left[\frac{bit}{byte}\right]\cdot N[bytes]_{payload}}{SPI\ data\ rate\left[\frac{bit}{s}\right]}$
T_{ESB}	Time Enhanced ShockBurst™ cycle	$T_{ESB} = T_{UL} + 2\cdot T_{stby2a} + T_{ACK} + T_{IRQ}$

Table 15. Timing equations

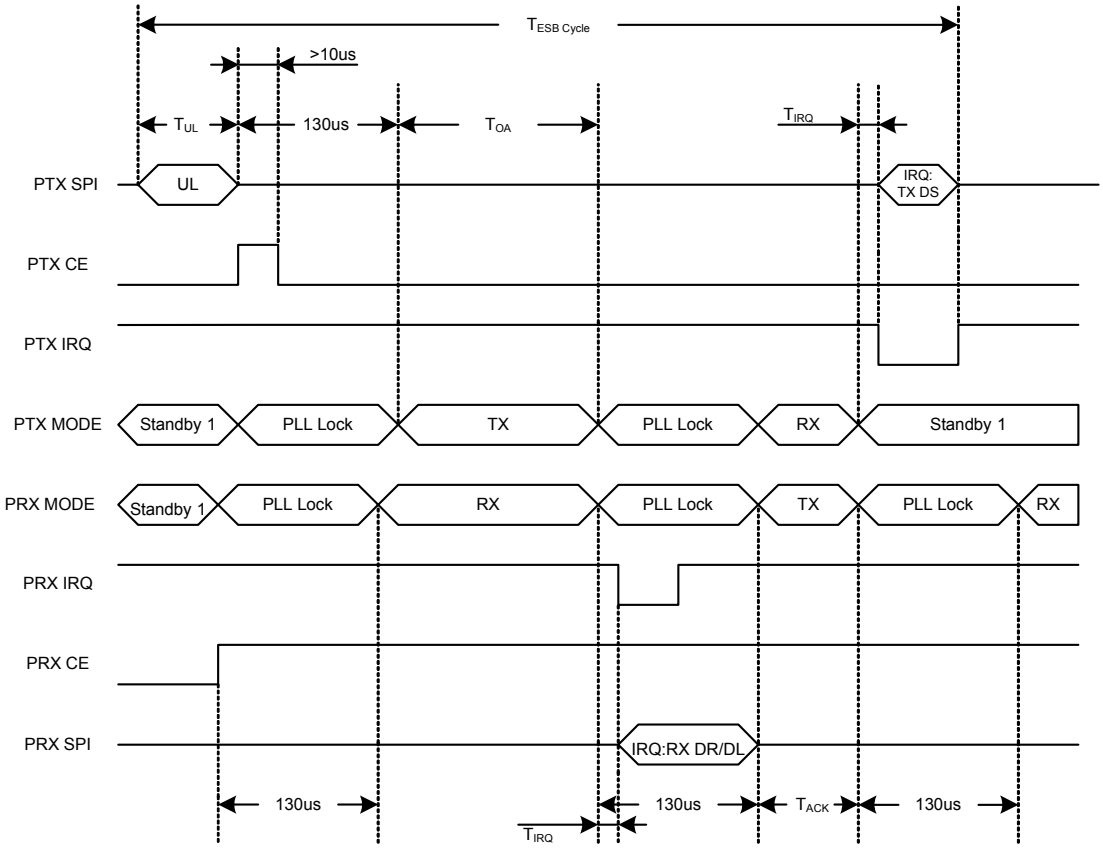


Figure 14. Timing of Enhanced ShockBurst™ for one packet upload (2Mbps)

In Figure 14. the transmission and acknowledgement of a packet are shown. The PRX device is turned into RX mode (CE=1), and the PTX device is set to TX mode (CE=1 for minimum 10µs). After 130µs the transmission starts and finishes after the elapse of T_{OA}.

When the transmission ends the PTX device automatically switches to RX mode to wait for the ACK packet from the PRX device. After the PTX device receives the ACK packet it responds with an interrupt to the MCU. When the PRX device receives the packet it responds with an interrupt to the MCU.

7.9 Enhanced ShockBurst™ transaction diagram

This section describes how several scenarios for the Enhanced ShockBurst™ automatic transaction handling. The call outs in this section's figures indicate the IRQs and other events. For MCU activity the event may be placed at a different timeframe.

Note: The figures in this section indicate the earliest possible download (DL) of the packet to the MCU and the latest possible upload (UL) of payload to the transmitter.

7.9.1 Single transaction with ACK packet and interrupts

In Figure 15. the basic auto acknowledgement is shown. After the packet is transmitted by the PTX and received by the PRX the ACK packet is transmitted from the PRX to the PTX. The `RX_DR` IRQ is asserted after the packet is received by the PRX, whereas the `TX_DS` IRQ is asserted when the packet is acknowledged and the ACK packet is received by the PTX.

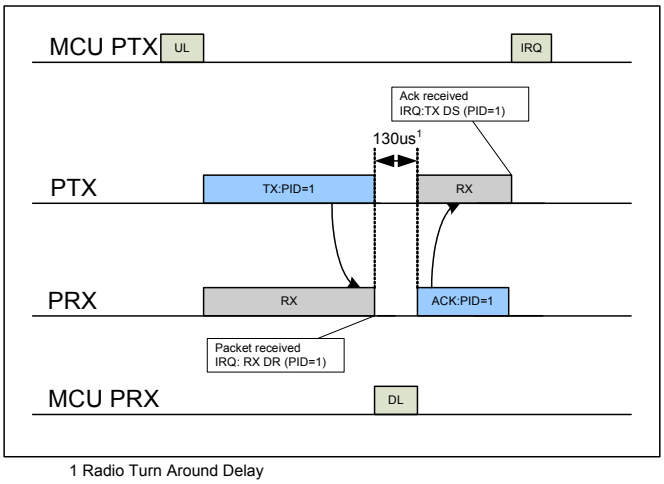


Figure 15. TX/RX cycles with ACK and the according interrupts

7.9.2 Single transaction with a lost packet

Figure 16. is a scenario where a retransmission is needed due to loss of the first packet transmit. After the packet is transmitted, the PTX enters RX mode to receive the ACK packet. After the first transmission, the PTX waits a specified time for the ACK packet, if it is not in the specific time slot the PTX retransmits the packet as shown in Figure 16.

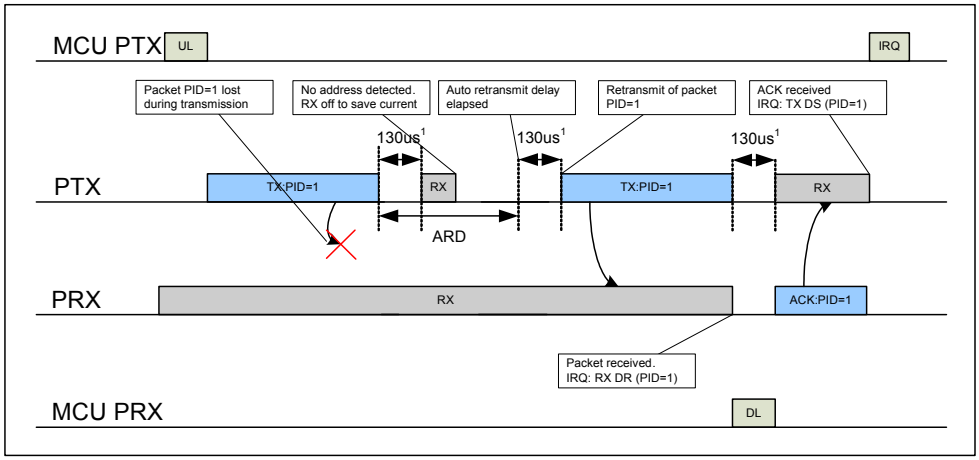


Figure 16. TX/RX cycles with ACK and the according interrupts when the first packet transmit fails

When an address is detected the PTX stays in RX mode until the packet is received. When the retransmitted packet is received by the PRX (see Figure 16.), the `RX_DR` IRQ is asserted and an ACK is transmitted back to the PTX. When the ACK is received by the PTX, the `TX_DS` IRQ is asserted.

7.9.3 Single transaction with a lost ACK packet

Figure 17. is a scenario where a retransmission is needed after a loss of the ACK packet. The corresponding interrupts are also indicated.

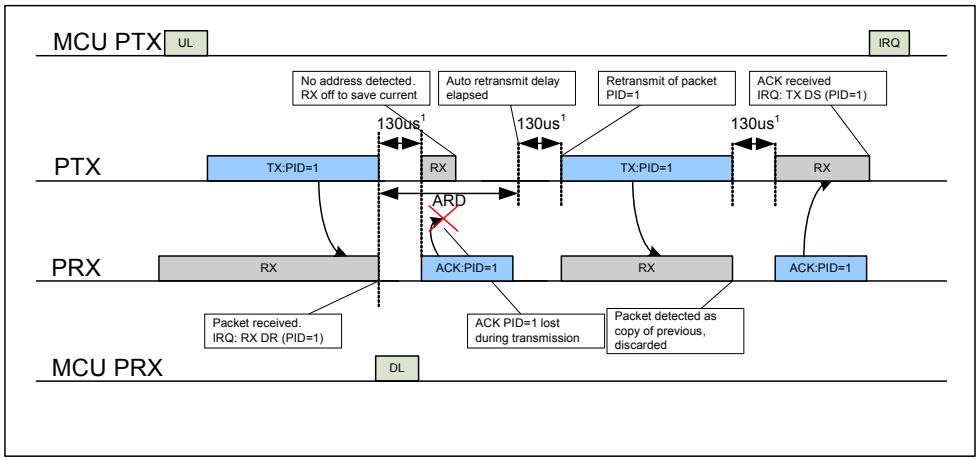
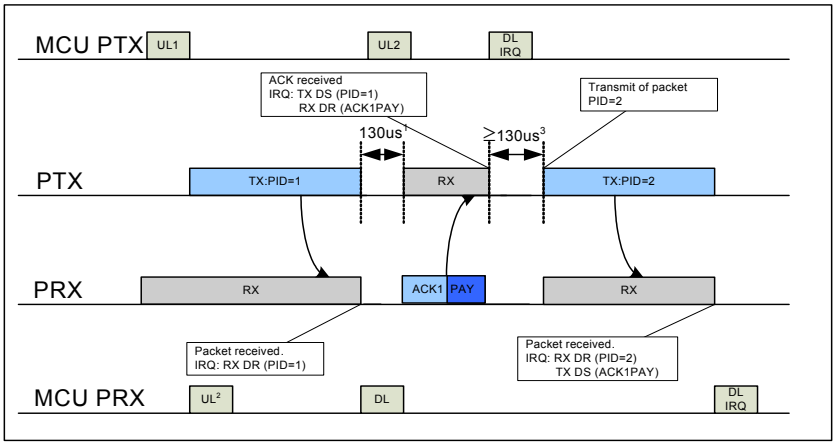


Figure 17. TX/RX cycles with ACK and the according interrupts when the ACK packet fails

7.9.4 Single transaction with ACK payload packet

Figure 18. is a scenario of the basic auto acknowledgement with payload. After the packet is transmitted by the PTX and received by the PRX the ACK packet with payload is transmitted from the PRX to the PTX. The RX_DR IRQ is asserted after the packet is received by the PRX, whereas on the PTX side the TX_DS IRQ is asserted when the ACK packet is received by the PTX. On the PRX side, the TX_DS IRQ for the ACK packet payload is asserted after a new packet from PTX is received. The position of the IRQ in Figure 18. shows where the MCU can respond to the interrupt.

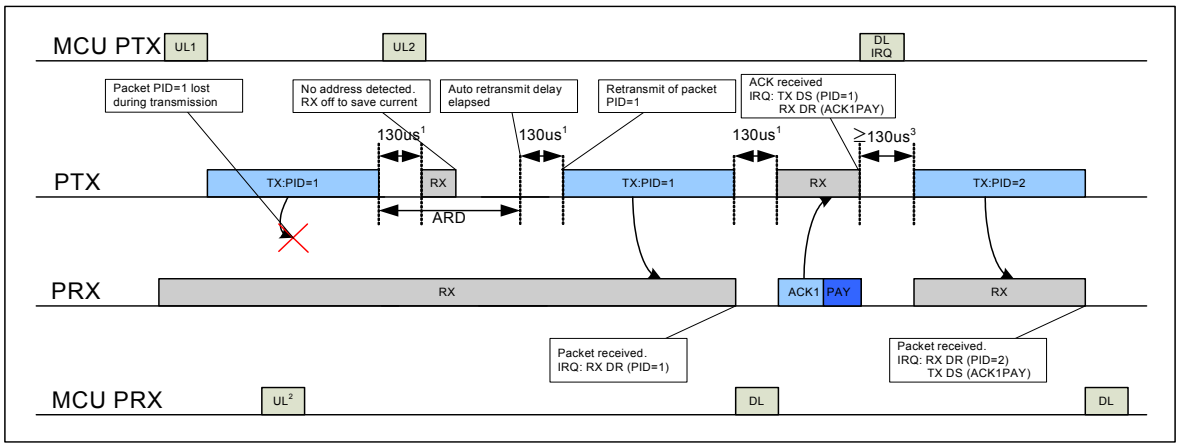


1 Radio Turn Around Delay
2 Uploading Paylod for Ack Packet
3 Delay defined by MCU on PTX side, ≥ 130us

Figure 18. TX/RX cycles with ACK Payload and the according interrupts

7.9.5 Single transaction with ACK payload packet and lost packet

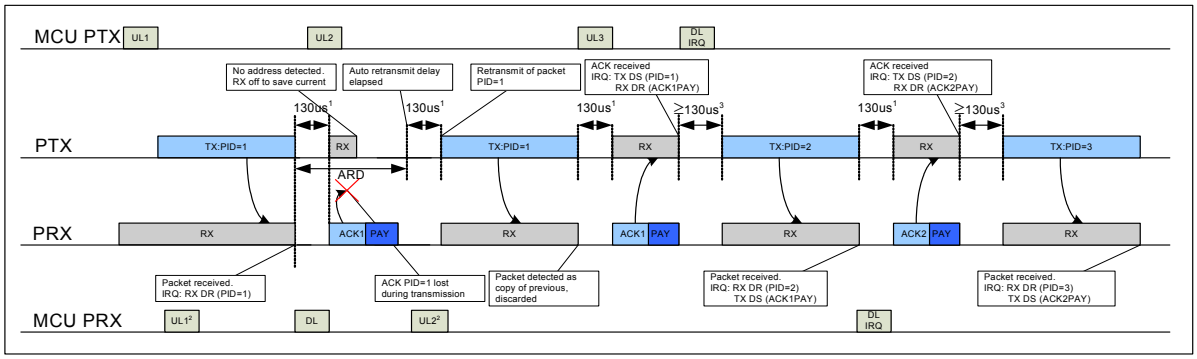
Figure 19. is a scenario where the first packet is lost and a retransmission is needed before the RX_DR IRQ on the PRX side is asserted. For the PTX both the TX_DS and RX_DR IRQ are asserted after the ACK packet is received. After the second packet (PID=2) is received on the PRX side both the RX_DR (PID=2) and TX_DS (ACK packet payload) IRQ are asserted.



1 Radio Turn Around Delay
2 Uploading Paylod for Ack Packet
3 Delay defined by MCU on PTX side, ≥ 130us

Figure 19. TX/RX cycles and the according interrupts when the packet transmission fails

7.9.6 Two transactions with ACK payload packet and the first ACK packet lost.

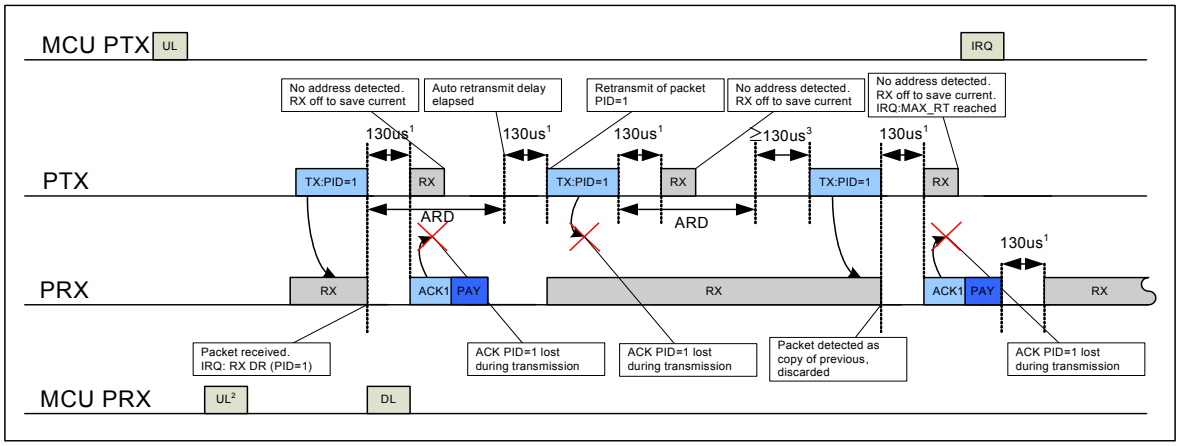


1 Radio Turn Around Delay
2 Uploading Paylod for Ack Packet
3 Delay defined by MCU on PTX side, ≥ 130us

Figure 20. TX/RX cycles with ACK Payload and the according interrupts when the ACK packet fails

In Figure 20. the ACK packet is lost and a retransmission is needed before the TX_DS IRQ is asserted, but the RX_DR IRQ is asserted immediately. The retransmission of the packet (PID=1) results in a discarded packet. For the PTX both the TX_DS and RX_DR IRQ are asserted after the second transmission of ACK, which is received. After the second packet (PID=2) is received on the PRX both the RX_DR (PID=2) and TX_DS (ACK1PAY) IRQ is asserted. The callouts explains the different events and interrupts.

7.9.7 Two transactions where max retransmissions is reached



1 Radio Turn Around Delay
2 Uploading Paylod for Ack Packet
3 Delay defined by MCU on PTX side, ≥ 130us

Figure 21. TX/RX cycles with ACK Payload and the according interrupts when the transmission fails. ARC is set to 2.

If the auto retransmit counter (ARC_CNT) exceeds the programmed maximum limit (ARC), the MAX_RT IRQ is asserted. In Figure 21. the packet transmission ends with a MAX_RT IRQ. The payload in TX FIFO is NOT removed and the MCU decides the next step in the protocol. A toggle of the **CE** starts a new sequence of transmitting the same packet. The payload can be removed from the TX FIFO using the FLUSH_TX command.

7.10 Compatibility with ShockBurst™

The nRF24L01 can have the Enhanced ShockBurst™ feature disabled in order to be backward compatible with the nRF2401A, nRF24E1, nRF2402 and nRF24E2.

Disabling the Enhanced ShockBurst™ features is done by setting register `EN_AA`=0x00 and the `ARC` = 0.

In addition, the nRF24L01 air data rate must be set to 1Mbps.

7.10.1 ShockBurst™ packet format

The ShockBurst™ packet format is described in this chapter. MSB to the left.

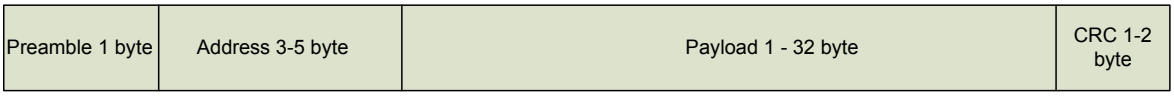


Figure 22. A ShockBurst™ packet compatible with nRF2401/nRF2402/nRF24E1/nRF24E2 devices.

The ShockBurst™ packet format has a preamble, address, payload and CRC field that is the same as in the Enhanced ShockBurst™ packet format described in section 7.3 on page 25.

The differences between the ShockBurst™ packet and the Enhanced ShockBurst™ packet are:

- The 9 bit Packet Control Field is not present in the ShockBurst™ packet format.
- The CRC is optional in the ShockBurst™ packet format and is controled by the `EN_CRC` bit in the `CONFIG` register.

8 Data and Control Interface

The data and control interface gives you access to all the features in the nRF24L01. The data and control interface consists of the following six 5Volt tolerant digital signals:

- **IRQ** (this signal is active low and is controlled by three maskable interrupt sources)
- **CE** (this signal is active high and is used to activate the chip in RX or TX mode)
- **CSN** (SPI signal)
- **SCK** (SPI signal)
- **MOSI** (SPI signal)
- **MISO** (SPI signal)

You can use the SPI to activate the nRF24L01 data FIFOs or the register map by 1 byte SPI commands during all modes of operation.

8.1 Features

- Special SPI commands for quick access to the most frequently used features
- 0-8Mbps 4-wire SPI serial interface
- 8 bit command set
- Easily configurable register map
- Full three level FIFO for both TX and RX direction

8.2 Functional description

The SPI is a standard SPI with a maximum data rate of 8Mbps.

8.3 SPI operation

This chapter describes the SPI commands and SPI timing.

8.3.1 SPI Commands

The SPI commands are shown in Table 16.. Every new command must be started by a high to low transition on **CSN**.

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

The serial shifting SPI commands is in the following format:

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

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

See Figure 23. on page 47 and Figure 24. on page 48 for timing information.

Command name	Command word (binary)	# Data bytes	Operation
R_REGISTER	000A AAAA	1 to 5 LSByte first	Read command and status registers. AAAAA = 5 bit Register Map Address
W_REGISTER	001A AAAA	1 to 5 LSByte first	Write command and status registers. AAAAA = 5 bit Register 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 always starts at byte 0. Payload is deleted from FIFO after it is read. Used in RX mode.
W_TX_PAYLOAD	1010 0000	1 to 32 LSByte first	Write TX-payload: 1 – 32 bytes. A write operation always starts at byte 0 used in TX payload.
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, that is, acknowledge package will not be completed.
REUSE_TX_PL	1110 0011	0	Used for a PTX device Reuse last transmitted payload. Packets are repeatedly retransmitted 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
ACTIVATE	0101 0000	1	This write command followed by data 0x73 activates the following features: • R_RX_PL_WID • W_ACK_PAYLOAD • W_TX_PAYLOAD_NOACK A new ACTIVATE command with the same data deactivates them again. *This is executable in power down or stand by modes only.* The R_RX_PL_WID, W_ACK_PAYLOAD, and W_TX_PAYLOAD_NOACK features registers are initially in a deactivated state; a write has no effect, a read only results in zeros on **MISO**. To activate these registers, use the ACTIVATE command followed by data 0x73. Then they can be accessed as any other register in nRF24L01. Use the same command and data to deactivate the registers again.
R_RX_PL_WID[a]	0110 0000		Read RX-payload width for the top R_RX_PAYLOAD in the RX FIFO.
W_ACK_PAYLOAD[a]	1010 1PPP	1 to 32 LSByte first	Used in RX mode. Write Payload to be transmitted together with ACK packet on PIPE PPP. (PPP valid in the range from 000 to 101). Maximum three ACK packet payloads can be pending. Payloads with same PPP are handled using first in - first out principle. Write payload: 1– 32 bytes. A write operation always starts at byte 0.

Command name	Command word (binary)	# Data bytes	Operation
W_TX_PAYLOAD_NOACK[a]	1011 000	1 to 32 LSByte first	Used in TX mode. Disables AUTOACK on this specific packet.
NOP	1111 1111	0	No Operation. Might be used to read the STATUS register

a. To activate this feature use the ACTIVATE SPI command followed by data 0x73. The corresponding bits in the FEATURE register shown in Table 24. on page 58 have to be set.

Table 16. Command set for the nRF24L01 SPI

The W_REGISTER and R_REGISTER commands can operate on single or multi-byte registers. When accessing multi-byte registers you read or write to the MSBit of LSByte first. You can terminate the writing before all bytes in a multi-byte register are written, leaving the unwritten MSByte(s) unchanged. For example, 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 is always read to **MISO** after a high to low transition on **CSN**.

Note: The 3 bit pipe information in the STATUS register is updated during the **IRQ** pin high to low transition. If the STATUS register is read during an **IRQ** pin high to low transition, the pipe information is unreliable.

8.3.2 SPI timing

SPI operation and timing is shown in Figure 23. on page 47 to Figure 25. on page 48 and in Table 18. on page 49 to Table 23. on page 50. nRF24L01 must be in one of the standby modes or in power down mode before writing to the configuration registers.

In Figure 23. on page 47 to Figure 25. on page 48 the following abbreviations are used:

Abbreviation	Description
Cn	SPI command bit
Sn	STATUS register bit
Dn	Data Bit (**Note:** LSByte to MSByte, MSBit in each byte first)

Table 17. Abbreviations used in Figure 23. to Figure 25.

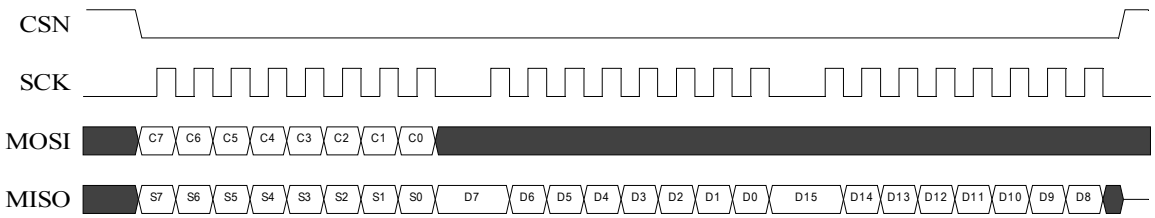


Figure 23. SPI read operation

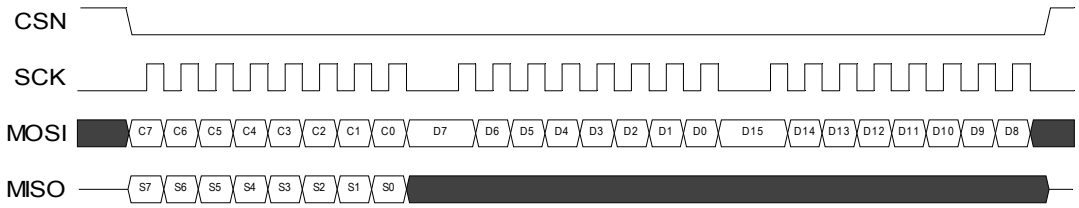


Figure 24. SPI write operation

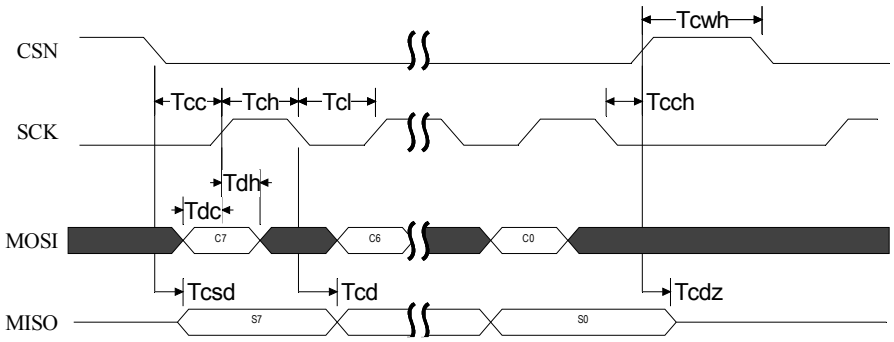


Figure 25. SPI NOP timing diagram

Figure 26. shows the R_{pull} and C_{load} that are referenced in Table 18. to Table 23.

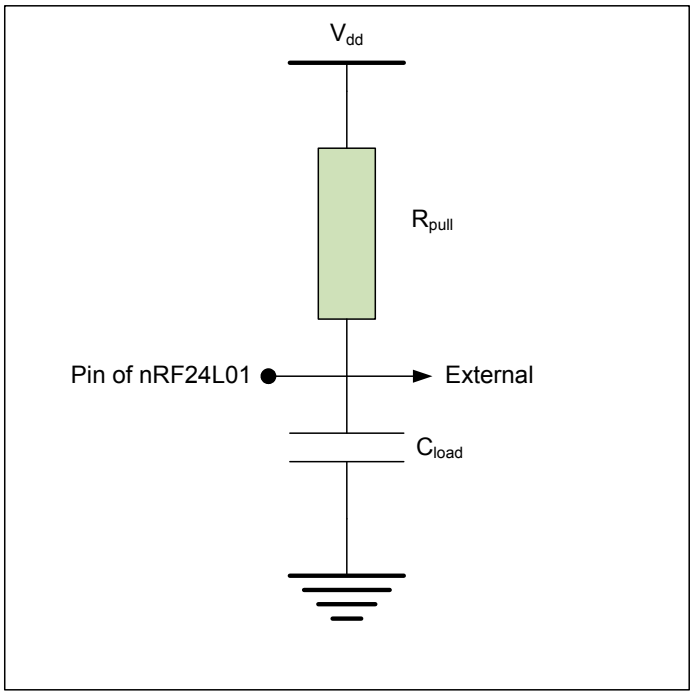


Figure 26. R_{pull} and C_{load}

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

Table 18. SPI timing parameters (C_{load} = 5pF)

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

Table 19. SPI timing parameters (C_{load} = 10pF)

Symbol	Parameters	Min	Max	Units
Tdc	Data to SCK Setup	2		ns
Tdh	SCK to Data Hold	2		ns
Tcsd	CSN to Data Valid		75	ns
Tcd	SCK to Data Valid		86	ns
Tcl	SCK Low Time	40		ns
Tch	SCK High Time	40		ns
Fsck	SCK Frequency	0	5	MHz
Tr,Tf	SCK Rise and Fall		100	ns
Tcc	CSN to SCK Setup	2		ns
Tcch	SCK to CSN Hold	2		ns
Tcwh	CSN Inactive time	50		ns
Tcdz	CSN to Output High Z		75	ns

Table 20. SPI timing parameters (R_{pull} = 10kΩ, C_{load} = 50pF)

Symbol	Parameters	Min	Max	Units
Tdc	Data to SCK Setup	2		ns
Tdh	SCK to Data Hold	2		ns
Tcsd	CSN to Data Valid		116	ns
Tcd	SCK to Data Valid		123	ns
Tcl	SCK Low Time	40		ns
Tch	SCK High Time	40		ns
Fsck	SCK Frequency	0	4	MHz
Tr,Tf	SCK Rise and Fall		100	ns
Tcc	CSN to SCK Setup	2		ns
Tcch	SCK to CSN Hold	2		ns
Tcwh	CSN Inactive time	50		ns
Tcdz	CSN to Output High Z		116	ns

Table 21. SPI timing parameters (R_{pull} = 10kΩ, C_{load} = 100pF)

Symbol	Parameters	Min	Max	Units
Tdc	Data to SCK Setup	2		ns
Tdh	SCK to Data Hold	2		ns
Tcsd	CSN to Data Valid		75	ns
Tcd	SCK to Data Valid		85	ns
Tcl	SCK Low Time	40		ns
Tch	SCK High Time	40		ns
Fsck	SCK Frequency	0	5	MHz
Tr,Tf	SCK Rise and Fall		100	ns
Tcc	CSN to SCK Setup	2		ns
Tcch	SCK to CSN Hold	2		ns
Tcwh	CSN Inactive time	50		ns
Tcdz	CSN to Output High Z		75	ns

Table 22. SPI timing parameters (R_{pull} = 50kΩ, C_{load} = 50pF)

Symbol	Parameters	Min	Max	Units
Tdc	Data to SCK Setup	2		ns
Tdh	SCK to Data Hold	2		ns
Tcsd	CSN to Data Valid		116	ns
Tcd	SCK to Data Valid		121	ns
Tcl	SCK Low Time	40		ns
Tch	SCK High Time	40		ns
Fsck	SCK Frequency	0	4	MHz
Tr,Tf	SCK Rise and Fall		100	ns
Tcc	CSN to SCK Setup	2		ns
Tcch	SCK to CSN Hold	2		ns
Tcwh	CSN Inactive time	50		ns
Tcdz	CSN to Output High Z		116	ns

Table 23. SPI timing parameters (R_{pull} = 50kΩ, C_{load} = 100pF)

8.4 Data FIFO

The data FIFOs are used to store payload that is transmitted (TX FIFO) or payload that is received and ready to be clocked out (RX FIFO). The FIFOs are accessible in both PTX mode and PRX mode.

The following FIFOs are present in nRF24L01:

- TX three level, 32 byte FIFO
- RX three level, 32 byte FIFO

Both FIFOs have a controller and are accessible through the SPI by using dedicated SPI commands. A TX FIFO in PRX can store payload for ACK packets to three different PTX devices. If the TX FIFO contains more than one payload to a pipe, payloads are handled using the first in - first out principle. The TX FIFO in a PRX is blocked if all pending payloads are addressed to pipes where the link to the PTX is lost. In this case, the MCU can flush the TX FIFO by using the `FLUSH_TX` command.

The RX FIFO in PRX may contain payload from up to three different PTX devices.

A TX FIFO in PTX can have up to three payloads stored.

The TX FIFO can be written to by three commands, `W_TX_PAYLOAD` and `W_TX_PAYLOAD_NO_ACK` in PTX mode and `W_ACK_PAYLOAD` in PRX mode. All three commands give access to the `TX_PLD` register.

The RX FIFO can be read by the command `R_RX_PAYLOAD` in both PTX and PRX mode. This command gives access to the `RX_PLD` register.

The payload in TX FIFO in a PTX is NOT removed if the `MAX_RT` IRQ is asserted. Figure 27. is a block diagram of the TX FIFO and the RX FIFO.

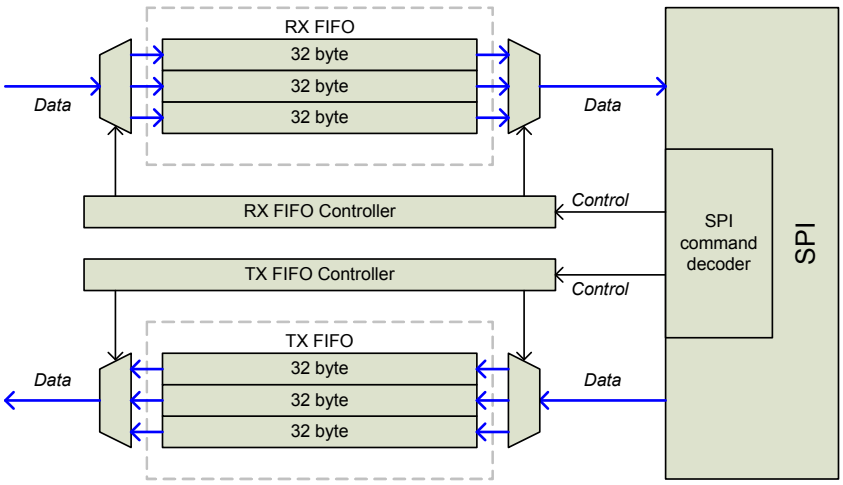


Figure 27. FIFO block diagram

In the `FIFO_STATUS` register it is possible to read if the TX and RX FIFO is full or empty. The `TX_REUSE` bit is also available in the `FIFO_STATUS` register. `TX_REUSE` is set by the SPI command `REUSE_TX_PL`, and is reset by the SPI commands `W_TX_PAYLOAD` or `FLUSH TX`.

8.5 Interrupt

The nRF24L01 has an active low interrupt (**IRQ**) pin. The **IRQ** pin is activated when TX_DS IRQ, RX_DR IRQ or MAX_RT IRQ are set high by the state machine in the STATUS register. The **IRQ** pin resets when MCU writes '1' to the IRQ source bit in the STATUS register. The IRQ mask in the CONFIG register is used to select the IRQ sources that are allowed to assert the **IRQ** pin. By setting one of the MASK bits high, the corresponding IRQ source is disabled. By default all IRQ sources are enabled.

Note: The 3 bit pipe information in the STATUS register is updated during the **IRQ** pin high to low transition. If the STATUS register is read during an **IRQ** pin high to low transition, the pipe information is unreliable.

9 Register Map

You can configure and control the radio chip by accessing the register map through the SPI by using read and write commands.

9.1 Register map table

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

Note: Addresses 18 to 1B are reserved for test purposes, altering them will make the chip malfunction.

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	RX/TX control 1: PRX, 0: PTX
01	EN_AA Enhanced ShockBurst™				Enable 'Auto Acknowledgment' Function Disable this functionality to be compatible with nRF2401, see page 65
	Reserved	7:6	00	R/W	Only '00' allowed
	ENAA_P5	5	1	R/W	Enable auto acknowledgement data pipe 5
	ENAA_P4	4	1	R/W	Enable auto acknowledgement data pipe 4
	ENAA_P3	3	1	R/W	Enable auto acknowledgement data pipe 3
	ENAA_P2	2	1	R/W	Enable auto acknowledgement data pipe 2
	ENAA_P1	1	1	R/W	Enable auto acknowledgement data pipe 1
	ENAA_P0	0	1	R/W	Enable auto acknowledgement 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.

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
	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 is used if address width is below 5 bytes
04	SETUP_RETR				Setup of Automatic Retransmission
	ARD	7:4	0000	R/W	Auto Retransmit Delay ‘0000’ – Wait 250µS ‘0001’ – Wait 500µS ‘0010’ – Wait 750µS …….. ‘1111’ – Wait 4000µS (Delay defined from end of transmission to start of next transmission)[a]
	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	Air Data Rate ‘0’ – 1Mbps ‘1’ – 2Mbps
	RF_PWR	2:1	11	R/W	Set RF output power in TX mode '00' – -18dBm '01' – -12dBm '10' – -6dBm '11' – 0dBm
	LNA_HCURR	0	1	R/W	Setup LNA gain

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
07	STATUS				Status Register (In parallel to the SPI command 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. Asserted when new data arrives RX FIFO[b]. Write 1 to clear bit.
	TX_DS	5	0	R/W	Data Sent TX FIFO interrupt. Asserted when packet transmitted on TX. If AUTO_ACK is activated, this bit is set high only when ACK is received. Write 1 to clear bit.
	MAX_RT	4	0	R/W	Maximum number of TX retransmits interrupt Write 1 to clear bit. If MAX_RT is asserted it must be cleared to enable further communication.
	RX_P_NO	3:1	111	R	Data pipe number for the payload 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 discontinues at max until reset. The counter is reset by writing to RF_CH. See page 65 and page 74.
	ARC_CNT	3:0	0	R	Count retransmitted packets. The counter is reset when transmission of a new packet starts. See page 65.
09	CD				
	Reserved	7:1	000000	R	
	CD	0	0	R	Carrier Detect. See page page 74.
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 is equal to RX_ADDR_P1[39:8]
0D	RX_ADDR_P3	7:0	0xC4	R/W	Receive address data pipe 3. Only LSB. MSBytes is equal to RX_ADDR_P1[39:8]
0E	RX_ADDR_P4	7:0	0xC5	R/W	Receive address data pipe 4. Only LSB. MSBytes is equal to RX_ADDR_P1[39:8]
0F	RX_ADDR_P5	7:0	0xC6	R/W	Receive address data pipe 5. Only LSB. MSBytes is equal to RX_ADDR_P1[39:8]

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
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 65.
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 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

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
	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 transmitted data packet if set high. The packet is repeatedly retransmitted as long as CE is high. TX_REUSE is set by the SPI command REUSE_TX_PL, and is reset by the SPI commands 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	ACK_PLD[c]	255:0	X	W	Written by separate SPI command ACK packet payload to data pipe number PPP given in SPI command Used in RX mode only Maximum three ACK packet payloads can be pending. Payloads with same PPP are handled first in first out.
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 three levels. Used in TX mode only

Address (Hex)	Mnemonic	Bit	Reset Value	Type	Description
N/A	RX_PLD	255:0	X	R	Read by separate SPI command RX data payload register. 1 - 32 bytes. This register is implemented as a FIFO with three levels. All RX channels share the same FIFO
1C	DYNPD[c]				Enable dynamic payload length
	Reserved	7:6	0	R/W	Only '00' allowed
	DPL_P5	5	0	R/W	Enable dyn. payload length data pipe 5. (Requires EN_DPL and ENAA_P5)
	DPL_P4	4	0	R/W	Enable dyn. payload length data pipe 4. (Requires EN_DPL and ENAA_P4)
	DPL_P3	3	0	R/W	Enable dyn. payload length data pipe 3. (Requires EN_DPL and ENAA_P3)
	DPL_P2	2	0	R/W	Enable dyn. payload length data pipe 2. (Requires EN_DPL and ENAA_P2)
	DPL_P1	1	0	R/W	Enable dyn. payload length data pipe 1. (Requires EN_DPL and ENAA_P1)
	DPL_P0	0	0	R/W	Enable dyn. payload length data pipe 0. (Requires EN_DPL and ENAA_P0)
1D	FEATURE[c]			R/W	Feature Register
	Reserved	7:3	0	R/W	Only '00000' allowed
	EN_DPL	2	0	R/W	Enables Dynamic Payload Length
	EN_ACK_PAY[d]	1	0	R/W	Enables Payload with ACK
	EN_DYN_ACK	0	0	R/W	Enables the W_TX_PAYLOAD_NOACK command

a. This is the time the PTX is waiting for an ACK packet before a retransmit is made. The PTX is in RX mode for a minimum of 250µS, but it stays in RX mode to the end of the packet if that is longer than 250µS. Then it goes to standby-I mode for the rest of the specified ARD. After the ARD it goes to TX mode and then retransmits the packet.

b. The RX_DR IRQ is asserted by a new packet arrival event. The procedure for handling this interrupt should be: 1) read payload through SPI, 2) clear RX_DR IRQ, 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)

c. To activate this feature use the ACTIVATE SPI command followed by data 0x73. The corresponding bits in the FEATURE register must be set.

d. If ACK packet payload is activated, ACK packets have dynamic payload lengths and the Dynamic Payload Length feature should be enabled for pipe 0 on the PTX and PRX. This is to ensure that they receive the ACK packets with payloads. If the payload in ACK is more than 15 byte in 2Mbps mode the ARD must be 500µS or more, and if the payload is more than 5byte in 1Mbps mode the ARD must be 500µS or more.

Table 24. Register map of nRF24L01

10 Peripheral RF Information

This chapter describes peripheral circuitry and PCB layout requirements that are important for achieving optimum RF performance from the nRF24L01.

10.1 Antenna output

The **ANT1** and **ANT2** output pins provide a balanced RF output to the antenna. The pins must have a DC path to **VDD_PA**, either through a RF choke or through the center point in a balanced dipole antenna. A load of 15Ω+j88Ω is recommended for maximum output power (0dBm). Lower load impedance (for instance 50Ω) can be obtained by fitting a simple matching network between the load and **ANT1** and **ANT2**. A recommended matching network for 50Ω load impedance is described in Appendix D on page 69.

10.2 Crystal oscillator

A crystal being used with the nRF24L01 must fulfil the specifications given in Table 8. on page 17.

To achieve a crystal oscillator solution with low power consumption and fast start-up time a crystal with a low load capacitance specification must be used. A lower C_0 also gives lower current consumption and faster start-up time, but may increase the cost of the crystal. 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, see Figure 30. on page 69. 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** pins respectively; the value is typically 1pF for each of these pins.

10.3 nRF24L01 sharing crystal with an MCU

When using an MCU to drive the crystal reference input **XC1** of the nRF24L01 transceiver the rules described in the following sections (10.3.1 and 10.3.2) must be followed.

10.3.1 Crystal parameters

The requirement of load capacitance C_L is only set by the MCU when the MCU drives the nRF24L01 clock input. The frequency accuracy of ±60ppm is still required to get a functional radio link. The nRF24L01 loads the crystal by 1pF in addition to the PCB routing.

10.3.2 Input crystal amplitude and current consumption

The input signal should not have amplitudes exceeding any rail voltage. Exceeding rail voltage excites the ESD structure and the radio performance is degraded below specification. You must use an external DC block if you are testing the nRF24L01 with a reference source that has no DC offset (which is often the case with a RF source).

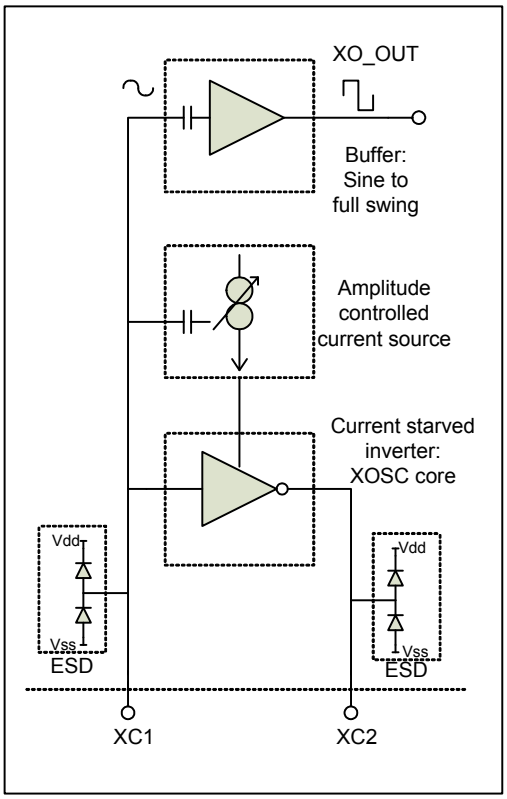


Figure 28. Principle of crystal oscillator

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

10.4 PCB layout and decoupling guidelines

A well designed PCB is necessary to achieve good RF performance. A poor layout can lead to loss of performance or functionality. A fully qualified RF-layout for the nRF24L01 and its surrounding components, including matching networks, can be downloaded from www.nordicsemi.no.

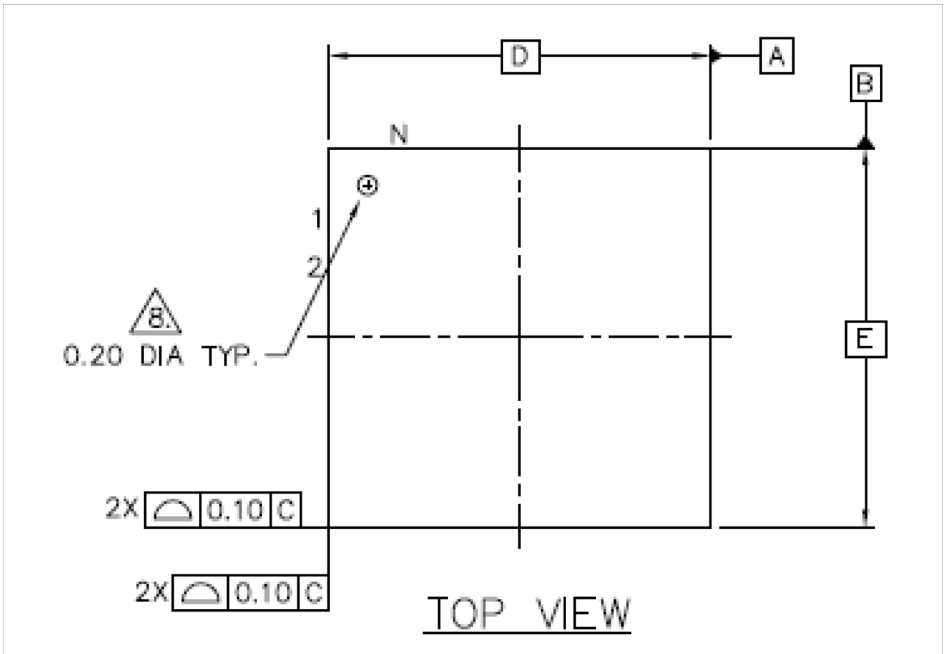
A PCB with a minimum of two layers including a ground plane is recommended for optimum performance. The nRF24L01 DC supply voltage should be decoupled as close as possible to the **VDD** pins with high performance RF capacitors, see Table 26. on page 69. It is preferable to mount a large surface mount capacitor (for example, 4.7µF ceramic) 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.

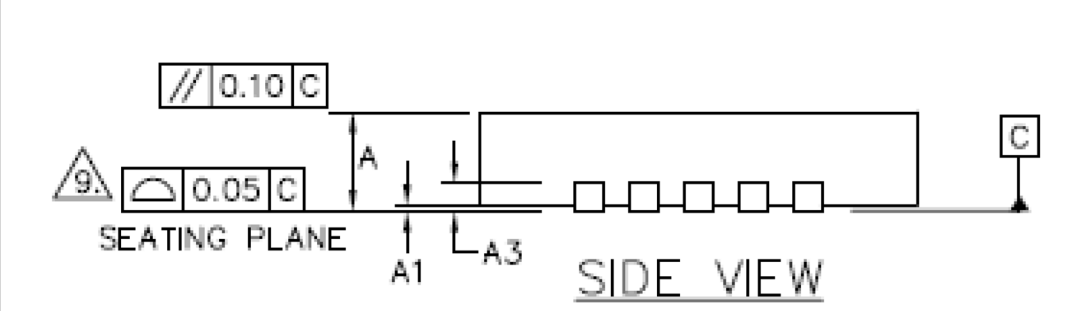
Long power supply lines on the PCB should be avoided. All device grounds, **VDD** connections and **VDD** bypass capacitors must be connected as close as possible to the nRF24L01 IC. For a PCB with a topside RF ground plane, the **VSS** pins should be connected directly to the ground plane. For a PCB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

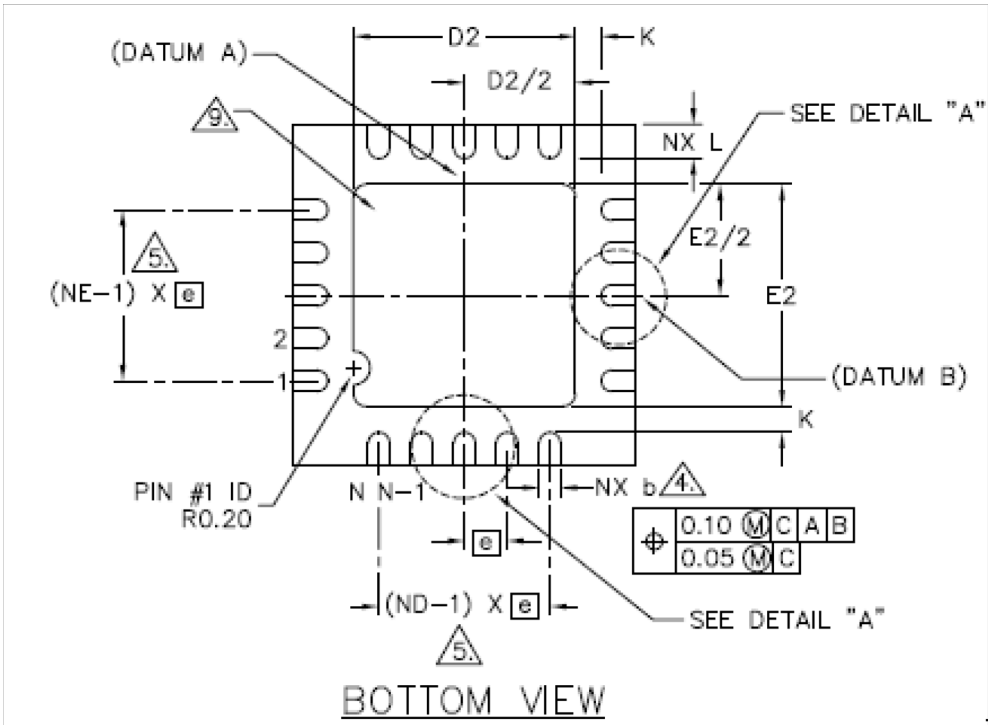
Full swing digital data or control signals should not be routed close to the crystal or the power supply lines. The exposed die attach pad is a ground pad connected to the IC substrate die ground and is intentionally not used in our layouts. It is recommended to keep it unconnected.

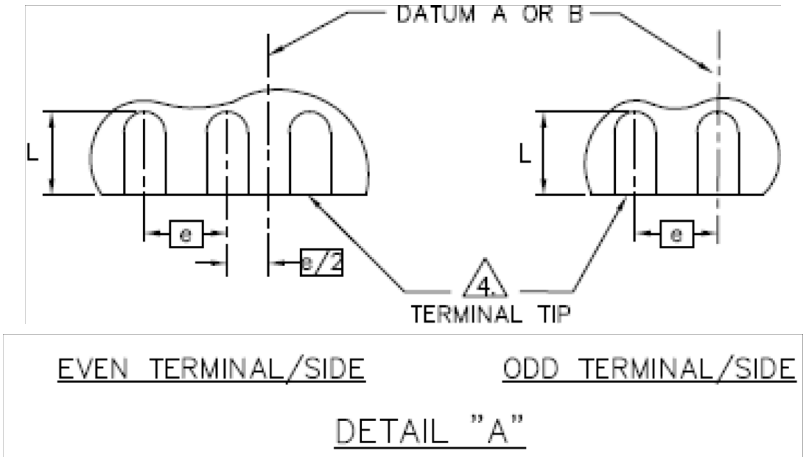
11 Mechanical specifications

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 (4x4 mm)	**Min**	0.80	0.00					2.50	0.35		0.18
	Typ.	0.85	0.02	0.20	0.20 min	4.0	0.5 BSC	2.60	0.40	0.15	0.25
	Max	0.95	0.05	REF.		BSC[a]		2.70	0.45	max	0.30

a. BSC: Basic Spacing between Centers, ref. JEDEC standard 95, page 4.17-11/A

Figure 29. nRF24L01 Package Outline

12 Ordering information

Ordering code	Description	Package	Container	MOQ[a]
nRF24L01-REEL	2/1Mbps Transceiver	20 pin QFN 4x4	Tape and reel[b]	4000
nRF24L01-REEL7	2/1Mbps Transceiver	20 pin QFN 4x4	Tape and reel	1500
nRF24L01	2/1Mbps Transceiver	20 pin QFN 4x4	Tray	490
nRF24L01-EVKIT	2 node evaluation	N/A	N/A	1

a. MOQ = Minimum order quantity
b. **Moisture Sensitivity Level:** MSL2@260ºC, three times reflow

12.1 Package marking

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

12.2 Abbreviations

Abbreviation	Definition
nRF	Fixed text
B	Variable Build Code, that is, 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

Attention!

Observe precaution for handling
Electrostatic Sensitive Device.

13 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	Microcontroller Unit
MISO	Master In Slave Out
MOSI	Master Out Slave In
MSB	Most Significant Bit
MSByte	Most Significant Byte
PCB	Printed Circuit Board
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 25. Glossary

Appendix A - Enhanced ShockBurst™ - Configuration and Communication Example

Enhanced ShockBurst™ Transmitting Payload

1. The configuration bit PRIM_RX has to be low.
2. When the application MCU has data to transmit, the address for the receiving node (TX_ADDR) and payload data (TX_PLD) has to be clocked into nRF24L01 through the SPI. 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 ACK packet. The RX address for data pipe 0 (RX_ADDR_P0) has to be equal to the TX address (TX_ADDR) in the PTX device. For the example in Figure 12. on page 37 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 10µs.
4. nRF24L01 ShockBurst™:
   - Radio is powered up.
   - 16MHz internal clock is started.
   - RF packet is completed (see the packet description).
   - Data is transmitted at high speed (1Mbps or 2Mbps configured by MCU).
5. If auto acknowledgement is activated (ENAA_P0=1) the radio goes into RX mode immediately, unless the NO_ACK bit is set in the received packet. 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 ACK packet is not received in the specified time window, the payload is retransmitted (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 is 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 ACK packet is received for a packet after the maximum number of retransmits, no further packets can be transmitted before the MAX_RT interrupt is cleared. The packet loss counter (PLOS_CNT) is incremented at each MAX_RT interrupt. That is, ARC_CNT counts the number of retransmits 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 retransmits.
6. nRF24L01 goes into standby-I mode if **CE** is low. Otherwise next payload in TX FIFO is transmitted. If TX FIFO is empty and **CE** is still high, nRF24L01 enters standby-II mode.
7. If nRF24L01 is in standby-II mode, it goes 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 CONFIG register to high. All data pipes that 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 is active

when RX_DR is high. RX_P_NO in STATUS register indicates what data pipe the payload has been received in.

5. If auto acknowledgement is enabled, an ACK packet is transmitted back, unless the NO_ACK bit is set in the received packet. If there is a payload in the TX_PLD FIFO, this payload is added to the ACK packet.
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 through the SPI.
8. nRF24L01 is now ready for entering TX or RX mode or power down mode.

Appendix B - Configuration for compatibility with nRF24XX

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

1. Use the same CRC configuration as the nRF2401/nRF2402/nRF24E1/nRF24E2
2. Set the PWR_UP and PRIM_RX bit to 1
3. Disable auto acknowledgement on the data pipe that is addressed
4. Use the same address width as the PTX device
5. Use the same frequency channel as the PTX device
6. Select data rate 1Mbps on both nRF24L01 and nRF2401/nRF2402/nRF24E1/nRF24E2
7. Set correct payload width on the data pipe that is addressed
8. Set CE high

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

1. Use the same CRC configuration as the nRF2401/nRF2402/nRF24E1/nRF24E2
2. Set the PRIM_RX bit to 0
3. Set the Auto Retransmit Count to 0 to disable the auto retransmit functionality
4. Use the same address width as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
5. Use the same frequency channel as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
6. Select data rate 1Mbps on both nRF24L01 and nRF2401/nRF2402/nRF24E1/nRF24E2
7. Set PWR_UP high
8. Clock in a payload that has the same length as the nRF2401/nRF2402/nRF24E1/nRF24E2 is configured to receive
9. Pulse CE to transmit the packet

Appendix C - Carrier wave output power

The output power of a radio is a critical factor for achieving wanted range. Output power is also the first test criteria needed to qualify for all telecommunication regulations.

Configuration

1. Set PWR_UP = 1 in the CONFIG register
2. Wait 1.5ms PWR_UP->standby
3. Clear the PRIM_RX in the CONFIG register
4. Set all auto acknowledgement functionality in the EN_AA register and the SETUP_RETR register to 0
5. Set output power
6. Set PLL_LOCK to 1
7. Configure TX address as 5 bytes with all 0xFF
8. Fill the TX payload with 32 bytes of 0xFF
9. Turn off CRC
10. Set the wanted RF channel
11. Transmit the packet by pulsing **CE** (minimum 10µs)
12. Wait until the transmission ends (indicated by IRQ going active, a fixed delay of 1ms can also be used)
13. Set **CE** high
14. Use the SPI command for re-use of last sent packet (REUSE_TX_PL)
15. Keep **CE** high as long as the carrier is needed

The nRF24L01 should now output a carrier.

Note: This is not a clean carrier but is slightly modulated by the preamble.

Appendix D - Application example

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

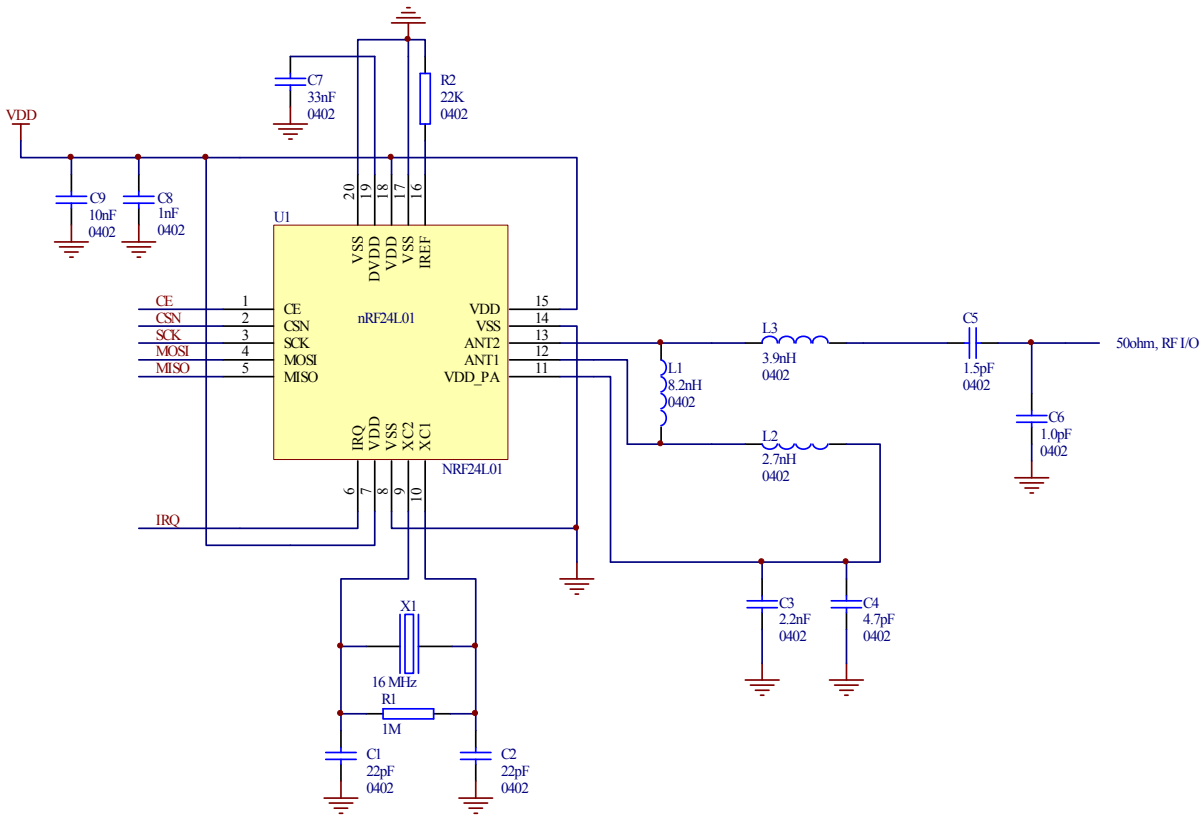


Figure 30. nRF24L01 schematic for RF layouts with single ended 50Ω RF output

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

a. C1 and C2 must have values that match the crystals load capacitance, C_L.

Table 26. Recommended components (BOM) in nRF24L01 with antenna matching network

PCB layout examples

Figure 31. on page 70, Figure 32. on page 71 and Figure 33. on page 71 show a PCB layout example for the application schematic in Figure 30. on page 69.

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.

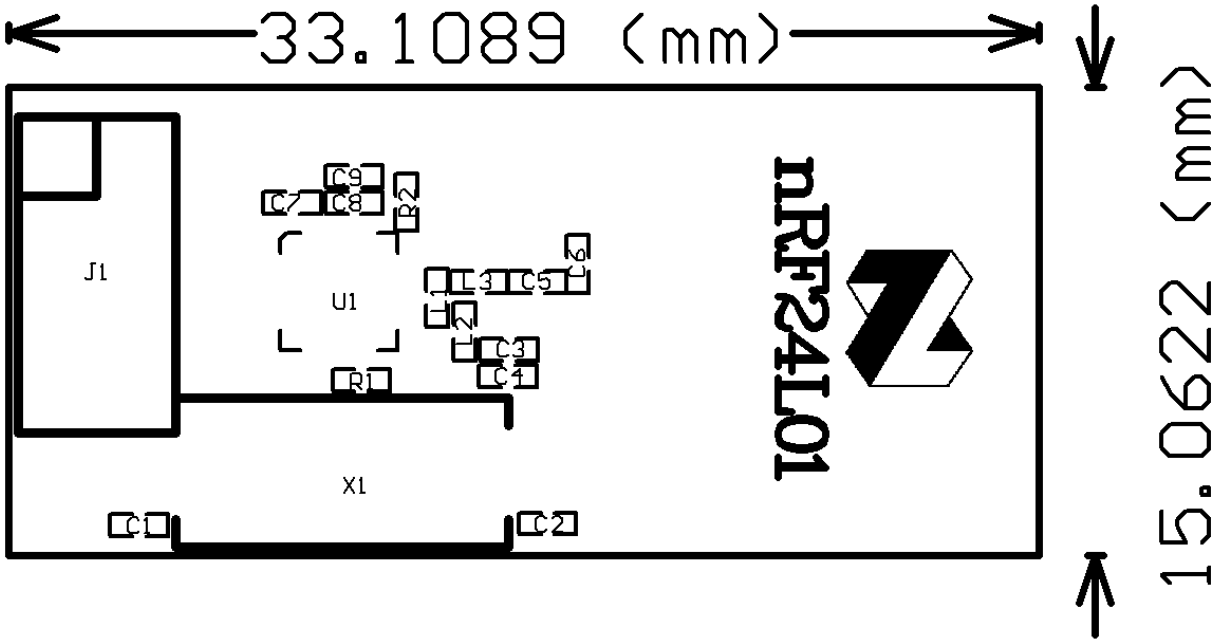


Figure 31. Top overlay (nRF24L01 RF layout with single ended connection to PCB antenna and 0402 size passive components)

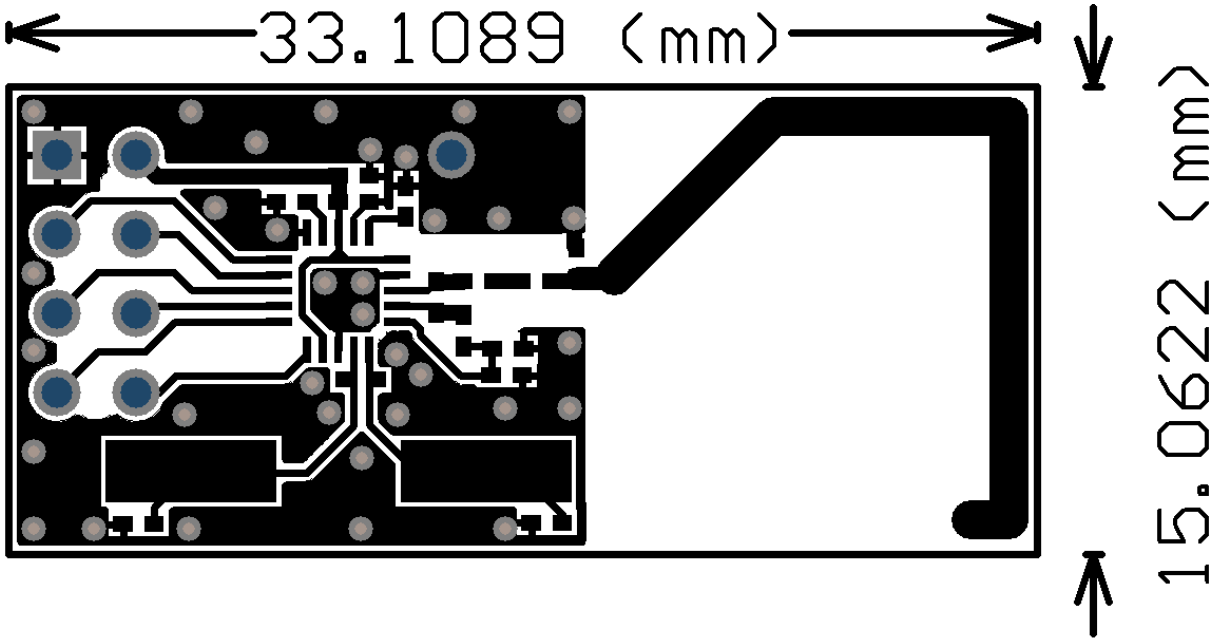


Figure 32. Top layer (nRF24L01 RF layout with single ended connection to PCB antenna and 0402 size passive components)

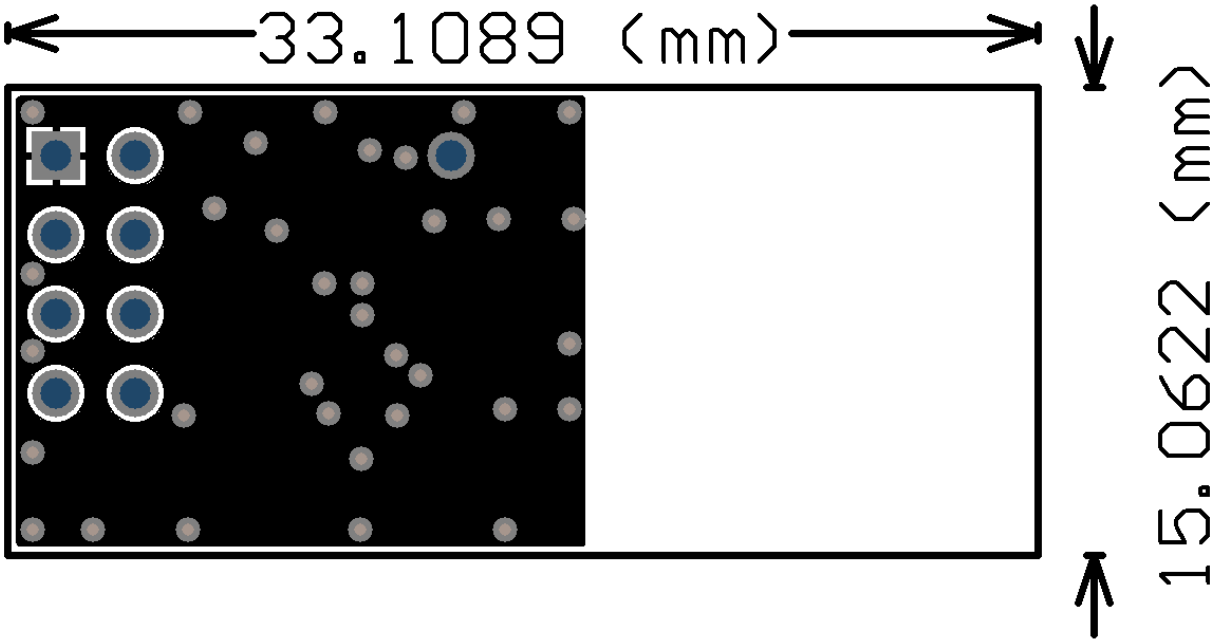


Figure 33. Bottom layer (nRF24L01 RF layout with single ended connection to PCB antenna and 0402 size passive components

The nest figure (Figure 34. on page 72, Figure 35. on page 72 and Figure 36. on page 73) is for the SMA output to have a board for direct measurements at a 50Ω SMA connector.

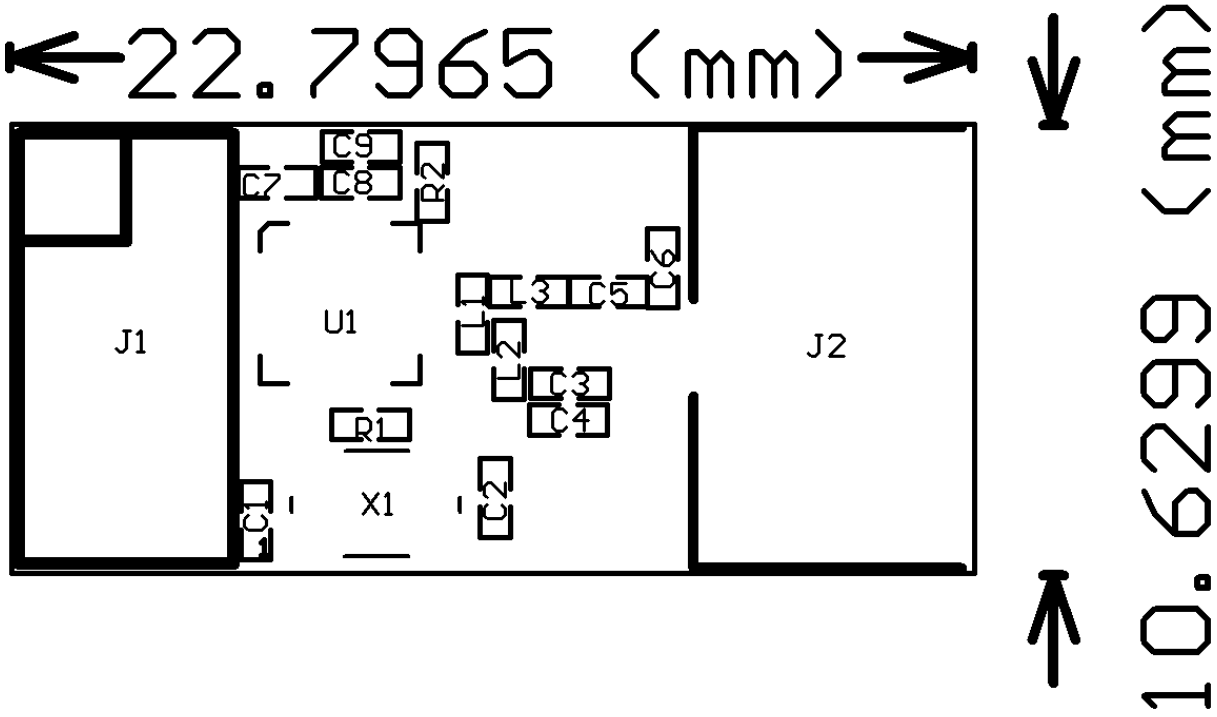


Figure 34. Top Overlay (Module with OFM crystal and SMA connector)

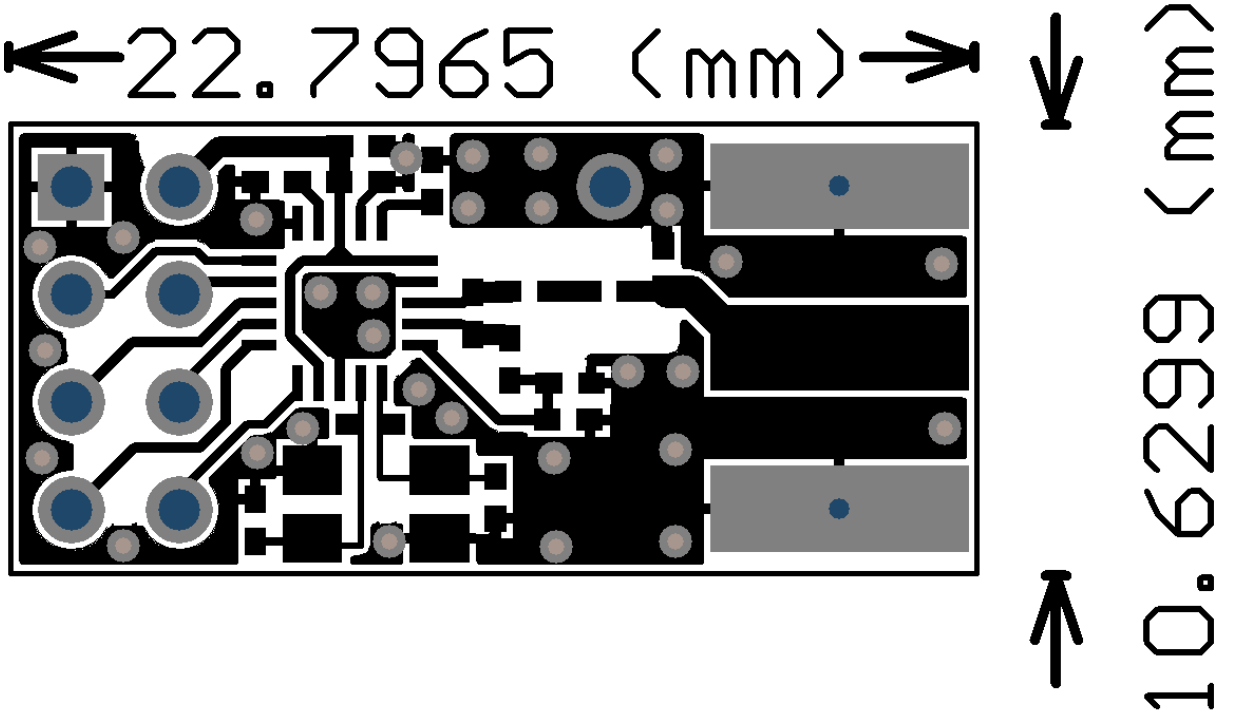


Figure 35. Top Layer (Module with OFM crystal and SMA connector)

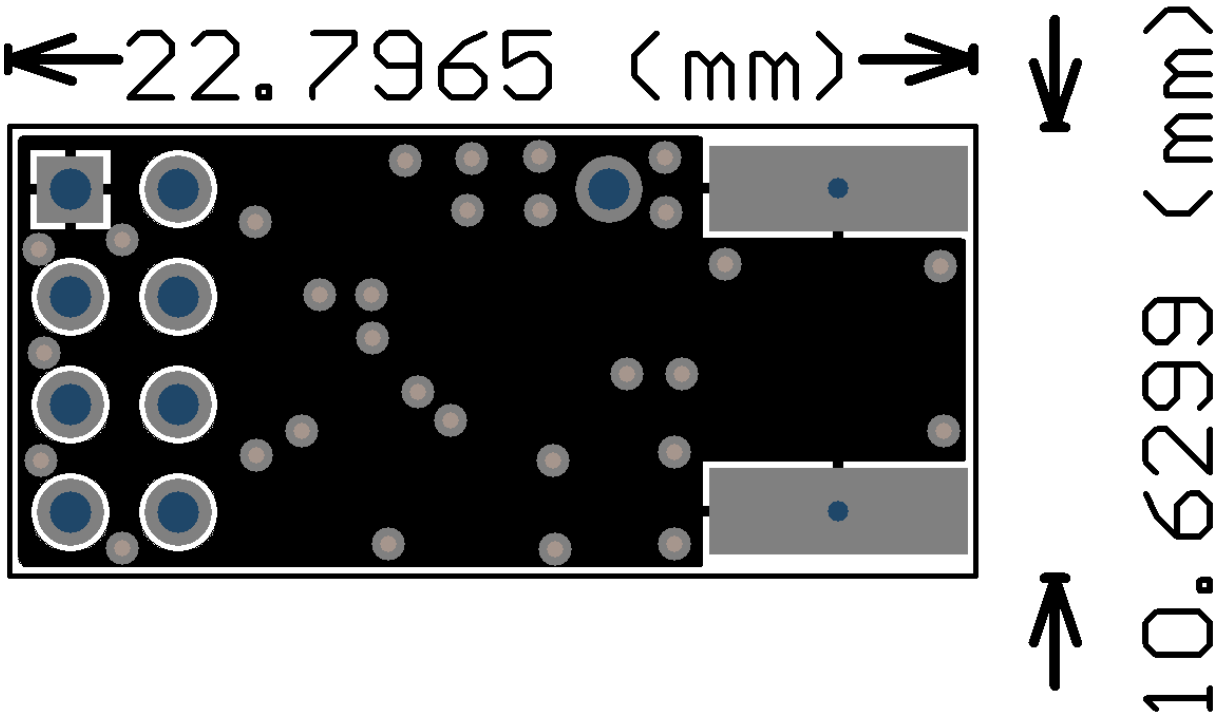


Figure 36. Bottom Layer (Module with OFM crystal and SMA connector)

Appendix E - Stationary disturbance detection

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 IRQ 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 a 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 or jammed by broadband signals like WLAN), it may continue on the same frequency channel, but you must perform other adjustments (a dummy write to the RF_CH clears the PLOS_CNT).

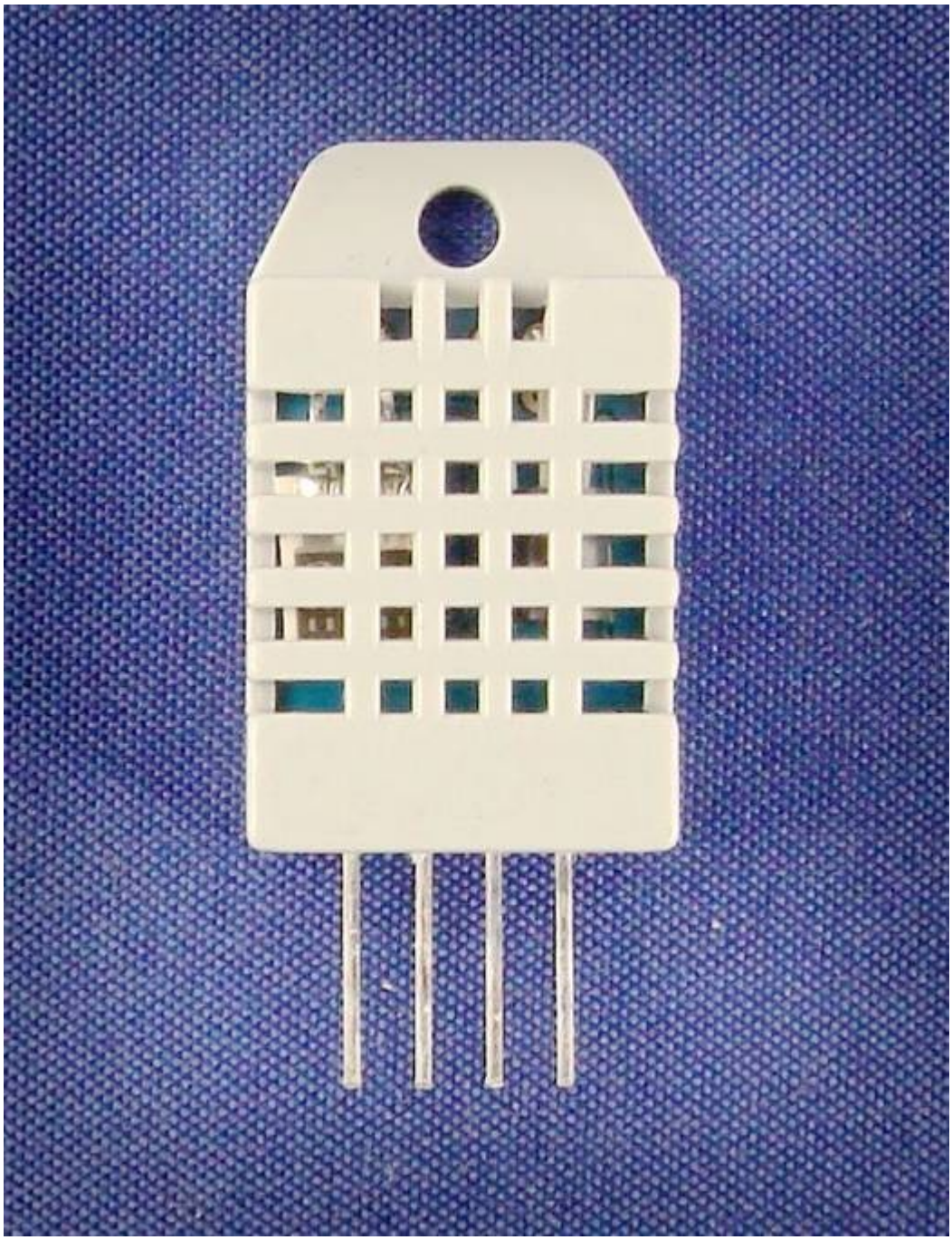
ANEXO 6
Sensor DHT 22

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

Digital-output relative humidity & temperature sensor/module

DHT22 (DHT22 also named as AM2302)



Capacitive-type humidity and temperature module/sensor

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn

Aosong Electronics Co.,Ltd

Your specialist in innovating humidity & temperature sensors

1. Feature & Application:

* Full range temperature compensated * Relative humidity and temperature measurement
* Calibrated digital signal *Outstanding long-term stability *Extra components not needed
* Long transmission distance * Low power consumption *4 pins packaged and fully interchangeable

2. Description:

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions.

Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

4. Dimensions: (unit----mm)

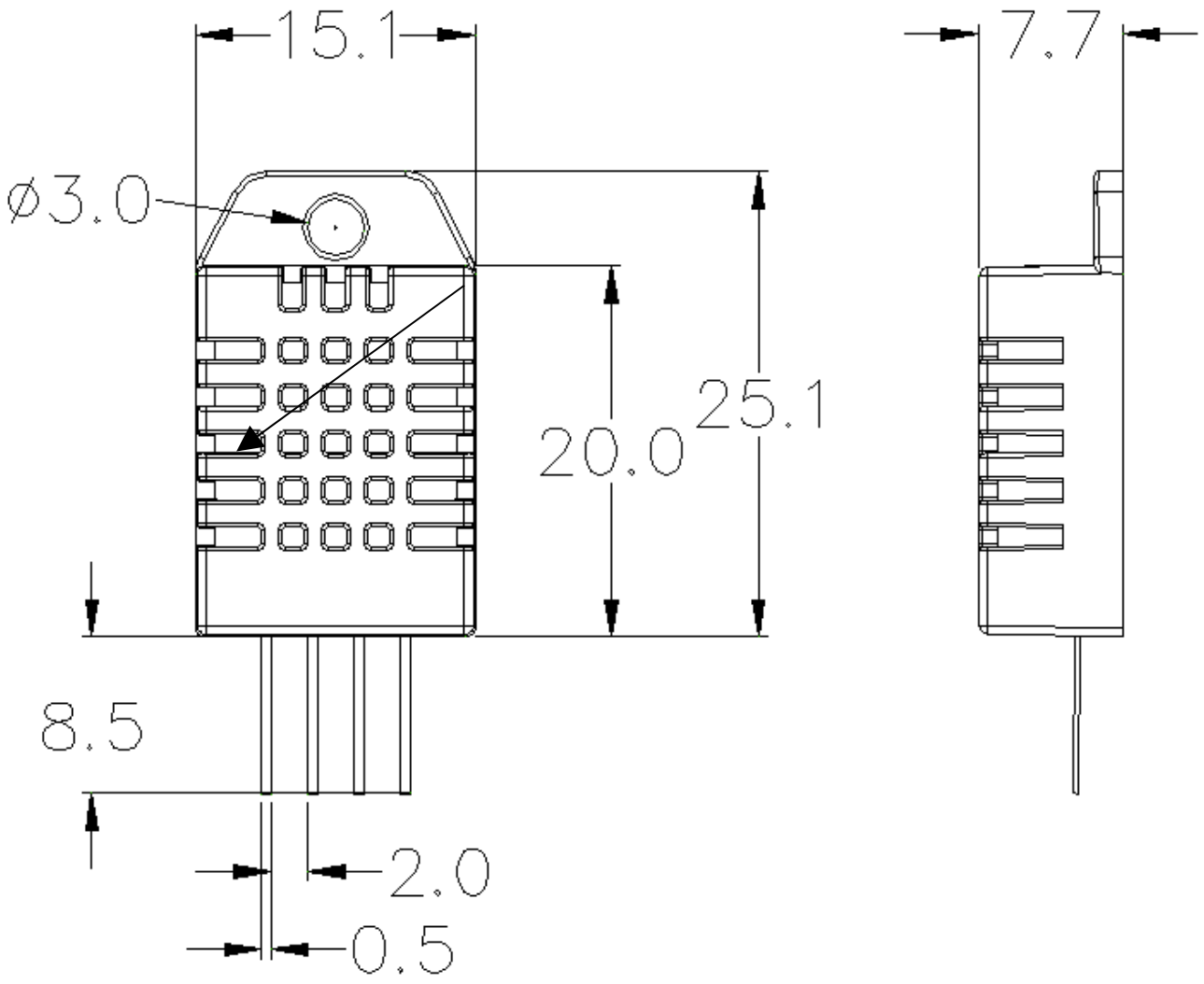
1) Small size dimensions: (unit----mm)

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn



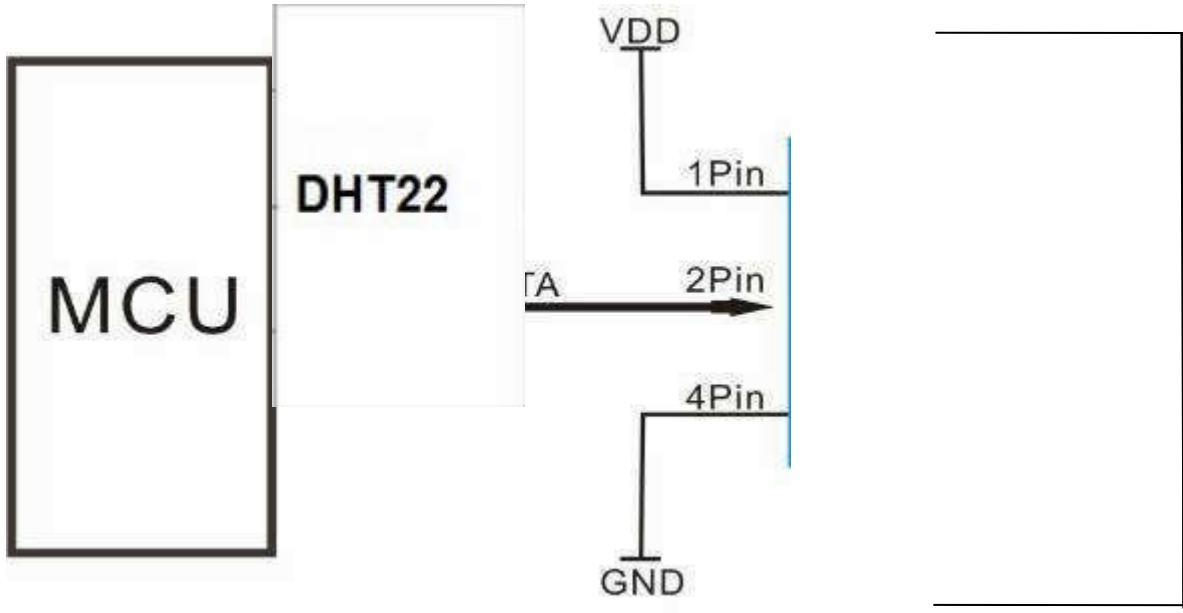
Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD----power supply
2	DATA--signal
3	NULL
4	GND

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn

5. Electrical connection diagram:



3Pin---NC, AM2302 is another name for DHT22

6. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-6V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

Single-bus data is used for communication between MCU and DHT22, it costs 5mS for single time communication.

Data is comprised of integral and decimal part, the following is the formula for data.

DHT22 send out higher data bit firstly!

DATA=8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data+8 bit check-sum

If the data transmission is right, check-sum should be the last 8 bit of "8 bit integral RH data+8 bit decimal RH data+8 bit integral T data+8 bit decimal T data".

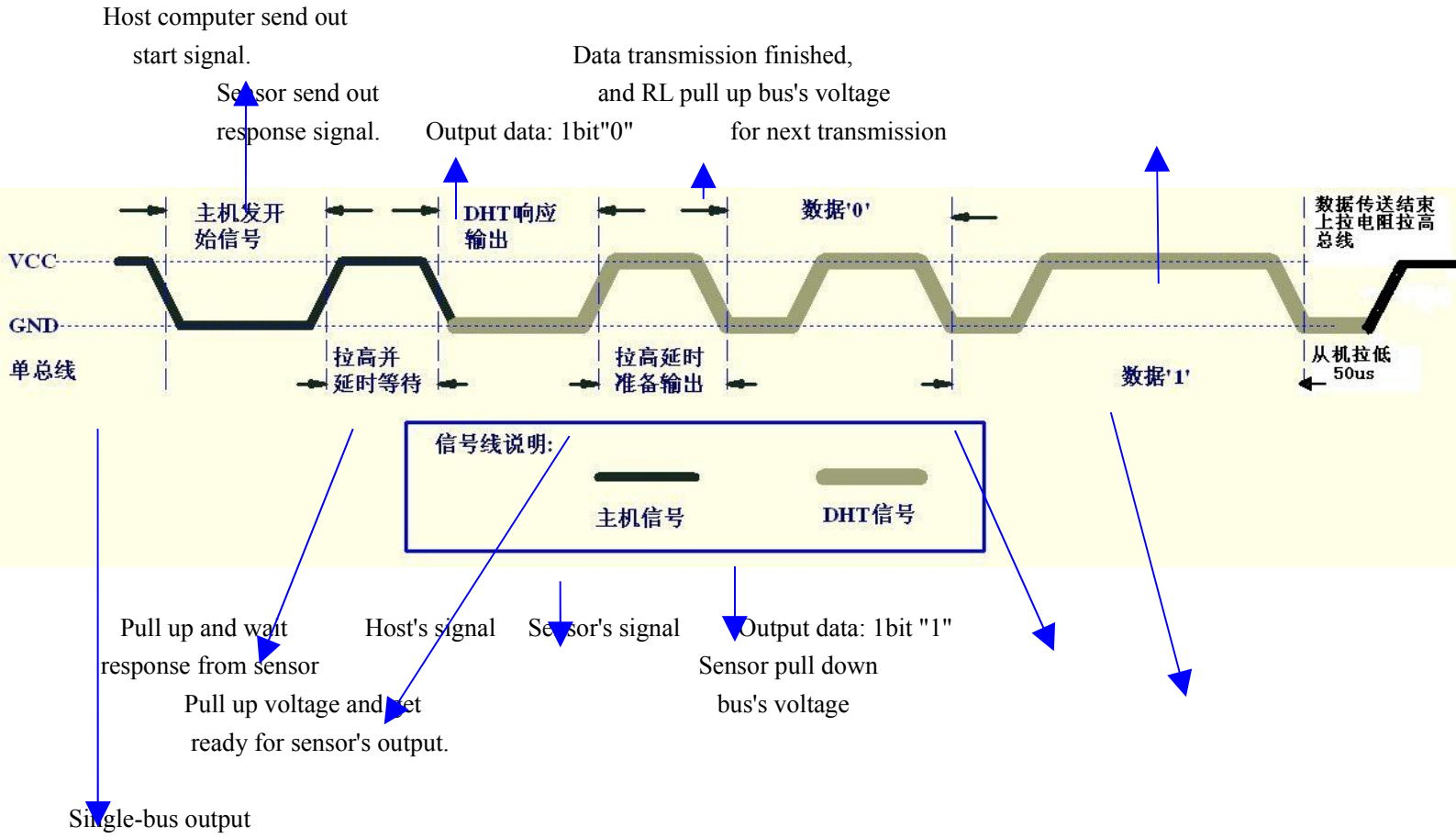
When MCU send start signal, DHT22 change from low-power-consumption-mode to running-mode. When MCU finishs sending the start signal, DHT22 will send response signal of 40-bit data that reflect the relative humidity

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn

and temperature information to MCU. Without start signal from MCU, DHT22 will not give response signal to MCU. One start signal for one time's response data that reflect the relative humidity and temperature information from DHT22. DHT22 will change to low-power-consumption-mode when data collecting finish if it don't receive start signal from MCU again.

1) Check bellow picture for overall communication process:



---

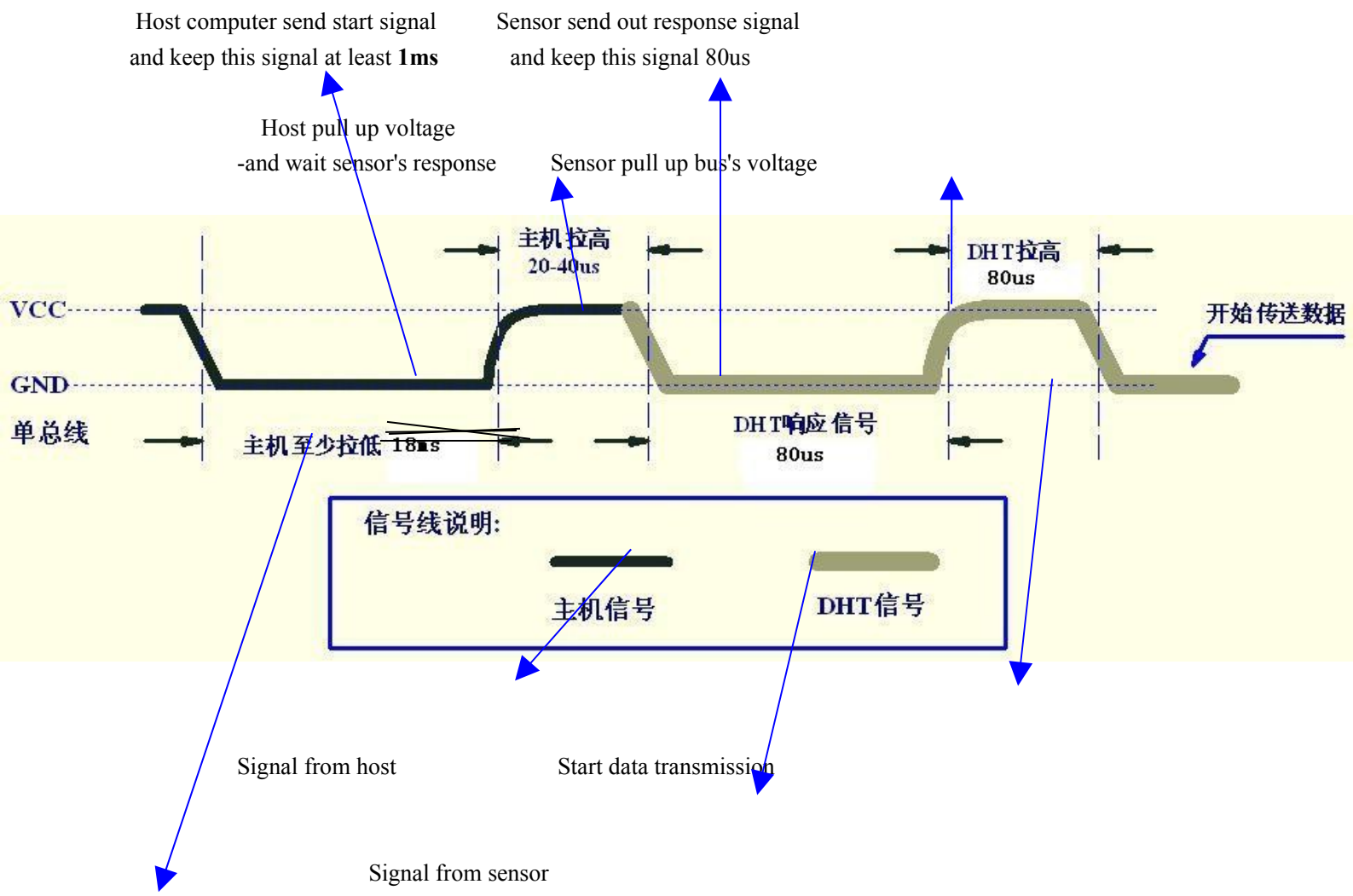
2) Step 1: MCU send out start signal to DHT22

Data-bus's free status is high voltage level. When communication between MCU and DHT22 begin, program of MCU will transform data-bus's voltage level from high to low level and this process must beyond at least 1ms to ensure DHT22 could detect MCU's signal, then MCU will wait 20-40us for DHT22's response.

Check bellow picture for step 1:

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn



Single-bus signal

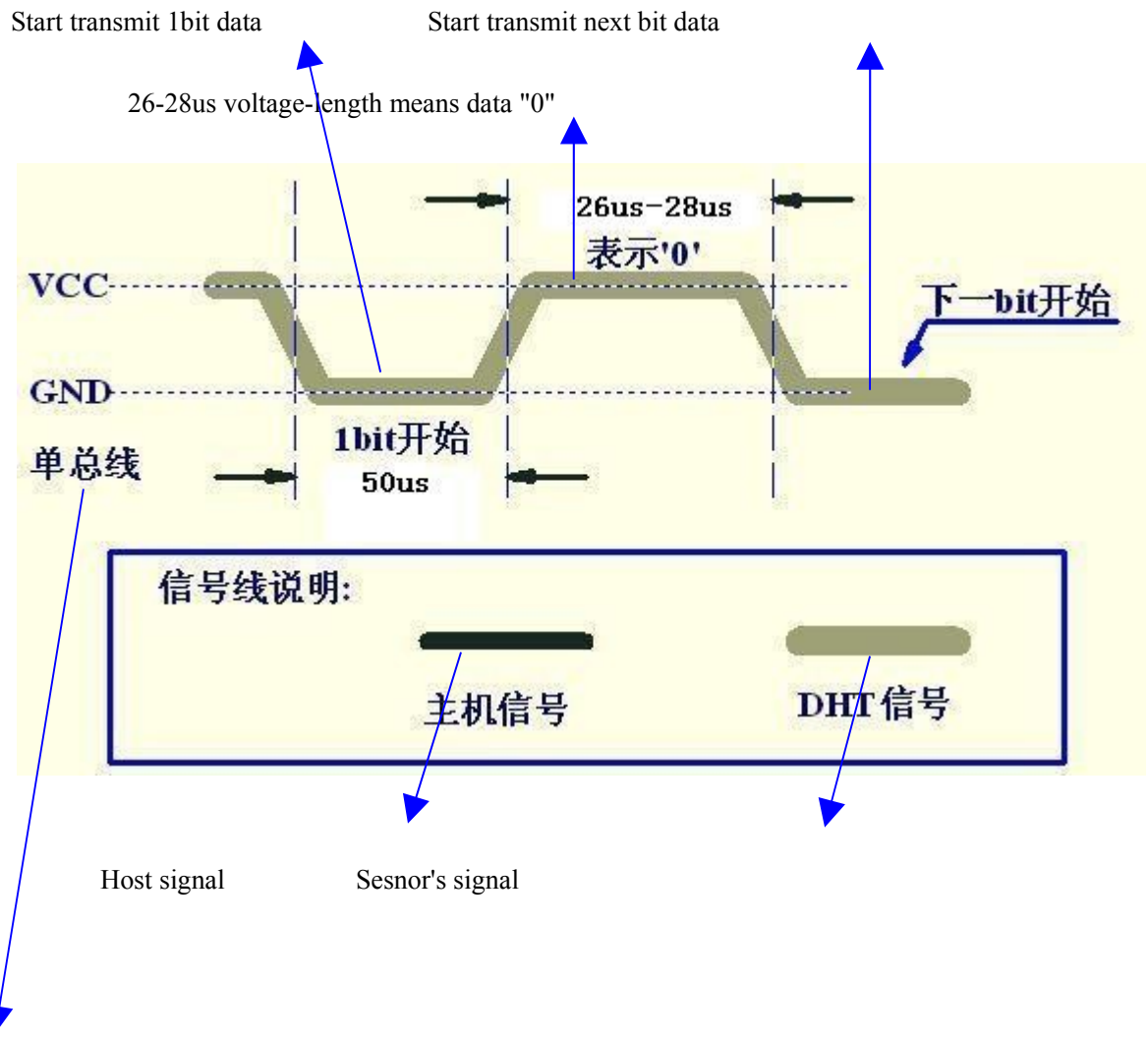
Step 2: DHT22 send response signal to MCU

When DHT22 detect the start signal, DHT22 will send out low-voltage-level signal and this signal last 80us as response signal, then program of DHT22 transform data-bus's voltage level from low to high level and last 80us for DHT22's preparation to send data.

Check bellow picture for step 2:

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn



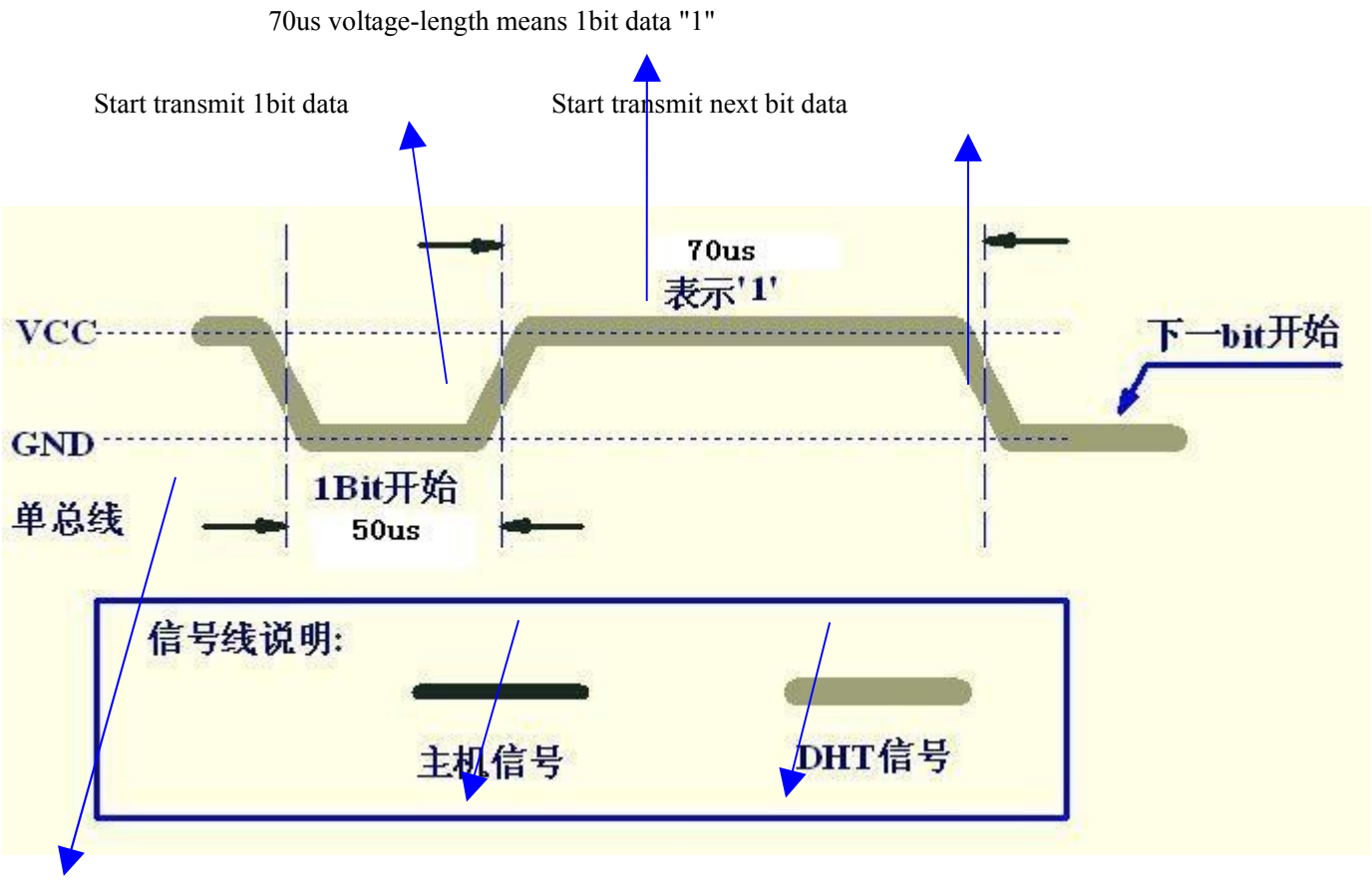
Step 3: DHT22 send data to MCU

When DHT22 is sending data to MCU, every bit's transmission begin with low-voltage-level that last 50us, the following high-voltage-level signal's length decide the bit is "1" or "0".

Check bellow picture for step 3:

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn



Host signal Sesnor's signal

Single-bus signal

If signal from DHT22 is always high-voltage-level, it means DHT22 is not working properly, please check the electrical connection status.

7. Electrical Characteristics:

Item	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

*Collecting period should be : >2 second.

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn

8. Attentions of application:

(1) Operating and storage conditions

We don't recommend the applying RH-range beyond the range stated in this specification. The DHT22 sensor can recover after working in non-normal operating condition to calibrated status, but will accelerate sensors' aging.

(2) Attentions to chemical materials

Vapor from chemical materials may interfere DHT22's sensitive-elements and debase DHT22's sensitivity.

(3) Disposal when (1) & (2) happens

Step one: Keep the DHT22 sensor at condition of Temperature 50~60Celsius, humidity <10%RH for 2 hours;

Step two: After step one, keep the DHT22 sensor at condition of Temperature 20~30Celsius, humidity >70%RH for 5 hours.

(4) Attention to temperature's affection

Relative humidity strongly depend on temperature, that is why we use temperature compensation technology to ensure accurate measurement of RH. But it's still be much better to keep the sensor at same temperature when sensing.

DHT22 should be mounted at the place as far as possible from parts that may cause change to temperature.

(5) Attentions to light

Long time exposure to strong light and ultraviolet may debase DHT22's performance.

(6) Attentions to connection wires

The connection wires' quality will effect communication's quality and distance, high quality shielding-wire is recommended.

(7) Other attentions

* Welding temperature should be bellow 260Celsius.
* Avoid using the sensor under dew condition.
* Don't use this product in safety or emergency stop devices or any other occasion that failure of DHT22 may cause personal injury.

Thomas Liu (Business Manager)

Email: thomasliu198518@yahoo.com.cn